Matching Theory Under Uncertainty

David Wajc

CMU-CS-20-125

August 2020

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Bernhard Haeupler, Chair
Anupam Gupta
R. Ravi
Cliff Stein, Columbia
Ola Svensson, EPFL

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy.*

This research was sponsored by the National Science Foundation award numbers: CCF-1910588, CCF-1814603, CCF-1618280, CCF-1527110, CCF-1750808; and by the Alfred P. Sloan Foundation Research Fellowship award number FG201911557. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity

Keywords: Matching, Matching Theory, Uncertainty, Online Algorithms, Dynamic Algorithms, Streaming Algorithms

To my family

Abstract

Traditionally, optimization in computer science has been studied in the full information setting: data is collected, a program is run, and then the output is used. However, the increasing pervasiveness of user-facing applications is increasingly shifting the focus to computation under *incomplete* information: data is generated continuously by users, who expect their new data to quickly affect the externalized solution. This modern computational paradigm motivates a renewed interest in computation under *uncertainty* (about the input), including online, dynamic and streaming algorithms.

Many problems providing the renewed impetus for studying algorithms under uncertainty come from the field of *matching theory*—the study of pairing agents/items. Examples abound, arising in disparate applications, from ride-sharing apps, to Internet advertising, to online gaming. This motivates the study of *matching theory under uncertainty*. Moreover, the study of matching theory has historically played a key role in the development of immensely influential techniques for computation more broadly. An additional motivation for studying matching theory under uncertainty, then, is its potential to provide similar fundamental insights for computation under uncertainty more broadly.

In this thesis we answer several longstanding open problems in the area of matching theory under uncertainty, and hint at some methods with potential broader applicability to computation under uncertainty. This again illustrates the pivotal role of matching theory, this time in modern settings. In Part I, we study *online algorithms*; here, the input is revealed in parts, in some adversarial order, in the form of *requests*, to which we must respond immediately and irrevocably. In Part II, we study online algorithms under *structural* and *stochastic* assumptions which are motivated by information about practical inputs of interest, allowing for better guarantees than possible for worst-case inputs. Finally, in Part III, we study *dynamic algorithms*, where the input is constantly changing, and the algorithm's choices, while not irrevocable, must be quick; in the same part, we study *streaming algorithms*, motivated by big-data applications, where choices are not irrevocable, but are restricted to only using a limited amount of memory compared to the (massive) input size.

Acknowledgments

My PhD would not have looked the same without a number of brilliant, energetic, and generous people who shared some of these qualities with me during my studies. Here I would like to thank the people who made grad school the fantastic experience it has been for me.

First and foremost, I would like to thank my advisor, Bernhard Haeupler, for his endless support. From sending me to conferences before I really knew what Theory *is*, to supporting my numerous academic visits in search of collaborators and new problems, Bernhard always selflessly had my back, while being hand-off enough to give me room to grow as a researcher. Almost paradoxically, Bernhard also exemplifies what a super hands-on collaborator should look like: from impromptu day-long research meetings to emails at all times of day (and night), Bernhard always seems to have endless energy for research. I can only hope to take some of these great qualities as an advisor and researcher with me going forward.

Next, I would like to thank my other thesis committee members: Anupam Gupta, R. Ravi, Cliff Stein, and Ola Svensson. Thank you for your time and feedback on this thesis, and advice about academia in general. I have learned a lot from you about research over the years: from picking problems, to distilling a problem to its essence, to presentation of the results. I look forward to continuing to learn from you in the future, and hopefully to collaborate again (or in Ravi's case, to collaborate, period).

I would also like to thank my hosts during different highly enjoyable visits and internships: Nitish Korula at Google Research NYC, Seffi Naor at the Technion, and Ola Svensson at EPFL. I would especially like to extend my thanks to the Simons Foundation for making the Simons Institute for Theoretical Computer Science at Berkeley the Mecca of TCS, where experts from around the world can spend time together and advance the field. Personally, I have gotten much from my three prolonged visits at Simons, and consider my first such visit, during the Algorithms and Uncertainty program, as my "academic birth". I was pleased to hear recently that the Simons Foundation will continue to support the Institute for another decade. Our community will be intellectually richer for it. Thank you.

Research is a fun process, made all the more so by sharing it and exchanging insights with brilliant colleagues. I am therefore deeply indebted to my coauthors who contributed so much to the journey, as well as the final destination, of my PhD, and for our joint papers (inside and outside of this thesis): Moab Arar, Sayan Bhattacharya, Ilan R. Cohen, Sarel Cohen, Shiri Chechik, Björn Feldkord, Matthias Feldotto, Buddhima Gamlath, Mohsen Ghaffari, Fabrizio Grandoni, Anupam Gupta, Guru Guruganesh, Bernhard Haeupler, D. Ellis Hershkowitz, Michael Kapralov, Amit Kumar, Roie Levin, Andreas Maggiori, Joseph (Seffi) Naor, Binghui Peng, Ariel D. Procaccia, Sören Riechers, Cliff Stein, Ola Svensson, Hanrui Zhang, and Goran Zuzic.

The social and group-forming aspects of collaborative research were mirrored by the social bonding within CMU, and in particular within the theory group. I consider myself extremely fortunate to have shared the grad school experience with such fun, energetic and driven people, most of whom would also gladly switch abruptly from talking about sports or politics to some random mathematical theorem. I will especially remember fun times with Sarah Allen, Dan Anderson, Ainesh Bakshi, Naama Ben-David, Vijay Bhattiprolu, Laxman Dhulipala, Bailey Flanigan, Paul Goelz, Guru Guruganesh, Nika Haghtalab, Ellis Hershkowitz, Raj Jayram, Angela Jiang, Ryan Kavanaugh, Euiwoong Lee, Jason Li, Jamie Morgenstern, Jakub Pachocki, Ram Raghunathan, Nicolas Resch, Sahil Singla, Ameya Velingker, Ellen Vitercik, Ruosong Wang, Colin White, David Witmer, John Wright, Xinyu Wu, Yu Zhao, and Goran Zuzic. Special shout out to Naama, Bailey and Paul for forming the best "quaranteam", without which quarantine would have been impossible to stand.

I would also like to thank the CSD and especially the theory group's faculty for creating such a welcoming research environment for their students. In particular I would like to thank Anupam Gupta, Mor Harchol-Balter, and Ariel Procaccia for lively conversations and sage advice. I would also like to thank the administrative staff at CMU, and in particular Deb Cavlovich, for making the bureaucratic side of academia such a smooth experience for CSD's students. Any organization can only dream of having someone running things as smoothly as Deb.

Outside of CMU, I would like to thank my friends (in the US and abroad), too many to list here. Spending time together, whether virtually or in person, sometimes after months or years apart, as though no time has elapsed, is one of my greatest joys in life.

Finally, I would like to thank my family for all their support and love: my parents, for providing such a nurturing environment growing up, and supporting me all the way across the globe; my sisters Maya and Bebelle, and my nephews and nieces who fill me with happiness: Joy, Zazoo, Ciel, Ido, Ella and Noga. Last, and definitely not least, I would like to thank Naama: my life is cleanly split into the period before and after having met you. In these times of great uncertainty, you are my one certainty. Thank you for everything.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Uncertainty is all around us; as individuals, we regularly have to make decisions with only partial information about the future.[1] Companies likewise increasingly have to make decisions under similar circumstances, due to the increasing prevalence of user-facing applications. In such applications, inputs that are *unknown* a priori are revealed in real time as they are generated by users. These users then expect prompt response to their data's effect on the problem input. Such dynamics are inherent to ride-sharing apps when matching drivers and passengers, Internet ad platforms when allocating ad slots to advertisers, and online gaming platforms when matching players.

A common feature of the above applications is their concern with pairing agents with agents, or agents with items. These applications are therefore all examples of problems in the area of *matching theory*. This area of combinatorial optimization, which has wide-ranging applications, most notably in economics and market design, has played a "catalytic role" in developing techniques and fundamental concepts in computation more broadly—to paraphrase Lovász and Plummer [200]. In particular, research on matching theory under full information has foreshadowed immensely influential techniques and concepts more broadly, including the primal-dual method [190], the field of polyhedral combinatorics [87], and the equation of polytime computability with tractability [88]. For many matching-theoretic problems, we know efficient exact algorithms, under full information. Unfortunately, these algorithms are not applicable when we must respond urgently to changes to partial input. In fact, this uncertainty and urgency makes problems *provably* harder than their full-information counterparts, ruling out all but *approximate* algorithms. The objective when considering models under uncertainty, then, is to guarantee the best achievable approximation compared to the full-information optimal solution.

Motivated by the abundance of applications involving matching theory in models of computation under uncertainty, and the pivotal role which matching theory has played in the development of fundamental techniques and notions in computer science, this thesis focuses on the intersection of these areas: *matching theory under uncertainty*. In this thesis, we answer several long-standing open problems in the area of matching theory under uncertainty, presenting improved online, dynamic and streaming algorithms for matching-theoretic problems in these models, or proofs that no such improved algorithms exist.

[1]Special thanks to COVID-19 for making this point abundantly apparent.

In the following three sections, we discuss our main contributions, contrasting them with prior state-of-the-art. In Section 1.4 we summarize these results, and briefly discuss our unifying techniques. In Section 1.5, we give a brief bibliography. We conclude this chapter with a discussion of how best to read this thesis, in Section 1.6.

1.1 Online Algorithms—Hedging One's Bets

In online problems, the algorithm's input is revealed incrementally in the form of *requests*, to which an algorithm responds with immediate and irrevocable decisions. Can such online algorithms' output be competitive with the hindsight-optimal solution? How well can such algorithms hedge their bets, by preparing for all possible eventualities? These are the kinds of questions we address in Part I.

Online matching under general arrivals (Chapters 3 and 4): A classic online problem, introduced by Karp et al. [179] in 1990, is the online bipartite matching problem. In this problem, impatient agents (e.g., passengers in a ride-sharing app) arrive and must be matched to compatible agents (e.g., drivers). As users are impatient and are likely to turn off the app in favor of a competing app, such matches must be made immediately and irrevocably. The abstract problem modeling this scenario consists of an (unknown) bipartite graphs, with vertices arriving on one side of the graph, together with their edges, and these must be matched (or left unmatched) forever upon arrival. The objective is to match as many vertices (serve as many ride requests) as possible. The trivial greedy online algorithm is $1/2$ competitive with respect to the hindsight optimum, and no deterministic algorithm can do better. Can randomization help us do better? Indeed it can! Karp et al. presented a beautiful randomized $(1 - 1/e)$-competitive algorithm, which they show is optimal among all algorithms.

In their seminal work, Karp et al. [179] raised the question of whether similar positive results are possible for *general graphs* under vertex arrivals, as generalizing their results for bipartite graphs. For such a generalization, vertices in a (general) graph arrive over time, revealing edges to their previously-arrived neighbors, and an algorithm can only match a new edge containing the newest-arrived vertex in the graph. Subsequent work, starting with Mehta's survey on online matching [206], suggested and studied a similarly natural generalization—that of online matching under *edge arrivals*. Here, edges arrive over time, with each edge either being added to the matching or refused, immediately and irrevocably upon arrival. This corresponds to perishable matching opportunities, for example fleeting collaboration opportunities between teams in a company. Greedy is $1/2$ competitive in both these more general models, too, and remains the optimal deterministic algorithm. Can one do better by using randomization?

Attempts at addressing the above question for these general arrival models were made over the years, under numerous relaxations and restrictions of the problem [55, 66, 95, 146, 163, 165, 165, 166, 191, 259, 267]. The problem was proven harder under these more general arrival models [55, 95, 165], in the sense that no randomized algorithm can achieve the $(1 - 1/e)$ competitive ratio of [179] for online bipartite matching. However, an answer to the question of whether randomized algorithms can outperform deterministic algorithm's optimal competitive ratio of $1/2$, in either of these models, remained elusive.

We answer the above question, for both of these general arrival models. In Chapter 3 we show that for matching under edge arrivals, greedy is essentially optimal, as no randomized algorithm is better than $1/2 + o(1)$ competitive; that is, we show that randomization (and even considering fractional relaxations of the problem) does not help for this problem. In Chapter 4 we show that for vertex arrivals randomization does help, and we present a $(1/2 + \Omega(1))$-competitive algorithm for this more general model. For our positive results, we show how to randomly round the fractional matching of a $(1/2+\Omega(1))$-competitive fractional algorithm of [267], without incurring too much loss in the competitive ratio. This result, as well as many other problems for which fractional competitive algorithms are known but randomized ones are not, motivate further study of online rounding, which we turn to in Chapter 5.

Rounding fractional bipartite matchings online (Chapter 5): A common design paradigm for approximation and online algorithms is to solve some fractional relaxation of the problem, and round this solution to a viable integral solution (online). For example, for offline matching in bipartite graphs, a classic rounding scheme of Gandhi et al. [122] allows to round a fractional matching $\vec{x} \in \mathbb{R}^{|E|}_{\geqslant 0}$ (for which $\sum_{e \ni v} x_e \leqslant 1$ for all $v \in V$), by outputting an integral (randomized) matching $\mathcal{M}$, such that each edge is matched in $\mathcal{M}$ with probability $\Pr[e \in \mathcal{M}] = x_e$. We note that rounding each edge independently does not result in a legal matching, and so the edges are rounded in a *dependent* manner by [122]. This dependent rounding scheme has been highly influential, with numerous applications and extensions over the years.

For *online* bipartite matching, however, the design paradigm of solving a fractional relaxation followed by dependent randomized rounding has gone largely overlooked. As we show in Chapter 5, there is good reason for this, since an *online* dependent rounding scheme paralleling that of [122] does not exist: we show that no online rounding scheme for matching problems can avoid losing a multiplicative factor depending on the magnitude of the fractional values of the input online fractional matching $\vec{x}$, specifically $1-\Omega(\sqrt{|x|_\infty})$. In our proof we show that for Δ-regular graphs (i.e., graphs where each vertex has degree Δ), despite the existence of a 1-competitive fractional matching algorithm, which assigns $1/\Delta$ to each edge, no randomized online algorithm is better than $1 - \Omega(1/\sqrt{\Delta})$ competitive. In Chapter 5, we present an online dependent rounding scheme which allows us to essentially meet this bound, presenting a $1 - \tilde{O}(1/\sqrt{\Delta})$-competitive randomized online matching algorithm for bipartite graphs. Such bounds were previously only known under stochastic or random-order arrivals [21, 176], and a bound converging to one as Δ increases is provably impossible for deterministic algorithm under adversarial arrivals, as we show in Chapter 7.

Beyond the above qualitative result for regular bipartite graphs—the most widely-studied graph family in the matching theory literature—the results of this chapter point at a more nuanced picture of dependent rounding in online settings. As mentioned before, we show the existence of families of online fractional matchings $\vec{x}$ such that any randomized matching algorithm outputting a matching $\mathcal{M}$ must have (many) edges e for which $\Pr[e \in \mathcal{M}] < x_e \cdot (1 - \Omega(\sqrt{|x|_\infty})$. In Chapter 5, we present an online dependent rounding scheme which matches this lower bound, up to the exact polynomial dependence on $|x|_\infty$. Our rounding scheme, on any fractional bipartite matching $\vec{x}$ presented online, outputs a matching $\mathcal{M}$ that matches each edge e with probability $\Pr[e \in \mathcal{M}] = x_e \cdot \left(1 - \tilde{O}\left(\sqrt[3]{|x|_\infty}\right)\right)$. This online dependent rounding scheme seems like a powerful tool, of wider applicability. Indeed, in Chapter 6, we use this tool to resolve an open problem concerning the online *edge coloring* problem, addressed below.

Online Edge Coloring (Chapter 6): A "dual" problem to that of computing a large matching is the edge coloring problem, i.e., the problem of decomposing the graph into a minimum number of matchings (colors). This corresponds to scheduling of two-agent tasks. For example, this is the problem used to schedule two-team sports games on different days, so that no team in the league plays more than once a day. König [184], in a century-old cornerstone result of matching theory, proved that every bipartite graph of maximum degree Δ can be edge colored using Δ color. (Clearly, no fewer colors suffice.) Moreover, this proof can be extended to yield an efficient algorithm which computes such a coloring.

In an online setting, where bipartite edge coloring can be used to model switch scheduling for Internet routers [5], the problem seems more challenging. In particular, in 1992 Bar-Noy et al. [25] showed that for small $\Delta = O(\log n)$, the trivial greedy online algorithm, which is $(2 - o(1))$ competitive, is optimal, at least for bounded-degree graphs. They further conjectured that better algorithms exist for higher-degree graphs, with $\Delta = \omega(\log n)$.

In Chapter 6 we resolve this conjecture affirmatively for the bipartite one-sided vertex arrival model of Karp et al., giving an optimal $(1 + o(1))$-competitive online edge coloring algorithm for graphs with maximum degree $\Delta = \omega(\log n)$, known a priori. On the other hand, we prove a dichotomy between the setting were Δ is known and when it isn't, proving that not knowing Δ makes the problem strictly harder, and no $(\frac{e}{e-1} - \Omega(1))$-competitive algorithm exist in this case. On the other hand, we show that in this case, too, the greedy algorithm is suboptimal for large $\Delta = \omega(\log n)$, for which we present an optimal $(\frac{e}{e-1} + o(1))$-competitive online algorithm. One key ingredient for our algorithms (both for the known and unknown Δ case) are our online rounding scheme for fractional matchings from Chapter 5, possibly hinting at further applicability of this scheme for other problems.

1.2 Online Algorithms—Beyond the Worst Case

A common criticism of online algorithms and competitive analysis is their pessimism. Focusing on worst-case inputs often results in guarantees which are not useful for inputs observed in practice. This has made online algorithms something of a poster child for the agenda of *beyond-worst-case analysis* (see [242, 243]), which aims to focus precisely on such practical inputs. One such approach assumes *structural* properties of inputs observed in practice. Another such approach is the assumption of various *stochastic* online models, where input is drawn from some distribution, or randomly permuted. This approach is strongly motivated by problems generated by millions of users over a long period of time, for which statistical information is often available based on prior data. What kind of improved guarantees can online algorithms obtain given such structural or distributional information? We address such questions in Part II.

Internet advertising (Chapter 7): Consider the problem of selling ads on the Internet. At the beginning of a day, advertisers announce their daily advertising budget to an ad exchange, and how much they are willing to pay to have their ad displayed to users of various market segments they target. Whenever a user u visits a webpage, an ad can be displayed in an ad slot, which must be sold immediately and irrevocably to an advertiser whose budget has not yet been exhausted. This extension of online bipartite matching, called the *AdWords* problem [207], is the driving economic force behind free content on the Internet. As such, it has been studied extensively over the years. In particular, experimental results have shown that simple heuristic online algorithms do surprisingly well—better than theory would suggest. How does one explain this empirical success?

In Chapter 7, we present a possible theoretical explanation of this practical success. Informally, we study instances with imbalanced thicknesses on the advertisers' and ad slots' sides. In more detail, we study the online bipartite matching and AdWords problems where advertisers (offline nodes) are interested in a large market segment, corresponding to *many* ad slots, at least some k, and ad slots (online nodes) mostly correspond to users in few targeted market segments at any point in time, say at most $d \leqslant k$. For such practical inputs, we show that greedy algorithms achieve a competitive ratio of $1 - \Theta(\frac{d}{k})$, tending to *one* as the imbalance, k/d, tends to infinity. Moreover, guided by this intuition, we show more elaborate algorithms whose competitive ratios tend *exponentially* faster to one as this imbalance grows, and show that this convergence rate is optimal among all deterministic algorithms for such inputs. The intuition driving our algorithm for online bipartite matching was used in practice by engineers at Google (private communication), and so our work serves as a theoretical justification of the empirical success of heuristics used in practice.

Online Metric Matching (Chapter 8): Consider a case of the ride-sharing problem where drivers may only drive one passenger a day. Passengers arrive at different locations of the city, requesting a ride. When a passenger arrives, she must be immediately and irrevocably matched to a driver. The ride-sharing app wishes to minimize the travel time of drivers to the passengers they are matched to. This is an instance of online min-cost perfect matching, where edge costs are given by a metric. This problem, first studied in the 90s [172, 182], is known to have an optimal competitive ratio $\mathrm{poly}(\log n)$ (the exact polylogarithmic term remains open). These worst-case bounds seem overly pessimistic and not very useful practically. What can be said if the input is generated by a stochastic process? For a request sequence which is randomly permuted, the optimal competitive ratio is known to be $\Theta(\log n)$ a result of [239]. While this is possibly better than under adversarial arrivals, it is still far from being practically useful. What can be done if we know more about the random process generating the requests? For example, what if each request is drawn independently from a known distribution (learned by the ride-sharing company over time)?

In Chapter 8, we address this problem, and present an $O((\log\log\log n)^2)$-competitive algorithm, i.e., *doubly exponentially* better than possible under adversarial arrivals, or even random-order arrivals, and arguably closer to a practically useful guarantee. In fact, we show that on structured metrics of relevance, our algorithm is $O(1)$ competitive, and we conjecture that our algorithm is $O(1)$ competitive for all metrics.

Random-Order Online Edge Coloring (Chapter 9): Returning to the online edge coloring problem, we recall that, by Chapter 6, under one-sided vertex arrivals in bipartite graphs, an edge coloring using $(1 + o(1))\Delta$ colors is possible to compute online for graphs with $\Delta = \omega(\log n)$. Such results are unknown under the more fine-grained *edge arrival* model. However, given the results of Chapters 3 and 4 for online matching, it is not implausible that the edge-coloring problem is strictly harder under such a fine-grained arrival model. Nonetheless, results matching those of Chapter 6 were conjectured for this more general arrival model by Bar-Noy et al. [25] some 30 years ago. This conjecture remains unresolved. To make progress on this problem, Motwani and co-authors revisited this problem under *random-order* edge arrivals [5, 22]. Both papers proved that a coloring using fewer than the trivial $2\Delta - 1$ colors (obtained by the greedy algorithm), can be achieved for graphs of high enough maximum degree Δ. Aggarwal et al. [5] gave a $(1 + o(1))$-competitive algorithm for n-node (multi)graphs with maximum degree $\Delta = \omega(n^2)$, and Bahmani et al. [22] gave a 1.26-competitive algorithm for simple graphs with $\Delta = \omega(\log n)$. Whether an algorithm achieving the best of both worlds with regards to these two works exists, thus resolving the conjecture of [25] under random-order edge arrivals, remained open.

In Chapter 9, we answer this question in the affirmative, presenting such an algorithm, by extending and adapting an algorithm of [231] from another area of computation under uncertainty—distributed algorithms—and showing how to implement this algorithm in our model.

1.3 Dynamic and Streaming Algorithms

So far, when discussing computation under uncertainty, we focused on online algorithms. However, there are many other computational models where algorithms are required to make decisions based on only partial information regarding the input. Examples include distributed algorithms, local computation algorithms, dynamic algorithms and streaming algorithms, to name a few. In Part III we focus on these models of computation under uncertainty above.

Dynamic Matching (Chapter 10): In the previous section the core difficulties were the urgency and irrevocable nature of choices made. But what if changes to choices made are possible, but must be performed quickly? Dynamic algorithms address precisely such challenges. One problem which has been intensely studied in dynamic settings in recent years is the (approximate) maximum matching problem. Here the goal is to maintain an approximately-maximum matching subject to edge insertions and deletions—referred to as *updates*—in the graph, while spending little computation time per update—referred to as *update time*. The fast worst-case update time guarantees known to date are $\text{poly}(\log n, 1/\varepsilon)$ update time for $(2 + \varepsilon)$-approximate matching [14, 40, 62] (the first of these references is a joint work of the author with Arar et al).

A limitation of the above randomized algorithms, and of all prior randomized dynamic matching algorithms, is their assumption that the update sequence is generated by an *oblivious* adversary, a priori, rather than by an *adaptive* adversary, which chooses the updates based on previous queries' outputs. This oblivious adversary assumption rules out the use of such algorithms in user-facing applications, where updates may be affected by previous queries' outputs. As observed by Mądry [217], the assumption of a non-adaptive update sequence also rules out the black-box usage of such dynamic algorithms to speed up *static* algorithms. As such, a major open problem in the field of dynamic algorithms is understanding whether guarantees achieved by randomized algorithms under the oblivious adversary assumption can be achieved against the stronger, adaptive adversary. In fact, for many problems, including the dynamic matching problem, it was open whether any randomized algorithms which work against adaptive adversaries can outperform (known) deterministic algorithms.

In Chapter 10 we resolve this question, presenting a number of randomized matching algorithms which are the first to work against adaptive adversaries and outperform known deterministic algorithms.

Streaming Algorithms (Chapter 11): Another widely-studied model of computation under uncertainty, motivated by big-data applications, whose data is too large to store in memory, is the *streaming* model. Here, computation is performed while scanning the input elements in an arbitrary order and using little memory—ideally of size proportional to the *output* size, rather than proportional to the (massive) input size. For example, storing $O(n)$ edges of a graph—as opposed to the graph's $O(n^2)$ edges—allows to compute a 2-approximate matching in such a streaming setting. Whether similar approximation is achievable for maximum-*weight* matching was a long-standing open question, resolved in a recent breakthrough by Paz and Schwartzman [232], who presented a $(2+\varepsilon)$-approximate algorithm storing $O(n \log n)$ edges. In joint work with Mohsen Ghaffari [127], we simplified this algorithm's analysis, and obtained a space-optimal algorithm for this problem, which stores only the minimum requisite $O(n)$ edges.

An even more general problem was studied in the literature—that of computing a high-valued matchings according to a *submodular* function, i.e., an objective exhibiting *diminishing returns* [58, 65, 111]. For this more general problem, upper and lower bounds were known, but the gap between these bounds was significant.

In Chapter 11, we improve on prior upper and lower bounds for streaming submodular matching problems. Our algorithms of this section generalize that of [127, 232], and with the right parameters, recreate the same guarantees for the maximum weight matching problem. For the more general submodular problem, we obtain our improved results by extending the (randomized) primal-dual method. Using the same approach, we give a unified analysis of previous algorithms [58, 111]. Our results and analyses hint at wider applicability of the primal-dual method for such submodular problems in the streaming model and beyond.

1.4 Summary of Main Contributions and Techniques

In this thesis, we tackle three common flavors of problems: online matching problems, online edge coloring problems, and matching problems in other models of computation under uncertainty, tackled by dynamic and streaming algorithms. For most of the problems we study, we either break a natural barrier, or present an optimal algorithm (or both). Our main results include the following.

- A characterization of the power of randomization for online matching under general arrivals (Chapters 3 and 4)
- Optimal online matching algorithms for bipartite regular graphs (Chapters 5 and 7)
- Optimal online edge coloring algorithms, both under adversarial and random-order arrivals (Chapter 6 and 9)
- Optimal online matching and AdWords algorithms for inputs arising in practice in Internet advertising applications (Chapter 7)
- Doubly-exponential improvements for stochastic online metric matching (Chapter 8)
- The first randomized dynamic matching algorithms which work against adaptive adversaries (Chapter 10)
- Numerous improved streaming algorithms for submodular matching problems (Chapter 11)

Our results are summarized more precisely and concisely in tabular form in Table 1.1, Table 1.2, Table 1.3 and Table 1.4. Interspersed between these tables are slightly more detailed summaries and discussions of these results. We conclude this section by discussing some key (overarching) techniques which helped us achieve these results, in Section 1.4.1.

Problem	Our Competitive Ratio	Best Prior Bounds
Edge arrivals	$1/2 + O(1/n)$, † Chapter 3	$[1/2 + \Omega(2^{-n}), 0.586)$ [55, 191], [165]
General vertex arrivals	$1/2 + \Omega(1)$ Chapter 4	$[1/2 + \Omega(2^{-n}), 0.592)$ [55, 191], [55]
Bipartite d-regular graphs (randomized)	$1 - \tilde{\Theta}(1/\sqrt{d})$, † Chapter 5	$[1 - 1/e, 1]$ [179]
Bipartite d-regular graphs (deterministic)	$1 - (1 - 1/d)^d > 1 - 1/e$, † Chapter 7	$[1/2, 1]$ [Folklore]
Bipartite (k, d)-bounded graphs (deterministic)	$1 - (1 - 1/d)^k > 1 - (1/e)^{k/d}$, † Chapter 7	$[1/2, 1]$ [Folklore]
Stochastic metric matching	$O((\log \log \log n)^2)$ Chapter 8	$O(\log n)$ [239]

Table 1.1: Our Results for Online Matching Problems.
Tight results, possibly up to $o(1)$ terms, are marked with a †.

For online matching (see Table 1.1), our first two results characterize the power of randomization under general arrival models which generalize the classic bipartite model of Karp et al. [179]. For these more general models, it is known that the optimal achievable competitive ratio for deterministic algorithm is $1/2$. For edge arrivals we prove that randomization cannot help beyond possibly increasing this bound by $o(1)$, while for general vertex arrivals, we show that randomization does help improve the optimal competitive ratio by $\Omega(1)$. Next, we consider online matching in more well-structured graphs, including bipartite d-regular graphs (possibly the most commonly-studied family of graphs in the matching theory literature), and graphs which arise naturally in Internet advertising applications. For these graphs we obtain improved (and indeed, optimal) competitive ratios: for regular graphs, we show a separation between deterministic and randomized algorithms for online matching in d-regular graphs: while for deterministic algorithms, the problem becomes more difficult as d increases, with a competitive ratio tending to $1 - 1/e$ from above as this degree increases, the problem becomes easier for randomized algorithms, with the optimal competitive ratio tending to one. For instances arising in Internet advertising we provide an explanation for empirical ease beyond what is suggested by theory, and provide an optimal algorithm for these inputs. Lastly, for the metric matching problem, we give a doubly-exponential improvement under stochastic arrivals compared to that achievable under random-order arrivals—a bound which naturally carries over to stochastic arrivals. In this result we therefore prove a sharp separation between these arrival models for this problem.

Problem	Our Competitive Ratio	Prior Best Ratio
Bipartite vertex arrivals (known Δ)	$1+o(1)$, † Chapter 6	2 [Folklore]
Bipartite vertex arrivals (unknown Δ)	$\frac{e}{e-1}+o(1)$, † Chapter 6	2 [Folklore]
Random-order edge arrival (known Δ)	$1+o(1)$, † Chapter 9	1.27 [22]

Table 1.2: Our Results for Online Edge Coloring Problems with $\Delta = \omega(\log n)$.
Tight results, possibly up to $o(1)$ terms, are marked with a †.

Our main results for online edge coloring are summarized in Table 1.2. For adversarial one-sided vertex arrivals in bipartite graphs, we show that contrary to the title of [25], the greedy algorithm is **not** optimal for online edge-coloring, provided the maximum degree is super-logarithmic (as conjectured by [25]). More precisely, we present optimal algorithms for both known and unknown Δ regimes, showing that the optimal competitive ratio for both is strictly less than 2. Along the way, we prove that a the problem with unknown Δ is strictly harder than known Δ. For random-order edge arrivals, previously studied by Motwani et al. [5, 22], we also present optimal bounds, showing that under more fine-grained arrival granularity, in the form of edge arrivals (provided in random order) one can achieve a near-ideal $(1+o(1))$-competitive solution.

Main Measure	Update Time	Prior Best
Worst-case update time (for any $2+\varepsilon$ approximation)	poly $\log n$ Chapter 10	$\min\{\sqrt[3]{m}, \sqrt{n}\}$ [42]
Amortized update time (for any $2+\varepsilon$ approximation)	$O(1)$ Chapter 10	poly $\log n$ [43]
Amortized update time (for any $2-\varepsilon$ approximation)	$n^{f(\varepsilon)}$ Chapter 10	$\sqrt[4]{m}$ [38]

Table 1.3: Our Results for Dynamic Matching against Adaptive Adversaries.
The number of edges and nodes is denoted by m and n, respectively.

For dynamic matching, we provide the first approach to obtain randomized algorithms which work against adaptive adversaries, from which we derive a number of algorithms (see Table 1.3). Our results yield significant running time improvements, ranging from polynomial improvements for approximation ratios below 2, to exponential speedups for approximation ratio $2+\varepsilon$.

Finally, we obtain results for matching problems in the streaming model, motivated by big-data applications (see Table 1.4). For this model, we obtain a number of improved results for streaming maximizing submodular objectives under matching constraints. More interestingly, we obtain our algorithmic results using one common unifying approach: an extension of the randomized primal-dual method. We elaborate on this point in Section 1.4.1.

Problem	Our Approximation ratio	Prior best
Monotone submodular matching	$3+2\sqrt{2}\approx 5.828$ Chapter 11	7.75 [58]
Monotone submodular b-matching	$3+2\sqrt{2}\approx 5.828$ Chapter 11	8 [58]
Monotone submodular matching	> 1.914 Chapter 11	$> \frac{e}{e-1}\approx 1.582$ [58, 174]
Non-monotone submodular matching	$4+2\sqrt{3}\approx 7.464$ Chapter 11	$5+2\sqrt{6}\approx 9.899$ [111]
Maximum weight b-matching	$3+\varepsilon$ Chapter 11	$4+\varepsilon$ [73]

Table 1.4: Our Results for Streaming Matching Problems
Lower bounds are marked with a greater-than sign

1.4.1 Recurring Themes and Techniques

In this section we briefly discuss some common recurring themes and techniques throughout this thesis, adding further thematic connections between the different chapters.

Randomized dependent rounding: A frequent algorithmic approximation algorithm approach is randomized rounding [238]. Here, one solves (or approximates) some fractional relaxation of the problem and then rounds this fractional solution. For most combinatorial problems, rounding values independently will not result in a feasible solution, and so values must be rounded in a dependent manner. While such approaches are fairly well understood under full-information settings, they present additional challenges in the uncertain computational models which we study. We overcome such challenges to obtain our results of chapters 4, 5, 6 and 10. Illustrating these challenges, in Chapter 5 we initiate the study of online dependent rounding for matching problems. We show the limitations of such rounding schemes compared to their offline counterparts, in that they must inevitably lose a term which can be thought of as a variance term. A common theme both in the challenges and our solutions for dependent rounding under uncertainty is therefore the need to strive for negative correlation (or at least very weak positive correlation) between edges' or nodes' matched status. To design and analyze dependent rounding procedures which guarantee such correlations, we often rely on the notion of *negative association*, which we discuss in Section 2.4.1.

LP Duality and The Primal-Dual Method: One of the most influential design patterns for approximation algorithms is the primal-dual method. Here, one relies on a linear programming (LP) relaxation, for which one computes a primal and dual solution, using the dual solution as a certificate of the optimality (or approximation ratio) of the primal solution. This method is foreshadowed by Kuhn's Hungarian method for the minimum weight perfect bipartite matching problem [190],[2] and has found numerous applications over the years (see e.g. the surveys [52, 135, 268]). We rely on this method to obtain our results for (k, d)-bounded graphs in Chapter 7. A number of recent results in the online matching literature [97, 143, 162–166, 256] rely on an extension of this method: the *randomized primal-dual method*, introduced by [79] in the context of online matching and its extensions. Here, the (random) dual solution need only be feasible in expectation. In Chapter 11 we further extend this method to submodular objectives, where our key extensions of this method crucially relies on the dual solution being feasible in expectation for a *randomized* LP. Linear programming duality also plays a role in other results throughout this thesis, most prominently in our lower bound proofs of Chapter 3 and Chapter 6.

Interplay between various matching-theoretic problems: In several of our results, we rely on the close relationship between various matching-theoretic problems, motivating the holistic study of such problems. Besides the relationship between matchings and vertex covers—implied by LP duality and the bounded integrality gap of the fractional matching polytope—we obtain a number of our results by studying matchings and edge colorings in tandem. For our online bipartite edge coloring results of Chapter 6, we use randomized matchings to consistently peel off a matching, such that every such matching decreases the graph's maximum degree by roughly one per color. For our dynamic matching algorithms of Chapter 10, we rely on edge colorings of well-chosen subgraphs to sample a small number of colors (matchings) whose union is a sparse subgraph which approximately preserves the maximum matching size (i.e., a matching sparsifier). We further make use of Vizing's edge coloring theorem [261] to bound the quality of some of our matching sparsifiers.

Connections between different models: Motivating a holistic study of algorithms under uncertainty are a number of exchanges of ideas between these different models throughout this thesis. For example, our random-order online edge coloring algorithm of Chapter 9 is a variant and adaptation of a *distributed* edge coloring algorithm, which we show how to implement in this online model. Similarly, as mentioned before, our improved streaming algorithms of Chapter 11 build on (and extend) a technique which has proven useful for online algorithms, namely the randomized primal-dual method. It is the author's hope that some of the techniques presented in this thesis (for example, this extension of the randomized primal-dual method) will similarly transcend the particular computational model for which they were developed,

1.5 Bibliographic Notes

Most of this thesis is based on previously published work.
Chapter 3 and Chapter 4 are based on the following publication.

- [120] "Matching with General Arrivals" (FOCS'19)
  with Buddhima Gamlath, Michael Kapralov, Andreas Maggiori and Ola Svensson.

Chapter 5 is based on a full version of the following publication.

- [69] "Randomized Online Matching in Regular Graphs" (SODA'18)
  with Ilan R. Cohen.

[2]Essentially the same algorithm was presented by Jacobi, whose death precedes the work of Kuhn by more than a century, in a posthumous note [169].

Chapter 6 is based on the following publication.

- [70] "Tight Bounds for Online Edge Coloring" (FOCS'19),
  also invited to Highlights of Algorithms 2020 (HALG'20)
  with Ilan R. Cohen and Binghui Peng.

Chapter 7 is based on the following publication.

- [222] "Near-Optimum Online Ad Allocation for Targeted Advertising" (EC'15),
  also invited to Transactions of Economics and Computation (TEAC'18)
  with Joseph (Seffi) Naor.

Chapter 8 is based on the following publication.

- [144] "Stochastic Online Metric Matching" (ICALP'19)
  with Anupam Gupta, Guru Guruganesh and Binghui Peng.

Chapter 9 is based on the following joint work.

- "Online Algorithms for Edge Coloring via the Nibble Method"
  with Sayan Bhattacharya and Fabrizio Grandoni.

Chapter 10 is based on the following publication.

- [266] "Rounding Dynamic Matchings Against an Adaptive Adversary" (STOC'20)
  Solo-authored paper.

The same chapter's results subsume those of the following publication by the author.

- [14] "Dynamic Matching: Reducing Integral Algorithms to Approximately-Maximal Fractional Algorithms" (ICALP'18)
  with Moab Arar, Shiri Chechik, Sarel Cohen and Cliff Stein.

Chapter 11 is based on the following joint work.

- "Streaming Submodular Matching Meets the Primal-Dual Method"
  with Roie Levin.

The same chapter's results subsume those of the following publication by the author.

- [127] "Simplified and Space-Optimal Semi-Streaming $(2+\varepsilon)$-Approximate Matching" (SOSA'19)
  with Mohsen Ghaffari.

Omitted Work: This thesis does not address a number of results of the author (and co-authors) obtained during his PhD. This includes work on other problems and models which also fall squarely under the wide umbrella of algorithms under uncertainty, including [14, 127], which are subsumed by Chapter 10 and Chapter 11, respectively, and the author's work on distributed graph algorithms [149, 151, 154], dynamic bin packing [105], and mechanism design [237], as well as other works which fall outside of this scope, on routing and network coding [152, 153].

1.6 A Reader's Manual

The dependencies between the different chapters of this thesis have been kept to a minimum, and the individual chapters can be read essentially in whatever order the reader wishes to follow. As for the technical prerequisites for this thesis, a reader familiar with the area of approximation algorithms should be able to follow most technical arguments without too much difficulty. More specialized technical tools needed for different chapters of this thesis, including a brief introduction to the use of linear programming for approximation algorithms, and the theory of negative association, are presented in Chapter 2. In the same chapter we give a brief introduction to the common matching-theoretic problems studied in this thesis. Other technical background is restricted to the relevant chapters.

Chapter 2

Technical Background

In this chapter we provide basic notation, definitions of problems we study (in Section 2.1), our main measure of algorithm quality (in Section 2.2), and useful known lemmas and techniques which we use throughout the thesis (in Section 2.3 and Section 2.4).

Some common notation: In this thesis , we will study undirected graphs, denoted by $G = (V, E)$, where V is the set of nodes and $E \subseteq \binom{V}{2}$ is the set of edges. We say a graph $G = (V, E)$ is *bipartite*, and denote it by $G = (L, R, E)$, if the nodes of V can be partitioned into two sets L and R, with no edges between nodes in the same part. We denote the number of nodes and edges in $G = (V, E)$ by $n := |V|$ and $m := |E|$. We denote the degree of node $v \in V$ in graph G by $d_G(v)$ and the graph's maximum degree by $\Delta(G) := \max_{v \in V} d_G(v)$. When G is clear from context, we will often use $d(v)$ and Δ for short. We say G is *regular* (or Δ-regular, to be explicit) if all nodes $v \in V$ have degree $d_G(v) = \Delta$. For a set of vertices $U \subseteq V$, we denote by $G[U] := (U, E \cap \binom{U}{2})$ the subgraph induced by U. Similarly, for a set of edges $F \subseteq E$, we denote by $G[F] := (V(F), F)$ the subgraph induced by F, where we denote by $V(F) := \{v \in V \mid v \in e \in F\}$ the nodes spanned by edges in F.

We denote by $\mathbb{R}$ and $\mathbb{Z}$ the set of real and integer numbers, respectively. For a positive integer k, the set $[k] := \{1, 2, \ldots, k\}$ consists of the first k positive integers. For real x, we let $x^+ := \max\{0, x\}$ denote the positive part of x. We use $c = a \pm b$ as shorthand for $c \in [a-b, a+b]$. When discussing complexity measures, we often find it useful to ignore multiplicative polylogarithmic factors, and we use $\tilde{O}(f(n))$ as shorthand for $O(f(n) \cdot \operatorname{poly}\log n)$. Throughout, we say an event happens with high probability (w.h.p.) if it happens with probability $1 - n^{-c}$ for some $c \geqslant 1$.

2.1 Matching Theory – A Primer

A *matching* in a graph $G = (V, E)$ is a subset of vertex-disjoint edges $M \subseteq E$. The cardinality of a maximum matching in G is denoted by $\mu(G)$. A *fractional matching* is a non-negative vector $\vec{x} \in \mathbb{R}^m_{\geqslant 0}$ satisfying the *fractional matching constraint*, $\sum_{e \ni v} x_e \leqslant 1 \ \forall v \in V$. That is, it is a point in the *fractional matching polytope* of G,

$$\mathcal{P}(G) := \left\{ \vec{x} \in \mathbb{R}^m_{\geqslant 0} \;\middle|\; \sum_{e \ni v} x_e \leqslant 1 \ \forall v \in V \right\}. \tag{2.1}$$

A matching M is *perfect* if is spans all vertices of G; that is, if $V(M) = V$. In weighted matching problems, edges are associated with a weight function, $w : E \to \mathbb{R}$. Here, we will consider the problem of computing a maximum weight matching (MWM), or, thinking of weights as costs, we will also consider the problem of computing a minimum cost perfect matching.

Closely related to the maximum matching problem is the *minimum vertex cover* problem. We say a set of vertices $U \subseteq V$ if $G[V \setminus U]$ is an empty graph. Put otherwise, all edges of G have at least one endpoint in V. A minimum vertex cover is a vertex cover of minimum cardinality, and its size is denoted by $\nu(G)$. For bipartite graphs G, by a classic theorem of König [185], we have that $\nu(G) = \mu(G)$, i.e., the cardinality of a minimum vertex cover is equal to the cardinality of a maximum matching. As we shall see in Section 2.3, a similar relation holds (though only in a relaxed sense) for general graphs, as well.

A third problem which will appear often in this thesis is the *edge coloring* problem. In a *k-edge-coloring* of a graph $G = (V, E)$, edges in E are assigned one of k colors, such that no two incident edges share the same color. That is, E is partitioned into k matchings. Here, the objective is to minimize this number k of matchings used. A seminal result due to König [184] states that for bipartite graphs, the minimum such number of colors k is equal to Δ, the maximum degree in G. (Clearly, no fewer colors suffice.) For general graphs, this bound is not always achievable; e.g., in an odd-length cycle graph, $\Delta + 1 = 3$ colors are needed. A classic theorem of Vizing [261] asserts that $\Delta + 1$ colors always suffice.

2.2 Approximation and Competitive Ratios

As mentioned previously, in the models of computation we consider, approximation is a necessary evil, as exact optimization of objectives under such uncertainty is often *provably* impossible. For example, for online matching algorithms, consider an online node u with two edges to neighbors a and b. If the next online neighbor to arrive, v, only neighbors b or a, then the (unique) optimal matching is $\{(u, a), (v, b)\}$ or $\{(u, b), (v, a)\}$, respectively. An online algorithm, which must decide which neighbor of u to match it to (if any), irrevocably before v arrives, cannot guarantee that v's sole neighbor will be free when v arrives, and can therefore not guarantee to output a maximum matching. Similar challenges arise when considering other models of computation under uncertainty. We therefore consider the natural next best thing—approximation.

We say an algorithm $\mathcal{A}$ for a maximization problem Π has *approximation ratio* $\alpha \in [0, 1]$, or is *α-approximate*, for short, if for any input $\mathcal{I} \in \Pi$, algorithm $\mathcal{A}$'s output value, $ALG(\mathcal{I})$, is at least α times the value of the optimum value, $OPT(\mathcal{I})$.

$$ALG(\mathcal{I}) \geqslant \alpha \cdot OPT(\mathcal{I}). \tag{2.2}$$

If $\mathcal{A}$ is randomized, the inequality can hold either in expectation or w.h.p. A similar definition, with $\alpha \geqslant 1$ and the inequality reversed, is of interest for minimization problems. We will sometimes (depending on the common notation in the relevant literature) refer to α-approximate algorithms with $\alpha \geqslant 1$, by which we mean that

$$ALG(\mathcal{I}) \geqslant \frac{1}{\alpha} \cdot OPT(\mathcal{I}). \tag{2.3}$$

For online algorithms, we refer to the approximation ratio of an algorithm as its *competitive ratio*, and call an algorithm with competitive ratio α an α-*competitive* algorithm.

Regardless of the common notation or nomenclature, or model of computation, our main objective will be to optimize the value of our solution, which corresponds to achieving an algorithm with approximation (or competitive) ratio as close to one as possible. When trying to characterize the limits of algorithms, we will refer to impossibility results as *lower bounds* (even when considering competitive ratios $\alpha \in [0,1]$, for which impossibility results are, strictly speaking, upper bounds on α). Symmetrically, we will refer to algorithmic results as *upper bounds*.

When searching for good upper bounds (i.e., algorithms with good approximation ratios), inequalities such as Equation (2.2) will often require us to obtain some useful intermediate bound $B(\mathcal{I})$ on $OPT(\mathcal{I})$. For maximization problems, such bounds are used to prove the following

$$ALG(\mathcal{I}) \geqslant \alpha \cdot B(\mathcal{I}) \geqslant \alpha \cdot OPT(\mathcal{I}).$$

One reliable source of such bounds on $OPT(\mathcal{I})$ is mathematical programming relaxations, and in particular, linear programming, which we now turn to.

2.3 Linear Programming

An immensely powerful tool for exact computation, linear programming has played a similar pivotal role in approximation algorithms. In this thesis, we will use linear programming (LP) theory to prove both upper and lower bounds. We briefly review some relevant background. For a more thorough introduction to the theory of Linear Programming, we refer to [246].

2.3.1 Relaxations

Many combinatorial optimization problems can be stated in terms of linear objectives and linear inequalities. For example, one way to state the maximum weight matching problem is as follows.

$$\max_{\vec{x} \in \{0,1\}^E \cap \mathcal{P}(G)} \left\{ \sum_e w_e \cdot x_e \right\}, \tag{2.4}$$

where $\mathcal{P}(G)$ is the fractional matching polytope, given by (2.1). Similarly, a set of linear constraints and integrality constraints capture the related minimum vertex cover problem, as follows.

$$\min_{\vec{y} \in \{0,1\}^E} \left\{ \sum_v y_v \;\middle|\; y_u + y_v \geqslant 1 \;\; \forall (u,v) \in E \right\}. \tag{2.5}$$

Unfortunately, the minimum vertex cover problem is known to be NP-complete (it is one of Karp's 21 NP-complete problems [178]), and so phrasing it in the above terms is unlikely to help solve it exactly. However, phrasing this problem and others in terms of integer linear programs proves useful for the design of *approximation* algorithms, if we *relax the integrality constraints*. Such relaxations, whose optima can only be better than those of their (more constrained) integral versions, serve as useful benchmarks for the design of approximation algorithms. In particular, denoting by $LP(\mathcal{I})$ the optimal LP value for the relaxation of some integral problem instance $\mathcal{I}$ of maximization problem Π, we trivially have

$$LP(\mathcal{I}) \geqslant OPT(\mathcal{I}). \tag{2.6}$$

Therefore, to obtain an α-approximate (or α-competitive) algorithm, it is sufficient to guarantee that for any instance $\mathcal{I}$, our algorithm's output value is at least $ALG(\mathcal{I}) \geqslant \alpha \cdot LP(\mathcal{I})$. Two successful ways to make use of this observation in the literature, which we will also rely on, are (randomized) rounding, and the primal-dual method. The latter of these approaches relies on LP duality, which we now briefly review.

2.3.2 LP Duality

One useful concept when designing approximation algorithms whose analysis relies on LPs is the notion of *LP duality*. For an LP (in matrix notation), which we refer to as the *primal* LP,

$$\max\{\vec{c} \cdot \vec{x} \mid A \cdot \vec{x} \leqslant \vec{b},\, \vec{x} \geqslant \vec{0}\}, \tag{2.7}$$

one associates a *dual* LP,

$$\min\{\vec{b} \cdot \vec{y} \mid A^T \cdot \vec{y} \geqslant \vec{c},\, \vec{y} \geqslant \vec{0}\}. \tag{2.8}$$

The optimal value of this dual program (2.8) is equal to the best upper bound on (2.7) obtained by considering linear combinations of the constraints of the primal LP.

As a concrete example, one pair of LPs which will prove useful for us are LP relaxation for the maximum weight matching (MWM) problem and its dual, given in Figure 2.1 below.

Primal		Dual	
maximize	$\sum_{e\in E} w_e \cdot x_e$	minimize	$\sum_{v\in V} y_v$
subject to:		subject to:	
$\forall v \in V$:	$\sum_{e\ni v} x_e \leqslant 1$	$\forall (u,v) \in E$:	$y_u + y_v \geqslant w_{(u,v)}$
$\forall e \in E$:	$x_e \geqslant 0$	$\forall v \in V$:	$y_v \geqslant 0$

Figure 2.1: The LP relaxation of the MWM problem and its dual

For maximum *cardinality* matching (MCM), where all weights are equal to $w_e = 1$, this dual is precisely the relaxation of the minimum vertex cover problem. As mentioned above, these problems' *integral* optima's values are equal in bipartite graphs. Indeed, as proven by Egerváry [89], the integral optima's values for the above LPs (for MWM and the corresponding dual) are also equal in bipartite graphs. While in general graphs the same does not hold (for example, in a triangle graph, $\mu(G) = 1$, while $\nu(G) = 2$), these problems' *fractional* optima are equal in general graphs, due to strong LP duality. For our needs, we will only rely on the fact that the dual's optimal value upper bounds the primal's optimal value, which follows by weak duality.

Lemma 2.3.1. *Let $\vec{x}$ and $\vec{y}$ be feasible solutions to primal and dual LPs of the forms* (2.7) *and* (2.8)*, respectively. Then, their objective values satisfy*

$$\vec{c} \cdot \vec{x} \leqslant \vec{b} \cdot \vec{y}.$$

Lemma 2.3.1 will prove useful in the design of approximation algorithms, as we now outline.

2.3.3 The (Randomized) Primal-Dual Method

Lemma 2.3.1 suggests a natural upper bound on the optimum (integral) solution of any maximization problem Π (and similarly for minimization problems); if we denote by $\mathcal{I} \in \Pi$ some instance, by $OPT(\mathcal{I})$, $LP(\mathcal{I})$, the optimal integral and fractional values for $\mathcal{I}$ (according to some LP relaxation), and by $D(\mathcal{I})$ the value of some feasible dual solution for $\mathcal{I}$, we have that

$$D(\mathcal{I}) \geqslant LP(\mathcal{I}) \geqslant OPT(\mathcal{I}). \tag{2.9}$$

Equation (2.9) suggests a schematic (high-level) approach for the design of approximation algorithms, referred to as the *primal-dual* method. Here, our algorithm computes a feasible (integral) primal solution and dual solutions of value $ALG(\mathcal{I}) \geqslant \alpha \cdot D(\mathcal{I})$. Concatenating this inequality with Equation (2.9) implies that this algorithm is α-approximate.

Example: A simple example of the primal-dual method is given by the vertex cover problem (whose relaxation is dual to MCM). A natural algorithm for vertex cover, attributed to Gavril in [123, pg. 134], initializes an empty cover $\mathcal{C}$, and then inspects the edges in some arbitrary order, adding both endpoints of every edge not covered by $\mathcal{C}$ prior to its inspection. To analyze this algorithm using the primal-dual method, we initially set primal and dual solutions to zero. When inspecting an edge $e = (u, v)$ that it is not already covered by $\mathcal{C}$, we set $x_e \leftarrow 1$ and $y_u, y_v \leftarrow 1$. It is not hard to see that $\vec{x}$ and $\vec{y}$ are primal and dual feasible, and trivially satisfy $|\mathcal{C}| = \sum_v y_v = 2 \cdot \sum_e x_e \leqslant 2 \cdot OPT$, where the inequality follows by weak LP duality.

The Randomized Primal-Dual Method: In 2013, Devanur, Jain, and Kleinberg [79] presented an extension of the primal-dual method. The simple underlying observation of this method is that dual feasibility need not hold *always* to imply a useful approximation. In particular, constructing a dual solution of value $D \leqslant \frac{1}{\alpha} \cdot ALG$ and which satisfies all constraints *in expectation* implies that, by linearity of the dual objective,

$$\mathbb{E}[ALG(\mathcal{I})] \geqslant \alpha \cdot \mathbb{E}[D] \geqslant \alpha \cdot LP(\mathcal{I}) \geqslant \alpha \cdot OPT(\mathcal{I}).$$

This method allowed Devanur et al. [79] to give a unified analysis of known results for online matching [6, 134, 179, 207]. Below we give a simple example of this approach.

Example: A simple example of the randomized primal-dual method is also given by the vertex cover problem. Here we consider an arguably even more natural algorithm due to Pitt [235], which initializes an empty cover $\mathcal{C}$, and then inspects the edges in some arbitrary order, adding *one single*, *random*, endpoint of every edge not covered by $\mathcal{C}$ prior to its inspection. To analyze this algorithm using the randomized primal-dual method, we initialize primal and dual solutions to zero. When inspecting an edge e that it is not previously covered by $\mathcal{C}$, and adding an endpoint $v \in e$ to $\mathcal{C}$, we set $x_e \leftarrow 1/2$ and $y_v \leftarrow 1$. For any vertex v, the expected number of edges of v not covered until v is added is at most two, since for each such edge the probability we pick v is one half, and so the number of such edges is dominated by a geometric variable with success probability $1/2$. Therefore $\vec{x}$ is feasible in expectation, since $\mathbb{E}[\sum_{e \ni v} x_e] \leqslant 2 \cdot 1/2 = 1$ for every vertex v. On the other hand, $\sum_v y_v = 2 \cdot \sum_e x_e$ for any realization, and similarly in expectation. Consequently, weak duality implies that $\mathbb{E}[|\mathcal{C}|] = \mathbb{E}[\sum_v y_v] = 2 \cdot \mathbb{E}[\sum_e x_e] \leqslant 2 \cdot OPT$.

2.3.4 Randomized Rounding

Another approach used often in approximation algorithms, pioneered by Raghavan and Tompson [238], is that of *randomized rounding*. Here, a (fractional) solution to an LP relaxation has each of its coordinates rounded to an integer value, possibly randomly. As the value of an optimal fractional solution is no worse than that of an integral solution, rounding in a way which incurs only a multiplicative blowup of α results in an α-approximate solution.

Example: Again, vertex cover provides an illustrative example. Consider a fractional vertex cover $\vec{y}$ in a bipartite graph $G = (L, R, E)$. Sample a uniform random variable $\tau \sim Uni(0, 1)$. For all $u \in L$, add u to the cover $\mathcal{C}$ is $y_u \geqslant \tau$, and for all $v \in R$, add v to the cover $\mathcal{C}$ if $y_v \geqslant 1 - \tau$. Since $y_u + y_v \geqslant 1$ for all edges $(u, v) \in E$, it is immediate that the obtained set $\mathcal{C}$ is indeed a vertex cover. On the other hand, since τ and $1 - \tau$ are both $Uni(0, 1)$ variables, each vertex belongs to $\mathcal{C}$ with probability $\Pr[v \in \mathcal{C}] = \int_0^{y_v} dx = y_v$, and so by linearity of expectation $\mathbb{E}[\mathcal{C}] = \sum_v \Pr[v \in \mathcal{C}] = \sum_v y_v$. Therefore, solving the fractional vertex cover LP in bipartite graphs and rounding it this way yields a 1-approximate integral solution.

More sophisticated approaches for rounding fractional matchings in bipartite graphs, due to [4, 122], proved immensely useful in the approximation algorithms literature over the years. We elaborate more on this approach in Chapter 5.

2.3.5 Integrality Gaps

A limit to the use of any LP relaxation (or indeed any mathematical relaxation) for approximation algorithms, is the *integrality gap* of the relaxation. Informally, this is the highest multiplicative gap between the optimal value of an integral solution of the problem and the optimal value of a feasible point w.r.t. the given relaxation. For example, the fractional matching polytope (2.1) has an integrality gap of $3/2$, obtained by considering the triangle graph. In this graph, the maximum matching size is trivially one, as no two edges of the triangle belong to a common matching, while assigning values $x_e = 1/2$ to all edges yields a fractional matching of value $3/2$. This implies that in general, when using the LP relaxation as our benchmark, we cannot hope to obtain an approximation ratio better than $3/2$.

A seminal result of Edmonds [87] characterizes the matching polytope, which is the convex hull of all (integer) matchings of a graph G. In particular, this polytope is defined by the following sets of inequalities

$$\begin{aligned} \sum_{e \ni v} x_e &\leqslant 1 && \forall v \in V \\ \sum_{e \in \binom{S}{2}} x_e &\leqslant (|S| - 1)/2 && \forall S \subseteq V, |S| \text{ odd.} \end{aligned}$$

This exponential-sized LP characterization underlies numerous fundamental results for matching theory in the full-information régime. This characterization, however, requires a somewhat less myopic view of the graphs than we will be able to consider in our partial-information settings. As such, we will mostly rely on the fractional matching polytope (2.1) when considering matching problems under uncertainty. This will require some care in order to not lose the same multiplicative factor of $3/2$ when rounding a fractional solution.

2.4 Probability Theory

In this section we review some useful probability theory background used in our analysis, starting with some basic probabilistic inequalities. For more basic background on probability theory, we refer the reader to [216].

2.4.1 Negative Association

A common tool in the analysis of randomized algorithms is the family of Chernoff-Hoeffding tail bounds for independent random variables. One tool we will often rely on for our analysis is concentration inequalities of *dependent* random variables. Specifically, in this section, based on notes by the author [265], we will study concentration of sums of *negatively associated* variables.

Definition 2.4.1 ([171, 183]). *A joint distribution $(X_1, \ldots, X_n)$ is* negatively associated (NA)*, and the variables $X_1, \ldots, X_n$ are NA, if every two monotone increasing functions f and g defined on disjoint subsets of the variables in $\vec{X}$ are negatively correlated. That is,*

$$\mathbb{E}[f \cdot g] \leqslant \mathbb{E}[f] \cdot \mathbb{E}[g]. \tag{2.10}$$

A trivial example of NA variables is given by *independent* random variables, for which (2.10) holds with equality for *any* functions f and g. (Consequently, all the tail bounds and other useful properties for negative association which we state in this section hold for independent variables as well.) More interesting, useful examples of NA for our use are given by the following two propositions.

Proposition 2.4.2 (0-1 Principle [83]). *Let $X_1, \ldots, X_n \in \{0, 1\}$ be binary random variables such that $\sum_i X_i \leqslant 1$ always. Then, the joint distribution $(X_1, \ldots, X_n)$ is NA.*

Proposition 2.4.3 (Permutation Distributions are NA [171]). *Let $x_1, \ldots, x_n$ be n values and let $X_1, \ldots, X_n$ be random variables taking on all permutations of $(x_1, \ldots, x_n)$ with equal probability. Then the joint distribution $(X_1, \ldots, X_n)$ is NA.*

More elaborate NA distributions can be obtained from simple NA distributions as those given by propositions 2.4.2 and 2.4.3 via the following closure properties.

Proposition 2.4.4 (NA Closure Properties [83, 171, 183]).

1. *Independent union. Let $(X_1, \ldots, X_n)$ and $(Y_1, \ldots, Y_m)$ be mutually independent NA joint distributions. Then, the joint distribution $(X_1, \ldots, X_n, Y_1, \ldots, Y_m)$ is also NA.*
2. *Function composition. Let $f_1, \ldots, f_k$ be monotone (all increasing or all decreasing) functions defined on disjoint subsets of the variables in $\vec{X}$. Then the joint distribution $(f_1(\vec{X}), \ldots, f_k(\vec{X}))$ is NA.*

Example NA distributions obtained using these closure properties are statistics of balls and bins processes. An illustrative example is given by the following proposition.

Proposition 2.4.5 (Balls and Bins is NA)**.** *Suppose m balls are thrown independently into one of n bins (not necessarily u.a.r., and not necessarily i.i.d). Let B_i be the number of balls placed in bin i in this process. Then the joint distribution $(B_1, B_2, \ldots, B_n)$ is NA.*

Proof. For each $b \in [m]$ and $i \in [n]$, let $X_{b,i}$ be an indicator variable for ball b landing in bin i. By the 0-1 Principle (Proposition 2.4.2), the variables $\{X_{b,i} \mid i \in [n]\}$ are NA. By closure of NA under independent union (Proposition 2.4.4.1), as each ball is placed independently of all other balls, $\{X_{b,i} \mid b \in [m], i \in [n]\}$ are NA. Finally, by closure of NA under monotone increasing functions on disjoint subsets (Proposition 2.4.4.2), the variables $B_i = \sum_b X_{b,i}$ are NA. □

As the variables $N_i := \min\{1, B_i\}$, indicating whether bin i is non-empty, are monotone increasing functions depending on disjoint subsets of the B_i variables (specifically, singletons), we obtain the following corollary.

Corollary 2.4.6. *The indicator variables N_i for bins being non-empty in a balls and bins process as in Proposition 2.4.5, are NA.*

Useful Properties of NA Distributions: We now outline some simple useful properties of NA distributions. For example, a special case of the definition of NA, taking $f_i(\vec{X}) = X_i$, we find that NA variables are negatively correlated.

Corollary 2.4.7 (NA implies Negative Correlation)**.** *Let $X_1, \ldots, X_n$ be n NA variables. Then, for all $i \neq j$, we have $\mathbb{E}[X_i \cdot X_j] \leqslant \mathbb{E}[X_i] \cdot \mathbb{E}[X_j]$. That is, $\mathrm{Cov}(X_i, X_j) \leqslant 0$.*

Another useful property of NA variables is Negative Orthant Dependence (NOD).

Corollary 2.4.8 (NA implies NOD)**.** *For any n NA variables $X_1, \ldots, X_n$ and n real values $x_1, \ldots, x_n$,*

$$\Pr\left[\bigwedge_i (X_i \geqslant x_i)\right] \leqslant \prod_i \Pr[X_i \geqslant x_i]$$

$$\Pr\left[\bigwedge_i (X_i \leqslant x_i)\right] \leqslant \prod_i \Pr[X_i \leqslant x_i].$$

The following corollary of NA will prove useful shortly. It is easily proved by induction on the number of functions, k, using Definition 2.4.1 for the inductive step.

Corollary 2.4.9. *Let $X_1, \ldots, X_n$ be NA variables. Then, for every set of k positive monotone increasing functions $f_1, \ldots, f_k$ depending on disjoint subsets of the X_i, it holds*

$$\mathbb{E}\left[\prod_i f_i(\vec{X})\right] \leqslant \prod_i \mathbb{E}[f_i(\vec{X})].$$

Chernoff-Hoeffding Bounds—Expect the Expected: As stated in Corollary 2.4.7, NA implies negative correlation. A much stronger, and particularly useful property of NA variables is the applicability of Chernoff-Hoeffding type concentration inequalities to sums of NA variables $X_1, \ldots, X_n$. This follows from monotonicity of the exponential function and Corollary 2.4.9 implying that $\mathbb{E}[\exp(\lambda \cdot \sum_i X_i)] \leqslant \prod_i \mathbb{E}[\exp(\lambda \cdot x_i)]$ (see also [83]). This is the crucial first step of proofs of such tail bounds. In particular, we will make use of the following tail bounds.

Lemma 2.4.10. *Let X be the sum of NA random variables $X_1, \ldots, X_m \in [0, 1]$. Then for all $\delta \in (0, 1)$, and $\kappa \geqslant \mathbb{E}[X]$,*

$$\Pr[X \leqslant (1 - \delta) \cdot \mathbb{E}[X]] \leqslant \exp\left(-\frac{\mathbb{E}[X] \cdot \delta^2}{2}\right),$$

$$\Pr[X \geqslant (1 + \delta) \cdot \kappa] \leqslant \exp\left(-\frac{\kappa \cdot \delta^2}{3}\right).$$

Lemma 2.4.11. *Let X be the sum of m NA random variables $X_1, \ldots, X_m$ with $X_i \in [a_i, b_i]$ for each $i \in [m]$. Then for all $t > 0$,*

$$\Pr[X \geqslant \mathbb{E}[X] + t] \leqslant \exp\left(-\frac{2t^2}{\sum_i (b_i - a_i)^2}\right),$$

$$\Pr[X \leqslant \mathbb{E}[X] - t] \leqslant \exp\left(-\frac{2t^2}{\sum_i (b_i - a_i)^2}\right).$$

Another tail bound obtained from $\mathbb{E}[\exp(\lambda \sum_i X_i)] \leqslant \prod_i \mathbb{E}[\exp(\lambda X_i)]$ is Bernstein's Inequality, which yields stronger bounds for sums of NA variables with bounded variance.

Lemma 2.4.12. *Let X be the sum of NA random variables $X_1, \ldots, X_k \in [-M, M]$. Then, for $\sigma^2 = \sum_{i=1}^k \mathrm{Var}(X_i)$ and all $a > 0$,*

$$\Pr[X > \mathbb{E}[X] + a] \leqslant \exp\left(\frac{-a^2}{2(\sigma^2 + aM/3)}\right).$$

As mentioned above, since independent random variables are (trivially) NA, lemmas 2.4.10, 2.4.11 and 2.4.12 hold for sums of independent random variables as a special case.

2.4.2 Other Useful Probabilistic Inequalities

Another useful inequality for our needs is Jensen's Inequality, which follows form the definition of convexity.

Lemma 2.4.13. *For any random variable X and convex function f,*

$$f(\mathbb{E}[X]) \leqslant \mathbb{E}[f(X)].$$

A useful corollary of Jensen's Inequality is that the mean average deviation of any random variable, $\mathbb{E}[|X - \mathbb{E}[X]|]$ is upper bounded by its standard deviation.

Lemma 2.4.14. *For any random variable X,*

$$\mathbb{E}[|X - \mathbb{E}[X]|] \leqslant \mathrm{Std}(X).$$

Proof. Applying Jensen's Inequality to the concave function $f(x) = \sqrt{x}$ and the random variable $Y = (X - \mathbb{E}[X])^2$, we obtain the desired inequality,

$$\mathbb{E}[|X - \mathbb{E}[X]|] = \mathbb{E}[\sqrt{(X - \mathbb{E}[X])^2}] \leqslant \sqrt{\mathbb{E}[(X - \mathbb{E}[X])^2]} = \sqrt{\mathrm{Var}(X)} = \mathrm{Std}(X). \qquad \square$$

Coupling Arguments: Some joint distributions of random variables $X_1, X_2, \ldots, X_n$ revealed by some iterative process are hard to reason about directly. However, if one can show that for any realization $\vec{x} \in \mathbb{R}^{i-1}$ of $X_1, X_2, \ldots, X_{i-1}$, the (conditional) variable $[Y_i \mid (X_1, X_2, \ldots, X_{i-1}) = \vec{x}]$ is dominated by a simple variable Z_i, say, a Bernoulli variable, then a simple coupling argument allows us to bound probabilities of events determined by these X variables in terms of events determined by independent copies of the variables Z_i. Such a statement is given by the following proposition.

Proposition 2.4.15. *Let $X_1, \ldots, X_m$ be random variables and $Y_1, \ldots, Y_m$ be binary random variables such that $Y_i = f_i(X_1, \ldots, X_i)$ for all i such that for all $\vec{x} \in \mathbb{R}^m$,*

$$\Pr\left[Y_i = 1 \,\middle|\, \bigwedge_{\ell \in [i]} (X_\ell = x_\ell)\right] \leqslant p_i.$$

Then, if $Z_i = Bernoulli(p_i)$ are independent random variables, we have

$$\Pr\left[\sum_i Y_i \geqslant k\right] \leqslant \Pr\left[\sum_i Z_i \geqslant k\right].$$

Conditional Union Bound: Finally, we will also need the following simple extension of the union bound, whose proof follows by application of the standard union bound to the variables $B_i := \overline{A_i} \wedge \bigwedge_{j<i} A_j$, corresponding to i being the first index for which $\overline{A_i}$ holds.

Proposition 2.4.16 (Conditional Union Bound)**.** *Let* $A_1, A_2, \ldots, A_n$ *be random indicator variables such that* $\Pr[\overline{A_i} \mid \bigwedge_{j<i} A_j] \leqslant p$*. Then*

$$\Pr\left[\bigvee_i \overline{A_i}\right] \leqslant n \cdot p.$$

Part I

Online Algorithms: Hedging One's Bets

Chapter 3

Online Matching: Edge Arrivals

This chapter, based on [120, Section 2], is by far the shortest chapter in this thesis. In it, we discuss our (surprisingly simple) resolution of the optimal competitive ratio for online matching under edge arrivals, highlighting this result with a dedicated chapter.

3.1 Background and Contribution

Arguably the most natural, and the least restricted, arrival model for online matching is the (adversarial) edge arrival model. In this model, edges are revealed one by one, and an online matching algorithm must decide immediately and irrevocably whether to match the edge on arrival, or whether to leave both endpoints free to be possibly matched later.

On the hardness front, the problem is known to be strictly harder than the one-sided vertex arrival model of Karp et al. [179], which admits a competitive ratio of $1 - 1/e \approx 0.632$. In particular, Epstein et al. [95] gave an upper bound of $1/(1+\ln 2) \approx 0.591$ for this problem, recently improved by Huang et al. [165] to $2-\sqrt{2} \approx 0.585$. (Both bounds apply even to online algorithms with preemption, which may remove edges from the matching in favor of a newly-arrived edge.) On the positive side, as pointed out by Buchbinder et al. [55], the edge arrival model has proven challenging, and results beating the $1/2$ competitive ratio were only achieved under various relaxations, including: random order edge arrival [146], bounded number of arrival batches [191], on trees, either with or without preemption [55, 259], and for bounded-degree graphs [55]. The above papers all asked whether there exists a randomized $(1/2 + \Omega(1))$-competitive algorithm for adversarial edge arrivals (see also Open Question 17 in Mehta's survey [206]).

In this chapter, we answer this open question, providing it with a strong negative answer. In particular, we show that no online algorithm for *fractional* matching (i.e., an algorithm which immediately and irrevocably assigns values x_e to edge e upon arrival such that $\vec{x}$ is in the fractional matching polytope $\mathcal{P} = \{\vec{x} \geqslant \vec{0} \mid \sum_{e \ni v} x_e \leqslant 1 \ \forall v \in V\}$) is better than $1/2$ competitive. As any randomized algorithm induces a fractional algorithm with the same competitive ratio, this rules out any randomized online matching algorithm which is better than deterministic algorithms.

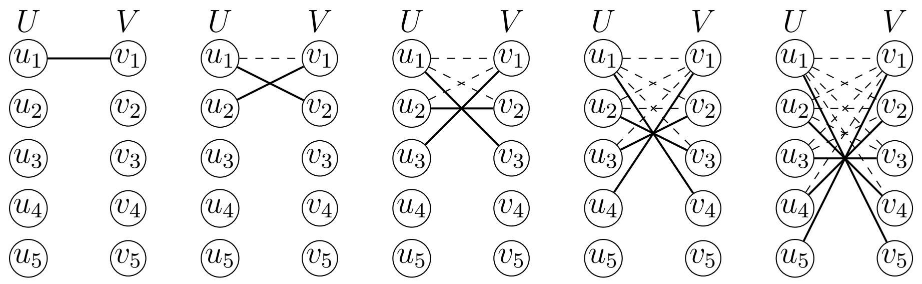
This result shows that the study of relaxed variants of online matching under edge arrivals is not only justified by the difficulty of beating the trivial bound for this problem, but rather by its *impossibility*.

3.2 The Lower Bound

Our main idea will be to provide a "prefix hardness" instance, where an underlying input and the arrival order is known to the online matching algorithm, but the prefix of the input to arrive (or "termination time") is not. Consequently, the algorithm must accrue high enough value up to each arrival time, to guarantee a high competitive ratio at all points in time. As we show, the fractional matching constraints rule out a competitive ratio of $1/2+\Omega(1)$ even in this model where the underlying graph is known.

Theorem 3.2.1 (Edge Arrival Lower Bound)**.** *There exists an infinite family of bipartite graphs with maximum degree n and edge arrival order for which any online matching algorithm is at best $\left(\frac{1}{2}+\frac{1}{2n+2}\right)$-competitive.*

Proof. We provide a family of graphs for which no fractional online matching algorithm has better competitive ratio. Since any randomized algorithm induces a fractional matching algorithm, this immediately implies our theorem. The n^{th} graph of the family, $G_n = (U, V, E)$, consists of a bipartite graph with $|U| = |V| = n$ vertices on either side. We denote by $u_i \in U$ and $v_i \in V$ the i^{th} node on the left and right side of G_n, respectively. Edges are revealed in n discrete rounds. In round $i = 1, 2, \ldots, n$, the edges of a perfect matching between the first i left and right vertices arrive in some order. I.e., a matching of $u_1, u_2, \ldots, u_i$ and $v_1, v_2, \ldots, v_i$ is revealed. Specifically, edges (u_j, v_{i-j+1}) for all $i \geqslant j$ arrive. (See Figure 3.1 for example.) Intuitively, the difficulty for an algorithm attempting to assign high value to edges of OPT is that the (unique) maximum matching OPT changes every round, and no edge ever re-enters OPT.



(a) round 1 (b) round 2 (c) round 3 (d) round 4 (e) round 5

Figure 3.1: G_5, together with arrival order.

Edges of current (prior) round are solid (dashed).

Consider some α-competitive fractional algorithm $\mathcal{A}$. We call the edge of a vertex w in the (unique) maximum matching of the subgraph of G_n following round i the i^{th} edge of w. For $i \geqslant j$, denote by $x_{i,j}$ the value $\mathcal{A}$ assigns to the i^{th} edge of vertex u_j (and of v_{i-j+1}); i.e., to (u_j, v_{i-j+1}). By feasibility of the fractional matching output by $\mathcal{A}$, we immediately have that $x_{i,j} \geqslant 0$ for all i, j, as well as the following matching constraints for u_j and v_j. (For the latter, note that the i^{th} edge of v_{i-j+1} is assigned value $x_{i,j} = x_{i,i-(i-j+1)+1}$ and so the i^{th} edge of v_j is assigned value $x_{i,i-j+1}$).

$$\sum_{i=j}^{n} x_{i,j} \leqslant 1. \qquad (u_j \text{ matching constraint}) \tag{3.1}$$

$$\sum_{i=j}^{n} x_{i,i-j+1} \leqslant 1. \qquad (v_j \text{ matching constraint}) \tag{3.2}$$

On the other hand, as $\mathcal{A}$ is α-competitive, we have that after some k^{th} round – when the maximum matching has cardinality k – algorithm $\mathcal{A}$'s fractional matching must have value at least $\alpha \cdot k$. (Else an adversary can stop the input after this round, leaving $\mathcal{A}$ with a worse than α-competitive matching.) Consequently, we have the following competitiveness constraints.

$$\sum_{i=1}^{k}\sum_{j=1}^{i} x_{i,j} \geqslant \alpha \cdot k \qquad \forall k \in [n]. \tag{3.3}$$

Combining constraints (3.1), (3.2) and (3.3) together with the non-negativity of the $x_{i,k}$ yields the following linear program, LP(n), whose optimal value upper bounds any fractional online matching algorithm's competitiveness on G_n, by the above.

$$\begin{array}{lll}
\text{maximize} & \alpha & \\
\text{subject to:} & \displaystyle\sum_{i=j}^{n} x_{i,j} \leqslant 1 & \forall j \in [n] \\
& \displaystyle\sum_{i=j}^{n} x_{i,i-j+1} \leqslant 1 & \forall j \in [n] \\
& \displaystyle\sum_{i=1}^{k}\sum_{j=1}^{i} x_{i,j} \geqslant \alpha \cdot k & \forall k \in [n] \\
& x_{i,j} \geqslant 0 & \forall i,j \in [n].
\end{array}$$

To bound the optimal value of LP(n), we provide a feasible solution its LP dual, which we denote by Dual(n). By weak duality, any dual feasible solution's value upper bounds the optimal value of LP(n), which in turn upper bounds the optimal competitive ratio. Using the dual variables ℓ_j, r_j for the degree constraints of the j^{th} left and right vertices respectively (u_j and v_j) and dual variable c_k for the competitiveness constraint of the k^{th} round, we get the following dual linear program. Recall here again that $x_{i,i-j+1}$ appears in the matching constraint of v_j, with dual variable r_j, and so $x_{i,j} = x_{i,i-(i-j+1)+1}$ appears in the same constraint for v_{i-j+1}.)

$$\begin{aligned}
\text{minimize} \quad & \sum_{j=1}^{n} (\ell_j + r_j) & \\
\text{subject to:} \quad & \sum_{k=1}^{n} k \cdot c_k \geqslant 1 & \\
& \ell_j + r_{i-j+1} - \sum_{k=i}^{n} c_k \geqslant 0 & \forall i \in [n], j \in [i] \\
& \ell_j, r_j, c_k \geqslant 0 & \forall j, k \in [n].
\end{aligned}$$

We provide the following dual solution.

$$c_k = \frac{2}{n(n+1)} \qquad \forall k \in [n]$$

$$\ell_j = r_j = \begin{cases} \frac{n-2(j-1)}{n(n+1)} & \text{if } j \leqslant n/2+1 \\ 0 & \text{if } n/2+1 < j \leqslant n. \end{cases}$$

We start by verifying feasibility of this solution. The first constraint is satisfied with equality. For the second constraint, as $\sum_{k=i}^{n} c_k = \frac{2(n-i+1)}{n(n+1)}$, it suffices to show that $\ell_j + r_{i-j+1} \geqslant \frac{2(n-i+1)}{n(n+1)}$ for all $i \in [n], j \in [i]$. Note that if $j > n/2+1$, then $\ell_j = r_j = 0 > \frac{n-2(j-1)}{n(n+1)}$. So, for all j we have $\ell_j = r_j \geqslant \frac{n-2(j-1)}{n(n+1)}$. Consequently, $\ell_j + r_{i-j+1} \geqslant \frac{n-2(j-1)}{n(n+1)} + \frac{n-2(i-j+1-1)}{n(n+1)} = \frac{2(n-i+1)}{n(n+1)}$ for all $i \in [n], j \in [i]$. Non-negativity of the ℓ_j, r_j, c_k variables is trivial, and so we conclude that the above is a feasible dual solution.

It remains to calculate this dual feasible solution's value. For even n, this is

$$\sum_{j=1}^{n}(\ell_j + r_j) = 2 \cdot \sum_{j=1}^{n} \ell_j = 2 \cdot \sum_{j=1}^{n/2+1} \frac{n-2(j-1)}{n(n+1)} = \frac{1}{2} + \frac{1}{2n+2},$$

completing the proof. □

Remark 1.: Recall that Buchbinder et al. [55] and Lee and Singla [191] presented better-than-1/2-competitive algorithms for bounded-degree graphs and few arrival batches, respectively. Our upper bound above shows that a deterioration of the competitive guarantees as the maximum degree and number of arrival batches increase (as in the algorithms of [55, 191]) is inevitable.

Remark 2.: Recall that the *asymptotic* competitive ratio of an algorithm is the maximum c such that the algorithm always guarantees value at least $ALG \geqslant c \cdot OPT - b$ for some fixed $b > 0$. Our proof rules out this weaker notion of competitiveness too, by revealing multiple copies of the family of Theorem 3.2.1 and letting x_{ik} denote the average of its counterparts over all copies.

Conclusion: The result of this chapter indicates that some fundamental matching-theoretic problems do not admit non-trivial competitive algorithms under online edge arrivals. The remainder of this thesis which is dedicated to online algorithms (with the exception of Chapter 9) will therefore focus on such problems under vertex arrivals.

Chapter 4

Online Matching: General Vertex Arrivals

In this chapter, based on [120] (joint work with Buddhima Gamlath, Michael Kapralov, Andreas Maggiori and Ola Svensson), we consider another generalization of the online matching problem introduced by Karp et al. [179]. In their seminal work, Karp et al. studied online matching in bipartite graphs, under vertex arrivals in one side of the graph. They also asked if better algorithms than the $1/2$-competitive greedy algorithm exist in general graphs. Here we study this problem in general (i.e., possibly non-bipartite) graphs, with arbitrary arrival order of nodes, and present a randomized $(1/2 + \Omega(1))$-competitive algorithm for this model.

4.1 Background

In the online matching problem under vertex arrivals, vertices are revealed one at a time, together with their edges to their previously-revealed neighbors. An online matching algorithm must decide immediately and irrevocably upon arrival of a vertex whether to match it (or keep it free for later), and if so, who to match it to. The one-sided bipartite problem studied by Karp et al. [179] is precisely this problem when all vertices of one side of a bipartite graph arrive first. For this one-sided arrival model, the problem is thoroughly understood (even down to lower-order error terms [103]). Wang and Wong [267] proved that general vertex arrivals are strictly harder than one-sided bipartite arrivals, providing an upper bound of $0.625 < 1-1/e$ for the more general problem, later improved by Buchbinder et al. [55] to $\frac{2}{3+\phi^2} \approx 0.593$. Clearly, the general vertex arrival model is no harder than the online edge arrival model but is it *easier*? The answer is "yes" for *fractional* algorithms, as shown by combining our Theorem 3.2.1 with the 0.526-competitive fractional online matching algorithm under general vertex arrivals of Wang and Wong [267]. For *integral* online matching, however, the problem has proven challenging, and the only positive results for this problem, too, are for various relaxations, such as restriction to trees, either with or without preemption [55, 66, 259], for bounded-degree graphs [55], or (recently) allowing vertices to be matched during some known time interval [163, 165].

We elaborate on the last relaxation above. In the model recently studied by Huang et al. [163, 165] vertices have both arrival and departure times, and edges can be matched whenever both their endpoints are present. (One-sided vertex arrivals is a special case of this model with all online vertices departing immediately after arrival and offline vertices departing at ∞.) We note that any α-competitive online matching under general vertex arrivals is α-competitive in the less restrictive model of Huang et al. As observed by Huang et al., for their model an optimal approach might as well be greedy; i.e., an unmatched vertex v should always be matched at its departure time if possible. In particular, Huang et al. [163, 165], showed that the RANKING algorithm of Karp et al. achieves a competitive ratio of ≈ 0.567. For general vertex arrivals, however, RANKING (and indeed any maximal matching algorithm) is no better than $1/2$ competitive, as is readily shown by a path on three edges with the internal vertices arriving first. Consequently, new ideas and algorithms are needed.

The natural open question for general vertex arrivals is whether a competitive ratio of $(1/2 + \Omega(1))$ is achievable by an *integral* randomized algorithm, without any assumptions (see e.g., [267]). In this chapter, we answer this question in the affirmative:

Theorem 4.1.1. *There exists a $(1/2 + \Omega(1))$-competitive randomized online matching algorithm for general adversarial vertex arrivals.*

4.1.1 Our Techniques

Here we outline the techniques underlying our results.

Our high-level approach here will be to round online a fractional online matching algorithm's output, specifically that of Wang and Wong [267]. While this approach sounds simple, there are several obstacles to overcome. First, the fractional matching polytope is not integral in general graphs, where a fractional matching may have value, $\sum_e x_e$, some $3/2$ times larger than the optimal matching size. (For example, in a triangle graph with value $x_e = 1/2$ for each edge e.) Therefore, any general rounding scheme must lose a factor of $3/2$ on the competitive ratio compared to the fractional algorithm's value, and so to beat a competitive ratio of $1/2$ would require an online fractional matching with competitive ratio $> 3/4 > 1 - 1/e$, which is impossible. To make matters worse, even in bipartite graphs, for which the fractional matching polytope is integral and offline lossless rounding is possible [4, 122], *online* lossless rounding of fractional matchings is impossible, even under one-sided vertex arrivals [69].

Despite these challenges, we show that a slightly better than $1/2$-competitive fractional matching computed by the algorithm of [267] can be rounded online without incurring too high a loss, yielding $(1/2+\Omega(1))$-competitive randomized algorithm for online matching under general vertex arrivals.

To outline our approach, we first consider a simple method to round matchings online. When vertex v arrives, we pick an edge $\{u, v\}$ with probability $z_u = x_{uv} / \Pr[u \text{ free when } v \text{ arrives}]$, and add it to our matching if u is free.

If $\sum_u z_u \leqslant 1$, this allows us to pick at most one edge per vertex and have each edge $e = \{u, v\}$ be in our matching with the right marginal probability, x_e, resulting in a lossless rounding. Unfortunately, we know of no better-than-$1/2$-competitive fractional algorithm for which this rounding guarantees $\sum_u z_u \leqslant 1$.

However, we observe that, for the correct set of parameters, the fractional matching algorithm of Wang and Wong [267] makes $\sum_u z_u$ close to one, while still ensuring a better-than-$1/2$-competitive fractional solution. Namely, as we elaborate later in Section 4.2.3, we set the parameters of their algorithm so that $\sum_u z_u \leqslant 1 + O(\varepsilon)$, while retaining a competitive ratio of $1/2 + O(\varepsilon)$. Now consider the same rounding algorithm with normalized probabilities: I.e., on v's arrival, sample a neighbor u with probability $z'_u = z_u / \max\{1, \sum_u z_u\}$ and match if u is free. As the sum of z_u's is slightly above one in the worst case, this approach does not drastically reduce the competitive ratio. But the normalization factor is still too significant compared to the competitive ratio of the fractional solution, driving the competitive ratio of the rounding algorithm slightly below $1/2$.

To account for this minor yet significant loss, we therefore augment the simple algorithm by allowing it, with small probability (e.g., say $\sqrt{\varepsilon}$), to sample a second neighbor u_2 for each arriving vertex v, again with probabilities proportional to z'_{u_2}: If the first sampled choice, u_1, is free, we match v to u_1. Otherwise, if the second choice, u_2, is free, we match v to u_2. What is the marginal probability that such an approach matches an incoming vertex v to a given neighbor u? Letting F_u denote the event that u is free when v arrives, this probability is precisely

$$\Pr[F_u] \cdot \left(z'_u + z'_u \cdot \sqrt{\varepsilon} \cdot \sum_w z'_w \cdot (1 - \Pr[F_w \mid F_u]) \right). \tag{4.1}$$

Here the first term in the parentheses corresponds to the probability that v matches to u via the first choice, and the second term corresponds to the same happening via the second choice (which is only taken when the first choice fails).

Ideally, we would like (4.1) to be at least x_{uv} for all edges, which would imply a lossless rounding. However, as mentioned earlier, this is difficult and in general impossible to do, even in much more restricted settings including one-sided bipartite vertex arrivals. We therefore settle for showing that (4.1) is at least $x_{uv} = \Pr[F_u] \cdot z_u$ for *most* edges (weighted by x_{uv}). Even this goal, however, is challenging and requires a nontrivial understanding of the correlation structure of the random events F_u. To see this, note that for example if the F_w events are perfectly positively correlated, i.e., $\Pr[F_w \mid F_u] = 1$, then the possibility of picking $e = \{u, v\}$ as a second edge does not increase this edge's probability of being matched *at all* compared to if we only picked a single edge per vertex. This results in e being matched with probability $\Pr[F_u] \cdot z'_u = \Pr[F_u] \cdot z_u / \sum_w z_w = x_{uv} / \sum_w z_w$, which does not lead to any gain over the $1/2$ competitive ratio of greedy. Such problems are easily shown not to arise if all F_u variables are independent or negatively correlated. Unfortunately, positive correlation does arise from this process, and so we the need to control these positive correlations.

The core of our analysis is therefore dedicated to showing that even though positive correlations do arise, they are, by and large, rather weak. Our main technical contribution consists of developing techniques for bounding such positive correlations. The idea behind the analysis is to consider the primary choices and secondary choices of vertices as defining a graph, and showing that after a natural pruning operation that reflects the structure of dependencies, most vertices are most often part of a very small connected component in the graph. The fact that connected components are typically very small is exactly what makes positive correlations weak and results in the required lower bound on (4.1) for most edges (in terms of x-value), which in turn yields our $1/2 + \Omega(1)$ competitive ratio.

4.2 General Vertex Arrivals

In this section we present a $(1/2 + \Omega(1))$-competitive randomized algorithm for online matching under general arrivals. As discussed in the introduction, our approach will be to round (online) a *fractional* online matching algorithm's output. Specifically, this will be an algorithm from the family of fractional algorithms introduced in [267]. In Section 4.2.1 we describe this family of algorithms. To motivate our rounding approach, in Section 4.2.2 we first present a simple lossless rounding method for a $1/2$-competitive algorithm in this family. In Section 4.2.3 we then describe our rounding algorithm for a better-than-$1/2$-competitive algorithm in this family. Finally, in Section 4.2.4 we analyze this rounding scheme, and show that it yields a $(1/2 + \Omega(1))$-competitive algorithm.

4.2.1 Finding a Fractional Solution

In this section we revisit the algorithm of Wang and Wong [267], which beats the $1/2$ competitiveness barrier for online fractional matching under general vertex arrivals. Their algorithm (technically, family of algorithms) applies the primal-dual method to compute both a fractional matching and a fractional vertex cover – the dual of the fractional matching relaxation. The LPs defining these dual problems are as follows.

$$\begin{array}{rlll} & \text{Primal-Matching} & & \\ \text{maximize} & \sum_{e\in E} x_e & & \\ \text{subject to:} & \sum_{u\in N(v)} x_{uv} \leqslant 1 & \forall u \in V \\ & x_e \geqslant 0 & \forall e \in E \end{array} \qquad \begin{array}{rlll} & \text{Dual-Vertex Cover} & & \\ \text{minimize} & \sum_{u\in V} y_u & & \\ \text{subject to:} & y_u + y_v \geqslant 1 & \forall e = \{u,v\} \in E \\ & y_u \geqslant 0 & \forall u \in V \end{array}$$

Before introducing the algorithm of [267], we begin by defining the fractional online vertex cover problem for vertex arrivals. When a vertex v arrives, if $N_v(v)$ denotes the previously-arrived neighbors of v, then for each $u \in N_v(v)$, a new constraint $y_u + y_v \geqslant 1$ is revealed, which an online algorithm should satisfy by possibly increasing y_u or y_v. Suppose v has its dual value set to $y_v = 1 - \theta$. Then all of its neighbors should have their dual increased to at least θ. Indeed, an algorithm may as well increase y_u to $\max\{y_u, \theta\}$. The choice of θ therefore determines an online fractional vertex cover algorithm. The increase of potential due to the newly-arrived vertex v is thus $1 - \theta + \sum_{u\in N_v(v)} (\theta - y_u)^+$. In [267] θ is chosen to upper bound this term by $1 - \theta + f(\theta)$ for some function $f(\cdot)$. The primal solution (fractional matching) assigns values x_{uv} so as to guarantee feasibility of $\vec{x}$ and a ratio of β between the primal and dual values of $\vec{x}$ and $\vec{y}$, implying $\frac{1}{\beta}$-competitiveness of this online fractional matching algorithm, by feasibility of $\vec{y}$ and weak duality. The algorithm, parameterized by a function $f(\cdot)$ and parameter β to be discussed below, is given formally in Algorithm 1. In the subsequent discussion, $N_v(u)$ denotes the set of neighbors of u that arrive before v.

Algorithm 1 is parameterized by a function f and a constant β. The family of functions considered by [267] are as follows.

Definition 4.2.1. *Let* $f_\kappa(\theta) := \left(\frac{1+\kappa}{2} - \theta\right)^{\frac{1+\kappa}{2\kappa}} \left(\theta + \frac{\kappa-1}{2}\right)^{\frac{\kappa-1}{2\kappa}}$. *We define* $W := \{f_\kappa \mid \kappa \geqslant 1\}$.

Algorithm 1 Online general vertex arrival fractional matching and vertex cover

```
Input: A stream of vertices $v_1, v_2, \dots v_n$. At step $i$, vertex $v_i$ and $N_{v_i}(v_i)$ are revealed
Output: A fractional vertex cover solution $\vec{y}$ and a fractional matching $\vec{x}$
1: let $y_u \leftarrow 0$ for all $u$, let $x_{uv} \leftarrow 0$ for all $u, v$
2: for each vertex $v$ in the stream do
3:     $\theta \leftarrow \max\{\theta \leqslant 1 \mid \sum_{u \in N_v(v)} (\theta - y_u)^+ \leqslant f(\theta)\}$
4:     for each neighbor $u \in N_v(v)$ do
5:         $x_{uv} \leftarrow \frac{(\theta - y_u)^+}{\beta} \left(1 + \frac{1-\theta}{f(\theta)}\right)$
6:         $y_u \leftarrow \max\{y_u, \theta\}$
7:     $y_v \leftarrow 1 - \theta$
```

As we will see, choices of β guaranteeing feasibility of $\vec{x}$ are related to the following quantity.

Definition 4.2.2. *For a given $f : [0,1] \to \mathbb{R}_+$ let*

$$\beta^*(f) := \max_{\theta \in [0,1]} 1 + f(1-\theta) + \int_\theta^1 \frac{1-t}{f(t)}\, dt.$$

For functions $f \in W$ this definition of $\beta^*(f)$ can be simplified to $\beta^*(f) = 1 + f(0)$, due to the observation (see [267, Lemmas 4,5]) that all functions $f \in W$ satisfy

$$\beta^*(f) = 1 + f(1-\theta) + \int_\theta^1 \frac{1-t}{f(t)}\, dt \qquad \forall \theta \in [0,1]. \tag{4.2}$$

As mentioned above, the competitiveness of Algorithm 1 for appropriate choices of f and β is obtained by relating the overall primal and dual values, $\sum_e x_e$ and $\sum_v y_v$. As we show (and rely on later), one can even bound individual vertices' contributions to these sums. In particular, for any vertex v's arrival time, each vertex u's contribution to $\sum_e x_e$, which we refer to as its *fractional degree*, $x_u := \sum_{w \in N_v(u)} x_{uw}$, can be bounded in terms of its dual value by this point, y_u, as follows.

Lemma 4.2.3. *For any vertices $u, v \in V$, let y_u be the potential of u prior to arrival of v. Then the fractional degree just before v arrives, $x_u := \sum_{w \in N_v(u)} x_{uw}$, is bounded as follows:*

$$\frac{y_u}{\beta} \leqslant x_u \leqslant \frac{y_u + f(1-y_u)}{\beta}.$$

Broadly, the lower bound on x_u is obtained by lower bounding the increase x_u by the increase to y_u/β after each vertex arrival, while the upper bound follows from a simplification of a bound given in [267, Invariant 1] (implying feasibility of the primal solution), which we simplify using (4.2). See Section 4.4 for a full proof.

Another observation we will need regarding the functions $f \in W$ is that they are decreasing.

Observation 4.2.4. *Every function $f \in W$ is non-increasing in the range $[0, 1]$.*

Proof. As observed in [267], differentiating (4.2) with respect to z yields $-f'(1-z) - \frac{1-z}{f(z)} = 0$, from which we obtain $f(z) \cdot f'(1-z) = z-1$. Replacing z by $1-z$, we get $f(1-z) \cdot f'(z) = -z$, or $f'(z) = -\frac{z}{f(1-z)}$. As $f(z)$ is positive for all $z \in [0, 1]$, we have that $f'(z) \leqslant 0$ for all $z \in [0, 1]$. □

The next lemma of [267] characterizes the achievable competitiveness of Algorithm 1.

Lemma 4.2.5. *Algorithm 1 with function $f \in W$ and $\beta \geqslant \beta^*(f) = 1 + f(0)$ is $\frac{1}{\beta}$ competitive.*

Wang and Wong [267] showed that taking $\kappa \approx 1.1997$ and $\beta = \beta^*(f_\kappa)$, Algorithm 1 is ≈ 0.526 competitive. In later sections we show how to round the output of Algorithm 1 with f_κ with $\kappa = 1 + 2\varepsilon$ for some small constant ε and $\beta = 2 - \varepsilon$ to obtain a $(1/2 + \Omega(1))$-competitive algorithm. But first, as a warm up, we show how to round this algorithm with $\kappa = 1$ and $\beta = \beta^*(f_1) = 2$.

4.2.2 Warmup: a $1/2$-Competitive Randomized Algorithm

In this section we will round the $1/2$-competitive fractional algorithm obtained by running Algorithm 1 with function $f(\theta) = f_1(\theta) = 1 - \theta$ and $\beta = \beta^*(f) = 2$. We will devise a lossless rounding of this fractional matching algorithm, by including each edge e in the final matching with a probability equal to the fractional value x_e assigned to it by Algorithm 1. Note that if v arrives after u, then if F_u denotes the event that u is free when v arrives, then edge $\{u, v\}$ is matched by an online algorithm with probability $\Pr[\{u, v\} \in M] = \Pr[\{u, v\} \in M \mid F_u] \cdot \Pr[F_u]$. Therefore, to match each edge $\{u, v\}$ with probability x_{uv}, we need $\Pr[\{u, v\} \in M \mid F_u] = x_{uv} / \Pr[F_u]$. That is, we must match $\{u, v\}$ with probability $z_u = x_{uv} / \Pr[F_u]$ conditioned on u being free. The simplest way of doing so (if possible) is to pick an edge $\{u, v\}$ with the above probability z_u always, and to match it only if u is free. Algorithm 2 below does just this, achieving a lossless rounding of this fractional algorithm. As before, $N_v(u)$ denotes the set of neighbors of u that arrive before v.

Algorithm 2 is well defined if for each vertex v's arrival, z is a probability distribution; i.e., $\sum_{u \in N_v(v)} z_u \leqslant 1$. The following lemma asserts precisely that. Moreover, it asserts that Algorithm 2 matches each edge with the desired probability.

Lemma 4.2.6. *Algorithm 2 is well defined, since for every vertex v on arrival, z is a valid probability distribution. Moreover, for each v and $u \in N_v(v)$, it matches edge $\{u, v\}$ with probability x_e.*

Algorithm 2 Online vertex arrival warmup randomized fractional matching

```
Input: A stream of vertices $v_1, v_2, \ldots v_n$. At step $i$, vertex $v_i$ and $N_{v_i}(v_i)$ are revealed
Output: A matching $M$
1: let $y_u \leftarrow 0$ for all $u$, let $x_{uv} \leftarrow 0$ for all $u, v$
2: let $M \leftarrow \emptyset$
3: for each $v$ in the stream do
4:     update $y_u$'s and $x_{uv}$'s using Algorithm 1 with $\beta = 2$ and $f = f_1$
5:     for each neighbor $u \in N_v(v)$ do
6:         $z_u \leftarrow \frac{x_{uv}}{\Pr[u \text{ is free when } v \text{ arrives}]}$        ▷ $z_u = x_{uv}/(1-y_u)$, as shown later
7:     sample (at most) one neighbor $u \in N_v(v)$ according to $z_u$
8:     if a free neighbor $u$ is sampled then
9:         add $\{u, v\}$ to $M$
```

Proof. We prove both claims in tandem for each v, by induction on the number of arrivals. For the base case (v is the first arrival), the set $N_v(v)$ is empty and thus both claims are trivial. Consider the arrival of a later vertex v. By the inductive hypothesis we have that each vertex $u \in N_v(v)$ is previously matched with probability $\sum_{w \in N_v(u)} x_{wu}$. But by our choice of $f(\theta) = f_1(\theta) = 1 - \theta$ and $\beta = 2$, if w arrives after u, then y_u and θ at arrival of w satisfy $x_{uw} = \frac{(\theta - y_u)^+}{\beta} \cdot \left(1 + \frac{1-\theta}{f(\theta)}\right) = (\theta - y_u)^+$. That is, x_{uw} is precisely the increase in y_u following arrival of w. On the other hand, when u arrived we have that its dual value y_u increased by $1 - \theta = \sum_{v' \in N_u(u)} (\theta - y_{v'})^+ = \sum_{v' \in N_u(u)} x_{uv'}$. To see this last step, we recall first that by definition of Algorithm 1 and our choice of $f(\theta) = 1 - \theta$, the value θ on arrival of v is chosen to be the largest $\theta \leqslant 1$ satisfying

$$\sum_{\forall u \in N_v(v)} (\theta - y_u)^+ \leqslant 1 - \theta. \tag{4.3}$$

But the inequality (4.3) is an equality whether or not $\theta = 1$ (if $\theta = 1$, both sides are zero). We conclude that $y_u = \sum_{v' \in N_v(u)} x_{uv'}$ just prior to arrival of v. But then, by the inductive hypothesis, this implies that $\Pr[u \text{ free when } v \text{ arrives}] = 1 - y_u$ (yielding an easily-computable formula for z_u). Consequently, by (4.3) we have that when v arrives z is a probability distribution, as

$$\sum_{u \in N_v(v)} z_u = \sum_{u \in N_v(v)} \frac{(\theta - y_u)^+}{1 - y_u} \leqslant \sum_{u \in N_v(v):\, y_u \leqslant \theta} \frac{(\theta - y_u)^+}{1 - \theta} = \sum_{u \in N_v(v)} \frac{(\theta - y_u)^+}{1 - \theta} \leqslant 1.$$

Finally, for u to be matched to a latter-arriving neighbor v, it must be picked and free when v arrives, and so $\{u, v\}$ is indeed matched with probability

$$\Pr[\{u, v\} \in M] = \frac{x_{uv}}{\Pr[u \text{ is free when } v \text{ arrives}]} \cdot \Pr[u \text{ is free when } v \text{ arrives}] = x_{uv}. \qquad \square$$

In the next section we present an algorithm which allows to round better-than-$1/2$-competitive algorithms derived from Algorithm 1.

4.2.3 An Improved Algorithm

In this section, we build on Algorithm 2 and show how to improve it to get a $(1/2 + \Omega(1))$ competitive ratio.

There are two concerns when modifying Algorithm 2 to work for a general function from the family W. The first is how to compute the probability that a vertex u is free when vertex v arrives, in Line 6. In the simpler version, we inductively showed that this probability is simply $1 - y_u$, where y_u is the dual value of u as of v's arrival (see the proof of Lemma 4.2.6). With a general function f, this probability is no longer given by a simple formula. Nevertheless, it is easily fixable: We can either use Monte Carlo sampling to estimate the probability of u being free at v's arrival to a given inverse polynomial accuracy, or we can in fact exactly compute these probabilities by maintaining their marginal values as the algorithm progresses. In what follows, we therefore assume that our algorithm can compute these probabilities exactly.

The second and more important issue is with the sampling step in Line 7. In the simpler algorithm, this step is well-defined as the sampling probabilities indeed form a valid distribution: I.e., $\sum_{u \in N_v(v)} z_u \leqslant 1$ for all vertices v. However, with a general function f, this sum can exceed one, rendering the sampling step in Line 7 impossible. Intuitively, we can normalize the probabilities to make it a proper distribution, but by doing so, we end up losing some amount from the approximation guarantee. We hope to recover this loss using a second sampling step, as we mentioned in Section 9.1.1 and elaborate below.

Suppose that, instead of $\beta = 2$ and $f = f_1$ (i.e., the function $f(\theta) = 1 - \theta$), we use $f = f_{1+2\varepsilon}$ and $\beta = 2 - \varepsilon$ to define x_{uv} and y_u values. As we show later in this section, for an ε sufficiently small, we then have $\sum_{u \in N_v(v)} z_u \leqslant 1 + O(\varepsilon)$, implying that the normalization factor is at most $1 + O(\varepsilon)$. However, since the approximation factor of the fractional solution is only $1/2 + O(\varepsilon)$ for such a solution, (i.e., $\sum_{\{u,v\} \in E} x_{uv} \geqslant (1/\beta) \cdot \sum_{u \in V} y_u$), the loss due to normalization is too significant to ignore.

Now suppose that we allow arriving vertices to sample a second edge with a small (i.e., $\sqrt{\varepsilon}$) probability and match that second edge if the endpoint of the first sampled edge is already matched. Consider the arrival of a fixed vertex v such that $\sum_{u \in N_v(v)} z_u > 1$, and let z'_u denote the normalized z_u values. Further let F_w denote the event that vertex w is free (i.e, unmatched) at the arrival of v. Then the probability that v matches u for some $u \in N_v(v)$ using either of the two sampled edges is

$$\Pr[F_u] \cdot \left(z'_u + z'_u \sqrt{\varepsilon} \cdot \sum_{w \in N_v(v)} z'_w \cdot (1 - \Pr[F_w \mid F_u]) \right), \tag{4.4}$$

which is the same expression from (4.1) from Section 9.1.1, restated here for quick reference. Recall that the first term inside the parentheses accounts for the probability that v matches u via the first sampled edges, and the second term accounts for the probability that the same happens via the second sampled edge. Note that the second sampled edge is used only when the first one is incident to an already matched vertex and the other endpoint of the second edge is free. Hence we have the summation of conditional probabilities in the second term, where the events are conditioned on the other endpoint, u, being free. If the probability given in (4.4) is x_{uv} for all $\{u, v\} \in E$, we would have the same guarantee as the fractional solution x_{uv}, and the rounding would be lossless. This seems unlikely, yet we can show that the quantity in (4.4) is at least $(1-\varepsilon^2)\cdot x_{uv}$ for most (not by number, but by the total fractional value of x_{uv}'s) of the edges in the graph, showing that our rounding is *almost* lossless. We postpone further discussion of the analysis to Section 4.2.4 where we highlight the main ideas before proceeding with the formal proof.

Algorithm 3 A randomized online matching algorithm under general vertex arrivals.

```
Input: A stream of vertices $v_1, v_2, \dots v_n$. At step $i$, vertex $v_i$ and $N_{v_i}(v_i)$ are revealed
Output: A matching $M$
 1: let $y_u \leftarrow 0$ for all $u$, let $x_{uv} \leftarrow 0$ for all $u, v$
 2: let $M \leftarrow \emptyset$
 3: for each $v$ in the stream do
 4:     update $y_u$'s and $x_{uv}$'s using Algorithm 1 with $\beta = 2-\varepsilon$ and $f = f_{1+2\varepsilon}$
 5:     for each neighbor $u \in N_v(v)$ do
 6:         $z_u \leftarrow \frac{x_{uv}}{\Pr[u \text{ is free when } v \text{ arrives}]}$    ▷ computing $\Pr[u$ is free when $v$ arrives$]$ as explained in Section 4.2.3
 7:     for each neighbor $u \in N_v(v)$ do
 8:         $z'_u \leftarrow z_u / \max\left\{1, \sum_{u\in N_v(v)} z_u\right\}$
 9:     pick (at most) one $u_1 \in N_v(v)$ with probability $z'_{u_1}$
10:     if $\sum_{u\in N_v(v)} z_u > 1$ then
11:         with probability $\sqrt{\varepsilon}$ do
12:             pick (at most) one $u_2 \in N_v(v)$ with probability $z'_{u_2}$
13:             drop $u_2$ with minimal probability ensuring $\{u_2, v\}$ is matched with probability at
                    most $x_{u_2 v}$    ▷ this probability can be computed using (4.4)
14:     if a free neighbor $u_1$ is sampled then
15:         add $\{u_1, v\}$ to $M$
16:     else if a free neighbor $u_2$ is sampled then
17:         add $\{u_2, v\}$ to $M$
```

Our improved algorithm is outlined in Algorithm 3. Up until Line 6, it is similar to Algorithm 2 except that it uses $\beta = 2-\varepsilon$ and $f = f_{1+2\varepsilon}$ where we choose $\varepsilon > 0$ to be any constant small enough such that the results in the analysis hold. In Line 8, if the sum of z_u's exceeds one we normalize the z_u to obtain a valid probability distribution z'_u. In Line 9, we sample the first edge incident to an arriving vertex v. In Line 12, we sample a second edge incident to the same vertex with probability $\sqrt{\varepsilon}$ if we had to scale down z_u's in Line 8. Then in Line 13, we drop the sampled second edge with the minimal probability to ensure that no edge $\{u, v\}$ is matched with probability more than x_{uv}. Since (4.4) gives the exact probability of $\{u, v\}$ being matched, this probability of dropping an edge $\{u, v\}$ can be computed by the algorithm. However, to compute this, we need the conditional probabilities $\Pr[F_w \mid F_u]$, which again can be estimated using Monte Carlo sampling, (Alternatively, it is also possible to compute them exactly if we allow the algorithm to take exponential time.) In the subsequent lines, we match v to a chosen free neighbor (if any) among its chosen neighbors, prioritizing its first choice.

For the purpose of analysis we view Algorithm 3 as constructing a greedy matching on a directed acyclic graph (DAG) H_τ defined in the following two definitions.

Definition 4.2.7 (Non-adaptive selection graph G_τ)**.** *Let τ denote the random choices made by the vertices of G. Let G_τ be the DAG defined by all the arcs (v, u_1), (v, u_2) for all vertices $v \in V$. We call the arcs (v, u_1)* primary *arcs, and the arcs (v, u_2) the* secondary *arcs.*

Definition 4.2.8 (Pruned selection graph H_τ)**.** *Now construct H_τ from G_τ by removing all arcs (v, u) (primary or secondary) such that there exists a* primary *arc (v', u) with v' arriving before v. We further remove a secondary arc (v, u) if there is a primary arc (v, u); i.e., if a vertex u has at least one incoming primary arc, remove all incoming primary arcs that came after the first primary arc and all secondary arcs that came after or from the same vertex as the first primary arc.*

It is easy to see that the matching constructed by Algorithm 3 is a greedy matching constructed on H_τ based on order of arrival and prioritizing primary arcs. The following lemma shows that the set of matched vertices obtained by this greedy matching does not change much for any change in the random choices of a single vertex v, which will prove useful later on. It can be proven rather directly by an inductive argument showing the size of the symmetric difference in matched vertices in G_τ and $G_{\tau'}$ does not increase after each arrival besides the arrival of v, whose arrival clearly increases this symmetric difference by at most two. See Section 4.3 for details.

Lemma 4.2.9. *Let G_τ and $G_{\tau'}$ be two realizations of the random digraph where all the vertices in the two graphs make the same choices except for* one *vertex v. Then the number of vertices that have different* matched status *(free/matched) in the matchings computed in H_τ and $H_{\tau'}$ at any point of time is at most two.*

4.2.4 Analysis

In this section, we analyze the competitive ratio of Algorithm 3. We start with an outline of the analysis where we highlight the main ideas.

High-Level Description of Analysis

As described in Section 4.2.3, the main difference compared to the simpler $1/2$-competitive algorithm is the change of the construction of the fractional solution, which in turn makes the rounding more complex. In particular, we may have at the arrival of a vertex v that $\sum_{u \in N_v(v)} z_u > 1$. The majority of the analysis is therefore devoted to such "problematic" vertices since otherwise, if $\sum_{u \in N_v(v)} z_u \leqslant 1$, the rounding is lossless due to the same reasons as described in the simpler setting of Section 4.2.2. We now outline the main ideas in analyzing a vertex v with $\sum_{u\in N_v(v)} z_u > 1$. Let F_w be the event that vertex w is free (i.e., unmatched) at the arrival of v. Then, as described in Section 4.2.3, the probability that we select edge $\{u,v\}$ in our matching is the minimum of x_{uv} (because of the pruning in Line 13), and

$$\Pr[F_u] \cdot \left(z'_u + z'_u \sqrt{\varepsilon} \cdot \sum_{w \in N_v(v)} z'_w \cdot (1 - \Pr[F_w \mid F_u]) \right).$$

By definition, $\Pr[F_u] \cdot z_u = x_{uv}$, and the expression inside the parentheses is at least z_u (implying $\Pr[\{u,v\} \in M] = x_{uv}$) if

$$1 + \sqrt{\varepsilon} \cdot \sum_{w \in N_v(v)} z'_w \cdot (1 - \Pr[F_w \mid F_u]) \geqslant \frac{z_u}{z'_u}. \tag{4.5}$$

To analyze this inequality, we first use the structure of the selected function $f = f_{1+2\varepsilon}$ and the selection of $\beta = 2 - \varepsilon$ to show that if $\sum_{u \in N_v(v)} z_u > 1$ then several structural properties hold (see Lemma 4.2.10 and Corollary 4.2.11 in Section 4.2.4). In particular, there are absolute constants $0 < c < 1$ and $C > 1$ (both independent of ε) such that

1. $\sum_{u \in N_v(v)} z_u \leqslant 1 + C\varepsilon$;
2. $z_u \leqslant C\sqrt{\varepsilon}$ for every $u \in N_v(v)$; and
3. $c \leqslant \Pr[F_w] \leqslant 1 - c$ for every $w \in N_v(v)$.

The first property implies that the right-hand-side of (4.5) is at most $1 + C\varepsilon$; and the second property implies that v has at least $\Omega(1/\sqrt{\varepsilon})$ neighbors and that each neighbor u satisfies $z'_u \leqslant z_u \leqslant C\sqrt{\varepsilon}$.

For simplicity of notation, we assume further in the high-level overview that v has exactly $1/\sqrt{\varepsilon}$ neighbors and each $u \in N_v(v)$ satisfies $z'_u = \sqrt{\varepsilon}$. Inequality (4.5) would then be implied by

$$\sum_{w \in N_v(v)} (1 - \Pr[F_w \mid F_u]) \geqslant C\,. \tag{4.6}$$

To get an intuition why we would expect the above inequality to hold, it is instructive to consider the unconditional version:

$$\sum_{w \in N_v(v)} (1 - \Pr[F_w]) \geqslant c|N_v(v)| = c/\sqrt{\varepsilon} \gg C\,,$$

where the first inequality is from the fact that $\Pr[F_w] \leqslant 1 - c$ for any neighbor $w \in N_v(v)$. The large slack in the last inequality, obtained by selecting $\varepsilon > 0$ to be a sufficiently small constant, is used to bound the impact of conditioning on the event F_u. Indeed, due to the large slack, we have that (4.6) is satisfied if the quantity $\sum_{w\in N_v(v)} \Pr[F_w | F_u]$ is not too far away from the same summation with unconditional probabilities, i.e., $\sum_{w \in N_v(v)} \Pr[F_w]$. Specifically, it is sufficient to show

$$\sum_{w \in N_v(v)} (\Pr[F_w|F_u] - \Pr[F_w]) \leqslant c/\sqrt{\varepsilon} - C\,. \tag{4.7}$$

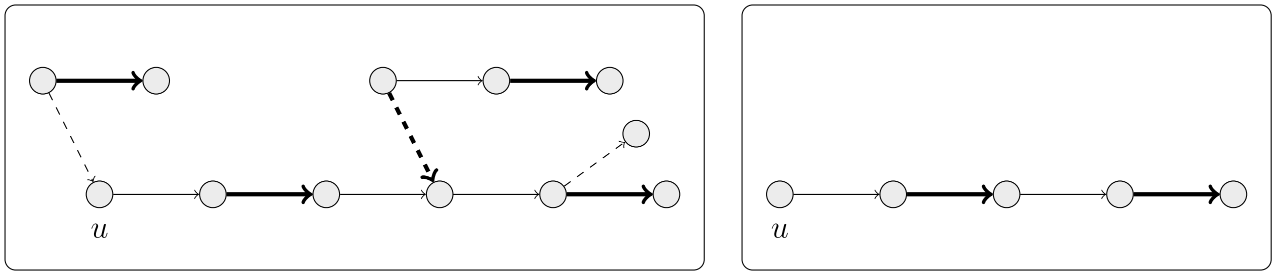


Figure 4.1: Two examples of the component of H_τ containing u

Vertices are depicted from right to left in the arrival order. Primary and secondary arcs are solid and dashed, respectively. The edges that take part in the matching are thick.

We do so by bounding the correlation between the events F_u and F_w in a highly non-trivial manner, which constitutes the heart of our analysis. The main challenges are that events F_u and F_w can be positively correlated and that, by conditioning on F_u, the primary and secondary choices of different vertices are no longer independent.

We overcome the last difficulty by replacing the conditioning on F_u by a conditioning on the component in H_τ (at the time of v's arrival) that includes u. As explained in Section 4.2.3, the matching output by our algorithm is equivalent to the greedy matching constructed in H_τ and so the component containing u (at the time of v's arrival) determines F_u. But how can this component look like, assuming the event F_u? First, u cannot have any incoming primary arc since then u would be matched (and so the event F_u would be false). However, u could have incoming secondary arcs, assuming that the tails of those arcs are matched using their primary arcs. Furthermore, u can have an outgoing primary and possibly a secondary arc if the selected neighbors are already matched. These neighbors can in turn have incoming secondary arcs, at most one incoming primary arc (due to the pruning in the definition of H_τ), and outgoing primary and secondary arcs; and so on. In Figure 4.1, we give two examples of the possible structure, when conditioning on F_u, of u's component in H_τ (at the time of v's arrival). The left example contains secondary arcs, whereas the component on the right is arguably simpler and only contains primary arcs.

An important step in our proof is to prove that, for most vertices u, the component is of the simple form depicted to the right with probability almost one. That is, it is a path P consisting of primary arcs, referred to as a primary path (see Definition 4.2.13) that further satisfies:

(i) it has length $O(\ln(1/\varepsilon))$; and
(ii) the total z-value of the arcs in the blocking set of P is $O(\ln(1/\varepsilon))$. The blocking set is defined in Definition 4.2.14. Informally, it contains those arcs that if appearing as primary arcs in G_τ would cause arcs of P to be pruned (or blocked) from H_τ.

Let $\mathcal{P}$ be the primary paths of above type that appear with positive probability as u's component in H_τ. Further let EQ_P be the event that u's component equals P. Then we show (for most vertices) that $\sum_{P\in\mathcal{P}} \Pr[\mathrm{EQ}_P \mid F_u]$ is almost one. For simplicity, let us assume here that the sum is equal to one. Then by the law of total probability and since $\sum_{P\in\mathcal{P}} \Pr[\mathrm{EQ}_P \mid F_u] = 1$,

$$\sum_{w\in N_v(v)} (\Pr[F_w \mid F_u] - \Pr[F_w]) = \sum_{P\in\mathcal{P}} \Pr[\mathrm{EQ}_P \mid F_u] \left(\sum_{w\in N_v(v)} (\Pr[F_w \mid F_u, \mathrm{EQ}_P] - \Pr[F_w]) \right)$$
$$= \sum_{P\in\mathcal{P}} \Pr[\mathrm{EQ}_P \mid F_u] \left(\sum_{w\in N_v(v)} (\Pr[F_w \mid \mathrm{EQ}_P] - \Pr[F_w]) \right),$$

where the last equality is because the component P determines F_u. The proof is then completed by analyzing the term inside the parentheses for each primary path $P \in \mathcal{P}$ separately. As we prove in Lemma 4.2.15, the independence of primary and secondary arc choices of vertices is maintained after conditioning on EQ_P.[1] Furthermore, we show that there is a bijection between the outcomes of the unconditional and the conditional distributions, so that the expected number of vertices that make different choices under this pairing can be upper bounded by roughly the length of the path plus the z-value of the edges in the blocking set. So, for a path P as above, we have that the expected number of vertices that make different choices in the paired outcomes is $O(\ln(1/\varepsilon))$ which, by Lemma 4.2.9, implies that the expected number of vertices that change matched status is also upper bounded by $O(\ln(1/\varepsilon))$. In other words, we have for every $P \in \mathcal{P}$ that

$$\sum_{w\in N_v(v)} (\Pr[F_w|\mathrm{EQ}_P] - \Pr[F_w]) \leqslant \sum_{w\in V} (\Pr[F_w|\mathrm{EQ}_P] - \Pr[F_w]) = O(\ln(1/\varepsilon)),$$

which implies (4.7) for a small enough choice of ε. This completes the overview of the main steps in the analysis. The main difference in the formal proof is that not all vertices satisfy that their component is a short primary path with probability close to 1. To that end, we define the notion of *good* vertices in Section 4.2.4, which are the vertices that are very unlikely to have long directed paths of primary arcs rooted at them. These are exactly the vertices v for which we can perform the above analysis for most neighbors u (in the proof of the "key lemma") implying that the rounding is almost lossless for v. Then, in Section 4.2.4, we show using a rather simple charging scheme that most of the vertices in the graph are good. Finally, in Section 4.2.4, we put everything together and prove Theorem 4.1.1.

Useful Properties of W Functions and Algorithm 3

For the choice of $f = f_{1+2\varepsilon}$ as we choose, we have $f(\theta) = (1+\varepsilon-\theta) \cdot \left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right)^{\frac{\varepsilon}{1+2\varepsilon}}$. In Section 4.5 we give a more manageable upper bound for $f(\theta)$ which holds for sufficiently small ε. Based on this simple upper bound on f and some basic calculus, we obtain the following useful structural properties for the conditional probabilities, z_u, of Algorithm 3. See Section 4.5.

[1]To be precise, we condition on a primary path P with a so-called *termination certificate* T, see Definition 4.2.13. In the overview, we omit this detail, and consider the event $\mathrm{EQ}_{P,T}$ (instead of EQ_P) in the formal proof.

Lemma 4.2.10. *(Basic bounds on conditional probabilities z_u) There exist absolute constants $c \in (0,1)$ and $C > 1/c > 1$ and $\varepsilon_0 \in (0,1)$ such that for every $\varepsilon \in (0, \varepsilon_0)$ the following holds: for every vertex $v \in V$, if y_u is the dual variable of a neighbor $u \in N_v(v)$ before v's arrival and θ is the value chosen by Algorithm 1 on v's arrival, then for z_u as defined in Algorithm 3, we have:*

(1) *If $\theta \notin (c, 1-c)$, then $\sum_{u \in N_v(v)} z_u \leqslant 1$,*

(2) *If $\theta \in [0,1]$, then $\sum_{u \in N_v(v)} z_u \leqslant 1 + C\varepsilon$,*

(3) *If $\sum_{u \in N_v(v)} z_u > 1$, then $z_u \leqslant C\sqrt{\varepsilon}$ for every $u \in N_v(v)$,*

(4) *If $\sum_{u \in N_v(v)} z_u > 1$, then for every $u \in N_v(v)$ such that $z_u > 0$, one has $y_u \in [c/2, 1 - c/2]$, and*

(5) *For all $u \in N_v(v)$, one has $z_u \leqslant 1/2 + O(\sqrt{\varepsilon})$.*

The following corollary will be critical to our analysis:

Corollary 4.2.11. *There exist absolute constants $c > 0$ and $\varepsilon_0 > 0$ such that for all $\varepsilon \in (0, \varepsilon)$, on arrival of any vertex $v \in V$, if z as defined in Algorithm 3 satisfies $\sum_{u \in N_v(v)} z_u > 1$, then for every $u \in N_v(v)$ we have*

$$c \leqslant \Pr[u \textit{ is free when } v \textit{ arrives}] \leqslant 1 - c.$$

Proof. By Lemma 4.2.10, **(1)** and **(4)** we have that if $\sum_{u \in N_v(v)} z_u > 1$, then $\theta \in (c, 1-c)$ (c is the constant from Lemma 4.2.10), and for every $u \in N_v(v)$ one has

$$y_u \in [c/2, 1 - c/2]. \tag{4.8}$$

On the other hand, by Lemma 4.2.3 one has

$$\frac{y_u}{\beta} \leqslant x_u \leqslant \frac{y_u + f(1 - y_u)}{\beta}, \tag{4.9}$$

where x_u is the fractional degree of u when v arrives.

We now note that by Lemma 4.2.10, **(2)**, we have that Algorithm 3 matches every vertex u with probability at least $x_u/(1 + C\varepsilon)$ (due to choices of primary arcs), and thus

$$\begin{aligned}
\Pr[u \text{ is free when } v \text{ arrives}] &\leqslant 1 - \frac{x_u}{1 + C\varepsilon} \\
&\leqslant 1 - \frac{y_u}{\beta(1 + C\varepsilon)} && \text{(by (4.9))} \\
&\leqslant 1 - \frac{c/2}{2(1 + C\varepsilon)} && \text{(by (4.8) and the setting } \beta = 2 - \varepsilon \leqslant 2) \\
&\leqslant 1 - c/5,
\end{aligned}$$

as long as ε is sufficiently small.

For the other bound we will use two facts. The first is that the since $f(y)$ is monotone decreasing by Observation 4.2.4 and since we picked $\beta > \beta^*(f) = 1 + f(0)$, we have that for any $y \leqslant 1 - c/2 \leqslant 1$,

$$y + f(1-y) \leqslant 1 - c/2 + f(0) < \beta - c/2. \tag{4.10}$$

Then, using the fact that by Line 13, Algorithm 3 matches every vertex u with probability at most x_u, we obtain the second bound, as follows.

$$\begin{aligned}
\Pr[u \text{ is free when } v \text{ arrives}] &\geqslant 1 - x_u \\
&\geqslant 1 - \frac{y_u + f(1-y_u)}{\beta} && \text{(by (4.9))} \\
&\geqslant 1 - \frac{\beta - c/2}{\beta} && \text{(by (4.8) and (4.10))} \\
&\geqslant c/5. && (\beta = 2 - \varepsilon < 2.5)
\end{aligned}$$

Choosing $c/5$ as the constant in the statement of the lemma, we obtain the result. □

Finally, for our analysis we will rely on the competitive ratio of the fractional solution maintained in Line 4 being $1/\beta$. This follows by Lemma 4.2.5 and the fact that for our choices of $\beta = 2 - \varepsilon$ and $f = f_{1+2\varepsilon}$ we have that $\beta \geqslant \beta^*(f)$. See Section 4.5 for a proof of this fact.

Fact 4.2.12. *For all sufficiently small $\varepsilon > 0$, we have that $2 - \varepsilon \geqslant \beta^*(f_{1+2\varepsilon})$.*

Structural Properties of G_τ and H_τ

In our analysis later, we focus on maximal primary paths (directed paths made of primary arcs) in H_τ, in the sense that the last vertex along the primary path has no outgoing primary arc in H_τ. The following definition captures termination certificates of such primary paths.

Definition 4.2.13 (Certified Primary Path)**.** *A tuple (P, T) is a certified primary path in H_τ if P is a directed path of primary arcs in H_τ and either*

(a) *the last vertex of P does not have an outgoing primary arc in G_τ and $T = \emptyset$, or*

(b) *the last vertex u of P has an outgoing primary arc (u, w) in G_τ and $T = (u', w)$ is a primary arc in H_τ such that u' precedes u in the arrival order.*

To elaborate, a certified primary path (P, T) is made of a (directed) path P of primary arcs in H_τ and T is a certificate of P's termination in H_τ that ensures the last vertex u in P has no outgoing primary arc in H_τ, either due to u not picking a primary arc with $T = \emptyset$, or due to the picked primary arc (u, w) being blocked by another primary arc $T = (u', w)$ which appears in H_τ.

As described, G_τ and H_τ differ in arcs (u, w) that are *blocked* by previous primary arcs to their target vertex w. We generally define sets of arcs which can block an edge, or a path, or a certified path from appearing in H_τ as in the following definition:

Definition 4.2.14 (Blocking sets)**.** *For an arc (u, w), define its* blocking set

$$B(u, w) := \{(u', w) \mid \{u', w\} \text{ is an edge and } u' \text{ arrived before } u\}$$

to be those arcs, the appearance of any of which as primary arc in G_τ blocks (u, v) from being in H_τ. In other words, an arc (u, v) is in H_τ as primary or secondary arc if and only if (u, v) is in G_τ and none of the arcs in its blocking set $B(u, v)$ is in G_τ as a primary arc. The blocking set of a path P is simply the union of its arcs' blocking sets,

$$B(P) := \bigcup_{(u,v)\in P} B(u, v)\,.$$

The blocking set *of a certified primary path (P, T) is the union of blocking sets of P and T,*

$$B(P, T) := B(P \cup T).$$

The probability of an edge, or path, or certified primary path appearing in H_τ is governed in part by the probability of arcs in their blocking sets appearing as primary arcs in G_τ. As an arc (v, u) is picked as primary arc by when v arrives with probability roughly z_u (more precisely, $z'_u \in [z_{vu}/(1 + C\varepsilon), z_{vu}]$, by Lemma 4.2.10), it will be convenient to denote by $z(v, u)$ and $z'(v, u)$ the values z_u and z'_u when v arrives, and by $z(S) = \sum_{s\in S} z(s)$ and $z'(S) = \sum_{s\in S} z(s)$ the sum of z- and z'-values of arcs in a set of arcs S.

Product distributions. Note that by definition the distribution over primary and secondary arc choices of vertices are product distributions (they are independent). As such, their joint distribution is defined by their marginals. Let p_w and s_w denote the distribution on primary and secondary arc choices of w, respectively. That is, for every $u \in N_w(w)$, $p_w(u)$ is the marginal probability that w selects (w, u) as its primary arc, and $s_w(u)$ is the marginal probability that w selects (w, u) as its secondary arc. Given our target bound (4.5), it would be useful to show that conditioning on F_u preserves the independence of these arc choices. Unfortunately, conditioning on F_u does not preserve this independence. We will therefore refine our conditioning later on the existence of primary paths in H_τ, which as we show below maintains independence of the arc choices.

Lemma 4.2.15. *For a certified primary path (P,T) let $\mathrm{EQ}_{(P,T)}$ be the event that the path P equals a maximal connected component in H_τ and the termination of P is certified by T. Then the conditional distributions of primary and secondary choices conditioned on $\mathrm{EQ}_{(P,T)}$ are product distributions; i.e., these conditional choices are independent. Moreover, if we let $\tilde{p}_w$ and $\tilde{s}_w$ denote the conditional distribution on primary and secondary choices of w, respectively, then*

$$\mathrm{TV}(p_w, \tilde{p}_w) \leqslant z(R(w)) \qquad \textit{and} \qquad \mathrm{TV}(s_w, \tilde{s}_w) \leqslant z(R(w)),$$

where $R(w) \subseteq \{w\} \times N_w(w)$ is the set of arcs leaving w whose existence as primary arcs in G_τ is ruled out by conditioning on $\mathrm{EQ}_{(P,T)}$, and the union of these $R(w)$, denoted by $R(P,T)$, satisfies

$$R(P,T) := \bigcup_w R(w) \subseteq B(P,T) \cup \{(w,r) \mid r \textit{ is root of } P\} \cup \bigcup_{w \in P \cup \{w : T = (w,w')\}} \{w\} \times N_w(w). \tag{4.11}$$

Proof. We first bound the total variation distance between the conditional and unconditional distributions. For primary choices, conditioning on $\mathrm{EQ}_{(P,T)}$ rules out the following sets of primary arc choices. For vertex $w \notin P$ arriving before the root r of P this conditioning rules out w picking any edge in $B(P,T)$ as primary arc. For vertices $w \notin P$ with w arriving after the root r of P this conditioning rules out picking arcs (w,r). Finally, this conditioning rules out some subset of arcs leaving vertices in $P \cup \{w : T = (w,w')\}$. Taking the union over these supersets of $R(w)$, we obtain (4.11). Now, the probability of each ruled out primary choice $(w,u) \in R(w)$ is zero under $\tilde{p}_w$ and $z'(w,u)$ under p_w, and all other primary choices have their probability increase, with a total increase of $\sum_{(w,u) \in R(w)} z'(w,u)$, from which we conclude that

$$\mathrm{TV}(p_w, \tilde{p}_w) = \frac{1}{2} \sum_{u \in N_w(w)} |p_w(u) - \tilde{p}_w(u)| = z'(R(w)) \leqslant z(R(w)).$$

The proof for secondary arcs is nearly identical, the only differences being that the sets of ruled out secondary arcs can be smaller (specifically, secondary arcs to w' such that $T = (u,w')$ are not ruled out by this conditioning), and the probability of any arc (w,u) being picked as secondary arc of w is at most $\sqrt{\varepsilon} \cdot z'(w,u) \leqslant z(w,u)$.

Finally, we note that primary and secondary choices for different vertices are independent. Therefore, conditioning on each vertex w not picking a primary arc in its ruled out set $R(w)$ still yields a product distribution, and similarly for the distributions over secondary choices. $\square$

It is easy to show that a particular certified primary path (P,T) with high value of $z(B(P,T))$ is unlikely to appear in H_τ, due to the high likelihood of arcs in its breaking set being picked as primary arcs. The following lemma asserts that the probability of a vertex u being the root of *any* primary certified path (P,T) with high $z(B(P,T))$ value is low.

Lemma 4.2.16. *For any $k \geqslant 0$ and any vertex u, we have the following*

$$\Pr[H_\tau \textit{ contains a certified primary path } (P,T) \textit{ rooted at } u \textit{ and } z(B(P,T)) \geqslant k] \leqslant e^{-k/2},$$

$$\Pr[H_\tau \textit{ contains a primary path } P \textit{ rooted at } u \textit{ with } z(B(P)) \geqslant k] \leqslant e^{-k/2}.$$

Proof. We first prove the bound for certified primary paths. For a certified primary path (P,T) where the last vertex of P is w, define P^* as follows:

$$P^* = \begin{cases} P & \text{if } T = \emptyset \\ P \cup \{(w,w'')\} & \text{if } T = (w',w''). \end{cases}$$

Observe that $z(B(P^*)) \geqslant k$ whenever $z(B(P,T)) \geqslant k$. This is trivial when $T = \emptyset$. To see this for the case $T = (w', w'')$, let w be the last vertex of P, and note that $B(w',w'') \subseteq B(w,w'')$, as w arrives after w'. Also note that for (P,T) to be in H_τ, we have that P^* must be in G_τ.

We say a directed primary path $P' = u \to u_1 \to \cdots \to u_{\ell-1} \to u_\ell$ is *k-minimal* if $z(B(P')) \geqslant k$ and $z(B(P' \setminus \{(u_{\ell-1},u_\ell)\})) < k$. For such a path P', define $B^*(P')$ as follows: Initially set $B^*(P') = B(P \setminus \{(u_{\ell-1},u_\ell)\})$. Then from $B(u_{\ell-1},u_\ell)$, the breaking set of the last arc of P', add arcs to $B^*(P')$ in reverse order of their sources' arrival until $z(B^*(P')) \geqslant k$.

Consider a certified primary path (P,T) with P rooted at u. If a k-minimal path rooted at u which is not a prefix of P^* is contained in G_τ, then (P,T) does not appear in G_τ, and therefore it does not appear in H_τ. On the other hand, if $z(B(P,T)) \geqslant k$ then for (P,T) to appear in H_τ, we must have that the (unique) k-minimal prefix P' of P^* must appear in G_τ, and that none of the edges of $B^*(P')$ appear in G_τ. Moreover, for any certified primary path with $z(B(P,T))$, conditioning on the existence of P' in G_τ does not affect random choices of vertices with outgoing arcs in $B^*(P')$, as these vertices are not in P'. Since by Lemma 4.2.10 each arc (w,w') appears in G_τ with probability $z'(v,u) \geqslant z(v,u)/(1+C\varepsilon) \geqslant z(v,u)/2$, we conclude that for any k-minimal primary path P' rooted at u, we have

$$\begin{aligned}
&\Pr[H_\tau \text{ contains any certified primary path } (P,T) \text{ with } z(B(P,T)) \geqslant k \mid P' \text{ is in } G_\tau] \\
&\leqslant \Pr[\text{No edge in } B^*(P') \text{ is in } G_\tau \mid P' \text{ is in } G_\tau] \\
&= \prod_{w \notin P'} (1 - \Pr[\text{Some primary edge in } B^*(P') \cap (\{w\} \times N_w(w)) \text{ is in } G_\tau]) \\
&\leqslant \prod_{w \notin P'} \exp\left(-\textstyle\sum_{(w,w') \in B(P,T) \times N_w(w)} z(w,w')/2\right) \\
&\leqslant \exp(-z(B^*(P'))/2) \leqslant e^{-k/2}.
\end{aligned}$$

Taking total probability $\mathcal{P}_u$, the set of all k-minimal primary paths P' rooted at u, we get that indeed, since u is the root of at most one k-minimal primary path in any realization of G_τ,

$$\begin{aligned}
&\Pr[H_\tau \text{ contains a certified primary path } (P,T) \text{ rooted at } u \text{ with } z(B(P,T)) \geqslant k] \\
&\leqslant \sum_{P' \in \mathcal{P}_u} \underbrace{\Pr[H_\tau \text{ contains a } (P,T) \text{ with } z(B(P,T)) \geqslant k \mid P' \text{ is in } G_\tau]}_{\leqslant e^{-k/2}} \cdot \Pr[P' \text{ is in } G_\tau] \leqslant e^{-k/2}.
\end{aligned}$$

The proof for primary path is essentially the same as the above, taking $P^* = P$. □

Analyzing Good Vertices

Consider the set of vertices that are unlikely to be roots of long directed paths of primary arcs in H_τ. In this section, we show that Algorithm 3 achieves almost lossless rounding for such vertices, and hence we call them *good* vertices. We start with a formal definition:

Definition 4.2.17 (Good vertices)**.** *We say that a vertex v is* good *if*

$$\Pr_\tau[H_\tau \textit{ has a primary path rooted at } v \textit{ of length at least } 2000 \cdot \ln(1/\varepsilon)] \leqslant \varepsilon^6.$$

Otherwise, we say v is bad.

As the main result of this section, for good vertices, we prove the following:

Theorem 4.2.18. *Let v be a good vertex. Then*

$$\Pr[v \textit{ is matched on arrival}] \geqslant (1-\varepsilon^2) \cdot \sum_{u \in N_v(v)} x_{uv}.$$

Notational conventions.: Throughout this section, we fix v and let z, z' be as in Algorithm 3. Moreover, for simplicity of notation, we suppose that the stream of vertices ends just before v's arrival and so quantities, such as G_τ and H_τ, refer to their values when v arrives. For a vertex u, we let F_u denote the event that u is free (i.e., unmatched) when v arrives. In other words, F_u is the event that u is free in the stream that ends just before v's arrival.

To prove the theorem, first note that it is immediate if $\sum_{u \in N_v(v)} z_u \leqslant 1$: in that case, we have $z' = z$ and so the probability to match v by a primary edge, by definition of z_u, is simply

$$\sum_{u \in N_v(v)} z_u \cdot \Pr[F_u] = \sum_{u \in N_v(v)} x_{uv}.$$

From now on we therefore assume $\sum_{u \in N_v(v)} z_u > 1$, which implies

(I) $\sum_{u \in N_v(v)} z'_u = 1$,

and moreover, by Lemma 4.2.10 and Corollary 4.2.11, for every $u \in N_v(v)$:

(II) $z_u \leqslant C\sqrt{\varepsilon}$,

(III) $z_u \leqslant (1 + C\varepsilon) \cdot z'_u$, and

(IV) $c \leqslant \Pr[F_u] \leqslant 1 - c$,

where c is the constant of Corollary 4.2.11 and C is the constant of Lemma 4.2.10.

We now state the key technical lemma in the proof of Theorem 4.2.18:

Lemma 4.2.19. *Consider a neighbor $u \in N_v(v)$ such that*

$$\Pr_\tau[H_\tau \text{ has a primary path rooted at } u \text{ of length} \geqslant 2000 \cdot \ln(1/\varepsilon) \mid F_u] \leqslant \varepsilon^2 \,. \tag{4.12}$$

Then,

$$\sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w \mid F_u] - \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w] \leqslant \varepsilon^{1/3} \,. \tag{4.13}$$

Note that the above lemma bounds the quantity $\sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w \mid F_u]$, which will allow us to show that (4.5) holds and thus the edge $\{u, v\}$ is picked in the matching with probability very close to x_{uv}. Before giving the proof of the lemma, we give the formal argument why the lemma implies the theorem.

Proof of Theorem 4.2.18. Define S to be the neighbors u in $N_v(v)$ satisfying

$$\Pr_\tau[H_\tau \text{ has a primary path rooted at } u \text{ of length} \geqslant 2000 \cdot \ln(1/\varepsilon) \mid F_u] > \varepsilon^2 \,.$$

In other words, S is the set of neighbors of v that violate (4.12). As v is good, we have

$$\begin{aligned}
\varepsilon^6 &\geqslant \Pr_\tau[H_\tau \text{ has a primary path rooted at } v \text{ of length at least } 2000 \cdot \ln(1/\varepsilon)] \\
&\geqslant \sum_{u \in N_v(v)} z'_u \cdot \Pr[F_u] \cdot \Pr_\tau[H_\tau \text{ has a primary path rooted at } u \text{ of length} \geqslant 2000 \cdot \ln(1/\varepsilon) - 1 \mid F_u] \\
&\geqslant \sum_{u \in N_v(v)} z'_u \cdot \Pr[F_u] \cdot \Pr_\tau[H_\tau \text{ has a primary path rooted at } u \text{ of length} \geqslant 2000 \cdot \ln(1/\varepsilon) \mid F_u] \\
&\geqslant \sum_{u \in S} z'_u \cdot \Pr[F_u] \cdot \varepsilon^2.
\end{aligned}$$

The second inequality holds because v selects the primary arc (u, v) with probability z'_u and, conditioned on F_u, u cannot already have an incoming primary arc, which implies that (u, v) is present in H_τ. The last inequality follows from the choice of S.

By Property (III), $z_u \leqslant (1 + C\varepsilon) \cdot z'_u$ and so by rewriting we get

$$\sum_{u \in S} x_{uv} = \sum_{u \in S} z_u \cdot \Pr[F_u] \leqslant (1 + C\varepsilon) \cdot \sum_{u \in S} z'_u \cdot \Pr[F_u] \leqslant (1 + C\varepsilon) \cdot \varepsilon^4 \leqslant \varepsilon^3.$$

In other words, the contribution of the neighbors of v in S to $\sum_{u \in N_v(v)} x_{uv}$ is insignificant compared to the contribution of all neighbors,

$$\sum_{u \in N_v(v)} x_{uv} = \sum_{u \in N_v(v)} z_u \cdot \Pr[F_u] \geqslant c, \tag{4.14}$$

where the inequality follows by the assumption $\sum_{u \in N_v(v)} z_u \geqslant 1$ and $\Pr[F_u] \geqslant c$ by Property (IV).

We proceed to analyze a neighbor $u \in N_v(v) \setminus S$. Recall that it is enough to verify (4.5) to conclude that edge $\{u, v\}$ is picked in the matching with probability x_{uv}. We have that

$$\begin{aligned}
&1 + \sqrt{\varepsilon} \sum_{w \in N_v(v)} z'_w \cdot (1 - \Pr[F_w \mid F_u]) \\
&\geqslant 1 + \sqrt{\varepsilon} \sum_{w \in N_v(v)} z'_w \cdot (1 - \Pr[F_w]) - \sqrt{\varepsilon} \cdot \varepsilon^{1/3} && \text{(by Lemma 4.2.19)} \\
&\geqslant 1 + \sqrt{\varepsilon} \sum_{w \in N_v(v)} z'_w \cdot c - \sqrt{\varepsilon} \cdot \varepsilon^{1/3} && (\Pr[F_w] \leqslant 1 - c \text{ by (IV)}) \\
&= 1 + \sqrt{\varepsilon} c - \sqrt{\varepsilon} \cdot \varepsilon^{1/3} && \left(\sum_{w \in N_v(v)} z'_w = 1 \text{ by (I)} \right) \\
&\geqslant 1 + C\varepsilon && \text{(for } \varepsilon \text{ small enough)} \\
&\geqslant z_u / z'_u . && \text{(by (III))}
\end{aligned}$$

Therefore, by definition of S and Lemma 4.2.19, we thus have that for every $u \in N_v(v) \setminus S$, the edge $\{u, v\}$ is taken in the matching with probability x_{uv}. Thus, the probability that v is matched on arrival is, as claimed, at least

$$\sum_{u \in N_v(v) \setminus S} x_{uv} = \sum_{u \in N_v(v)} x_{uv} - \sum_{u \in S} x_{uv} \geqslant \sum_{u \in N_v(v)} x_{uv} - \varepsilon^3 \geqslant (1 - \varepsilon^2) \sum_{u \in N_v(v)} x_{uv} \,,$$

where the last inequality holds because we have $\sum_{u \in N_v(v)} x_{uv} \geqslant c$, as calculated in (4.14). □

Proof of the Key Lemma

It remains to prove the key lemma, Lemma 4.2.19, which we do here.

Proof of Lemma 4.2.19. For a certified primary path (P, T) let $\mathrm{EQ}_{(P,T)}$ be the event as defined in Lemma 4.2.15, and let $\mathrm{IN}_{(P,T)}$ be the event that P is a maximal *primary* path in H_τ and the termination of P is certified by T. Further, let

$$\mathcal{C} = \{(P, T) : (P, T) \text{ is a certified primary path rooted at } u \text{ with } \Pr[\mathrm{IN}_{(P,T)}] > 0\}$$

be the set of certified primary paths rooted at u that have a nonzero probability of being maximal in H_τ. Then, by the law of total probability and since $\sum_{(P,T) \in \mathcal{C}} \Pr[\mathrm{IN}_{(P,T)} \mid F_u] = 1$ (since conditioning on F_u implies in particular that u has no incoming primary arc), we can rewrite the expression to bound, $\sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w \mid F_u] - \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w]$, as

$$\sum_{(P,T) \in \mathcal{C}} \Pr[\mathrm{IN}_{(P,T)} \mid F_u] \left(\sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w \mid F_u, \mathrm{IN}_{(P,T)}] - \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w] \right) . \tag{4.15}$$

We analyze this expression in two steps. First, in the next claim, we show that we can focus on the case when the certified path (P, T) is very structured and equals the component of u in H_τ. We then analyze the sum in that structured case.

Lemma 4.2.20. *Let $\mathcal{P} \subseteq \mathcal{C}$ contain those certified primary paths (P,T) of $\mathcal{C}$ that satisfy: P has length less than $2000 \cdot \ln(1/\varepsilon)$ and $z(B(P,T)) \leqslant 2\ln(1/\varepsilon)$. Then,* (4.15) *is at most*

$$\sum_{(P,T)\in\mathcal{P}} \Pr[\mathrm{EQ}_{(P,T)} \mid F_u] \left(\sum_{w\in N_v(v)} z'_w \cdot \Pr[F_w \mid \mathrm{EQ}_{(P,T)}] - \sum_{w\in N_v(v)} z'_w \cdot \Pr[F_w] \right) + \varepsilon^{1/3}/2.$$

Proof. Define the following subsets of certified primary paths rooted at u:

$$\mathcal{C}_1 := \{(P,T) \in \mathcal{C} \mid P \text{ is of length at least } 2000 \cdot \ln(1/\varepsilon)\}$$
$$\mathcal{C}_2 := \{(P,T) \in \mathcal{C} \setminus \mathcal{C}_1 \mid z(B(P,T)) > 2\ln(1/\varepsilon)\}$$

Note that $\mathcal{P} = \mathcal{C} \setminus (\mathcal{C}_1 \cup \mathcal{C}_2)$. Since u satisfies (4.12), we have that

$$\sum_{(P,T)\in\mathcal{C}_1} \Pr[\mathrm{IN}_{(P,T)} \mid F_u] \leqslant \varepsilon^2 \leqslant \varepsilon^{1/3}/6.$$

On the other hand, by Lemma 4.2.16 and $\Pr[F_u] \geqslant c$ (by Property (IV)), we have that

$$\sum_{(P,T)\in\mathcal{C}_2} \Pr[\mathrm{IN}_{(P,T)} \mid F_u] \leqslant c^{-1} \cdot \sum_{(P,T)\in\mathcal{C}_2} \Pr[\mathrm{IN}_{(P,T)}] \leqslant c^{-1} \cdot \varepsilon \leqslant \varepsilon^{1/3}/6.$$

In other words, almost all probability mass lies in those outcomes where one of the certified paths $(P,T) \in \mathcal{P}$ is in H_τ. It remains to prove that, in those cases, we almost always have that the component of u in H_τ equals the path P (whose termination is certified by T). Specifically, let $\overline{\mathrm{EQ}_{(P,T)}}$ denote the complement of $\mathrm{EQ}_{(P,T)}$. We now show that

$$\Pr\left[\overline{\mathrm{EQ}_{(P,T)}} \mid \mathrm{IN}_{(P,T)}\right] \leqslant \varepsilon^{1/3}/7\,. \tag{4.16}$$

To see this, note that by the definition of the event $\mathrm{IN}_{(P,T)}$, if we restrict ourselves to primary edges then the component of u in H_τ equals P. We thus have that for the event $\overline{\mathrm{EQ}_{(P,T)}}$ to be true at least one of the vertices in P must have an incoming or outgoing *secondary* edge. Hence the expression $\Pr\left[\overline{\mathrm{EQ}_{(P,T)}} \mid \mathrm{IN}_{(P,T)}\right]$ can be upper bounded by

$$\Pr[\text{a vertex in } P \text{ has an incoming or outgoing secondary arc in } G_\tau \mid \mathrm{IN}_{(P,T)}] \tag{4.17}$$

Note that event $\mathrm{IN}_{(P,T)}$ is determined solely by choices of primary arcs. By independence of these choices and choices of secondary arcs, conditioning on $\mathrm{IN}_{(P,T)}$ does not affect the distribution of secondary arcs. So the probability that any of the nodes in P selects a secondary edge is at most $\sqrt{\varepsilon}$. Thus, by union bound, the probability that any of the $|P| \leqslant 2000 \cdot \ln(1/\varepsilon)$ vertices in P pick a secondary arc is at most $\sqrt{\varepsilon} \cdot 2000 \cdot \ln(1/\varepsilon)$. We now turn our attention to incoming secondary arcs. First, considering the secondary arcs that go into u, we have

$$c \leqslant \Pr[F_u] \leqslant \prod_{(w,u)\in B(v,u)} (1 - z(w,u)/2) \leqslant \exp(-z(B(v,u))/2),$$

because any arc $(w,u) \in B(v,u)$ appears as a primary arc in G_τ independently with probability at least $z(w,u)/2$ and the appearance of such an arc implies that u has an incoming primary arc in H_τ and is therefore matched; i.e., the event F_u is false in this case. We thus have $z(B(v,u)) \leqslant 2\ln(1/c)$. Further, since $(P,T) \notin \mathcal{C}_2$, we have $z(B(P)) \leqslant z(B(P,T)) \leqslant 2\ln(1/\varepsilon)$. Again using that the conditioning on $\mathrm{IN}_{(P,T)}$ does not affect the distribution of secondary edges, we have that the probability of an incoming secondary arc to any vertex in P is at most $\sqrt{\varepsilon} \cdot (2\ln(1/c) + 2\ln(1/\varepsilon))$. Thus, by union bound, the probability that any vertex in P has an incoming or outgoing secondary arc conditioned on $\mathrm{IN}_{(P,T)}$ is at most

$$\sqrt{\varepsilon} \cdot 2000 \cdot \ln(1/\varepsilon) + \sqrt{\varepsilon} \cdot (2\ln(1/c) + 2\ln(1/\varepsilon)) \leqslant \varepsilon^{1/3}/7,$$

for sufficiently small ε, which implies (4.16) via (4.17).

We now show how the above concludes the proof of the claim. We have shown that each one of the two sets $\mathcal{C}_1, \mathcal{C}_2$ contributes at most $\varepsilon^{1/3}/6$ to (4.15) (where we use that $\sum_{w\in N_v(v)} z'_w = 1$). Hence, (4.15) is at most

$$\sum_{(P,T)\in\mathcal{P}} \Pr[\mathrm{IN}_{(P,T)} \mid F_u] \left(\sum_{w\in N_v(v)} z'_w \cdot \Pr[F_w \mid \mathrm{IN}_{(P,T)}, F_u] - \sum_{w\in N_v(v)} z'_w \cdot \Pr[F_w] \right) + 2\varepsilon^{1/3}/6.$$

This intuitively concludes the proof of the claim as (4.16) says that $\Pr[\mathrm{EQ}_{(P,T)} | \mathrm{IN}_{(P,T)}]$ is almost 1. The formal calculations are as follows. Since the event $\mathrm{EQ}_{(P,T)}$ implies the event $\mathrm{IN}_{(P,T)}$, we have that

$$\Pr[\mathrm{EQ}_{(P,T)}] = \Pr[\mathrm{EQ}_{(P,T)} \wedge \mathrm{IN}_{(P,T)}] = \Pr[\mathrm{IN}_{(P,T)}] - \Pr[\overline{\mathrm{EQ}_{(P,T)}} \wedge \mathrm{IN}_{(P,T)}],$$

which by (4.16) implies

$$\Pr[\mathrm{EQ}_{(P,T)}] = \Pr[\mathrm{IN}_{(P,T)}] \left(1 - \Pr\left[\overline{\mathrm{EQ}_{(P,T)}} \mid \mathrm{IN}_{(P,T)}\right]\right) \geqslant \Pr[\mathrm{IN}_{(P,T)}] \left(1 - \varepsilon^{1/3}/7\right). \tag{4.18}$$

We are now ready to rewrite and upper bound Equation (4.15), namely

$$\Pr[\mathrm{IN}_{(P,T)} \mid F_u] \left(\sum_{w\in N_v(v)} z'_w \cdot \Pr[F_w \mid \mathrm{IN}_{(P,T)}, F_u] - \sum_{w\in N_v(v)} z'_w \cdot \Pr[F_w] \right).$$

Specifically, by law of total probability, this expression can be rewritten as the sum of the expressions (4.19) and (4.20) below:

$$\begin{aligned} \Pr[\mathrm{EQ}_{(P,T)} \wedge \mathrm{IN}_{(P,T)} \mid F_u] &\left(\sum_{w\in N_v(v)} z'_w \cdot \Pr[F_w \mid \mathrm{EQ}_{(P,T)}, \mathrm{IN}_{(P,T)}, F_u] - \sum_{w\in N_v(v)} z'_w \cdot \Pr[F_w] \right) \\ = \Pr[\mathrm{EQ}_{(P,T)} \mid F_u] &\left(\sum_{w\in N_v(v)} z'_w \cdot \Pr[F_w \mid \mathrm{EQ}_{(P,T)}] - \sum_{w\in N_v(v)} z'_w \cdot \Pr[F_w] \right) \end{aligned} \tag{4.19}$$

and

$$\Pr[\overline{\mathrm{EQ}_{(P,T)}} \wedge \mathrm{IN}_{(P,T)} \mid F_u] \left(\sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w \mid \overline{\mathrm{EQ}_{(P,T)}}, \mathrm{IN}_{(P,T)}, F_u] - \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w] \right), \tag{4.20}$$

where (4.20) can be upper bounded as follows:

$$\begin{aligned}
(4.20) &\leqslant \Pr[\overline{\mathrm{EQ}_{(P,T)}} \wedge \mathrm{IN}_{(P,T)} \mid F_u] && \text{(by } \textstyle\sum_{w \in N_v(v)} z'_w \leqslant 1) \\
&\leqslant c^{-1} \cdot \Pr[\overline{\mathrm{EQ}_{(P,T)}} \wedge \mathrm{IN}_{(P,T)}] && \text{(by } c \leqslant \Pr[F_u]) \\
&= c^{-1} \cdot \Pr[\mathrm{IN}_{(P,T)}] \cdot \Pr\left[\overline{\mathrm{EQ}_{(P,T)}} \mid \mathrm{IN}_{(P,T)}\right] \\
&\leqslant c^{-1} \cdot \frac{\Pr[\mathrm{EQ}_{(P,T)}]}{1 - \varepsilon^{1/3}/7} \cdot (\varepsilon^{1/3}/7) && \text{(by (4.16) and (4.18))} \\
&\leqslant \Pr[\mathrm{EQ}_{(P,T)}] \cdot \varepsilon^{1/3}/6. && \text{(for } \varepsilon \text{ small enough)}
\end{aligned}$$

As at most one of the events $\{\mathrm{EQ}_{(P,T)}\}_{(P,T) \in \mathcal{P}}$ is true in any realization of G_τ, we have that $\sum_{(P,T) \in \mathcal{P}} \Pr[\overline{\mathrm{EQ}_{(P,T)}} \wedge \mathrm{IN}_{(P,T)} \mid F_u] \leqslant \sum_{(P,T) \in \mathcal{P}} \left(\Pr[\mathrm{EQ}_{(P,T)}] \cdot \varepsilon^{1/3}/6\right) \leqslant \varepsilon^{1/3}/6$. Thus, again using that $\sum_{w \in N_v(v)} z_w \leqslant 1$, we have that (4.15) is at most

$$\begin{aligned}
&\sum_{(P,T) \in \mathcal{P}} \Pr[\mathrm{IN}_{(P,T)} \mid F_u] \left(\sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w \mid \mathrm{IN}_{(P,T)}, F_u] - \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w] \right) + 2\varepsilon^{1/3}/6 \\
\leqslant &\sum_{(P,T) \in \mathcal{P}} \Pr[\mathrm{EQ}_{(P,T)} \mid F_u] \left(\sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w \mid \mathrm{EQ}_{(P,T)}] - \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w] \right) + 3\varepsilon^{1/3}/6,
\end{aligned}$$

as claimed. □

The previous claim bounded the contribution of certified primary paths in $\mathcal{C} \setminus \mathcal{P}$ to (4.15). The following claim bounds the contribution of paths in $\mathcal{P}$.

Lemma 4.2.21. *Let $\mathcal{P} \subseteq \mathcal{C}$ contain those certified primary paths (P, T) of $\mathcal{C}$ that satisfy: P has length less than $2000 \cdot \ln(1/\varepsilon)$ and $z(B(P,T)) \leqslant 2\ln(1/\varepsilon)$. Then, we have*

$$\sum_{(P,T) \in \mathcal{P}} \Pr[\mathrm{EQ}_{(P,T)}] \left(\sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w \mid \mathrm{EQ}_{(P,T)}] - \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w] \right) \leqslant \varepsilon^{1/3}/2.$$

Proof. We prove the claim in two steps: first we construct a chain of distributions that interpolates between the unconditional distribution of H_τ and its conditional distribution, and then bound the expected number of vertices that change their matched status along that chain. For the remainder of the proof we fix the certified primary path (P, T).

Constructing a chain of distributions.: Let $H_\tau^{(0)}$ denote the unconditional distribution of H_τ when v arrives, and let $H_\tau^{(n)}$ denote the distribution of H_τ conditioned on $\mathrm{EQ}_{(P,T)}$ when v arrives. Here $n = |V|$ is the number of vertices in the input graph. For every $w \in V$ let $F_w^{(0)}$ denote the indicator of w being free when v arrives (unconditionally) and let $F_w^{(n)}$ denote the indicator variables of w being free when v arrives conditioned on $\mathrm{EQ}_{(P,T)}$. Note that $F^{(0)}$ is determined by $H_\tau^{(0)}$ and $F^{(n)}$ is determined by $H_\tau^{(n)}$. For $t = 0, \ldots, n$, we define distributions $H_\tau^{(t)}$ that interpolate between $H_\tau^{(0)}$ and $H_\tau^{(n+1)}$ as follows.

As in Lemma 4.2.15, for every $w \in V$ we denote the unconditional distribution of its primary choice by p_w, and the unconditional distribution of its secondary choice by s_w. Similarly, we denote the conditional distribution given $\mathrm{EQ}_{(P,T)}$ of the primary choice by $\widetilde{p}_w$ and the conditional distribution of the secondary choice by $\widetilde{s}_w$. For every $t = 0, \ldots, n$ the primary choice of vertices $w_j, j = 1, \ldots, t$ are sampled independently from $\widetilde{p}_{w_j}$, and the primary choices of vertices $w_j, j = t+1, \ldots, n$ are sampled independently from the unconditional distribution p_{w_t}. Similarly, secondary choices of vertices $w_j, j = 1, \ldots, t$ are sampled independently from $\widetilde{s}_{w_j}$ and secondary choices of vertices $w_j, j = t+1, \ldots, n$ are sampled independently from s_{w_j}. Note that $H_\tau^{(0)}$ is sampled from the unconditional distribution of H_τ, and $H_\tau^{(n)}$ is sampled from the conditional distribution (conditioned on $\mathrm{EQ}_{(P,T)}$), as required, due to the independence of the conditional probabilities $\tilde{p}_{w_j}$ and $\tilde{s}_{w_j}$, by Lemma 4.2.15. For $t = 0, \ldots, n$ let M_t denote the matching constructed by our algorithm on $H_\tau^{(t)}$, and let $F_w^{(t)}$ be the indicator variable for w being free when v arrives in the DAG sampled from $H_\tau^{(t)}$.

Coupling the distributions of $H_\tau^{(t)}$.: We now exhibit a coupling between the $H_\tau^{(t)}, t = 0, \ldots, n$. Specifically, we will show that for every such t the following holds.

$$\mathbb{E} * \sum_{q \in V} |F_q^{(t+1)} - F_q^{(t)}| \leqslant 4z(R(w_{t+1})), \tag{4.21}$$

where $R(w_{t+1})$ is as defined in Lemma 4.2.15 with regard to the certified primary path $R(P, T)$. Recall that $z(R(w_{t+1}))$ is the total probability assigned to arcs leaving w_{t+1} which are ruled out from being primary arcs in G_τ by conditioning on $\mathrm{EQ}_{(P,T)}$.

We construct the coupling by induction. The base case corresponds to $t = 0$ and is trivial. We now give the inductive step ($t \to t + 1$). We write $w := w_{t+1}$ to simplify notation. Let $Z^p \in N_w(w)$ denote the primary choice of w in $H_\tau^{(t)}$, and let $Z^s \in N_w(w)$ denote the secondary choice of w in $N_w(w)$ (they are sampled according to the unconditional distributions p_w and s_w respectively). Let $\widetilde{Z}^p \in N_w(w)$ and $\widetilde{Z}^s \in N_w(w)$ be sampled from the conditional distributions $\widetilde{p}_w$ and $\widetilde{s}_w$ respectively, such that that the joint distributions $(Z^p, \widetilde{Z}^p)$ and $(Z^s, \widetilde{Z}^s)$ satisfy

$$\Pr[Z^p \neq \widetilde{Z}^p] = \mathrm{TV}(p_w, \widetilde{p}_w) \text{ and } \Pr[Z^s \neq \widetilde{Z}^s] = \mathrm{TV}(s_w, \widetilde{s}_w). \tag{4.22}$$

First, we note that if $Z^p = \widetilde{Z}^p$ and $Z^s = \widetilde{Z}^s$, then $w = w_{t+1}$ is matched to the same neighbor under $H_\tau^{(t)}$ and $H_\tau^{(t+1)}$, and so $M_t = M_{t+1}$, due to the greedy nature of the matching constructed.

Otherwise, by Lemma 4.2.9, at most two vertices have different matched status in M_t and M_{t+1} in the latter case (in the former case every vertex has the same matched status). To summarize, we have, for $R(w)$ determined by (P,T) as in Lemma 4.2.15, that

$$\begin{aligned}\mathbb{E}\left[\sum_{q\in V}|F_q^{(t+1)}-F_q^{(t)}|\right] &\leqslant 2\cdot\Pr[Z^p\neq\widetilde{Z}^p \text{ or } Z^s\neq\widetilde{Z}^s]\\ &\leqslant 2(\mathrm{TV}(p_w,\widetilde{p}_w)+\mathrm{TV}(s_w,\widetilde{s}_w)) \quad \text{(by (4.22) and union bound)}\\ &\leqslant 4z(R(w)). \quad \text{(by Lemma 4.2.15)}\end{aligned} \tag{4.23}$$

This concludes the proof of the inductive step, and establishes (4.21). In particular, we get

$$\begin{aligned}\mathbb{E}\left[\sum_{q\in V}|F_q^{(n)}-F_q^{(0)}|\right] &\leqslant \sum_{t=0}^{n-1}\mathbb{E}\left[\sum_{q\in V}|F_q^{(t+1)}-F_q^{(t)}|\right]\\ &\leqslant \sum_{t=0}^{n-1}4z(R(w_{t+1})) \quad \text{(by (4.23))}\\ &= 4z(R(P,T)),\end{aligned} \tag{4.24}$$

by the definition of $R(P,T)=\bigcup_w R(w)$ in Lemma 4.2.15.

We now finish the claim. First note that for any (P,T) such that P has length at most $2000\cdot\ln(1/\varepsilon)$ and $z(B(P,T))\leqslant 2\ln(1/\varepsilon)$ one has $\sum_w z(R(w))=z(R(P,T))=O(\ln(1/\varepsilon))$. Indeed, by Lemma 4.2.15 and linearity of z, recalling that u is the root of P and that no vertex appears after v (and thus $B(v,u)=\{(w,u)\mid w \text{ arrives between } u \text{ and } v\}$), we have

$$z(R(P,T))\leqslant z(B(P,T))+z(B(v,u))+\sum_{w\in P\cup\{w:\,T=(w,w')\}} z\left(\{w\}\times N_w(w)\right). \tag{4.25}$$

We now bound the contribution to the above upper bound on $\sum_w z(R(w))=z(R(P,T))$ in (4.25). First, we have that $z(B(P,T))\leqslant 2\ln(1/\varepsilon)$ by assumption of the lemma. To bound the contribution of $z(B(v,u))$, we note that by Property IV, we have

$$c\leqslant\Pr[F_u]=\prod_{e\in B(v,u)}(1-z_e)\leqslant\exp\left(-\sum_{(w,u)\in B(v,u)}z(w,u)/2\right)\leqslant\exp(-z(B(v,u))/2),$$

because any arc $e=(w,u)$ appears as a primary arc in G_τ with probability $z'(w,u)\geqslant z(w,u)/2$, independently of other such arcs, and the appearance of any such an edge implies that u has an incoming primary edge in H_τ when v arrives and is therefore matched; i.e., the event F_u is false in this case. We thus have $z(B(v,u))\leqslant 2\ln(1/c)$. Finally, it remains to note that for every one of the at most $2000\cdot\ln(1/\varepsilon)+1$ vertices $w\in P\cup\{w: T=(w,w')\}$ the contribution of $z(\{w\}\times N_w(w))$ to the right hand side of (4.25) is at most $1+C\varepsilon\leqslant 2$, by Lemma 4.2.10, **(2)**. Putting these bounds together, we get that for sufficiently small ε,

$$z(R(P,T))\leqslant 2\ln(1/\varepsilon)+2\ln(1/c)+2\cdot 2000\cdot\ln(1/\varepsilon)+2=O(\ln(1/\varepsilon)). \tag{4.26}$$

The term we wish to upper bound is at most

$$
\begin{aligned}
&\sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w \mid \mathrm{EQ}_{(P,T)}] - \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w] \\
\leqslant &\left(\max_{w \in N_v(v)} z'_w\right) \cdot \sum_{w \in N_v(v)} \left|\Pr[F_w \mid \mathrm{EQ}_{(P,T)}] - \Pr[F_w]\right| \\
\leqslant &C\sqrt{\varepsilon} \cdot \sum_{w \in N_v(v)} \left|\Pr[F_w \mid \mathrm{EQ}_{(P,T)}] - \Pr[F_w]\right| && \text{(by Lemma 4.2.10, } \mathbf{(3)}\text{)} \\
= &C\sqrt{\varepsilon} \cdot \mathbb{E}\left[\sum_{w \in N_v(v)} |F_w^{(n)} - F_w^{(0)}|\right] && \text{(by definition of } F^{(0)} \text{ and } F^{(n)}\text{)}
\end{aligned}
$$

then, using (4.24) and (4.26), we find that the term we wish to upper bound is at most

$$
\begin{aligned}
&C\sqrt{\varepsilon} \cdot \mathbb{E}\left[\sum_{w \in V} |F_w^{(n)} - F_w^{(0)}|\right] \\
\leqslant &C\sqrt{\varepsilon} \cdot z(R(P,T)) && \text{(by (4.24))} \\
= &O(\sqrt{\varepsilon} \cdot \log(1/\varepsilon)) && \text{(by (4.26))} \\
\leqslant &\varepsilon^{1/3}/2,
\end{aligned}
$$

completing the proof. □

Finally, we obtain Lemma 4.2.19 by combining Lemma 4.2.20 and Lemma 4.2.21, to find that, as claimed

$$
\begin{aligned}
(4.15) \leqslant &\sum_{(P,T) \in \mathcal{P}} \Pr[\mathrm{EQ}_{(P,T)} \mid F_u] \left(\sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w \mid \mathrm{EQ}_{(P,T)}] - \sum_{w \in N_v(v)} z'_w \cdot \Pr[F_w]\right) + \varepsilon^{1/3}/2 \\
\leqslant &\varepsilon^{1/3}/2 + \varepsilon^{1/3}/2 = \varepsilon^{1/3}.
\end{aligned}
$$
□

Bounding the Impact of Bad Vertices

In this section, we show that we can completely ignore the bad vertices without losing too much. From the definition of good vertices, for a bad vertex v, we have that

$$\Pr_{\tau}[H_\tau \text{ has a primary path rooted at } v \text{ of length at least } 2000 \cdot \ln(1/\varepsilon)] \geqslant \varepsilon^6.$$

As the main result of this section, we prove the following theorem:

Theorem 4.2.22. *The number of bad vertices is at most* $\varepsilon^3 \cdot \sum_{e \in E} x_e$.

To prove this, we first describe a charging mechanism in which, for each bad vertex, a charge of one is distributed among a subset of other vertices. Then, using the following supplementary lemma, we show that the total distributed charge over all vertices in the graph is at most $\varepsilon^3 \cdot \sum_{(u,v)\in E} x_{uv}$.

Lemma 4.2.23. *We call a primary path P a* primary predecessor path *(PPP) of v if it ends at v. That is, $P = v_\ell \to v_{\ell-1} \to \cdots \to v_1 = v$. We have*

$$\Pr_\tau[v \textit{ has any PPP } P \textit{ with } z(B(P)) \leqslant 20 \cdot \ln(1/\varepsilon) \textit{ and } |P| \geqslant 1000 \cdot \ln(1/\varepsilon)] \leqslant \varepsilon^{10}.$$

Proof. We use the principle of deferred decisions and traverse the path backwards. Let b be the current vertex, which is initially set to v. Consider all incoming arcs to b, say $(a_1, b), \ldots, (a_k, b)$ where we index a's by time of arrival; i.e., a_i arrives before a_j if $i < j$ (and b arrived before any a_i).

First consider the random choice of a_1 and see if it selected the arc (a_1, b).

- If it does, then the path including b in H_τ will use the arc (a_1, b).
- Otherwise, if a_1 does not select the arc (a_1, b), then go on to consider a_2 and so on.

If no $a_1, \ldots, a_k$ selects b, then the process stops; i.e., the primary path starts at this vertex since b has no incoming primary arc. Otherwise let i be the first index so that (a_i, b) was selected. Then (a_i, b) is in the primary path ending at v in H_τ. Now, observe that *no* $a_1, \ldots, a_{i-1}$ may be in the path in this case, because these vertices arrived before a_i and after b. Moreover, we have not revealed any randomness regarding $a_{i+1}, \ldots, a_k$ that may appear later in the path. We can therefore repeat the above process with b now set to a_i and "fresh" randomness for all vertices we consider, as the random choices of arcs of all vertices are independent. We now show that this process, with good probability, does not result in a long predecessor path P of low $z(B(P))$ value.

Recall from Lemma 4.2.10, **(5)**, that $z(u, v) \leqslant 3/5$ for all $(u, v) \in V \times V$. Suppose that $\sum_{i=1}^k z(a_i, b) \geqslant 4/5$. Let j be the first index such that $\sum_{i=1}^j z(a_i, b) \geqslant 1/5$. Thus $\sum_{i=1}^j z(a_i, b) \leqslant 4/5$, and hence the probability that none of the first j vertices select b is at least $\prod_{i=1}^j (1 - z(a_i, b)) \geqslant 1 - \sum_{i=1}^j z(a_i, b) \geqslant 1/5$. Consequently, with probability at least $1/5$, vertex b either has no predecessor or the increase to $z(B(P))$ is at least $1/5$.

In the other case, we have $\sum_{i=1}^k z(a_i, b) \leqslant 4/5$. Then the probability that b has *no* predecessor is $\prod_{i=1}^k (1 - z(a_i, b)) \geqslant 1 - \sum_{i=1}^k z(a_i, b) \geqslant 1/5$.

Therefore, at any step in the above random process, with probability at least $1/5$, we either stop or increase $z(B(P))$ by $1/5$. Let Z_i be an indicator variable for the random process either stopping or increasing $z(B(P))$ by at least $1/5$ at step i, and notice that according to the above random process, each Z_i is lower bounded by an independent Bernoulli variable with probability $1/5$. Thus if we define $Z = \sum_{i\in[1000\cdot\ln(1/\varepsilon)]} Z_i$, we have $\mathbb{E}[Z] \geqslant 200 \cdot \ln(1/\varepsilon)$, and thus by standard coupling arguments and Chernoff bounds, we have that

$$\Pr[Z \leqslant 100 \cdot \ln(1/\varepsilon)] \leqslant \Pr\left[Z \leqslant (1 - 1/2) \cdot \mathbb{E}[Z]\right] \leqslant e^{-(1/2)\cdot(1/2)^2\cdot 200\cdot\ln(1/\varepsilon)} \leqslant \varepsilon^{10}.$$

But if the path does not terminate within $1000 \cdot \ln(1/\varepsilon)$ steps and $Z \geqslant 100 \cdot \ln(1/\varepsilon)$, then $z(B(P)) \geqslant 20 \cdot \ln(1/\varepsilon)$. □

We now prove Theorem 4.2.22.

Proof of Theorem 4.2.22. By Lemma 4.2.16, the probability that H_τ has a primary path P with $z(B(P)) \geqslant 20 \cdot \ln(1/\varepsilon)$ starting at v is at most ε^{10}. Thus, for a bad vertex u, the probability that H_τ has some primary path P rooted at u with $|P| \geqslant 2000 \cdot \ln(1/\varepsilon)$ and $z(B(P)) \leqslant 20 \cdot \ln(1/\varepsilon)$ is at least $\varepsilon^6 - \varepsilon^{10} \geqslant \varepsilon^6/2$.

Let $k = 20 \cdot \ln(1/\varepsilon)$ and $\ell = 2000 \cdot \ln(1/\varepsilon)$. Let $\mathcal{P}_u$ be the set of all primary paths P rooted at u such that $z(B(P)) \leqslant k$ and $|P| = \ell$ starting at u. Since all such primary paths with length more than ℓ are extensions of those with length exactly ℓ, we have $\sum_{P \in \mathcal{P}_u} \Pr[P \text{ is in } H_\tau] \geqslant \varepsilon^6/2$. For each such path $P \in \mathcal{P}_u$, consider the two vertices w_ℓ^P and $w_{\ell-1}^P$ at distances ℓ and $\ell - 1$ respectively from u. For each such vertex w_j^P ($j \in \{\ell-1, \ell\}$), charge $(2/\varepsilon^6) \cdot \Pr[P \text{ is in } H_\tau] \cdot y_{w_j^P}$. Then the sum of these charges is

$$\sum_{P \in \mathcal{P}_u} (2/\varepsilon^6) \cdot \Pr[P \text{ is in } H_\tau] \cdot \underbrace{(y_{w_\ell^P} + y_{w_{\ell-1}^P})}_{\geqslant 1} \geqslant (2/\varepsilon^6) \cdot \sum_{P \in \mathcal{P}_u} \Pr[P \text{ is in } H_\tau] \geqslant 1.$$

Notice that the fact $(y_{w_\ell^P} + y_{w_{\ell-1}^P}) \geqslant 1$ follows because y_w's form a feasible dual solution (to the vertex cover problem).

On the other hand, consider how many times each vertex is charged. For this, for every vertex w, let $\mathcal{Q}_w$ be the set of primary predecessor paths Q of u such that $|Q| = \ell - 1$ and $z(B(P)) \leqslant k$. As $|Q| = \ell - 1 \geqslant 1000 \cdot \ln(1/\varepsilon)$ for all $Q \in \mathcal{Q}_w$, by Lemma 4.2.23, $\sum_{Q \in \mathcal{Q}_w} \Pr[Q \text{ is in } H_\tau] \leqslant \varepsilon^{10}$. For a primary predecessor path $Q \in \mathcal{Q}_w$ (or one of its extensions), the vertex w can be charged at most twice according to the above charging mechanism. Since any predecessor path of w with length more than $\ell - 1$ must be an extension of one with length exactly $\ell - 1$, we have that the amount w is charged is at most

$$\sum_{Q \in \mathcal{Q}_w} 2 \cdot 2 \cdot \Pr[Q \text{ is in } H_\tau] \cdot y_w/\varepsilon^6 \leqslant 4 \cdot (\varepsilon^{10}/\varepsilon^6) \cdot y_w \leqslant 4 \cdot \varepsilon^4 \cdot y_w.$$

Summing over all $w \in V$ and using Lemma 4.2.3, the total charge is at most

$$\sum_{w \in v} 4 \cdot \varepsilon^4 \cdot y_w \leqslant 4 \cdot \varepsilon^4 \cdot \beta \cdot \sum_{e \in E} x_e \leqslant \varepsilon^3 \sum_{e \in E} x_e.$$ □

Calculating the Competitive Ratio of Algorithm 3

We now show that the competitive ratio of Algorithm 3 is indeed $(1/2 + \alpha)$ competitive for some sufficiently small absolute constant $\alpha > 0$, thus proving Theorem 4.1.1. This essentially combines the facts that for good vertices, the matching probability is very close to the fractional values of incident edges, and that the number of bad vertices is very small compared to the total value of the fractional algorithm (over the entire graph).

Proof of Theorem 4.1.1. Let OPT denote the size of the maximum cardinality matching in the input graph G. Then, by Lemma 4.2.5 and our choice of $f = f_{1+2\varepsilon}$ and $\beta = 2 - \varepsilon \geqslant \beta^*(f_{1+2\varepsilon})$, we have that $\sum_e x_e \geqslant (1/\beta) \cdot \mathrm{OPT} \geqslant (1/2 + \varepsilon/4) \cdot \mathrm{OPT}$, where the x_e's are the fractional values we compute in Algorithm 3.

Now let M be the matching output by Algorithm 3. We have

$$\begin{aligned}
\mathbb{E}[|M|] &= \sum_{e \in E} \Pr[e \text{ is matched}] \\
&\geqslant \sum_{\text{good } v \in V} (1 - \varepsilon^2) \cdot \sum_{u \in N_v(v)} x_{uv} && \text{(By Theorem 4.2.18)} \\
&\geqslant (1 - \varepsilon^2) \cdot \left(\sum_{e \in E} x_e - \sum_{\text{bad } v \in V} \sum_{u \in N_v(v)} x_{uv} \right) \\
&\geqslant (1 - \varepsilon^2) \cdot \left(\sum_{e \in E} x_e - \sum_{\text{bad } v \in V} 1 \right) && \left(\text{By } \sum_{u \in N_v(v)} x_{uv} \leqslant 1 \right) \\
&\geqslant (1 - \varepsilon^2) \cdot \left(\sum_{e \in E} x_e - \varepsilon^3 \sum_{e \in E} x_e \right) && \text{(By Theorem 4.2.22)} \\
&\geqslant (1 - 2\varepsilon^2) \cdot \sum_{e \in E} x_e \\
&\geqslant (1 - 2\varepsilon^2) \cdot (1/2 + \varepsilon/4) \cdot \mathrm{OPT} \\
&\geqslant (1/2 + \varepsilon/5) \cdot \mathrm{OPT},
\end{aligned}$$

where the last line holds for a sufficiently small constant $\varepsilon > 0$. □

4.3 Deferred Proofs of Section 4.2.3

Here we prove that a change of the realized arc choices of any vertex does not change the matched status of more than two vertices (at any point in time). This is Lemma 4.2.9, restated below.

Lemma 4.2.9. *Let G_τ and $G_{\tau'}$ be two realizations of the random digraph where all the vertices in the two graphs make the same choices except for* one *vertex v. Then the number of vertices that have different* matched status *(free/matched) in the matchings computed in H_τ and $H_{\tau'}$ at any point of time is at most two.*

Proof. We consider the evolution, following each vertex arrival, of the matchings M_τ and $M_{\tau'}$ computed in H_τ and $H_{\tau'}$, respectively, as well as the set of vertices with different matched status in these matchings, denoted by $D := (M_\tau \setminus M_{\tau'}) \cup (M_{\tau'} \setminus M_\tau)$. The set D is empty before the first arrival and remains empty until the arrival of v, as all earlier vertices than v have the same primary and secondary arcs and have the same set of free neighbors in H_τ and $H_{\tau'}$ (as $D = \emptyset$, by induction). Now, if immediately after v arrives it remains free in both M_τ and

$M_{\tau'}$, or it is matched to the same neighbor in both matchings, then clearly D remains empty. Otherwise, either v is matched to different neighbors in M_τ and $M_{\tau'}$, or v is matched in one of these matchings but not in the other. Both these cases result in $|D| = 2$. We now show by induction that the cardinality of D does not increase following subsequent arrivals, implying the lemma.

Let u be some vertex which arrives after v. If when u arrives u is matched to the same neighbor w in M_τ and $M_{\tau'}$ or if u remains free in both matchings, then D is unchanged. If u is matched to some w on arrival in M_τ, but not in $M_{\tau'}$, then since the arcs of u are the same in G_τ and $G_{\tau'}$, this implies that w must have been free in M_τ but not in $M_{\tau'}$, and so $D \ni w$. Therefore, after u arrives, we have $D \leftarrow (D \setminus \{w\}) \cup \{u\}$, and so D's cardinality is unchanged. Finally, if u is matched to two distinct neighbors, denoted by w and w', respectively, then one of (u, w) and (u, w') must be the primary arc of u in both G_τ and $G_{\tau'}$. Without loss of generality, say (u, w) is this primary arc. Since u is matched to w in M_τ but not in $M_{\tau'}$, then w must be free in M_τ when u arrives, but not in $M_{\tau'}$, and so $D \ni w$. Consequently, we have that after u arrives we have $D \leftarrow S$ for some set $S \subseteq (D \setminus \{w\}) \cup \{w'\}$, and so D's cardinality does not increase. $\square$

4.4 Deferred Proofs of Section 4.2.1

Here we prove the bound on the fractional degree x_u in terms of its dual value, restated below.

Lemma 4.2.3. *For any vertices $u, v \in V$, let y_u be the potential of u prior to arrival of v. Then the fractional degree just before v arrives, $x_u := \sum_{w \in N_v(u)} x_{uw}$, is bounded as follows:*

$$\frac{y_u}{\beta} \leqslant x_u \leqslant \frac{y_u + f(1 - y_u)}{\beta}.$$

Proof. Let y_0 be u's potential after u's arrival. For the lower bound, note that it suffices to prove that every increase in the fractional degree is bounded below by the increase in the potential divided by β. When vertex u first arrived, we consider two cases.

1. $y_0 > 0$ (thus $y_0 = 1 - \theta > 0$, and so $\theta < 1$), then the increase in u's fractional degree was:

$$\sum_{v \in N_u(u)} \frac{(\theta - y_v)^+}{\beta} \left(1 + \frac{1 - \theta}{f(\theta)}\right) = \frac{f(\theta) + 1 - \theta}{\beta} = \frac{f(1 - y_0) + y_0}{\beta} \geqslant \frac{y_0}{\beta}.$$

2. $y_0 = 0$ (thus $\theta = 1$), then the increase in u's fractional degree was:

$$\sum_{v \in N_u(u)} \frac{(\theta - y_v)^+}{\beta} \left(1 + \frac{1 - \theta}{f(\theta)}\right) = \sum_{v \in N_u(u)} \frac{(\theta - y_v)^+}{\beta} \geqslant 0 = \frac{y_0}{\beta}.$$

For every subsequent increase of the fractional degree due to a newly-arrived vertex we have that:

$$\frac{(\theta - y_u^{old})^+}{\beta}\left(1 + \frac{1-\theta}{f(\theta)}\right) \geqslant \frac{(\theta - y_u^{old})^+}{\beta},$$

Which concludes the proof for the lower bound.

For the upper bound, by [267, Invariant 1], we have that

$$\beta \cdot x_u \leqslant y_c + f(1-y_0) + \int_{y_0}^{y_c} \frac{1-x}{f(x)}\,dx. \tag{4.27}$$

This upper bound can be simplified by using Equation (4.2), as follows. Taking (4.27), adding and subtracting $1+f(1-y_u)$ and writing the integral $\int_{y_0}^{y_u} \frac{1-x}{f(x)}\,dx$ as the difference of two integrals $\int_{y_0}^{1} \frac{1-x}{f(x)}\,dx$ - $\int_{y_u}^{1} \frac{1-x}{f(x)}\,dx$, and relying on Equation (4.2), we find that

$$\begin{aligned}\beta \cdot x_u &\leqslant y_c + f(1-y_0) + \int_{y_0}^{y_c} \frac{1-x}{f(x)}\,dx \\ &= \left(1 + f(1-y_0) + \int_{y_0}^{1} \frac{1-x}{f(x)}\,dx\right) - 1 + y_c + \int_{1}^{y_c} \frac{1-x}{f(x)}\,dx \\ &= \beta^*(f) + y_c - \left(1 + f(1-y_c) + \int_{y_c}^{1} \frac{1-x}{f(x)}\,dx\right) + f(1-y_c) \\ &= \beta^*(f) + y_c - \beta^*(f) + f(1-y_c) \\ &= y_c + f(1-y_c),\end{aligned}$$

from which the lemma follows. □

4.5 Deferred Proofs of Section 4.2.4

In this section we present the proofs deferred from Section 4.2.4. We start by presenting a more manageable form for the function $f = f_{1+2\varepsilon}$ which we use.

A function in the WW family is determined by a parameter $k \geqslant 1$ and takes the following form

$$f_\kappa(\theta) = \left(\frac{1+\kappa}{2} - \theta\right)^{\frac{1+\kappa}{2\kappa}} \left(\theta + \frac{\kappa-1}{2}\right)^{\frac{\kappa-1}{2\kappa}}.$$

Letting $\kappa = 1 + 2\varepsilon$, we get that $f := f_\kappa$ is of the form

$$\begin{aligned}f(\theta) &= (1+\varepsilon-\theta)^{\frac{1+\varepsilon}{1+2\varepsilon}} \cdot (\theta+\varepsilon)^{\frac{\varepsilon}{1+2\varepsilon}} \\ &= (1+\varepsilon-\theta) \cdot \left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right)^{\frac{\varepsilon}{1+2\varepsilon}}.\end{aligned}$$

Clearly this is water filling when $\varepsilon = 0$ and otherwise we have that the first term is like water filling and then the second term is less than 1 for $z \leqslant 1/2$ and greater than 1 if $z > 1/2$.

By Taylor expansion, we obtain the following more manageable form for f.

Lemma 4.5.1. *There exists $\varepsilon_0 \in (0,1)$ such that for every $\varepsilon \in (0, \varepsilon_0)$ and every $\theta \in [0,1]$, we have*

$$f(\theta) \leqslant (1-\theta)\left(1+\varepsilon \ln\left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right)\right) + 1.01\varepsilon.$$

Proof. Taking the Taylor expansion of e^x, we find that

$$\begin{aligned}
f(\theta) &= (1+\varepsilon-\theta)\cdot\left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right)^{\frac{\varepsilon}{1+2\varepsilon}} \\
&= (1+\varepsilon-\theta)\cdot\sum_{i=0}^{\infty}\frac{\left(\ln\left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right)\cdot\frac{\varepsilon}{1+2\varepsilon}\right)^i}{i!} \\
&= (1+\varepsilon-\theta)\left(1+\ln\left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right)\cdot\frac{\varepsilon}{1+2\varepsilon}\right)+o(\varepsilon) \\
&= (1+\varepsilon-\theta)+(1-\theta)\ln\left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right)\cdot\frac{\varepsilon}{1+2\varepsilon}+o(\varepsilon) \\
&= (1+\varepsilon-\theta)+(1-\theta)\varepsilon\ln\left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right)+o(\varepsilon) \\
&= (1-\theta)\left(1+\varepsilon\ln\left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right)\right)+\varepsilon+o(\varepsilon).
\end{aligned}$$

To be precise, for $\theta \in [0,1]$ and $0 < \varepsilon \leqslant \varepsilon_0 \leqslant 1$ (implying for example $\frac{\theta+\varepsilon}{1+\varepsilon-\theta} \leqslant \frac{2}{\varepsilon}$), we will show that terms dropped in the third, fourth and fifth lines are all at most some $O((\ln(\frac{1}{\varepsilon})\cdot\varepsilon)^2) = o(\varepsilon)$, from which the lemma follows as the sum of these terms is at most 0.01ε for $\varepsilon \leqslant \varepsilon_0$ and ε_0 sufficiently small.

Indeed, in the third line, we dropped

$$(1+\varepsilon-\theta)\cdot\sum_{i=2}^{\infty}\frac{\left(\ln\left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right)\cdot\frac{\varepsilon}{1+2\varepsilon}\right)^i}{i!} \leqslant 2\cdot\sum_{i=2}^{\infty}\frac{(\ln(\frac{2}{\varepsilon})\cdot\varepsilon)^i}{i!} = O((\ln(1/\varepsilon)\cdot\varepsilon)^2),$$

where the last step used that $\ln(1/\varepsilon)\cdot\varepsilon \leqslant 1$ holds for all $\varepsilon \geqslant 0$. In the fourth line, we dropped

$$\varepsilon\cdot\ln\left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right)\cdot\frac{\varepsilon}{1+2\varepsilon} \leqslant \varepsilon^2\cdot\ln\left(2/\varepsilon\right) = O((\ln(1/\varepsilon)\cdot\varepsilon)^2).$$

Finally, in the fifth line, we dropped

$$(1-z)\cdot\left(\varepsilon-\frac{\varepsilon}{1+2\varepsilon}\right)\cdot\ln\left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right) \leqslant 1\cdot(\varepsilon^2/(1+2\varepsilon))\cdot\ln\left(2/\varepsilon\right) = O((\ln(1/\varepsilon)\cdot\varepsilon)^2).\square$$

Given this more manageable form for f, we can now turn to prove Lemma 4.2.10, restated below.

Lemma 4.2.10. *(Basic bounds on conditional probabilities z_u) There exist absolute constants $c \in (0,1)$ and $C > 1/c > 1$ and $\varepsilon_0 \in (0,1)$ such that for every $\varepsilon \in (0, \varepsilon_0)$ the following holds: for every vertex $v \in V$, if y_u is the dual variable of a neighbor $u \in N_v(v)$ before v's arrival and θ is the value chosen by Algorithm 1 on v's arrival, then for z_u as defined in Algorithm 3, we have:*

(1) *If $\theta \notin (c, 1-c)$, then $\sum_{u \in N_v(v)} z_u \leqslant 1$,*
(2) *If $\theta \in [0,1]$, then $\sum_{u \in N_v(v)} z_u \leqslant 1 + C\varepsilon$,*
(3) *If $\sum_{u \in N_v(v)} z_u > 1$, then $z_u \leqslant C\sqrt{\varepsilon}$ for every $u \in N_v(v)$,*
(4) *If $\sum_{u \in N_v(v)} z_u > 1$, then for every $u \in N_v(v)$ such that $z_u > 0$, one has $y_u \in [c/2, 1 - c/2]$, and*
(5) *For all $u \in N_v(v)$, one has $z_u \leqslant 1/2 + O(\sqrt{\varepsilon})$.*

Proof. We begin by getting a generic upper bound for z_u. We note that each edge e is matched by Algorithm 3 with probability at most x_e by Line 13. Therefore, u is matched before v arrives with probability at most $x_u := \sum_{w \in N_v(u) \setminus \{v\}} x_{wu}$, the fractional degree of u before v arrives. Therefore, by Lemma 4.2.3, the probability that u is free is at least

$$\Pr[u \text{ free when } v \text{ arrives}] \geqslant 1 - x_u \geqslant 1 - \frac{y_u + f(1-y_u)}{\beta}, \tag{4.28}$$

from which, together with the definition of $x_{uv} = \frac{1}{\beta}(\theta - y_u)^+ \left(1 + \frac{1-\theta}{f(\theta)}\right)$, we obtain the following upper bound on z_u:

$$z_u = \frac{x_{uv}}{\Pr[u \text{ is free when } v \text{ arrive}]} \leqslant \frac{\frac{1}{\beta}(\theta - y_u)^+ \left(1 + \frac{1-\theta}{f(\theta)}\right)}{1 - \frac{y_u + f(1-y_u)}{\beta}} = \frac{(\theta - y_u)\left(1 + \frac{1-\theta}{f(\theta)}\right)}{\beta - (y_u + f(1-y_u))}. \tag{4.29}$$

We start by upper bounding $\sum_{u \in N_v(v)} z_u$, giving a bound which will prove useful in the proofs of both **(1)** and **(2)**. Recall that θ is defined as the largest $\theta \leqslant 1$ such that

$$\sum_{u \in N_v(v)} (\theta - y_u)^+ \leqslant f(\theta). \tag{4.30}$$

Summing (4.29) over all $u \in N_v(v)$, we find that

$$\begin{aligned}\sum_{u \in N_v(v)} z_u &\leqslant \sum_{u \in N_v(v)} \frac{(\theta - y_u)^+ \cdot (1 + \frac{1-\theta}{f(\theta)})}{\beta - (\theta + f(1-\theta))} && (f(\cdot) \text{ is non-increasing, by Observation 4.2.4}) \\ &\leqslant \frac{f(\theta) + 1 - \theta}{\beta - (\theta + f(1-\theta))} && (\text{by (4.30) and } \beta \geqslant \beta^*(f) = 1 + f(0) \geqslant \theta + f(1-\theta))\end{aligned}$$

We therefore wish to upper bound $\frac{f(\theta)+1-\theta}{\beta - \theta - f(1-\theta)}$. To this end let $\gamma(\theta, \varepsilon) := \varepsilon \ln\left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right)$. Before proceeding to the proof, it would be useful to summarize some properties of the function $\gamma(\theta, \varepsilon)$.

1. $\gamma(\theta, \varepsilon) = -\gamma(1-\theta, \varepsilon)$ for all $\theta \in [0,1]$.
2. For c, ε_0 sufficiently small we have for all $\theta \in [0, c)$ that $\gamma(\theta,\varepsilon) \leqslant \varepsilon \ln\left(\frac{c+\varepsilon}{1+\varepsilon-c}\right) \leqslant -20\cdot\varepsilon$, and for all $\theta \in (1-c, 1]$ that $\gamma(\theta,\varepsilon) \geqslant \varepsilon \ln\left(\frac{1-c+\varepsilon}{1+\varepsilon-(1-c)}\right) \geqslant 20\cdot\varepsilon$.
3. $\gamma(\theta,\varepsilon)\cdot(1-2\theta) \leqslant 0$ for $\theta \in [0,1]$, since $\gamma(\theta,\varepsilon) \leqslant 0$ for $\theta \leqslant 1/2$ and $\gamma(\theta,\varepsilon) \geqslant 0$ for $\theta \geqslant 1/2$.
4. $\theta\cdot\gamma(\theta,\varepsilon) \geqslant -\varepsilon$ for all $\theta \in [0,1]$.
   The last property follows from $\ln\left(\frac{1+\varepsilon-\theta}{\theta+\varepsilon}\right) \leqslant \ln\left(\frac{1+\varepsilon+\theta}{\theta+\varepsilon}\right) \leqslant \ln\left(1+\frac{1}{\theta+\varepsilon}\right) \leqslant \frac{1}{\theta+\varepsilon} \leqslant \frac{1}{\theta}$, which implies in particular that $\theta\cdot\gamma(\theta,\varepsilon) = \theta\cdot\varepsilon\cdot\left(-\ln\left(\frac{1+\varepsilon-\theta}{\theta+\varepsilon}\right)\right) \geqslant -\varepsilon$.

We will use γ as shorthand for $\gamma(\theta,\varepsilon)$. Recalling that $\beta = 2-\varepsilon$ and using Lemma 4.5.1, we have

$$\begin{aligned}\frac{f(\theta)+1-\theta}{\beta-(\theta+f(1-\theta))} &\leqslant \frac{(1-\theta)\left(1+\varepsilon\ln\left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right)\right)-\theta+1+1.01\varepsilon}{2-\varepsilon-\theta-\theta\left(1+\varepsilon\ln\left(\frac{1-\theta+\varepsilon}{\theta+\varepsilon}\right)\right)-1.01\varepsilon} \\ &\leqslant \frac{(1-\theta)(2+\gamma)+2\varepsilon}{2-2\theta+\theta\gamma-3\varepsilon} \\ &= 1+\frac{\gamma(1-2\theta)+5\varepsilon}{2-2\theta+\theta\gamma-3\varepsilon}.\end{aligned} \tag{4.31}$$

We will continue by proving that the second term is negative. First we prove that the denominator is positive. To this end, first consider the case when $\theta \in [0,c)$. In this case for ε_0, c sufficiently small one has that: $2-2\theta+\theta\gamma-2\varepsilon > 2-2\theta-\varepsilon-2\varepsilon > 0$ from Item 4. Moreover, when $\theta \in (1-c,1]$ one has that $\theta > \frac{1}{2}$ (since c is small) and $\gamma \geqslant 20\varepsilon$ from Item 2. Thus $2-2\theta+\theta\gamma-2\varepsilon \geqslant \theta\gamma-2\varepsilon \geqslant \frac{1}{2}\cdot 20\varepsilon-3\varepsilon = 7\varepsilon > 0$. Now, it remains to prove that the numerator is always negative. When $\theta \in [0,c)$ we have that $1-2\theta \geqslant 3/4$(since c is small) and $\gamma \leqslant -20\varepsilon$ from Item 2, therefore $\gamma(1-2\theta)+5\varepsilon \leqslant \gamma\cdot\frac{3}{4}+5\cdot(-\frac{\gamma}{20}) = \frac{\gamma}{2} < 0$. In the case where $\theta \in (1-c,1]$, we have that $1-2\theta < -3/4$, and $\theta > 1/2$ (since c is small), and $\gamma \geqslant 20\varepsilon$ from Item 2, thus $\gamma(1-2\theta)+5\varepsilon \leqslant -\frac{3}{4}\cdot 20\varepsilon+5\varepsilon = -10\varepsilon < 0$.

We now turn to **(2)**. We assume that $\theta \in (c, 1-c)$, since otherwise the claim is trivial, by **(1)**. We have by (4.31) that $\frac{f(\theta)+1-\theta}{\beta-(\theta-f(1-\theta))} \leqslant 1+\frac{\gamma(1-2\theta)+5\varepsilon}{2-2\theta+\theta\gamma-3\varepsilon}$. We have that $\gamma(1-2\theta)+5\varepsilon \leqslant 5\varepsilon$ from Item 3. Furthermore, using Item 4 we have that $2-2\theta+\theta\gamma-3\varepsilon \geqslant 2c+-4\varepsilon > c$ for a sufficiently small ε_0. Overall, the second term is bounded above by $\frac{5}{c}\cdot\varepsilon < C\cdot\varepsilon$, for $C > \frac{5}{c} > \frac{1}{c}$ as required.

We now prove **(3)**. Note that by **(1)**, $\sum_{u\in N_v(v)} z_u > 1$ implies that $\theta \in (c, 1-c)$. Now, for every $u \in N_v(v)$, let $\alpha_u := \frac{(\theta-y_u)^+}{f(\theta)}$, so that $y_u = \theta - f(\theta)\cdot\alpha_u$ if $y_u \leqslant \theta$. We also note that by definition of α_u and our choice of θ, we have $\sum_{u\in N_v(v)}\alpha_u = \sum_{u\in N_v(v)}\frac{(\theta-y_u)^+}{f(\theta)} \leqslant 1$. In the proof of **(3)** and **(4)** we will assume for notational simplicity that all $u \in N_v(v)$ have $y_u \leqslant \theta$, implying

$z_u \geqslant 0$. Summing up (4.29) over all $u \in N_v(v)$ and substituting in α_u, we thus find that

$$\begin{aligned}
\sum_{u\in N_v(v)} z_u &\leqslant \sum_{u\in N_v(v)} \frac{(\theta - y_u)^+(1+\frac{1-\theta}{f(\theta)})}{\beta - (y_u + f(1-y_u))} \\
&= \sum_{u\in N_v(v)} \alpha_u \cdot \frac{f(\theta)+1-\theta}{\beta - (y_u + f(1-y_u))} \\
&\leqslant \sum_{u\in N_v(v)} \alpha_u \cdot \frac{f(\theta)+1-\theta}{2 - y_u - f(1-y_u) - 2.01\varepsilon} \qquad \text{(by Lemma 4.5.1 and } \beta = 2-\varepsilon) \\
&\leqslant \sum_{u\in N_v(v)} \alpha_u \cdot \frac{f(\theta)+1-\theta}{2-4\varepsilon-2y_u},
\end{aligned}$$

In the last transition we used again (as in Item 4) that $y_u \cdot \varepsilon \ln\left(\frac{1-y_u+\varepsilon}{y_u+\varepsilon}\right) \leqslant \varepsilon$, which implies $f(\theta) \leqslant 1-\theta+\varepsilon$ for all $\theta \in [0,1]$. Substituting $y_u = \theta - f(\theta)\cdot\alpha_u$ into the above upper bound on $\sum_{u\in N_v(v)} z_u$, we get

$$\begin{aligned}
\sum_{u\in N_v(v)} z_u &\leqslant \sum_{u\in N_v(v)} \alpha_u \cdot \frac{f(\theta)+1-\theta}{2-4\varepsilon-2\theta+2f(\theta)\cdot\alpha_u} \\
&= \sum_{u\in N_v(v)} \alpha_u \cdot \frac{f(\theta)+1-\theta}{2-4\varepsilon-2\theta} - \sum_{u\in N_v(v)} \frac{(f(\theta)+1-\theta)\cdot 2f(\theta)\cdot\alpha_u^2}{(2-4\varepsilon-2\theta)\cdot(2-4\varepsilon-2\theta+2f(\theta)\cdot\alpha_u)},
\end{aligned} \tag{4.32}$$

using the elementary identity $\frac{1}{a+b} = \frac{1}{a} - \frac{b}{a(a+b)}$ for appropriate a and b. Now, both terms in the last line of (4.32) can be significantly simplified, as follows. For the former term, again using that $f(\theta) \leqslant 1-\theta+\varepsilon$, together with $\sum_{u\in N_v(v)} \alpha_u \leqslant 1$ noted above, we find that

$$\sum_{u\in N_v(v)} \alpha_u \cdot \frac{f(\theta)+1-\theta}{2-4\varepsilon-2\theta} \leqslant \sum_{u\in N_v(v)} \alpha_u \cdot \frac{2+\varepsilon-2\theta}{2-4\varepsilon-2\theta} = \sum_{u\in N_v(v)} \alpha_u \cdot \left(1 + \frac{5\varepsilon}{2-4\varepsilon-2\theta}\right) \leqslant 1 + O(\varepsilon), \tag{4.33}$$

where in the last step we used that $\theta \leqslant 1-c$ and c is some fixed constant. For the second term in the last line of (4.32), we note that

$$\sum_{u\in N_v(v)} \frac{(f(\theta)+1-\theta)\cdot 2f(\theta)\cdot\alpha_u^2}{(2-4\varepsilon-2\theta)\cdot(2-4\varepsilon-2\theta+2f(\theta)\cdot\alpha_u)} = \Omega(1)\cdot\left(\sum_{u\in N_v(v)} \alpha_u^2\right). \tag{4.34}$$

To see this, first note that for $\theta \in (c, 1-c)$, the numerator of each summand of the LHS is at least $2f(c)^2\cdot\alpha_u^2 \geqslant \Omega(\alpha_u^2)$, since f is decreasing by Observation 4.2.4 and $f(c) \geqslant \frac{1}{2}\cdot(1+\varepsilon-c) \geqslant \Omega(1)$ for c and ε sufficiently small. To verify the first inequality of this lower bound for $f(c)$, recall that $f(c) = (1+\varepsilon-c)\cdot\left(\frac{c+\varepsilon}{1+\varepsilon-c}\right)^{\frac{\varepsilon}{1+2\varepsilon}}$. Now, for ε tending to zero and $c < 1/2$, the term $\left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right)^{\frac{\varepsilon}{1+2\varepsilon}}$

tends to one as ε tends to zero. Therefore for ε sufficiently small we have $f(c) \geqslant \frac{1}{2} \cdot (1 + \varepsilon - c)$ for all $c < 1/2$. We now turn to upper bounding the denominator of each summand in the LHS of Equation (4.34). Indeed, substituting $y_u = \theta - f(\theta) \cdot \alpha_u$, we find that each such denominator is at most $(2 - 4\varepsilon - 2\theta) \cdot (2 - 4\varepsilon - 2\theta + 2f(\theta) \cdot \alpha_u) \leqslant (1/2) \cdot (2 - 4\varepsilon - 2y_u) \leqslant (1/2) \cdot (2 - 4\varepsilon - 2c) \leqslant O(1)$ for c and ε sufficiently small. Note that both numerator and denominator are positive for sufficiently small c and ε_0. Substituting the bounds of (4.33) and (4.34) into (4.32), we obtain

$$\sum_{u \in N_v(v)} z_u \leqslant 1 + O(\varepsilon) - \Omega(1) \cdot \left(\sum_{u \in N_v(v)} \alpha_u^2 \right). \tag{4.35}$$

From Eq. (4.35) and $\sum_{u \in N_v(v)} z_u > 1$ by assumption of **(3)**, we get that

$$\sum_{u \in N_v(v)} \alpha_u^2 \leqslant C' \varepsilon \tag{4.36}$$

for an absolute constant $C' > 1$, since otherwise $\sum_{u \in N_v(v)} z_u \leqslant 1$. Finally, we note that

$$\begin{aligned}
\sum_{u \in N_v(v)} z_u^2 &= \sum_{u \in N_v(v)} \left(\frac{\alpha_u \cdot (f(\theta) + 1 - \theta)}{\beta - (y_u + f(1 - y_u))} \right)^2 \\
&\leqslant \left(\sum_{u \in N_v(v)} \alpha_u^2 \right) \cdot \left(\frac{f(\theta) + 1 - \theta}{\beta - (\theta + f(1 - \theta))} \right)^2 && \text{(by Observation 4.2.4 and } y_u \leqslant \theta) \\
&\leqslant \left(\sum_{u \in N_v(v)} \alpha_u^2 \right) \cdot \left(\frac{f(\theta) + 1 - \theta}{\beta - (1 - c + f(c))} \right)^2 && \text{(by Observation 4.2.4 and } \theta \leqslant 1 - c) \\
&\leqslant \left(\sum_{u \in N_v(v)} \alpha_u^2 \right) \cdot \left(\frac{1 - \theta + \varepsilon + 1 - \theta}{\beta - (1 - c + 1 - c + \varepsilon)} \right)^2 && (f(c) \leqslant 1 - c + \varepsilon) \\
&\leqslant \left(\sum_{u \in N_v(v)} \alpha_u^2 \right) \cdot \frac{2}{2c - 2\varepsilon} \\
&\leqslant C\varepsilon,
\end{aligned}$$

for some constant $C \geqslant \frac{2}{2c - 2\varepsilon}$. Thus $z_u^2 \leqslant \sum_{u \in N_v(v)} z_u \leqslant C\varepsilon$ and so $z_u \leqslant \sqrt{C \cdot \varepsilon} \leqslant C\sqrt{\varepsilon}$, as claimed.

We now prove **(4)**. Since $\sum_{u \in N_v(v)} z_u > 1$ implies $\theta \in (c, 1-c)$ by **(1)**, using the definition of α_u's from the proof of **(3)** together with the fact that $\alpha_u \leqslant C' \sqrt{\varepsilon}$ for every $u \in N_v(v)$ by (4.36) and the fact that $f(\theta) \leqslant 2$ for all $\theta \in [0, 1]$ (by Lemma 4.5.1), we get that for sufficiently small $\varepsilon_0 > 0$,

$$y_u = \theta - f(\theta) \cdot \alpha_u \in [c - O(\sqrt{\varepsilon}), 1 - c] \subseteq [c/2, 1 - c/2].$$

As for **(5)**, simplifying (4.29) and using the fact that $\theta - y_u \leqslant f(\theta)$, we get

$$z_u \leqslant \frac{\theta - y_u + 1 - \theta}{\beta - y_u - f(1 - y_u)} = \frac{1 - y_u}{\beta - y_u - f(1 - y_u)}.$$

Recall from Lemma 4.5.1 that for all $\theta \in [0,1]$, we have $f(\theta) \leqslant (1-\theta)\left(1 + \varepsilon \ln\left(\frac{\theta+\varepsilon}{1+\varepsilon-\theta}\right)\right) + 1.01\varepsilon$, which implies the following:

1. For all $\theta \in [0,1]$, we have $f(\theta) \leqslant 1 - \theta + \sqrt{\varepsilon}$, and
2. For $\theta < e^{-10}$, we have $f(\theta) \leqslant (1-\theta)(1 + \varepsilon(\ln((e^{-10} + \varepsilon)/(1 - e^{-10} + \varepsilon)) + 1.01\varepsilon \leqslant 1 - \theta - 2\varepsilon$.

Suppose that $y_u \leqslant 1 - e^{-10}$. Then using Item 1, we have

$$\begin{aligned} z_u &\leqslant \frac{1 - y_u}{\beta - y_u - f(1 - y_u)} \leqslant \frac{1 - y_u}{2 - \varepsilon - y_u - y_u - \sqrt{\varepsilon}} \\ &\leqslant \frac{1 - y_u}{2(1 - y_u) - 2\sqrt{\varepsilon}} \leqslant 1/2 + \frac{2\sqrt{\varepsilon}}{2e^{-10} - 2\sqrt{\varepsilon}} \leqslant 1/2 + O(\sqrt{\varepsilon}). \end{aligned}$$

Now suppose that $y_u > 1 - e^{-10}$. Then $1 - y_u < e^{-10}$, and so by Item 2, $f(1 - y_u) \leqslant 1 - y_u - 2\varepsilon$. Thus we have the final required inequality,

$$z_u \leqslant \frac{1 - y}{\beta - y_u - f(1 - y_u)} \leqslant \frac{1 - y_u}{2 - \varepsilon - y_u - (y_u - 2\varepsilon)} = \frac{1 - y_u}{2(1 - y_u) + \varepsilon} \leqslant 1/2. \qquad \square$$

Finally, we rely on Lemma 4.5.1 to prove that the fractional solution maintained by Line 4 is $1/\beta$ competitive, as implied by Lemma 4.2.5 and the following restated fact.

Fact 4.2.12. *For all sufficiently small $\varepsilon > 0$, we have that $2 - \varepsilon \geqslant \beta^*(f_{1+2\varepsilon})$.*

Proof. Let us denote as before $f = f_{1+2\varepsilon}$. Recall that $\beta^*(f) = 1 + f(0)$. By Lemma 4.5.1, this is at most $1 + f(0) \leqslant 1 + \left(1 + \varepsilon \ln\left(\frac{\varepsilon}{1+\varepsilon}\right)\right) + 1.01\varepsilon$. But for small enough ε, we have that $\ln\left(\frac{\varepsilon}{1+\varepsilon}\right) \leqslant -2.01$, implying that $1 + f(0) \leqslant 2 - \varepsilon$, as claimed. $\square$

4.6 Conclusion and Open Questions

In this chapter we gave the first randomized online matching algorithm under general vertex arrivals to beat the optimal $1/2$ competitive ratio achievable by deterministic algorithms. This chapter suggests a number of open questions, the most natural of which is what is improving on our competitive ratio, and ideally achieving the optimal such ratio. Our algorithm's competitive ratio is greater than $1/2$ by some constant which we did not try to optimize, and so this unspecified constant improvement is likely not large. Can one obtain a more sizable improvement over the natural bound of $1/2$? Can one match the 0.526 bound for fractional matchings given by [267]? Can one do better? We note that matching the bound of [267] could be obtained by an online rounding scheme which rounds this fractional matching algorithm's output *losslessly*, as opposed to our approach, which incurred some constant multiplicative loss. In the next chapter we discuss the abstract problem of lossless online rounding of fractional matchings, showing that this approach may be challenging, but also hinting that it is possibly not hopeless.

Chapter 5

Online Dependent Rounding

In this chapter, based on [69] (joint with Ilan R. Cohen), we give asymptotically near optimal online bipartite matching algorithms for regular graphs. This result required us to study the abstract problem of *online* dependent rounding. Indeed, underlying our results for regular graphs, is a near-lossless online dependent rounding scheme for bounded fractional bipartite matchings which we present, from which we obtain our results for regular graphs. This online dependent rounding scheme also proves useful in Chapter 6, where we study online bipartite edge coloring.

5.1 Background

A particularly well-studied class of graphs in the context of the maximum matching problem are Δ-regular bipartite graphs; i.e., bipartite graphs in which each vertex neighbors Δ other vertices. This class of graphs has been studied in many contexts, including expander graph constructions, scheduling, routing in switch fabrics, and task assignment [5, 72, 218]. In the context of matching theory, a consequence of Hall's Theorem [155] implies that such graphs can be decomposed into Δ disjoint perfect matchings. This result, which is equivalent to the existence of a perfect matching in every regular bipartite graph, was first proved by König more than a century ago [184], and is one of the seminal results in matching theory. In the traditional, offline model of computation, numerous linear and near-linear time algorithms for computing such a perfect matching are known [10, 71, 72, 74, 130, 131, 245], as well as a *sublinear*, $O(n \log n)$-time randomized algorithm for this problem [133].

In online models of computation, we shall show in Chapter 7, the optimal deterministic competitive ratio for such graphs is $1 - (1 - 1/\Delta)^{\Delta}$; that is, better than the optimal (randomized) $1 - {}^{1}\!/_{e}$ bound for arbitrary bipartite graphs [179], but tending to this bound from above as Δ grows. For random-order and stochastic arrival models, it is known that a competitive ratio of $1 - O(1/\sqrt{\Delta})$, i.e. tending to *one* is possible [21, 176]. This begs the question, "what is the optimal competitive ratio for randomized algorithms for regular graphs under adversarial arrivals?" Before addressing our results for this class of graphs, we discuss an abstract problem underlying our solution of this problem, which is useful in its own right, as we shall see in this chapter and in Chapter 6.

5.1.1 An Abstract Problem: Online Dependent Rounding

A successful design paradigm for online algorithms (and approximation algorithms more broadly) is the *randomized rounding* method, first advocated for by Raghavan and Tompson [238]. Broadly, this approach starts with an α-approximate fractional solution, which is rounded to an integral solution, incurring a multiplicative loss of β on the approximation ratio, resulting in an $\alpha \cdot \beta$ approximation ratio. Let us consider this approach for the online bipartite matching problem, starting with its offline counterpart.

Let us fix some graph $G = (V, E)$ and (re)consider its fractional matching polytope,

$$\mathcal{P}(G) := \left\{ \vec{x} \in \mathbb{R}^{|E|}_{\geqslant 0} \;\middle|\; \sum_{e \ni v} x_e \leqslant 1 \quad \forall v \in V \right\}.$$

We recall that for bipartite graphs, this LP relaxation has an integrality gap of *one*; that is, for any fractional matching of value $\sum_e x_e = k$, there exists an integral matching of size at least k [247]. Moreover, in an offline setting, such an integral matching can be computed efficiently [4, 122]. The matching constraints are not satisfied by randomly rounding each edge independently. However, Gandhi et al. [122] showed a *dependent* rounding scheme which outputs a (random) integral matching $\mathcal{M}$ such that each edge $e \in E$ is matched with probability $\Pr[e \in \mathcal{M}] = x_e$, thus rounding the fractional matchings $\vec{x}$ *losslessly*, even preserving the marginals for every edge. This dependent rounding scheme and its extensions have proven immensely useful over the years (see, e.g., [24, 56, 57, 63, 64]).

We now consider this approach in an *online* setting, starting with the problem of computing a fractional matching. Here, an optimal, $(1 - 1/e)$-competitive, fractional algorithm is known [173]. For Δ-regular graphs, there is a trivial 1-competitive fractional algorithm, which simply sets $x_e = \frac{1}{\Delta}$ for each edge e. We now consider *online* rounding of fractional bipartite matchings. For this problem, the online contention resolution schemes (OCRSes) of Feldman et al. [110] output a matching $\mathcal{M}$ matching each edge e with probability at least $\Pr[e \in \mathcal{M}] \geqslant x_e \cdot 1/2e$. Unfortunately, then, such OCRSes result in a possibly worse competitive ratio than greedy's ratio of $1/2$. However, this does not rule out better results using another online rounding scheme. Can we obtain an optimal randomized algorithm by rounding the fractional algorithm of [173]? Can we obtain 1-competitive algorithms for regular graphs by rounding the trivial 1-competitive fractional algorithm's output? Even better, can we, similarly to [122], on *any* input fractional matching $\vec{x}$ revealed in an online fashion, output a random matching $\mathcal{M}$ such that each edge e is matched in $\mathcal{M}$ with probability $\Pr[e \in \mathcal{M}] = x_e$?

Unfortunately, for all these questions, the answer is no, as can be seen by inspecting the optimal fractional algorithm in, say, 8-cycles with two diametrically-opposed online nodes arriving first [79]. For this family of graphs, both the fractional algorithm of [173] and the trivial algorithm for 2-regular graphs are 1-competitive, while the best randomized algorithm is at best $7/8$-competitive. So, perfect rounding is *impossible*.

In this chapter, we explore the power and limitations of online dependent rounding for online bipartite matching. In particular, we obtain such an online rounding scheme which is in some sense best possible, and use it to design online bipartite matching algorithms for regular graphs with an asymptotically near optimal competitive ratio, up to sub-logarithmic multiplicative terms in their error term.

5.1.2 Our Contributions

Extending the 8-cycle example of [79] (who credit this example to Nick Harvey), we first prove the following lower bound for online matching in regular graphs.

Theorem 5.1.1. *Any randomized online bipartite matching algorithm has competitive ratio* $(1 - \Omega(\sqrt{1/\Delta}))$ *on* Δ*-regular graphs under adversarial one-sided vertex arrivals.*

We note that most prior lower bounds for online matching [19, 55, 95, 120, 165, 173, 179] were stated, or can be recast, as lower bounds for fractional algorithms. This "first moment" method, however, is insufficient for our needs, where one must explicitly consider variance, since there is a separation between fractional and randomized algorithms for this problem, as we show.

Considering the 1-competitive fractional matching algorithm for regular graphs which assigns a value of $\frac{1}{\Delta}$ to each edge, we obtain the following lower bound for online dependent rounding schemes, relating their rounding loss to the ℓ_∞ of the fractional matching to round.

Corollary 5.1.2. *For any real* $\delta > 0$*, any online dependent rounding scheme* $\mathcal{A}$ *has some graph* G *and fractional matching* $\vec{x}$ *in* G *with* $|x|_\infty \leqslant \delta$*, such that the randomized matching* $\mathcal{M}$ *output by* $\mathcal{A}$ *on* $(G, \vec{x})$ *has expected size at most*

$$\mathbb{E}[|\mathcal{M}|] \leqslant \sum_e x_e \cdot (1 - \Omega(\sqrt{\delta})).$$

Consequently, G *has some edge* e *for which*

$$\Pr[e \in \mathcal{M}] \leqslant x_e \cdot (1 - \Omega(\sqrt{\delta})),$$

In Section 5.3 we show that an online rounding scheme whose multiplicative loss has a similar polynomial dependence on $|\vec{x}|_\infty$ can be achieved.

Theorem 5.1.3. *There exists an online algorithm, which, given online fractional bipartite matching* $\vec{x}$ *satisfying* $|x|_\infty \leqslant \delta$*, outputs a random matching* $\mathcal{M}$ *satisfying*

$$\Pr[e \in \mathcal{M}] = x_e \cdot (1 - O(\sqrt[3]{\delta \cdot \log(1/\delta)})) \quad \forall e \in E.$$

Consequently, this output matching $\mathcal{M}$ *has expected size at least*

$$\mathbb{E}[\mathcal{M}] \geqslant \sum_e x_e \cdot (1 - O(\sqrt[3]{\delta \cdot \log(1/\delta)})).$$

Moreover, with high probability, the output matching has size at least

$$|\mathcal{M}| \geqslant \mathbb{E}[|\mathcal{M}|] - O(\sqrt{n \log n}).$$

In Section 5.4 we discuss applications of this online dependent rounding scheme for online matching in bounded-degree graphs, including regular and near-regular graphs. In particular, we give asymptotically near optimal algorithms for regular graphs, as in the following theorem.

Theorem 5.1.4. *There exists a* $(1 - O(\sqrt{(\log \Delta)/\Delta}))$*-competitive randomized online bipartite matching algorithm for* Δ*-regular graphs under adversarial one-sided vertex arrivals.*

Up to a sub-logarithmic factor in Δ in the loss term, the competitive ratio of Theorem 5.1.4 matches the lower bound of Theorem 5.1.1, as well as the results previously known for regular graphs under *stochastic* and *random-arrival* online models [21, 176]. Moreover, we show that while for deterministic algorithms online matching in regular graphs becomes harder as Δ increases, with the optimal competitive ratio tending to $1 - {}^1/_e$ as Δ increases (see Chapter 7), the problem becomes easier for randomized algorithms, for which the optimal competitive ratio converges to *one*.

Most of this chapter is dedicated to proving Theorem 5.1.3. Before laying the groundwork for this theorem, we tun to proving Theorem 5.1.1.

5.2 Impossibility of Lossless Online Rounding

In this section we give an infinite family of graphs and fractional matchings demonstrating that perfect online dependent rounding is impossible. In particular, we consider Δ-regular graphs, where each node has degree Δ. Such graphs admit a trivial 1-competitive fractional matching algorithm, by assigning $1/\Delta$ to each edge. As we show, no randomized algorithm can match this fractional algorithm's performance.

Theorem 5.1.1. *Any randomized online bipartite matching algorithm has competitive ratio* $(1 - \Omega(\sqrt{1/\Delta}))$ *on* Δ*-regular graphs under adversarial one-sided vertex arrivals.*

Proof. We appeal to Yao's Lemma [270], giving a distribution over inputs for which no deterministic algorithm achieves competitive ratio better than the above bound, implying our claimed result. Without loss of generality, we may assume that the deterministic algorithm is maximal; i.e., the algorithm always matches when possible. The input consists of $n = \Delta^2$ offline nodes, partitioned into Δ many Δ-tuples of offline nodes. During the first phase, each of these Δ-tuples' nodes all neighbor $\Delta/2$ common online neighbors. (Taking disjoint copies of this example, the number of nodes n can grow arbitrarily large compared to Δ.) Following the first phase, the Δ offline nodes of each of the offline Δ-tuples are randomly permuted and correspondingly numbered 1 through Δ. Next, a second phase begins, during which, for each $i \in [\Delta]$, all the Δ offline nodes numbered i neighbor $\Delta/2$ common online nodes. By the maximality of the algorithm, each offline node is matched with probability $1/2$ during the first phase. Therefore, for each $i \in [\Delta]$, if we denote by X_i the number of nodes of the i-th tuple which are not matched during the first phase, we find that X_i is distributed binomially, $X_i \sim Bin(\Delta, 1/2)$. In particular, X_i's expectation is $\mathbb{E}[X_i] = \Delta/2$. On the other hand, at most $\Delta/2$ nodes numbered i can be matched during the second phase, and so the algorithm leaves $U_i = \max\{0, X_i - \Delta/2\}$ unmatched nodes among these Δ nodes. But as $X_i \sim Bin(\Delta, 1/2)$ is binomially distributed, then by the normal approximation of the binomial distribution, for large Δ, this X_i is approximately distributed $N(\Delta/2, \sqrt{\Delta}/2)$ and so the expectation of $|X_i - \mathbb{E}[X_i]|$ is $\mathbb{E}[|X_i - \mathbb{E}[X_i]|] \approx \sqrt{\Delta}/2 \cdot \sqrt{2/\pi} = \sqrt{\frac{\Delta}{2\pi}}$ ([126]). But as X_i is symmetric around its mean, the expected number of i-numbered nodes left unmatched after the second phase is

$$\mathbb{E}[U_i] = \mathbb{E}[\max\{0, X_i - \Delta/2\}] = \sqrt{\frac{\Delta}{2\pi}} \cdot \frac{1}{2} = \frac{\sqrt{\Delta}}{\sqrt{8\pi}}.$$

That is, for any $i \in [\Delta]$, the expected fraction of the Δ offline nodes numbered i and left unmatched is at least $(\sqrt{\Delta/8\pi})/\Delta = 1/\sqrt{8\pi\Delta}$. Consequently, the competitive ratio of any deterministic algorithm on this distribution of inputs is at most $1 - 1/\sqrt{8\pi\Delta}$. $\square$

As stated above, there exists a 1-competitive fractional matching algorithm for Δ-regular graphs: assign $x_e = 1/\Delta$ to each edge $e \in E$. Consequently, by linearity of expectation, this theorem implies Corollary 5.1.2, relating the loss of online dependent rounding to the ℓ_∞ norm of the fractional matching $\vec{x}$. In the following section, we prove a converse, positive result, namely that there exists an online dependent rounding scheme which matches each edge e with probability x_e up to a multiplicative error term which tends to zero with $|x|_\infty$.

5.3 Rounding Bounded Fractional Matchings

In this section we present and analyze our near-lossless online dependent rounding scheme for bounded fractional matchings. That is, we prove Theorem 5.1.3.

5.3.1 The Online Dependent Rounding Scheme

We start by presenting our online dependent rounding scheme, Algorithm MARKING, which is parameterized by some $\varepsilon \in [0, 1]$. A key concept used by this algorithm is the notion of *marking* offline nodes. Initially, all offline nodes are unmarked. Only unmarked nodes can be matched to an arriving online node. Whenever such an offline node i is matched, we mark it. In addition, the algorithm will sometimes mark unmatched offline nodes, so as to guarantee that each offline vertex i has a probability of exactly $x_{i,t} \cdot (1 - \varepsilon)$ of becoming marked following the arrival of online node t. In order to do so, at time t the algorithm associates a weight with each unmarked offline neighbor i of t which is inversely proportional to i's probability of not being marked prior. (We have a closed-form for this probability, by construction.) The algorithm then chooses a single candidate vertex to match t to (and mark), chosen with probability proportional to its weight. If the probability of nodes to be marked due to this first step is less than $x_{i,t} \cdot (1 - \varepsilon)$ (this happens if the sum of weights is greater than one), then we mark each neighbor of t with the appropriate correcting probability. The pseudocode for this algorithm, which uses the variables

$F_{i,t} := \mathbb{1}[i \text{ not marked by time } t]$ to denote whether an offline node i is free before the arrival of t, is given in Algorithm 4.

Algorithm 4 MARKING

```
1: Init: set $M \leftarrow \emptyset$
2: Init: set $F_{i,t} \leftarrow 1$ for all $i, t$                    ▷ all offline nodes initially unmarked
3: for all online nodes $t$, on arrival do
4:     read $\{x_{i,t} \mid i \in N(t)\}$
5:     for all neighbors $i$ of $t$ do
6:         let $c_{i,t} := \frac{x_{i,t} \cdot (1-\varepsilon)}{1-\sum_{t'<t} x_{i,t'} \cdot (1-\varepsilon)}$
7:         set $W_{i,t} \leftarrow c_{i,t} \cdot F_{i,t}$.
8:     pick at most one neighbor $i$ of $t$ with probability $p_{i,t} := \frac{W_{i,t}}{\max\{1, \sum_i W_{i,t}\}}$
9:     if some neighbor $i$ picked in Line 8 then
10:        $M \leftarrow M \cup \{(i,t)\}$
11:        set $F_{i,t'} \leftarrow 0$ for all $t' > t$ (implicitly)                    ▷ mark $i$
12:    if $\sum_i W_{i,t} > 1$ then
13:        for all free neighbors $i$ of $t$ do
14:            with probability $\frac{W_{i,t}-p_{i,t}}{1-p_{i,t}}$ do
15:                set $F_{i,t'} \leftarrow 0$ for all $t' > t$ (implicitly)            ▷ mark $i$
```

Intuition Behind the Analysis: At a high level, Algorithm 4 guarantees two useful properties. The first is that for each edge (i,t), the probability that i is marked at time t, denoted by $M_{i,t} := F_{i,t} - F_{i,t+1}$, is precisely $\Pr[M_{i,t}] = x_{i,t} \cdot (1-\varepsilon)$. This gives us a closed-form solution for the probability that i is free at time t, namely $\Pr[F_{i,t}] = 1 - \sum_{t'<t} \Pr[M_{i,t'}] = 1 - \sum_{t'<t} x_{i,t'}(1-\varepsilon)$. From this we obtain that $\mathbb{E}[W_{i,t}] = x_{i,t} \cdot (1-\varepsilon)$, and therefore $\mathbb{E}[\sum_i W_{i,t}] = \sum_i x_{i,t} \cdot (1-\varepsilon) \leqslant 1-\varepsilon$, with the last step following from the fractional matching constraint. As such, we might expect that the condition of Line 12 should be met rarely, which would imply that most times an offline node is marked are due to it being matched, and so $\mathbb{E}[|\mathcal{M}|] \approx \sum_e x_e \cdot (1-\varepsilon)$. Of course, the fact that $\mathbb{E}[\sum_i W_{i,t}] \leqslant 1-\varepsilon$ does not account for deviation of this sum from its mean. To address this, the core of our analysis will be to prove that Algorithm 4 guarantees strong negative correlation for the variables $F_{i,t}$, and consequently, for their weighted counterparts, $W_{i,t}$. This negative correlation underlies the results of Theorem 5.1.4. In particular, the strong negative dependence of these $F_{i,t}$ allow us to prove both per-edge and per-vertex matching guarantees, as well as concentration of the rounding loss of this scheme.

Before proceeding to analyze Algorithm 4, we prove that this algorithm is well defined, and in particular that the probabilities used throughout this algorithm are indeed probabilities.

Lemma 5.3.1. *Algorithm 4 is well defined and outputs a matching.*

Proof. First, we note that the sum of probabilities in Line 8 is at most one, and so we can indeed pick (at most) one neighbor, with each neighbor i of t picked with probability $p_{i,t}$. Next, we show that the terms $\frac{W_{i,t}-p_{i,t}}{1-p_{i,t}}$ in Line 15 are also probabilities. On the one hand, since $W_{i,t} \geqslant \frac{W_{i,t}}{\max\{1,\sum_i W_{i,t}\}} = p_{i,t}$, we have that $\frac{W_{i,t}-p_{i,t}}{1-p_{i,t}} \geqslant 0$. On the other hand, the fractional matching constraints imply $x_{i,t} \leqslant 1 - \sum_{t'<t} x_{i,t'}$, which in turn implies that the value $\frac{W_{i,t}-p_{i,t}}{1-p_{i,t}}$ is well defined (i.e., $1-p_{i,t} \neq 0$), and is at most one, since

$$p_{i,t} \leqslant W_{i,t} = \frac{x_{i,t}\cdot(1-\varepsilon)}{1-\sum_{t'<t} x_{i,t'}\cdot(1-\varepsilon)} < \frac{x_{i,t}\cdot(1-\varepsilon)}{(1-\varepsilon)-\sum_{t'<t} x_{i,t'}\cdot(1-\varepsilon)} \leqslant 1.$$

Finally, to show that Algorithm 4 outputs a valid matching, we note that each online node t is matched to at most one neighbor. On the other hand, each online node t can only be matched to a previously unmarked neighbor. Since offline nodes are marked once they are matched, no offline node i is matched more than once. We conclude that Algorithm 4 outputs a valid matching. ☐

5.3.2 Basic Properties of MARKING

We start by introducing some notation which will be useful when analyzing Algorithm 4. For every arrival time t (i.e., prior to t and its edges being processed), we let $F_t := \{i \mid F_{i,t} = 1\}$ be the set of free (i.e., unmarked) offline nodes by arrival time t. In addition, for all i, t, we let $M_{i,t} := F_{i,t} - F_{i,t+1}$ be an indicator variable for i becoming marked at time t.

One aim of algorithm MARKING will be to guarantee exact marginal probabilities for $\Pr[M_{i,t}]$. Indeed, the probabilities of Line 15 are chosen precisely so that the probability of i being marked at Line 15 but not in Line 11 is precisely $(1-p_{i,t})\cdot\frac{W_{i,t}-p_{i,t}}{1-p_{i,t}} = W_{i,t}-p_{i,t}$, implying the following.

Observation 5.3.2. *For each offline node $i \in L$, time t, and set of offline nodes $S \ni i$,*

$$\Pr[M_{i,t} \mid F_t = S] = [W_{i,t} \mid F_{i,t} = S] = c_{i,t}.$$

By total probability over all possible sets of free nodes $S \ni i$ by time t, the above observation yields the following corollary, giving a closed form for i's probability of being marked during time t (conditioned on it being free beforehand).

Corollary 5.3.3. *For any offline node i and time t with $(i,t) \in E$, we have*

$$\Pr[M_{i,t} \mid F_{i,t}] = c_{i,t}.$$

A simple induction on t, using Corollary 5.3.3 for the inductive step, yields the following closed form expressions for the probability of an edge being marked and consequently for an offline node i being free at time t.

Lemma 5.3.4. *For any offline node i and time t the following identities hold*

$$\Pr[M_{i,t}] = x_{i,t} \cdot (1-\varepsilon).$$

$$\Pr[F_{i,t}] = 1 - \sum_{t'<t} x_{i,t'} \cdot (1-\varepsilon).$$

Proof. We prove the second identity by induction on $t \geqslant 1$ and prove the first identity as a byproduct. The base case is trivial, since clearly $\Pr[F_{i,1}] = 1 = 1 - \sum_{t<1} x_{i,t} \cdot (1-\varepsilon)$. For the inductive step, we have by the inductive hypothesis that $\Pr[F_{i,t}] = 1 - \sum_{t'<t} x_{i,t'} \cdot (1-\varepsilon)$. Consequently, by Corollary 5.3.3 and by our choice of $c_{i,t} = \frac{x_{i,t}\cdot(1-\varepsilon)}{1-\sum_{t'<t} x_{i,t'}\cdot(1-\varepsilon)}$, we have

$$\begin{aligned}\Pr[M_{i,t}] &= \Pr[M_{i,t} \mid F_{i,t}] \cdot \Pr[F_{i,t}] \\ &= c_{i,t} \cdot \left(1 - \sum_{t'<t} x_{i,t'} \cdot (1-\varepsilon)\right) \\ &= x_{i,t} \cdot (1-\varepsilon).\end{aligned}$$

Therefore, by the inductive hypothesis and linearity of expectation, we have that

$$\begin{aligned}\Pr[F_{i,t+1}] &= \Pr[F_{i,t}] - \Pr[M_{i,t}] \\ &= 1 - \sum_{t'<t} x_{i,t'} \cdot (1-\varepsilon) - x_{i,t} \cdot (1-\varepsilon) \\ &= 1 - \sum_{t'<t+1} x_{i,t'} \cdot (1-\varepsilon).\end{aligned}$$ □

As stated in our outline of the intuition behind the analysis of Algorithm 4, Lemma 5.3.4 implies that for each edge (i,t), we have $\mathbb{E}[W_{i,t}] = x_{i,t} \cdot (1-\varepsilon)$. Consequently, $\sum_i \mathbb{E}[W_{i,t}] = \sum_i x_{i,t} \cdot (1-\varepsilon) \leqslant 1-\varepsilon$. Therefore, if we can argue that this sum is concentrated around its mean, we find that the condition of Line 12, whereby $\sum_i W_{i,t} > 1 \geqslant \mathbb{E}[\sum_i W_{i,t}] + \varepsilon$ is met infrequently, and therefore most nodes marked are also matched. Coupled with Lemma 5.3.4, this would imply the expected competitive ratio of our algorithm. For this, we now turn to proving the desired concentration, obtained by proving strong negative dependence between the variables $\{F_{i,t}\}_i$.

5.3.3 Negative Dependence of $\{F_{i,t}\}$

In this section we prove the key property of our algorithm, which asserts that for any set of offline nodes I at any time t is Negative Upper Orthant Dependant (NUOD).

Lemma 5.3.5. *For any set of offline nodes $I \subseteq L$ and any time t, we have*

$$\Pr\left[\bigwedge_{i\in I} F_{i,t}\right] \leqslant \prod_{i\in I} \Pr[F_{i,t}].$$

In order to prove this lemma, we will need to argue that the variables $M_{i,t}$ conditioned the set of free nodes at time t are themselves negatively correlated. Indeed, we will show that these conditional variables are *negatively associated (NA)* (see Section 2.4.1).

Lemma 5.3.6. *For any time t and set of offline nodes $I \subseteq L$ and set $I \subseteq S \subseteq L$, the variables $\{M_{i,t} \mid i \in I\}$ are NA conditioned on $F_t = S$.*

Proof. Fix the set of free nodes $F_t = S \supseteq I$. For any such S, we define the following random variables. For all $i \in I$, let A_i be an indicator variable for whether i was marked (and matched) due to bing picked in Line 8 at time t, and let B_i be independent Bernoulli variables with success probabilities $\frac{W_{i,t}-p_{i,t}}{1-p_{i,t}}$. Clearly, the A_i are binary variables with $\sum_{i\in I} A_i \leqslant 1$, so by the zero-one rule the variables $\{A_i \mid i \in I\}$ are NA. Furthermore, the variables $\{A_i \mid i \in I\}$ are independent, and as such are NA. Moreover, the joint distributions $\{A_i \mid i \in I\}$ and $\{B_i \mid k \in I\}$ are independent of each other and so, by closure of NA under distributions under disjoint union (Proposition 2.4.4.1), the joint distribution $\mathcal{D} = \{A_i, B_i \mid i \in I\}$ is NA. Finally, the set $\{M_{i,t}\}$ are the output of monotone increasing functions defined by disjoint subsets of a set of NA variables (as $M_{i,t} = A_i \vee B_i$, since i is marked either in Line 11 or in Line 15), and so the variables $M_{i,t}$ are NA, by closure of NA distributions under application of concordant functions of disjoint variables (Proposition 2.4.4.2). □

The above lemma gives us the following upper bound on all nodes in I not becoming marked at time t, conditioned on these nodes all being free at time t.

Corollary 5.3.7. *For any time t and set $I \subseteq L$, with $I \subseteq N(t)$, we have*

$$\Pr\left[\bigwedge_{i\in I} \overline{M_{i,t}} \,\middle|\, \bigwedge_{i\in I} F_{i,t}\right] \leqslant \prod_{i\in I}(1-c_{i,t}).$$

Proof. By Lemma 5.3.6, for all $S \supseteq I$ the variables $\{M_{i,t} \mid i \in I\}$ conditioned on $F_t = S$ are NA, and so by Corollary 2.4.8 they are negative orthant dependent. Together with Lemma 5.3.4, the above implies that

$$\Pr\left[\bigwedge_{i\in I} \overline{M_{i,t}} \,\middle|\, F_t = S\right] \leqslant \prod_{i\in I} \Pr\left[\overline{M_{i,t}} \mid F_t = S\right] = \prod_{i\in I}\left(1-c_{i,t}\right).$$

Taking total probability over all possible sets of free nodes $S \supseteq I$, the corollary follows. □

Given Lemma 5.3.4 and Corollary 5.3.7 we can now show this section's main result—that the indicators for being free for any set of offline nodes I at any time t is Negative Upper Orthant Dependant (NUOD), as stated in Lemma 5.3.5.

Proof of Lemma 5.3.5. We prove this lemma by induction on $t \geqslant 1$. First, clearly $\Pr[\bigwedge_{i\in I} F_{i,1}] = 1 = \prod_{i\in I} \Pr[F_{i,1}]$. Next, we assume the desired inequality $\Pr[\bigwedge_{i\in I} F_{i,t}] \leqslant \prod_{i\in I} \Pr[F_{i,t}]$ holds for t and prove that this implies the corresponding inequality for $t+1$. Let $J := I \cap N(t)$. For each $i \in I \setminus J$ we clearly have that $\Pr[F_{i,t+1}] = \Pr[F_{i,t}]$. On the other hand, by Corollary 5.3.3, for any $i \in J$ we have that $\Pr[F_{i,t+1}] = \Pr[\overline{M_{i,t}} \mid F_{i,t}] \cdot \Pr[F_{i,t}] = (1 - c_{i,t}) \cdot \Pr[F_{i,t}]$. Combining these bounds with the inductive step and Corollary 5.3.7, we obtain the desired inequality,

$$\begin{aligned}\Pr\left[\bigwedge_{i\in I} F_{i,t+1}\right] &= \Pr\left[\bigwedge_{i\in J} \overline{M_{i,t}} \,\middle|\, \bigwedge_{i\in I} F_{i,t}\right] \cdot \Pr\left[\bigwedge_{i\in I} F_{i,t}\right] \\ &\leqslant \prod_{i\in J} (1 - c_{i,t}) \cdot \prod_{i\in I} \Pr[F_{i,t}] \\ &= \prod_{i\in I} \Pr[F_{i,t+1}]. \qquad \square\end{aligned}$$

5.3.4 Analysis of Algorithm 4

In this section we finally turn to analyzing the performance of Algorithm 4. We recall that our key objective will be to prove that $\sum_i W_{i,t}$ is rarely higher than one, and so the test of Line 12 is rarely satisfied, which intuitively implies that most edges marked are in fact matched. To this end, we first bound the first moments of $\sum_i W_{i,t}$, and prove concentration of $\sum_i W_{i,t}$ around its mean,

Bounding the First Moments of $\sum_i W_{i,t}$: As a first step, we use Lemma 5.3.4 and Lemma 5.3.5 to obtain the following bounds on the expectation and variance of $\sum_i W_{i,t}$, which will prove useful in bounding the expected loss due to the random choices of Algorithm 4.

Lemma 5.3.8. *For each online node t, we have the following.*

$$\mathbb{E}\left[\sum_i W_{i,t}\right] = \sum_i x_{i,t} \cdot (1 - \varepsilon). \tag{5.1}$$

$$\operatorname{Var}\left(\sum_i W_{i,t}\right) \leqslant \sum_i \operatorname{Var}(W_{i,t}) \leqslant \sum_i \frac{x_{it}^2}{\Pr[F_{i,t}]} \leqslant \sum_i \frac{x_{it}^2}{\varepsilon}. \tag{5.2}$$

Proof. By Lemma 5.3.4 we have that $W_{i,t} = c_{i,t} \cdot F_{i,t} = \frac{x_{i,t}\cdot(1-\varepsilon)}{\Pr[F_{i,t}]} \cdot F_{i,t}$. Consequently, we have that $\mathbb{E}[W_{i,t}] = x_{i,t} \cdot (1 - \varepsilon)$, from which Equation (5.1) follows by linearity of expectation.

To prove Equation (5.2), we first note that the above expression for $W_{i,t}$ implies the following bound on its variance,

$$\mathrm{Var}(W_{i,t}) = \left(\frac{x_{i,t}\cdot(1-\varepsilon)}{\Pr[F_{i,t}]}\right)^2 \cdot \Pr[F_{i,t}]\cdot(1-\Pr[F_{i,t}]) \leqslant \frac{x_{i,t}^2}{\Pr[F_{i,t}]}.$$

By Lemma 5.3.5, if we restrict our attention to sets I of size $|I| = 2$, we find that the variables $\{F_{i,t} \mid i \in N(t)\}$ are pairwise negatively correlated, and thus so are their weighted counterparts, $\{W_{i,t} = c_{i,t}\cdot F_{i,t} \mid i \in N(t)\}$. So, by subadditivity of variance of pairwise negatively-correlated variables, we have that

$$\mathrm{Var}\left(\sum_i W_{i,t}\right) \leqslant \sum_i \mathrm{Var}(W_{i,t}) \leqslant \sum_i \frac{x_{i,t}^2}{\Pr[F_{i,t}]}.$$

The last inequality of Equation (5.2) follows from the fractional matching constraint of node i implying $\sum_{t'<t} x_{i,t'} \leqslant 1$, which implies by Lemma 5.3.4 that

$$\Pr[F_{i,t}] = 1 - \sum_{t'<t} x_{i,t'}\cdot(1-\varepsilon) \geqslant 1-(1-\varepsilon) = \varepsilon. \qquad \square$$

Recalling that we wish to prove that $\sum_i W_{i,t} \leqslant 1$ often, we now upper bound the probability of $\sum_i W_{i,t}$ deviating from its expectation, $\mathbb{E}[\sum_i W_{i,t}] (\leqslant 1-\varepsilon)$.

Concentration of $\sum_i W_{i,t}$: So far, we have bounded the mean and variance of $\sum_i W_{i,t}$. The following lemma asserts that $\sum_i W_{i,t}$ is sharply concentrated around its mean.

Lemma 5.3.9. *Let $k > 0$. For any online vertex t, let*

$$g(t,k) := \sqrt{\mathrm{Var}\left(\sum_i W_{i,t}\right)\cdot \log k + \left(\max_{i,t} x_{i,t}/3\varepsilon\right)\cdot \log k}. \tag{5.3}$$

Then, for any $c \geqslant \frac{1}{4}$ we have

$$\Pr\left[\sum_i W_{i,t} \geqslant \sum_i \mathbb{E}[W_{i,t}] + 4c\cdot g(t,k)\right] \leqslant 1/k^c.$$

Proof. By Lemma 5.3.5, the variables $\{F_{i,t} \mid i \in L\}$ are NUOD. Consequently, so are the variables $\{W_{i,t} = w_{i,t}\cdot F_{i,t} \mid i \in L\}$. It is well known that Chernoff-Hoeffding type bounds hold for the sums of NUOD scaled Bernoulli variables (see [230]). We may therefore apply Bernstein's Inequality, as stated in Lemma 2.4.12 for NA variables, to the sum of these $W_{i,t}$.

On the one hand, as the fractional matching constraint implies that $1-\sum_{t'<t} x_{i,t'}\cdot(1-\varepsilon) \leqslant \varepsilon$, we have for each i that

$$|W_{i,t}| \leqslant c_{i,t} = \frac{x_{i,t}\cdot(1-\varepsilon)}{1-\sum_{t'<t} x_{i,t'}\cdot(1-\varepsilon)} \leqslant \max_{i,t} x_{i,t}/\varepsilon.$$

On the other hand, by Lemma 5.3.8 we know that $\mathbb{E}[\sum_i W_{i,t}] = \sum_i x_{i,t} \cdot (1-\varepsilon)$. Plugging these values into Bernstein's Inequality, we have that for all $a > 0$,

$$\Pr\left[\sum_i W_{i,t} \geqslant \sum_i \mathbb{E}[W_{i,t}] + a\right] \leqslant \exp\left(\frac{-a^2}{2(\mathrm{Var}(\sum_i W_{i,t}) + a \cdot \max_{i,t} x_{i,t}/3\varepsilon)}\right). \tag{5.4}$$

Let $\hat{g} = 4 \cdot g(t,k)$, with $g(t,k)$ as defined in Equation (5.3). For this $\hat{g}$ we have the following.

$$\mathrm{Var}\left(\sum_i W_{i,t}\right) \leqslant \frac{g(t,k)^2}{\log k} = \frac{\hat{g}^2}{16c^2 \log k} \leqslant \frac{\hat{g}^2}{4c \log k}. \tag{5.5}$$

$$\frac{\max_{i,t} x_{i,t}}{3\varepsilon} \leqslant \frac{g(t,k)}{\log k} = \frac{\hat{g}}{4c \log k}. \tag{5.6}$$

Plugging inequalities (5.5) and (5.6) into Inequality (5.4) with $a = \hat{g}$, we conclude that

$$\Pr\left[\sum_i W_{i,t} \geqslant \sum_i \mathbb{E}[W_{i,t}] + \hat{g}\right] \leqslant \exp\left(\frac{-\hat{g}^2}{2(\hat{g}^2/4c\log k + \hat{g}^2/4c\log k)}\right) = \frac{1}{k^c}. \qquad \square$$

Equipped with Lemma 5.3.8 and Lemma 5.3.9, we now to bounding the overall, as well as per-edge and per-node, loss of the rounding of Algorithm 4. Put more positively, we are now ready to bound the gain of our algorithm.

Per-Edge Guarantees

We start by proving that Algorithm 4 run with an appropriately-chosen ε matches each edge e with probability roughly equal to the value x_e prescribed by the fractional matching $\vec{x}$.

Lemma 5.3.10. *Let $\mathcal{M}$ be the matching output by Algorithm 4 with $\varepsilon = \sqrt[3]{(\log\delta)/\delta}$ when run on graph $G = (L,R,E)$ and fractional matching $\vec{x}$ with $|x|_\infty \leqslant \delta$. Then, for each edge $e \in E$ there is some $\alpha_e \in [1,11]$ such that e is matched in $\mathcal{M}$ with probability*

$$\Pr[e \in \mathcal{M}] = x_e \cdot (1 - \alpha_e \cdot \sqrt[3]{(\log\delta)/\delta}).$$

The upper bound on the probability of any edge being matched is near immediate, as the probability of an edge $e = (i,t)$ to be matched is at most the probability it is marked, and hence, by Lemma 5.3.4, this probability is at most

$$\Pr[(i,t) \in \mathcal{M}] \leqslant \Pr[M_{i,t}] = x_e \cdot (1-\varepsilon) = x_e \cdot (1 - \sqrt[3]{(\log\delta)/\delta}). \tag{5.7}$$

Proving the complementary inequality, namely that each edge e is matched with probability at least $\Pr[e \in \mathcal{M}] \geqslant x_e \cdot (1 - 11\sqrt[3]{(\log\delta)/\delta})$, will require more work. We start with the following lower bound on $\Pr[e \in \mathcal{M}]$ in terms of $g(t,k)$ defined in Equation (5.3).

Lemma 5.3.11. *Let $\vec{x}$ be a fractional matching and $\hat{g} := \max_t 8 \cdot g(t, 1/\varepsilon)$, with $g(t,k)$ as defined in Equation (5.3). Then, Algorithm 4 run with parameter ε on input $\vec{x}$ matches each edge $(i,t) \in E$ with probability at least*

$$\Pr[(i,t) \in \mathcal{M}] \geqslant x_{i,t} \cdot (1 - 2\varepsilon - \hat{g}) .$$

Proof. Fix some edge (i,t). Denote by $A_t := \mathbb{1}[\sum_i W_{i,t} \geqslant \sum_i \mathbb{E}[W_{i,t}] + \hat{g}]$ the event that $\sum_i W_{i,t}$ exceeds its expectation by at least $\hat{g} = 8 \cdot g(t, 1/\varepsilon)$. By Lemma 5.3.9, we have that $\Pr[A_t] \leqslant \varepsilon^2$. But, by Lemma 5.3.4 and the fractional matching constraint, we also have that $\Pr[F_{i,t}] = 1 - \sum_{t'<t} x_{i,t'} \cdot (1-\varepsilon) \geqslant \varepsilon$. Combining both bounds we find that

$$\begin{aligned}
\Pr[F_{i,t} \wedge \overline{A_t}] &= \Pr[F_{i,t}] - \Pr[F_{i,t} \wedge A_t] \\
&\geqslant \Pr[F_{i,t}] - \Pr[A_t] \\
&\geqslant \Pr[F_{i,t}] - \varepsilon^2 \\
&\geqslant \Pr[F_{i,t}] \cdot (1-\varepsilon).
\end{aligned}$$

On the other hand, since $p_{i,t} = \frac{c_{i,t}}{\max\{1, \sum_i W_{i,t}\}}$, the probability of (i,t) being matched conditioned on i being free at time t and event $\overline{A_t}$, whereby $\sum_i W_{i,t} \leqslant \sum_i \mathbb{E}[W_{i,t}] + \hat{g} \leqslant 1 + \hat{g}$, is at least

$$\Pr[(i,t) \in \mathcal{M} \mid F_{i,t} \wedge \overline{A_t}] = \mathbb{E}\left[p_{i,t} \,\middle|\, F_{i,t} \wedge \overline{A_t}\right] \geqslant \frac{c_{i,t}}{1+\hat{g}} \geqslant c_{i,t} \cdot (1 - \hat{g}),$$

where the last inequality relied on $\hat{g} \geqslant 0$.

Recalling that by Observation 5.3.2 and Lemma 5.3.4, $c_{i,t} = \frac{x_{i,t} \cdot (1-\varepsilon)}{\Pr[F_{i,t}]}$, we find that for every edge $(i,t) \in E$ the probability of (i,t) being matched is indeed at least

$$\begin{aligned}
\Pr[(i,t) \in \mathcal{M}] &= \Pr[(i,t) \in \mathcal{M} \wedge F_{i,t}] \\
&\geqslant \Pr[(i,t) \in \mathcal{M} \wedge F_{i,t} \wedge \overline{A_t}] \\
&= \Pr[(i,t) \in \mathcal{M} \mid F_{i,t} \wedge \overline{A_t}] \cdot \Pr[F_{i,t}, \overline{A_t}] \\
&\geqslant c_{i,t} \cdot (1-\hat{g}) \cdot \Pr[F_{i,t}] \cdot (1-\varepsilon) \\
&\geqslant x_{i,t} \cdot (1-\varepsilon) \cdot (1-\hat{g}) \cdot (1-\varepsilon) \\
&\geqslant x_{i,t} \cdot (1 - 2\varepsilon - \hat{g}) .
\end{aligned}$$

□

Making a judicious choice of ε, we prove the following per-edge guarantees for our online dependent rounding scheme of Algorithm 4.

Lemma 5.3.12. *For all $\delta \in [0,1]$, let $\vec{x}$ be a fractional matching such that $\max_{i,t} x_{i,t} \leqslant \delta$. Then, Algorithm 4 with parameter $\varepsilon = \sqrt[3]{\delta \cdot \log(1/\delta)}$ run on $\vec{x}$ matches each edge $(i,t) \in E$ with probability at least*

$$\Pr[(i,t) \in \mathcal{M}] \geqslant x_{i,t} \cdot (1 - 11\varepsilon) = x_{i,t} \cdot \left(1 - 11\sqrt[3]{\delta \cdot \log(1/\delta)}\right) .$$

Proof. By Lemma 5.3.11, for $\hat{g} = 8 \cdot g(t, 1/\varepsilon)$ with $g(t, 1/\varepsilon)$ as defined in Equation (5.3), we have that

$$\Pr[(i,t) \in M] \geqslant 1 - 2\varepsilon - \hat{g}. \tag{5.8}$$

We therefore need to bound $\hat{g}$.

By Lemma 5.3.8 and the fractional matching constraint, we have that $\mathrm{Var}(W_t) \leqslant \sum_j x_{j,t}^2/\varepsilon \leqslant (\sum_j x_{j,t}) \cdot \delta/\varepsilon \leqslant \delta/\varepsilon$. Consequently, we have that $\hat{g} = 8 \cdot g(t, 1/\varepsilon)$ is at most

$$\begin{aligned}8 \cdot g(t, 1/\varepsilon) &= 8 \cdot \left(\sqrt{Var\left(\sum_i W_{i,t}\right) \cdot \log(1/\varepsilon)} + (\max_{j,t} x_{j,t}/3\varepsilon) \cdot \log(1/\varepsilon)\right) \\ &\leqslant 8 \cdot (\sqrt{(\delta/\varepsilon) \cdot \log(1/\varepsilon)} + (\delta/3\varepsilon) \cdot \log(1/\varepsilon)).\end{aligned}$$

To upper bound the above term, we first note that, as $\varepsilon = \sqrt[3]{\delta \cdot \log(1/\delta)}$, we have that

$$(\delta/\varepsilon) \cdot \log(1/\varepsilon) = \delta^{2/3} \cdot \frac{1}{\sqrt[3]{\log(1/\delta)}} \cdot \frac{1}{3} \cdot (\log(1/\delta) + \log\log(1/\delta)) \leqslant \frac{2}{3} \cdot (\delta \cdot \log(1/\delta))^{2/3}. \tag{5.9}$$

Therefore, letting $B := \frac{2}{3} \cdot (\delta \cdot \log(1/\delta))^{2/3}$, we have that

$$\hat{g} \leqslant 8 \cdot (\sqrt{B} + B/3).$$

Now, noting that $(\delta \cdot \log(1/\delta)) \leqslant 1/e$ for all $\delta \in [0,1]$, and therefore $B \leqslant 2/(3e)$, we have that $B/3 \leqslant \sqrt{B} \cdot \sqrt{2/(27e)}$. Consequently, we find that

$$\hat{g} \leqslant 8 \cdot (1 + \sqrt{2/27e})\sqrt{B} \leqslant 8 \cdot (1 + \sqrt{2/27e})\sqrt{2/3} \cdot (\delta \cdot \log(1/\delta))^{1/3} \leqslant 8 \cdot (\delta \cdot \log(1/\delta))^{1/3}.$$

Plugging the above bound into Equation (5.8), and using $\varepsilon = \sqrt[3]{\delta \cdot \log(1/\delta)}$, we find that

$$\Pr[(i,t) \in \mathcal{M}] \geqslant x_{i,t} \cdot (1 - 2\varepsilon - \hat{g}) \geqslant x_{i,t} \cdot (1 - 11\sqrt[3]{\delta \cdot \log(1/\delta)}).$$ □

Lemma 5.3.10 then follows by combining Lemma 5.3.12 and Equation (5.7).

Per-Vertex Guarantees

The per-edge guarantees of Lemma 5.3.10 immediately bounds on the probability of any vertex to be matched, by linearity of expectation. Here we give refined bounds on the per-vertex probabilities of being matched for online nodes.

Lemma 5.3.13. *For all online vertex t, the probability of t being matched is at least*

$$\Pr[t \text{ matched}] \geqslant \sum_i x_{i,t} \cdot (1 - \varepsilon) - \mathrm{Std}\left(\sum_i W_{i,t}\right).$$

Proof. The probability that t is matched is precisely

$$\Pr[t \in V(\mathcal{M})] = \mathbb{E}\left[\sum_i p_{i,t}\right] = \mathbb{E}\left[\min\{1, \sum_i W_{i,t}\}\right].$$

On the other hand, by Lemma 5.3.8, we know that $1 \geqslant \sum_i x_{i,t} \cdot (1-\varepsilon) = \sum_i \mathbb{E}[W_{i,t}]$, and therefore

$$\begin{aligned}
\Pr[t \in V(\mathcal{M})] &= \mathbb{E}\left[\min\left\{1, \sum_i W_{i,t}\right\}\right] \\
&\geqslant \mathbb{E}\left[\min\left\{\sum_i \mathbb{E}[W_{i,t}], \sum_i W_{i,t}\right\}\right] \\
&= \sum_i \mathbb{E}\left[W_{i,t}\right] + \mathbb{E}\left[\min\left\{0, \sum_i W_{i,t} - \sum_i \mathbb{E}\left[W_{i,t}\right]\right\}\right] \\
&\geqslant \sum_i x_{i,t} \cdot (1-\varepsilon) - \mathbb{E}\left[\left|\sum_i W_{i,t} - \sum_i \mathbb{E}\left[W_{i,t}\right]\right|\right] \\
&\geqslant \sum_i x_{i,t} \cdot (1-\varepsilon) - \mathrm{Std}\left(\sum_i W_{i,t}\right),
\end{aligned}$$

where the last inequality follows from Lemma 2.4.14, which asserts that the mean average deviation of any variable X being at most $\mathbb{E}[|X - \mathbb{E}[X]|] \leqslant \mathrm{Std}(X)$. □

Lemma 5.3.13 will prove useful when designing online matching algorithms for regular and near-regular graphs. Before addressing this application of our online dependent rounding scheme, we discuss the competitive ratios obtained by running Algorithm 4.

Global Guarantees

We now turn to outlining some more "global" guarantees of Algorithm 4, analyzing the size of the matching $\mathcal{M}$ output by this algorithm in expectation and with high probability.

First, Lemma 5.3.10 together with linearity of expectation immediately implies the following bound on the expected size of $\mathcal{M}$.

Corollary 5.3.14. *Let $\mathcal{M}$ be the matching output by running Algorithm 4 with $\varepsilon = \sqrt[3]{(\log \delta)/\delta}$ on graph $G = (L, R, E)$ and fractional matching $\vec{x}$ with $\delta = |x|_\infty$. Then, the matching $\mathcal{M}$ has expected size*

$$\Pr[|\mathcal{M}|] = \sum_e x_e \cdot (1 - \Theta(\sqrt[3]{\log \delta)/\delta}).$$

On the other hand, since the number of free nodes by the algorithm's termination is the sum of NUOD variables, by Lemma 5.3.5, we can rather directly show that $|M|$ is sharply concentrated around its mean, as follows.

Lemma 5.3.15. *W.h.p., the matching M output by Algorithm 4 satisfies*

$$|\mathcal{M}| = \mathbb{E}[|\mathcal{M}|] \pm O(\sqrt{n \log n}).$$

Proof. By definition, $|\mathcal{M}| = |L| - |F|$, for F the set of free vertices by the algorithm's termination. On the other hand, we have that $|F| = \sum_i F_{i,|R|+1}$ is the sum of $|L| \leqslant n$ NUOD binary variables, by Lemma 5.3.5. Therefore, Chernoff-Hoeffding bounds apply to this sum (see [230]), and in particular Lemma 2.4.11 implies that $F = \mathbb{E}[F] \pm O(n \log n)$ w.h.p. Since $|\mathcal{M}|$ is the difference of a constant term, $|L|$, and this random variable $|F|$, the lemma follows. □

5.4 Application to Near-Regular Graphs

In this section we explore some consequences of Theorem 5.1.3 to online matching in regular and near-regular graphs.

Recall that for such graphs, there exists a trivial 1-competitive online fractional matching algorithm, which assigns value $\frac{1}{\Delta}$ to each edge. This factional matching $\vec{x}$ has $|x|_\infty = \frac{1}{\Delta}$. Consequently, by Corollary 5.3.14, running Algorithm 4 with an appropriate choice of ε on this fractional matching immediately yields a $1 - \tilde{O}(\sqrt[3]{1/\Delta})$-competitive algorithm for regular graphs. We now discuss a refinement of this bound.

Theorem 5.4.1. *Running Algorithm 4 with $\varepsilon = O(\sqrt{(\log \Delta)/\Delta})$ on the fractional matching assigning $x_e = \frac{1}{\Delta}$ for each edge $e \in E$ yields a randomized online matching algorithm which is $1 - O(\sqrt{\log \Delta}/\sqrt{\Delta})$ competitive on Δ-regular graphs.*

Proof. Let the number of nodes on either side of the regular bipartite graph be $n = |L| = |R|$. By Lemma 5.3.13 and Lemma 5.3.8, the probability of any online node t to be matched is at least

$$\Pr[t \textit{ matched}] \geqslant \sum_i x_{i,t} \cdot (1-\varepsilon) - \mathrm{Std}\left(\sum_i W_{i,t}\right) = (1-\varepsilon) - \mathrm{Std}\left(\sum_i W_{i,t}\right).$$

Therefore, the resulting matching size is at least

$$\mathbb{E}[|\mathcal{M}|] \geqslant \sum_t \left((1-\varepsilon) - \mathrm{Std}\left(\sum_i W_{i,t}\right)\right). \tag{5.10}$$

We therefore wish to upper bound $\sum_t \mathrm{Std}\left(\sum_i W_{i,t}\right)$. To this end, we recall that by Lemma 5.3.8 and Lemma 5.3.4, we have that if t is the k-th online neighbor of i, then, for any positive ε,

$$\mathrm{Var}(W_{i,t}) \leqslant \frac{x_{i,t}^2}{\Pr[F_{it}]} = \frac{1}{\Delta^2} \cdot \frac{1}{1 - \sum_{t'<t}(1/\Delta)\cdot(1-\varepsilon)} \leqslant \frac{1}{(\Delta - k + 1)\cdot \Delta}.$$

Therefore, for any offline vertex i, summing over all the variances of edges (i,t), denoting by $H_\Delta = \sum_{k=1}^{\Delta} = \Theta(\log \Delta)$ the Δ-th harmonic number, we have that

$$\sum_t \mathrm{Var}(W_{i,t}) \leqslant \sum_{k=1}^{\Delta} \frac{1}{\Delta \cdot (\Delta - k + 1)} = \sum_{k=1}^{\Delta} \frac{1}{\Delta \cdot k} = \frac{H_\Delta}{\Delta}.$$

Summing over all edges (i,t), and recalling that $\mathrm{Var}(\sum_i W_{i,t}) \leqslant \sum_i \mathrm{Var}(W_{i,t})$ by Lemma 5.3.8, we have that

$$\sum_t \mathrm{Var}\left(\sum_i W_{i,t}\right) \leqslant \sum_t \sum_i \mathrm{Var}\left(W_{i,t}\right) \leqslant n \cdot \frac{H_\Delta}{\Delta}.$$

Put otherwise, over a uniformly random choice of online node t, we have that

$$\mathbb{E}_t\left[\mathrm{Var}\left(\sum_i W_{i,t}\right)\right] \leqslant \frac{H_\Delta}{\Delta}.$$

Therefore, by Jensen's Inequality applied to the concave function $f(x) = \sqrt{x}$, we find that

$$\mathbb{E}_t\left[\mathrm{Std}\left(\sum_i W_{i,t}\right)\right] = \mathbb{E}_t\left[\sqrt{\mathrm{Var}\left(\sum_i W_{i,t}\right)}\right] \leqslant \sqrt{\mathbb{E}_t\left[\mathrm{Var}\left(\sum_i W_{i,t}\right)\right]} \leqslant \sqrt{\frac{H_\Delta}{\Delta}}.$$

Plugging in the obtained bound on the expected standard deviation of $\sum_i W_{i,t}$ into Equation (5.10), we find that the expected matching's size is at least

$$\begin{aligned}\mathbb{E}[|\mathcal{M}|] &\geqslant \sum_t \left((1-\varepsilon) - \mathrm{Std}\left(\sum_i W_{i,t}\right)\right) \\ &\geqslant n \cdot \left(1 - \varepsilon - \frac{H_\Delta}{\Delta}\right) \\ &= n \cdot (1 - O(\sqrt{(\log \Delta)/\Delta}).\end{aligned}$$

As the optimum matching size is trivially no more than n, the theorem follows. $\square$

A simple extension of the above result to *near-regular* graphs, is obtained by considering the fractional matching assigning values $\frac{1}{\Delta + c\cdot(\sqrt{\Delta})}$ and running Algorithm 4 with $\varepsilon = \Theta(\sqrt{(\log \Delta)/\Delta}$, and generalizing the above proof.

Theorem 5.4.2. *For any $c \geqslant 0$, running Algorithm 4 with $\varepsilon = \Theta(\sqrt{(\log \Delta)/\Delta})$ on the fractional matching assigning $x_e = \frac{1}{\Delta + c\cdot\sqrt{\Delta}}$ for each edge $e \in E$ yields a randomized online matching algorithm which is $1 - O(\sqrt{\log \Delta}/\sqrt{\Delta})$ competitive on graphs with degrees in the range $[\Delta - c\cdot(\sqrt{\Delta}), \Delta + c \cdot (\sqrt{\Delta})]$ is $1 - O(\sqrt{\log \Delta}/\sqrt{\Delta})$-competitive.*

Before concluding, we note that a competitive ratio essentially equal to that of Theorem 5.4.1 and Theorem 5.4.2 can be shown to hold w.h.p. In particular, running Algorithm 4 with $\varepsilon = O(\sqrt{(\log n)/\Delta})$ yields a $1-O((\log n)/\sqrt{\Delta})$-competitive matching for regular (and near-regular) graphs, with high probability.

5.5 Conclusion and Open Questions

In this chapter, we presented optimal randomized online matching algorithms for Δ-regular graphs, showing that for such graphs the optimal competitive ratio is $1 - \tilde{\Theta}(1/\sqrt{\Delta})$, and that a similar bound can be achieved w.h.p. Beyond a pleasing resolution to the question of the optimal competitive ratio for online matching in this widely-studied graph classes, this chapter suggests that randomized (dependent) rounding presents both new challenges in online settings, as well as possible algorithmic opportunities for such settings. In the next chapter, we further substantiate the applicability of our online dependent rounding scheme, using it to obtain optimal online bipartite edge coloring algorithms. We conjecture that this is not the last such application of our rounding scheme and its ilk.

Chapter 6

Online Bipartite Edge Coloring

In this chapter, based on [70] (joint work with Ilan R. Cohen and Binghui Peng), we move from matchings to the study of *edge colorings* in bipartite graphs, under the one-sided vertex arrival model studied by Karp et al. [179] for online matching. For this problem, we present optimal algorithms and matching lower bounds. Along the way, we prove a dichotomy for this problem under known and unknown maximum degree, proving that the latter scenario is strictly harder, though here, too, the greedy algorithm is suboptimal. We obtain our algorithmic results in part by leveraging our online dependent rounding scheme of Chapter 5, which we show is useful in the context of online edge coloring, in addition to online matching.

6.1 Background

Edge coloring is the problem of assigning colors edges of a multigraph so that no two edges with a common endpoint have the same color. This classic problem, even restricted to bipartite graphs, can be used to model scheduling problems arising in sensor networks [121], switch routing [5], radio-hop networks [257] and optical networks [241], among others. The edge coloring problem can trace its origins back to the 19th-century works of Tait [255] and Petersen [234], who studied this problem in the context of the four color theorem. König [184] showed that any simple bipartite graph is can be edge colored using Δ colors. (Clearly, no fewer colors suffice.) Shannon [249] later studied edge coloring in the context of color coding wires in electrical units, and proved that any multigraph G of maximum degree $\Delta = \Delta(G)$ admits a $\lfloor \frac{3\Delta}{2} \rfloor$-edge-coloring; i.e., a coloring using at most $\lfloor \frac{3\Delta}{2} \rfloor$ colors. (This is tight.) Inspired by this result, Vizing [261] proved that any simple graph can be edge colored using $\Delta + 1$ colors.

On the algorithmic front, for bipartite graphs, which can always be colored with Δ colors (and no fewer), near-linear-time Δ-edge-coloring algorithms are known [10, 72, 133]. For general graphs, polytime $(\Delta + 1)$-edge-coloring algorithms are known [118, 215, 261], and this too is likely optimal, as determining whether a general graph is Δ-edge-colorable is NP-hard [160]. Besides these optimal polytime algorithms, a folklore quasilinear-time greedy algorithm, which colors each edge with the lowest color unused by its adjacent edges, is known to output a $(2\Delta - 1)$-edge-coloring. The greedy algorithm is implementable in many restricted models of computation, and improving upon its coloring guarantees, or even matching them quickly in such models, has been the subject of intense research. Examples include PRAM [196], NC and RNC [32, 177, 219], dynamic [46, 82, 266] and distributed algorithms (e.g., [61, 84, 92, 113, 129, 231]). In this chapter, we study bipartite edge coloring in an *online* model.

In the online bipartite edge coloring problem, a bipartite graph $G = (L, R, E)$ is presented in an online fashion. In particular, nodes of the offline side, L are known a priori, while nodes of the online side, R, are revealed one at a time, together with their edges. An online (bipartite) edge-coloring algorithm must decide, immediately and irrevocably, which color to assign to each revealed edge, before the arrival of the next online node. One simple such online edge coloring algorithm is the greedy algorithm, which assigns, for all edges of an arriving vertex in some arbitrary order, the lowest color which is unused by edges of both endpoints. This algorithm uses at most $2\Delta - 1$ colors on any graph of maximum degree Δ. On the other hand, each bipartite graph can be colored using Δ colors. We say an online edge-coloring algorithm $\mathcal{A}$ has competitive ratio α if the number of colors it uses is at most $\alpha \cdot \Delta$ for each graph of maximum degree Δ. We say the algorithm has competitive ratio α with high probability if this guarantee holds with probability $1 - 1/\operatorname{poly}(n)$. So, for example, the naïve greedy algorithm is $(2\Delta - 1)/\Delta \approx 2$-competitive. Do better algorithms exist?

6.1.1 Bad News?

As it turns out, the greedy algorithm is in a sense the best possible. Indeed, as shown by Bar-Noy et al. [25] in a note titled simply "the greedy algorithm is optimal for on-line edge coloring", no online algorithm for edge coloring uses fewer than $2\Delta - 1$ colors used by the greedy algorithm. That is, Bar-Noy et al. show that no algorithm can even save *one* color compared to this bound. It therefore might seem that online edge coloring is not a particularly fruitful problem to study. Nonetheless, as we shall see, there is room for improvement. To motivate the above, we briefly outline a proof of the lower bound of Bar-Noy et al.

Theorem 6.1.1 ([25]). *There exists no online bipartite $(2\Delta - 2)$-edge-coloring algorithm.*

For simplicity, we only outline the proof of this lower bound for deterministic algorithms, deferring a generalization of the lower bound of [25] for randomized algorithms to Chapter 9.

Proof. Let $\mathcal{A}$ be a deterministic edge-coloring algorithm. Consider a bipartite graph consisting of $K = \Delta \cdot \binom{2\Delta-2}{\Delta-1}$ stars of degree $\Delta - 1$, with the centers of these stars as offline nodes, and the leaves of these stars arriving before any other online nodes. Suppose algorithm $\mathcal{A}$ uses some $2\Delta - 2$ or fewer colors when coloring these stars (if it uses more, we are done). By the pigeonhole principle, as there are only $\binom{2\Delta-2}{\Delta-1}$ possible choices of colors $\mathcal{A}$ for the $\Delta - 1$ edges of each star, some $K/\binom{2\Delta-2}{\Delta-1} = \Delta$ stars must have their $\Delta - 1$ edges colored using the same set S of $\Delta - 1$ colors. Finally, a new online node v arrives, neighboring the centers of these Δ stars which use the same colors. All Δ edges of v must use unique colors, and each such color cannot belong to S, as these edges' other endpoint already has edges colored using these colors. Therefore, algorithm $\mathcal{A}$ uses at least $2\Delta - 1$ distinct colors on this graph. □

6.1.2 Not Such Bad News?

The careful reader might note that the above lower bound required the number of nodes n to be exponential in the maximum degree. Specifically, it requires $n \geqslant \Delta^2 \cdot \binom{2\Delta-2}{\Delta-1} = \Omega(4^\Delta)$. Taking logarithms, this exponential lower bound on n translates into a logarithmic upper bound on the maximum degree, $\Delta = O(\log n)$. Put otherwise, the impossibility result of [25] should not be read as a multiplicative lower bound of 2 (on the competitive ratio), but rather as an *additive* lower bound of $\Omega(\log n)$. That is, the $2\Delta - 2$ colors in the above theorem should be interpreted instead as a lower bound of $\Delta + \Omega(\log n)$ on the number of colors used by any algorithm. This motivates the study of improved edge-coloring algorithms for large $\Delta = \omega(\log n)$. Indeed, this observation was not lost on Bar-Noy et al. [25], who conjectured a near-*ideal* online edge-coloring algorithm for such large Δ.

Conjecture 6.1.2 ([25]). *There exists an online $(1 + o(1))$-competitive edge-coloring algorithm for graphs of maximum degree $\Delta = \omega(\log n)$.*

Bar-Noy et al. [25] conjectured the above in a stricter model than the we study here, namely the adversarial *edge-arrival* model. As can be deduced from chapters 3 and 4, this level of granularity of arrivals is often harder than vertex arrivals. On the other hand, the lower bound of Theorem 6.1.1 holds in our bipartite one-sided vertex arrival model, too. It is therefore natural to ask whether Conjecture 6.1.2 holds for this model.

6.1.3 Our Contributions

In this chapter, we address bipartite the edge coloring problem in the high-degree régime, presenting optimal algorithms for this problem.

Our first result is a resolution of Conjecture 6.1.2 for bipartite edge coloring.

Theorem 6.1.3. *There exits an online $(1+o(1))$-competitive bipartite edge-coloring algorithm for graphs of maximum degree $\Delta = \omega(\log n)$ revealed via one-side vertex arrivals.*

We also show that this is optimal, up to the exact $o(1)$ term in the competitive ratio.

Next, we show a (perhaps surprising) dichotomy for this problem, between the case where the algorithm knows Δ a priori, and when Δ is unknown. In particular, we show that not knowing Δ makes the problem strictly harder, ruling out a competitive ratio below a constant bounded away from one, even if Δ is large.

Theorem 6.1.4. *There exists no $\left(\frac{e}{e-1} - \Omega(1)\right)$-competitive edge coloring algorithm for graphs of* unknown *maximum degree Δ.*

On the other hand, we show that, while the problem becomes strictly harder when Δ is unknown, the greedy algorithm is still suboptimal if Δ is large. In particular, we present an algorithm for unknown Δ which matches the above lower bound, up to lower-order terms.

Theorem 6.1.5. *There exits an online* $\left(\frac{e}{e-1}+o(1)\right)$*-competitive bipartite edge-coloring algorithm for graphs of* unknown *maximum degree* $\Delta = \omega(\log n)$ *revealed via one-side vertex arrivals.*

6.1.4 Related Work

Several previous works studied edge coloring in online settings [5, 22, 25, 90, 100, 101, 211, 212]. Due to the strong lower bounds given by [25], these works mostly focus on relaxations of the problem. Mikkelsen [211, 212] studied the online edge coloring problem, with advice about the future. Favrholdt et al. [90, 100, 101] studied the "dual" problem of maximizing the number of edges colored using a fixed number of colors. Most relevant to this chapter is the work of Motwani et al. [5, 22, 25], including the aforementioned lower bound of [25]. Aggarwal et al. [5] presented a $(1+o(1))$-competitive algorithm for multigraphs with known $\Delta = \omega(n^2)$. Bahmani et al. [22], inspired by the distributed algorithm of Panconesi and Srinivasan [231], gave a 1.26-competitive algorithm for multigraphs with known $\Delta = \omega(\log n)$. Both algorithms require *random order* edge arrivals, and fall short of the guarantees of those conjectured by Bar-Noy et al. [25], either in the competitive ratio or in the requirement of Δ. In contrast, in this chapter we consider *vertex* arrivals, but under the stricter *adversarial* arrival order, for which we match these conjectured bounds for known Δ, and also achieve optimal bounds for (harder) unknown Δ.

6.2 Known Δ

In this section we present a $(1+o(1))$-competitive algorithm for known large $\Delta = \omega(\log n)$, and show this is best possible for any Δ, up to the exact $o(1)$ term.

6.2.1 Our Algorithmic Approach

We start by presenting our approach in an offline setting. Iterating over $c \in [\Delta]$, we compute and color a matching M_c in the uncolored subgraph $G \setminus \bigcup_{c'=1}^{c-1} M_{c'}$. We then color the remaining uncolored subgraph with new colors using the greedy algorithm. This approach can be implemented online, by iteratively running online matching algorithms on the relevant uncolored subgraphs to compute and color matchings. More concretely, when a vertex v arrives, we iterate over $c \in [\Delta]$ and update M_c in the current uncolored graph $G \setminus \bigcup_{c'=1}^{c-1} M_{c'}$, as follows. We run the next step of the online matching algorithm used to compute M_c in the current uncolored graph after v's arrival in this subgraph. We then color v's newly-matched edge (if any) using color c. Finally, we run steps of the greedy algorithm on the remaining uncolored edges of v.

For our analysis, we will analyze the above algorithm according to its offline description. Since the greedy algorithm requires a number of colors linear in its input graph's maximum degree, our objective will be to reduce the uncolored subgraph's maximum degree to $o(\Delta)$ w.h.p. after computing and coloring the first Δ matchings. In particular, this will require us to match each maximum-degree vertex in G with probability roughly one for each of these Δ matchings. One way of matching vertices v of degree Δ in the uncolored subgraph with probability roughly one is to guarantee each edge $e \ni v$ a probability of roughly $\frac{1}{\Delta}$ of being matched. An online matching algorithm which does just this is obtained from our online dependent rounding scheme of Lemma 5.3.10, applied to the trivial fractional matching which assigns a value of $\frac{1}{d}$ to each edge in graphs of maximum degree d. We will refer by MARKING_d to the obtained online matching algorithm for graphs of (known) maximum degree at most d.

Corollary 6.2.1. *Algorithm* MARKING_d *is an online matching algorithm which in graphs of maximum degree at most d outputs a matching $\mathcal{M}$ which matches each edge e with probability*

$$\frac{1}{d} \cdot \left(1 - 11\sqrt[3]{(\log d)/d}\right) \leqslant \Pr[e \in \mathcal{M}] \leqslant \frac{1}{d}.$$

The first natural approach given Corollary 6.2.1 is to iteratively run MARKING_Δ on the uncolored subgraph. However, this approach can be shown to be suboptimal, as the probability of coloring an edge of a maximum-degree node decreases as this node's degree decreases in the uncolored subgraph. Instead, we will increase the probability of high-degree vertices in the uncolored subgraph to have an edge colored, by running MARKING_d with a tighter upper bound d than Δ for the uncolored graph's maximum degree for each phase. Unfortunately, upon arrival of some vertex v, we do not know the uncolored graph's maximum degree for all phases, as this depends on *future* arrivals and random choices of our algorithm. To obtain a tight (up to $o(\Delta)$) bound d on the uncolored graph's maximum degree for each phase, we divide the Δ coloring iterations into *phases* of $\ell = \sqrt{\Delta \log n}$ iterations each, during which we use the same upper bound. As $\ell = o(\Delta)$ and $\ell = \omega(\log n)$, this gives us sharply concentrated upper bounds d_{i+1} on the resulting uncolored graph's maximum degree at the end of each phase i, which in turn serves as a tight upper bound for the next phase. This results in the desired rate of decrease in the uncolored graph's maximum degree, namely $1 - o(1)$ per iteration. Greedy thus runs on a subgraph of maximum degree $o(\Delta)$. Our $1 + o(1)$ competitive ratio follows.

6.2.2 The Algorithm

We now present our online edge coloring algorithm, starting with an offline description. Our algorithm consists of Δ iterations, equally divided into $\sqrt{\Delta/\log n}$ phases. During each iteration of phase i, we color a matching output by MARKING_{d_i} run on U_i – the uncolored subgraph prior to phase i, for $d_i := \Delta - i \cdot (\ell - 8\sqrt{\ell \log n})$. After all phases, we run greedy with new colors, starting with $\Delta + 1$. In the online implementation, after each online vertex v's arrival, for phase $i = 1, 2, \ldots$, we run the next step of $\ell = \sqrt{\Delta \log n}$ independent runs of MARKING_{d_i} in U_i, color newly-matched edges and update $U_{i'}$ for $i' > i$ accordingly. We then greedily color v's remaining uncolored edges with new colors. The algorithm's pesudocode is given in Algorithm 5.

Algorithm 5 Improved Algorithm for Known Δ

```
Input: Online bipartite graph $G(L, R, E)$ with maximum degree $\Delta = \omega(\log n)$
Output: Integral $(1 + o(1))\Delta$ edge coloring, w.h.p.
1:  let $\ell := \lfloor \sqrt{\Delta \log n} \rfloor$                                  ▷ phase length
2:  let $d_i := \Delta - i \cdot (\ell - 8\sqrt{\ell \log n})$ for $i \in [0, \Delta/\ell]$    ▷ degree upper bound for each phase
3:  for all $i$, denote by $U_i$ the online subgraph of $G$ not colored by colors $[i \cdot \ell]$
4:  for each arrival of a vertex $v \in R$ do
5:      for phase $i = 0, 1, \ldots, \lfloor \Delta/\ell \rfloor - 1$ do
6:          for colors $c \in [i \cdot \ell + 1, (i + 1) \cdot \ell]$ do
7:              $M_c \leftarrow$ output of copy $c$ of MARKING$_{d_i}$ on current $U_i$  ▷ next step of MARKING$_{d_i}$
8:              if some $e \in M_c$ is previously uncolored then
9:                  color $e$ using color $c$                                   ▷ note: $e \ni v$
10:     run greedy on all uncolored edges of $v$, using new colors starting from $\Delta + 1$
```

6.2.3 Analysis

The crux of our analysis is that for each phase i, we have $d_i \geqslant \Delta(U_i)$ w.h.p. Consequently, the final uncolored subgraph after the Δ iterations (and colors) has maximum degree at most $d_{\lfloor \Delta/\ell \rfloor} = o(\Delta)$, so greedily coloring this subgraph requires a further $o(\Delta)$ colors. The following lemma asserts that if $d_i \geqslant \Delta(U_i)$, then $d_{i+1} \geqslant \Delta(U_{i+1})$, w.h.p.

Lemma 6.2.2. *For all $i \in [0, \Delta/\ell - 1]$, if $\Delta(U_i) \leqslant d_i$, then*

$$\Pr[\Delta(U_{i+1}) > d_{i+1}] \leqslant 1/n^3.$$

Proof. If $\Delta(U_i) \leqslant d_i - \ell$, the claim is trivial, as then $d_{i+1} \geqslant d_i - \ell \geqslant \Delta(U_i) \geqslant \Delta(U_{i+1})$. We therefore focus on the case $d_i - \ell \leqslant \Delta(U_i) \leqslant d_i$. For this latter case, we will rely on the fact that for all $i \leqslant \Delta/\ell$, we have $d_i = \Delta - i \cdot (\ell - 8\sqrt{\ell \log n}) \geqslant (\Delta/\ell) \cdot 8\sqrt{\ell \log n} > 3\Delta^{3/4} \log^{1/4} n$.

Vertices of degree less than $\Delta(U_i) - \ell \leqslant d_i - \ell < d_{i+1}$ clearly have degree at most d_{i+1} in U_{i+1}, as we only decrease their degree in the uncolored subgraph over time. We therefore turn our attention to vertices v of degree at least $\Delta(U_i) - \ell$ in U_i. Such a vertex v cannot have more than ℓ edges colored during phase i, regardless of our algorithm's random choices, as at most one of v's edges is colored per iteration. So, before each color c used in the phase, v has at least $\Delta(U_i) - 2\ell \geqslant d_i - 3\ell$ uncolored edges, each of which is matched by MARKING$_{d_i}$ with probability at least $\frac{1}{d_i}(1 - 11\sqrt[3]{(\log d_i)/d_i})$. Therefore, if we let $X_c := \mathbb{1}[\bigvee_{e \ni v} \text{e colored } c]$ be an indicator for the event that v has an edge colored c, then, regardless of the realization $\vec{x}$ of variables $X_{c-1}, X_{c-2}, \ldots, X_{(i-1)\cdot \ell+1}$ corresponding to previous iterations of the i^{th} phase, vertex v will have an edge colored c with probability at least

$$\begin{aligned}\Pr[X_c = 1 \mid (X_{c-1}, X_{c-2}, \dots, X_{(i-1)\cdot\ell+1}) = \vec{x}] &\geqslant (d_i - 3\ell)\cdot\frac{1}{d_i}\left(1 - 11\sqrt[3]{(\log d)/d}\right)\\ &\geqslant 1 - \frac{3\ell}{d_i} - 11\sqrt[3]{(\log d_i)/d_i}\\ &\geqslant 1 - \left(\frac{3}{8} + \frac{11}{2}\right)\sqrt[4]{(\log n)/\Delta}\\ &\geqslant 1 - 6\sqrt[4]{(\log n)/\Delta},\end{aligned}$$

where the penultimate inequality follows from $\ell \leqslant \sqrt{\Delta \log n}$ and $n \geqslant d_i \geqslant 8\Delta^{3/4}\log^{1/4} n$.

Therefore, the expected decrease of v's degree in the uncolored graph during the ℓ iterations of the i^{th} phase is at least $\mathbb{E}\left[\sum_{c=i\cdot\ell+1}^{(i+1)\cdot\ell} X_c\right] \geqslant \ell\cdot\left(1 - 6\sqrt[4]{(\log n)/\Delta}\right) \geqslant \ell - 6\sqrt{\ell\log n}$. But the probability of v having an edge colored c is at least $1 - 6\sqrt[4]{(\log n)/\Delta}$ *independently* of previous colors during the phase. Consequently, we can appeal to standard coupling arguments (Proposition 2.4.15) together with Hoeffding's inequality (Lemma 2.4.11) to show that the sum of these ℓ binary variables satisfies

$$\Pr\left[\sum_{c=i\cdot\ell+1}^{(i+1)\cdot\ell} X_c \leqslant \ell - 6\sqrt{\ell\log n} - \sqrt{2\ell\log n}\right] \leqslant \exp\left(-\frac{2(\sqrt{2\ell\log n})^2}{\ell}\right) = 1/n^4.$$

Put otherwise, v's degree in the uncolored subgraph decreases during phase i by less than $\ell - 6\sqrt{\ell\log n} - \sqrt{2\ell\cdot\log n} > \ell - 8\sqrt{\ell\log n}$ with probability at most $1/n^4$. Thus, as v has degree at most $\Delta(U_i)$ in U_i by definition, we find that vertex v's degree in U_{i+1}, denoted by D_v, satisfies

$$\Pr[D_v \geqslant d_{i+1}] = \Pr[D_v \geqslant d_i - \ell + 8\sqrt{\ell\log n}] \leqslant \Pr[D_v \geqslant \Delta(U_i) - \ell + 8\sqrt{\ell\log n}] \leqslant 1/n^4.$$

Taking union bound over all vertices, the lemma follows. □

The above lemma implies this section's main result, given by the following theorem.

Theorem 6.2.3. *Algorithm 5 is $(1 + O(\sqrt[4]{(\log n)/\Delta}))$-competitive w.h.p. in n-vertex bipartite graphs with known maximum degree $\Delta = \Omega(\log n)$.*

Proof. Algorithm 5 computes a feasible edge coloring. It colors each edge, by Line 12, and each color class – computed during iterations or by greedy – constitutes a matching (here we rely on the colors used by greedy and the phases being disjoint). It remains to bound the number of colors this algorithm uses. Each phase requires at most ℓ colors, so the phases require at most Δ colors. The number of colors the greedy step requires is at most twice the maximum degree of the remaining uncolored subgraph after the phases, which we now bound.

Let $A_i := \mathbb{1}[\Delta(U_i) \leqslant d_i]$ be an indicator for the event that d_i upper bounds $\Delta(U_i)$. By Lemma 6.2.2 we have that $\Pr[\overline{A_i} \mid A_{i-1}, A_{i-2}, \dots] = \Pr[\overline{A_i} \mid A_{i-1}] \leqslant 1/n^3$. Also, trivially

$\Pr[\overline{A_0}] = 0$. By the conditional union bound (Proposition 2.4.16) over all i, we find that the probability of any A_i not being one is at most

$$\Pr\left[\bigvee_{i=0}^{\Delta/\ell} \overline{A_i}\right] \leqslant (\Delta/\ell)/n^3 \leqslant 1/n^2.$$

Consequently, all applications of $\text{MARKING}_{d_{i+1}}$ during phase $i+1$ match each edge of U_{i+1} with probability at least $\frac{1}{d_{i+1}}(1-11\sqrt[3]{(\log d_{i+1})/d_{i+1}})$, as required by our analysis for phase $i+1$. Moreover, $d_i \geqslant \Delta(U_i)$ for all $i \in [0, \Delta/\ell]$ w.h.p. implies that the uncolored subgraph following the Δ/ℓ phases has maximum degree at most

$$\begin{aligned} d_{\lfloor \Delta/\ell \rfloor} &= \Delta - \lfloor \Delta/\ell \rfloor \cdot (\ell - 8\sqrt{\ell \log n}) \\ &\leqslant \ell + (\Delta/\ell) \cdot 8\sqrt{\ell \log n} \\ &\leqslant \Delta^{1/2} \log^{1/2} n + 8\Delta^{3/4} \log^{1/4} n \\ &\leqslant 9\Delta^{3/4} \log^{1/4} n. \end{aligned}$$

The greedy algorithm therefore colors the remaining uncolored graph using at most a further $18\Delta^{3/4}\log^{1/4} n - 1$ colors. That is, it uses $\Delta \cdot O(\sqrt[4]{(\log n)/\Delta})$ colors in the range $\Delta+1, \Delta+2, \ldots$. The theorem follows, including the stated bound for $\Delta = \omega(\log n)$. ☐

Remark: Algorithm 5 is $(1 + O(\sqrt[4]{(\log n)/\Delta}))$ competitive w.h.p. for all $\Delta = \Omega(\log n)$ sufficiently large, and so for any $\Delta = \Omega(\log n)$ sufficiently large, it yields a constant competitive ratio strictly smaller than 2. For $\Delta = \omega(\log n)$, this algorithm achieves competitive ratio $(1 + o(1))$.

6.2.4 A Matching Lower Bound

In the preceding section, we gave a $(1 + o(1))$-competitive algorithm for large known Δ. Before proceeding to the harder régime of *unknown* Δ, we briefly note that knowledge of Δ alone (even when large) does not allow for arbitrarily-good competitive ratio. That is, we show that our algorithm's competitive ratio of $(1 + o(1))$ for known Δ is optimal (up to the exact $o(1)$ term).

Observation 6.2.4. *No online edge coloring algorithm is $(1 + o(1/\sqrt{\Delta}))$-competitive.*

Proof. Let $\mathcal{A}$ be a $(1+\varepsilon)$-competitive edge coloring algorithm. Consider an online matching algorithm $\mathcal{A}'$ which on any Δ-regular $2n$-node graph, runs $\mathcal{A}$ and randomly picks one of the $(1+\varepsilon)\cdot\Delta$ color classes upon initialization as its output matching. This online matching algorithm's output matching has expected size $\frac{\Delta \cdot n}{(1+\varepsilon)\cdot\Delta} \geqslant (1-\varepsilon)\cdot n$ on Δ-regular graphs. But by Theorem 5.1.1, and $2n$-node bipartite regular graphs having maximum matching size n [184], we have that no online matching algorithm outputs a matching of expected size $(1 - o(1/\sqrt{\Delta}))\cdot n$ in Δ-regular bipartite graphs under one-sided vertex arrivals. We conclude that $\varepsilon = \Omega(1/\sqrt{\Delta})$. ☐

6.3 Unknown Δ

In the previous section we established that having a large (known) maximum degree $\Delta = \omega(\log n)$ allows for $(1 + o(1))$-competitive algorithms. In this section we show that not knowing Δ makes the problem significantly harder, and no algorithm, even a fractional one (with regards to a relaxation we define shortly), is better than $\frac{e}{e-1}$-competitive. We then present a fractional algorithm which matches this bound. Finally, by repeatedly rounding (parts of copies of) solutions of this fractional algorithm, we obtain optimal randomized algorithms, with competitive ratio $\left(\frac{e}{e-1} + o(1)\right)$. We start by discussing the fractional relaxation we use for our results for unknown Δ.

6.3.1 Our Fractional Relaxation

In this section, we define the online fractional edge coloring relaxation we study and discuss several of its properties.

The Classic Fractional Relaxation: The classic relaxation for edge coloring has a nonnegative variable x_M for each matching M in $G = (V, E)$, corresponding to the (fractional) extent to which this matching is used in the solution. The objective is to minimize $\sum_M x_M$ subject to $\sum_{M \ni e} x_M = 1$ for each edge $e \in E$. This relaxation clearly lower bounds the chromatic index; i.e., the minimum number of matchings needed to cover G. A long-standing conjecture of Goldberg and Seymour is that this relaxation is at most one lower than the chromatic index [136, 248]. (See [200, Chapter 7.4] for more discussion of this relaxation.) Unfortunately, this relaxation seems somewhat unwieldy in an online setting, as we outline below.

The Relaxation We Study: The standard fractional edge coloring relaxation is difficult to use in online settings, where we do not know the edges which will arrive in the future, let alone which matchings G will contain. This motivates us to study a more "myopic" relaxation, which allows us to make our (fractional) assignments immediately upon an edge's arrival (due to one of its endpoints' arrival). Specifically, rather than relax the integrality of the extent to which we use integral matchings, we relax the integrality of the matchings used. That is, while the classic relaxation fractionally uses integral matchings to color edges, our relaxation *integrally* uses *fractional* matchings to color edges. As we will see, a useful property of this relaxation is that it allows us to rely on machinery for rounding fractional matchings online.

The edge coloring relaxation we consider is thus the following. We say a graph $G(V, E)$ is *fractionally k-edge-colorable* if there is a feasible solution to the following linear program.

$$\begin{aligned} \sum_{c \in [k]} x_{e,c} &= 1 && \forall e \in E \\ \sum_{e \ni v} x_{e,c} &\leqslant 1 && \forall v \in V, c \in [k] \\ x_{e,c} &\geqslant 0 && \forall e \in E, c \in [k] \end{aligned}$$

For any graph G, the minimal number of fractional colors k is equal to G's maximum degree, Δ. We note that in bipartite graphs this relaxation and the classic relaxation are equivalent in an offline sense, in that any solution to one can be transformed to a solution of equal value to the other (for general graphs, there can be a gap of one between the two, as exemplified by the triangle graph). In an online sense it is not clear how to go from one relaxation to the other, and so we will rely only on our new relaxation.

An LP Formulation.: For notational simplicity, rather than discuss fractional algorithms using some $k = \alpha \cdot \Delta$ colors, we will instead use $k = \Delta$ colors and relax the second constraint to

$$\sum_{e \ni v} x_{e,c} \leqslant \alpha \qquad\qquad \forall v \in V, c \in [\Delta]$$

When dealing with fractional solutions, it is easy to "stretch" such a solution to obtain a feasible edge coloring (i.e., satisfying $\sum_{e \ni v} x_{e,c} \leqslant 1$) while using $\lceil \alpha \cdot \Delta \rceil \leqslant \alpha \cdot \Delta + 1$ colors, and this can be done online. Therefore, our goal will be to minimize α — the competitive ratio.

Online Algorithms for the LP Relaxation: An online fractional edge coloring algorithm must assign $x_{e,c}$ values for all edges e upon arrival, immediately and irrevocably. For example, if Δ is known a priori, assigning each edge-color pair a value of $\frac{1}{\Delta}$ trivially yields a 1-competitive online fractional algorithm. If Δ is *unknown*, the situation is not so simple, as we now show.

6.3.2 Lower Bounds for Unknown Δ

In this section we present our lower bounds for online edge coloring with unknown Δ, proving that this problem is strictly harder than its known-Δ counterpart.

Our first lower bound concerns fractionally edge coloring bipartite graphs.

Theorem 6.3.1. *No fractional online edge coloring algorithm is better than $\frac{e}{e-1}$ competitive on bipartite graphs under one-sided arrivals.*

Proof. Consider the following construction. For any m, we construct a bipartite graph $G_m = (L_m, R_m, E_m)$, where L_m is the offline side and R_m is the online side. The offline side, L_m, contains $m!$ vertices, denoted by $v_1, \cdots, v_{m!}$. The online side, R_m, arrives over m phases. In phase k ($k \in [m]$), some $m!/k$ vertices of degree k arrive. Each vertex u_i which arrives in phase k ($i \in [m!/k]$) neighbors offline vertices $v_i, v_{m!/k+i}, \cdots, v_{m!(k-1)/k+i}$. We can see that each offline vertex has exactly one more neighbor in phase k and the maximum degree in phase k is exactly k. See Figure 6.1 for an illustrative example. The algorithm will have to be α competitive after each phase, as the adversarial sequence can "terminate early", after essentially presenting disjoint copies of $G_{m'}$ for some $m' \leqslant m$.

We use $x_{kj} := \frac{\sum_{e \in \text{phase } k} x_{e,j}}{|\{e \in \text{phase } k\}|}$ to denote the average assignment of color j to edges of phase k.

The average load for online vertices of phase k for color j is $k \cdot x_{kj}$, as each such online vertex has k edges. Consequently, as their average load is at most α, we have the following constraints.

$$k \cdot x_{kj} \leqslant \alpha \quad \forall 1 \leqslant j \leqslant k. \tag{6.1}$$

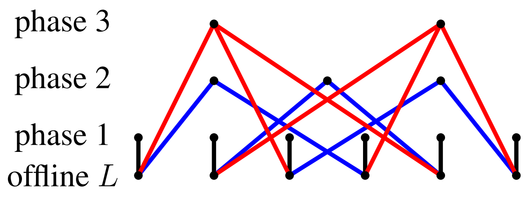


Figure 6.1: The hard instance for bipartite graphs for $m = 3$

Moreover, since each offline vertex has one more edge during phase k, the average assignment to all edges should cover all edges of phase k, implying the following constraint.

$$\sum_{j=1}^{k} x_{kj} \geqslant 1 \quad \forall k. \tag{6.2}$$

Finally, as the load of all offline vertices (which have only one edge in phase k) for any color j cannot exceed α (and so neither can their average), we have the following constraint.

$$\sum_{k=j}^{m} x_{kj} \leqslant \alpha \quad \forall j. \tag{6.3}$$

Combining constraints (6.1)-(6.3), yields the following linear program LP$_m$, which lower bounds α.

$$\begin{aligned}
\text{LP}_m := \min \alpha & \\
\sum_{j=1}^{k} x_{kj} \geqslant 1 & \qquad 1 \leqslant k \leqslant m \\
k \cdot x_{kj} \leqslant \alpha & \qquad 1 \leqslant j \leqslant k \leqslant m \\
\sum_{k=j}^{m} x_{kj} \leqslant \alpha & \qquad 1 \leqslant j \leqslant m \\
x_{kj} \geqslant 0 & \qquad 1 \leqslant j \leqslant k \leqslant m.
\end{aligned}$$

To lower bound α, we construct a series of solutions to the dual LP, which is as follows.

$$\begin{aligned}
\max \sum_{k=1}^{m} y_k & \\
\sum_{k=1}^{m} \sum_{j=1}^{k} z_{kj} + \sum_{j=1}^{m} w_j \leqslant 1 & \\
-k \cdot z_{kj} - w_j + y_k \leqslant 0 & \qquad 1 \leqslant j \leqslant k \leqslant m \\
y_k, w_j, z_{kj} \geqslant 0 & \qquad 1 \leqslant j \leqslant k \leqslant m.
\end{aligned}$$

Let $c(m) := \lfloor m/e \rfloor$. We know that $\lim_{m\to\infty} c(m)/m \to 1/e$. Let $t := 1/(m+1+c(m)\cdot(H_{c(m)} - H_m))$, where $H_k := \sum_{i=1}^{k} 1/k$ satisfies $\lim_{m\to\infty} H_{c(m)} - H_m \to \log(c(m)/m) \to -1$. We construct a feasible dual solution as follows: We let $y_1 = \cdots y_m = t$, and

$$w_j = \begin{cases} t & 1 \leqslant j \leqslant c(m) \\ 0 & \text{otherwise} \end{cases}$$

$$z_{kj} = \begin{cases} t/k & c(m)+1 \leqslant j \leqslant k \leqslant m \\ 0 & \text{otherwise.} \end{cases}$$

For any $1 \leqslant j \leqslant k \leqslant m$, we have that $k \cdot z_{kj} + w_j = t = y_k$. For the first dual constraint, we have

$$\begin{aligned}
&\sum_{k=1}^{m} w_k + \sum_{k=1}^{m}\sum_{j=1}^{k} z_{kj} \\
&= c(m)\cdot t + \sum_{k=c(m)}^{m} \left(\frac{k-c(m)}{k}\right)\cdot t \\
&= c(m)\cdot t + (m - c(m) + 1)\cdot t - c(m)\cdot t\cdot (H_m - H_{c(m)})) \\
&= \big(m + 1 + c(m)\cdot (H_{c(m)} - H_m\big)\cdot t = 1.
\end{aligned}$$

The above is therefore a feasible dual solution, of value

$$\begin{aligned}
\sum_{i=k}^{m} y_k = m\cdot t &= \frac{m}{m + c(m)\cdot (H_{c(m)} - H_m)} \\
&= \frac{1}{1 + \frac{c(m)}{m}\cdot (H_{c(m)} - H_m)}.
\end{aligned}$$

When $m \to \infty$, this tends to $\frac{1}{1-1/e} = \frac{e}{e-1}$. Consequently, $\lim_{m\to\infty} \mathrm{LP}_m \geqslant e/(e-1)$, implying our claimed lower bound for fractional online edge coloring of bipartite graphs. □

Making the Graph Dense: The above construction yields a sparse graph, as the number of vertices in this graph, $n = m! + m!(1 + \frac{1}{2} + \cdots + \frac{1}{m}) \approx m! \log m$, is exponential in its maximum degree, m. However, the following change yields a dense graph where the same lower bound still holds. Fix any integer $t > 0$, in the hard instance, we replace each vertex with t identical copies, and correspondingly, connecting all copies of pairs (u, v) which are adjacent in the sparse graph. The obtained graph is still bipartite and the maximum degree and the number of vertices both increase by a factor of t, to $t\cdot m$ and $t\cdot m! \log m$, respectively. Since we can take t to be arbitrarily large, the graph has maximum degree as high as $\Omega(n)$. In order to show that the lower bound still holds, we only need to slightly change the meaning of x_{kj} to be the average assignment of colors $(j-1)t+1, (j-1)t+2, \ldots jt$ during phase k. Constraints (6.1)-(6.3) still hold with this new meaning in the denser graph. Thus, we conclude that Theorem 6.3.1 holds for graphs of arbitrarily high degree.

Next, we present a lower bound for general graphs. The lower bound is based on the construction for bipartite graphs, but with more alterations. More specifically, recall that in the construction for bipartite graphs, when the online vertices of phase k arrive, we always connect them to k offline vertices. However, in general graphs, we have more freedom. In phase k, there can be two possible futures: in one we continue the sequence for bipartite graphs; in the other we connect all vertices which arrive during phases $k, k+1, \dots$ to the vertices which arrived in phase $k-1$. This example yields a lower bound of 1.606, showing a separation between bipartite and general graphs. (In [70], we show that this example is a tight instance for Algorithm 6, which is 1.777 competitive on it.)

Theorem 6.3.2. *No fractional online edge coloring algorithm is better than* 1.606 *competitive in general graphs.*

Proof. The adversarial instance has $m+1$ possible futures. Neither the number of phases m nor the choice of future are known to the online algorithm. There is a state associated with the input, and there are two possible states, "old" and "new". Initially, the graph contains $m!$ vertices and the state is "old". There are $m' \leqslant m$ phases in total. We use V_k to denote the set of online vertices which arrive in phase k ($k \in [m']$) and V_0 to denote the initial $m!$ vertices which arrive in phase 0. Moreover, we use v_i^k to denote the i^{th} vertex arrived in phase k. In phase k, newly-arrived vertices have degree k. If the state is "old", $m!/k$ vertices arrive and the i^{th} vertex, v_i^k, is adjacent to $v_i^0, v_{m!/k+i}^0, \cdots, v_{(k-1)m!/k+i}^0$. On the other hand, if the state is "new" and it changed from "old" to "new" at the end of phase t ($k > t$), then $m!/kt$ vertices arrive and the i^{th} vertex, v_i^k, will neighbor $v_i^t, v_{m!/kt+i}^t, \cdots v_{m!(k-1)/kt+i}^t$. At the end of phase k, the adversary decide whether to switch state to "new". Notice that the state can only transition from "old" to "new".

Again, we let x_{kj} denote the average assignment of color j to edges of phase k, but this time only if the state is "old" during this phase. The following constraints still hold for the same reason as Constraints (6.1) and (6.2) for the bipartite hard instance of Theorem 6.3.1.

$$\sum_{j=1}^{k} x_{k,j} \geqslant 1 \quad \forall k. \tag{6.4}$$

$$\sum_{k=j}^{m} x_{i,j} \leqslant \alpha \quad \forall j. \tag{6.5}$$

Furthermore, We use $y_{k,j}^t$ to denote the average assignment of color j to edges between V_k and V_t when the state transitions from "old" to "new" in phase t. (I.e., this is the average assignment of color j to edges of phase $k > t$, for t the phase at which the transition occurred.)

Again, as each edge between a V_t vertex and its neighbor in V_k ($k > t$) must be fractionally colored, we have

$$\sum_{j=1}^{k} y_{k,j}^t \geqslant 1 \quad \forall t < k \leqslant m. \tag{6.6}$$

Moreover, the maximum load of every vertex for every color is at most α, and so we have

$$t \cdot x_{t,j} + \sum_{k=t+1}^{m} y_{k,j}^{t} \leqslant \alpha \quad \forall 1 \leqslant j \leqslant t \leqslant m \tag{6.7}$$

$$\sum_{k=j}^{m} y_{k,j}^{t} \leqslant \alpha \quad \forall 1 \leqslant t < j \leqslant m \tag{6.8}$$

$$k \cdot y_{k,j}^{t} \leqslant \alpha \quad \forall 1 \leqslant t < k \leqslant m. \tag{6.9}$$

To summarize, constraints (6.4)-(6.9) for any m are all satisfied by any α-competitive online fractional edge coloring algorithm on this distribution of inputs. Therefore, the optimal value of an LP with objective of minimizing α subject to these constraints is a lower bound on the optimal competitive ratio α of any such online algorithm on general graphs. Using commercial solvers, we solve this LP for $m = 50$ and find that its optimal value, which lower bounds any algorithm's competitive ratio on general graphs, is 1.606. Again, using the same trick as Section 6.3.2, we find that this lower bound also holds for dense graphs. □

6.3.3 An Optimal Fractional Algorithm

Our LP relaxation asks to minimize the maximum *load* of any vertex u in color c, $L_u(c) := \sum_{e \ni u} x_{e,c}$. The greedy water-filling algorithm, upon arrival of edge e, increases all $x_{e,c}$ for all colors c minimizing the maximum load of either endpoint of e. This natural algorithm is no better than the integral greedy algorithm, however. In our algorithm, upon arrival of a vertex v, we run a variant of the water-filling algorithm on each edge (u, v) in an arbitrary order. One difference in our algorithm compared to the greedy one is that its greedy choice is *asymmetric*, and is only determined by the current loads of the previously-arrived endpoint, u. The second difference is that we set a *bound constraint* of β/Δ for each color per edge, where Δ is the **current** maximal degree, and β is a parameter of the algorithm which will be determined later. The bound constraints result in bounded load trivially for the online vertex, and by careful analysis, also for the offline vertex. In addition, the bound constraints result in a more balanced allocation, which uses more colors for each edge, but fewer colors overall. A formal description of our algorithm is given in Algorithm 6. Our algorithm is described as a continuous process, but can be discretized easily.

6.3.4 Basic properties of the algorithm

Our water filling algorithm preserves important monotonicity properties on the loads of any previously-arrived vertex v. In particular, the order obtained by sorting colors by their loads for v remains invariant following its future neighbors' arrivals. More formally, for each vertex v, we define an order permutation $\sigma_v : Z^+ \to Z^+$, where $\sigma_v(i)$ is the index of v's i^{th} most loaded color index after the vertex v arrives and its edges are fractionally colored (e.g., $\sigma_v(1)$ is the most-loaded color index). In addition, we define the load of a color in a vertex with respect to this order; i.e., we denote by $\ell_u^t(i)$ the load of color $\sigma_u(i)$ for vertex u after its t^{th} neighbor

Algorithm 6 Bounded Water Filling

```
Input: Online graph $G(V,E)$ with unknown maximum degree $\Delta(G)$ under vertex arrivals, parameter $\beta \in (1,2)$
Output: Fractional edge coloring $\{x_{e,c} \mid e \in E, c \in [\Delta(G)]\}$
1: (Implicitly) $x_{e,c} \leftarrow 0$ for all $e \in E, c \in \mathbb{N}$
2: for each arrival of a vertex $v$ do
3:     $\Delta \leftarrow \max\{\text{current } d(u) \mid u \in V\}$        ▷ $\Delta$ = current max. degree
4:     for each $e = (u,v) \in E$ do
5:         while $\sum_{c\in[\Delta]} x_{e,c} < 1$ do
6:             let $U := \{c \in [\Delta] \mid x_{e,c} < \beta/\Delta\}$        ▷ "unsaturated" colors for $e$
7:             let $C := \{c \in U \mid L_u(c) = \min_{c\in U} L_u(c)\}$        ▷ "currently active" colors for $e$
8:             for all $c \in C$ do
9:                 increase $x_{e,c}$ continuously        ▷ update $L_u(c), L_v(c), U$ and $C$
```

arrives – which we refer to as *step* t. In this notation, our monotonicity property will be that $\ell_u^t(i) \geqslant \ell_u^t(i+1)$ for each u and $i, t \in \mathbb{Z}^+$.

We denote by δ_u^t the global maximum degree after the arrival of the t^{th} neighbor of vertex u and denote by A_u the degree of u when it arrives (e.g., $A_u = 0$ for offline vertices in bipartite graphs). Next, we prove properties of the load of a specific vertex u after its arrival (i.e., for steps $t > A_u$), at which point the order σ_u is already set. For ease of notation we omit the subscript u from variables ℓ, δ and A whenever it will be clear from context (i.e., when considering a single vertex u). In addition, as σ will be clear from context, we will use *color* k as shorthand notation to $\sigma(k)$. Moreover, due to space constraints, we defer most proofs to Section 6.5.

We first observe that for our bounded water-filling algorithm (as for its unbounded counterpart), the load of u is monotone decreasing with respect to the σ_u order, and for each step t, the increase in the load for $i \leqslant \delta^t$ is monotone increasing in the σ_u order.

Observation 6.3.3. *For all color indices i, and any $t > A$,*
- $\ell^t(i) \geqslant \ell^t(i+1)$.
- $\ell^t(i) - \ell^{t-1}(i) \geqslant \ell^t(i-1) - \ell^{t-1}(i-1)$, *for all $i \leqslant \delta^t$.*

In our analysis, we focus on the *critical* colors at step T – colors whose load increased at step T and is higher than the following color load. Formally, color k is *critical* with respect to vertex u and its T^{th} neighbor if $\ell^T(k) > \ell^{T-1}(k)$ and $\ell^T(k) > \ell^T(k+1)$. Clearly, in order to upper bound the load at step T, it is sufficient to upper bound the load for critical colors k for T. If we let $V_1^k := \sum_{i=1}^k \ell^T(i)$ be the total load on colors $1, 2, \ldots, k$ and $V_2^k := \sum_{i=k+1}^{\delta^T} \ell^T(i)$ be the total load on colors $k+1, \ldots, \delta^T$, we will upper bound the load of color k by

$$\ell^T(k) \leqslant \frac{V_1^k}{k} \leqslant \frac{\delta^T - V_2^k}{k}, \tag{6.10}$$

where the first inequality is due to the monotonicity of the loads, and the second inequality is due to the total load being at most δ^T. Therefore, we will upper bound the load by proving a lower bound on the index of any critical color, and a lower bound on the total load after this index.

The next lemma plays a key role in both lower bounds. We show that for any color k critical at step T and for all steps $A < t \leqslant T$ during which k's load increases, all colors after k that could be increased (i.e. $k < i \leqslant \delta^T$) have their load increase by the maximum allowable amount, β/δ^t.

Lemma 6.3.4. *For a color k critical at step T, for all $A < t \leqslant T$ such that $\ell^t(k) > \ell^{t-1}(k)$, we have*

$$\ell^t(i) - \ell^{t-1}(i) = \beta/\delta^t \qquad \forall k < i \leqslant \delta^t.$$

Using the previous lemma, we bound the minimal index of a critical color at step T.

Lemma 6.3.5. *If k is a critical color at step T, then $k > \delta^T \cdot (1 - 1/\beta)$.*

Next, using Lemma 6.3.4 and some useful claims in Section 6.5 we prove a lower bound on V_2^k.

Lemma 6.3.6. *If k is a critical color at step T and $k^* \geqslant \max\{k, \delta^A\}$, then*

$$V_2^k \geqslant \sum_{j=k+1}^{\delta^T} \left(\ell^T(j) - \ell^{k^*}(j)\right) \geqslant \beta \cdot \left(\delta^T - k^* - k \log \frac{\delta^T}{k^*}\right).$$

Bounding the maximum load: Next, we use the previous lemmas in order to bound the maximum load after an assignment of an edge. Specifically, we will bound the load of ℓ_u and ℓ_v after coloring the edge (v, u), where v is the newly-arrived vertex. First, it is easy to bound the load of a vertex v for each color after its arrival, since we bound each edge-color pair's value $x_{e,c}$ by $\beta/\delta_v^{A_v} \leqslant \beta/A_v$ at arrival of v (when it has A_v neighbors).

Observation 6.3.7. $\ell_v^{A_v}(i) \leqslant \beta$ *for all* $i \in [\delta^{A_v}]$.

We now use Lemma 6.3.6 and Equation (6.10) to bound the load of a previously-arrived vertex u.

Lemma 6.3.8. *If $k > \delta_u^{A_u}$ is a critical color at step T w.r.t. u, then $\ell_u^T(k) \leqslant \beta \log \frac{\beta}{\beta-1}$.*

Lemma 6.3.9. *If $k \leqslant \delta_u^{A_u}$ is a critical color at step T w.r.t. u, then $\ell_u^T(k) \leqslant \beta^2 - \beta + \beta \log \frac{1}{\beta-1}$.*

Upper Bounding Algorithm 6's Competitive Ratio: We are now ready to bound the competitive ratio of Algorithm 6. First, we show that Algorithm 6 is $\frac{e}{e-1}$ competitive for one-sided bipartite graphs. That is, $G(L, R, E)$ is a bipartite graph and the offline vertices L arrive before the algorithm starts (i.e., $A_u = 0$ for all $u \in L$).

Theorem 6.3.10. *For bipartite graphs under one-sided arrivals, Algorithm 6 is* $\max\{\beta, \beta \log \frac{\beta}{\beta-1}\}$ *competitive. Setting* $\beta = \frac{e}{e-1}$*, we get an* $(\frac{e}{e-1})$*-competitive algorithm.*

Proof. We bound the load after coloring of edge (v, u), where $v \in R$ is the T^{th} online neighbor of u. First, we bound the load for any color i of v. By Observation 6.3.7, we have $\ell_v(i) = \ell_v^{A_v}(i) \leqslant \beta$. For vertex u, we have $A_u = \delta^{A_u} = 0$. Thus, by Lemma 6.3.8 we have that $\max_i \ell_u^T(i) \leqslant \beta \log \frac{\beta}{\beta-1}$. □

Finally, in Section 6.5 we bound our algorithm's competitive ratio on general graphs, proving that it is better than greedy.

Theorem 6.3.11. *For any graph, Algorithm 6 is* $\beta^2 - \beta + \beta \log \frac{1}{\beta-1}$ *competitive. Setting* $\beta = 1.586$*, we obtain a* 1.777*-competitive algorithm.*

6.3.5 An Optimal Integral Algorithm

In this section we show how to round fractional edge-coloring algorithms' output online, from which we obtain optimal integral online edge coloring algorithms for unknown Δ. Specifically, we will round fractional edge colorings provided by algorithms which assign at most some (small) value ε to each edge-color pair, which we refer to as ε*-bounded* algorithms. (As we shall see, the optimal fractional algorithms we will plug into this rounding scheme both satisfy this property.) We now state our main technical result of this section: a nearly-lossless rounding process for bounded algorithms on graphs with high enough lower bound on Δ.

Theorem 6.3.12. *For all* $\alpha \in [1, 2]$ *and* $\varepsilon \leqslant 1$*, if there exists an* ε*-bounded* α*-competitive fractional algorithm* $\mathcal{A}$ *for bipartite graphs with unknown maximum degree* $\Delta \geqslant \Delta' \geqslant 2/\varepsilon$*, then there exists a randomized integral algorithm* $\mathcal{A}'$ *which is* $(\alpha + O(\sqrt[12]{(\log n)/\Delta'})$*-competitive w.h.p on bipartite graphs of unknown maximum degree* $\Delta \geqslant \Delta' \geqslant c \cdot \log n$ *for some constant* c*.*

To make use of this theorem, we note that our optimal fractional algorithm for unknown Δ, Algorithm 6, can be made $2/\Delta'$ bounded by setting our initial lower bound on Δ to be Δ' in Line 3, without worsening the competitive ratio. (This is equivalent to adding a dummy star which does not increase the maximum degree.) Plugging this bounded fractional edge coloring algorithm into Theorem 6.3.12, we get an optimal randomized algorithm for edge coloring graphs with unknown Δ.

Theorem 6.3.13. *There exists an* $(\frac{e}{e-1} + O(\sqrt[12]{(\log n)/\Delta'}))$*-competitive algorithm for* n*-vertex bipartite graphs* G *with unknown maximum degree* $\Delta \geqslant \Delta' \geqslant c \cdot \log n$ *for some absolute constant* c*.*

Remark. The algorithm of Theorem 6.3.13 requires only a lower bound $\Delta' \leqslant \Delta$ for some $\Delta' = \omega(\log n)$ in order to output an $(\frac{e}{e-1} + o(1)) \cdot \Delta$ coloring, and not the exact value of Δ. Alternatively, our algorithm uses $(\frac{e}{e-1} + o(1)) \cdot \max\{\Delta, \Delta'\}$ colors for *any* unknown Δ, where the multiplicative approximation ratio is clearly only worse than $(\frac{e}{e-1} + o(1))$ for small $\Delta < \Delta'$ – in which case the additive approximation term is only $O(\Delta')$. This result can therefore be read as an asymptotic approximation scheme, trading off between the additive term and the asymptotic competitive ratio.

To describe our rounding scheme for fractional matchings, we make use of our online rounding scheme for bounded fractional matchings of Chapter 5, which motivates our study of bounded fractional edge colorings. We will in particular rely on the guarantees of that chapter's dependent rounding scheme given in Lemma 5.3.10, and restated her for ease of reference.

Lemma 6.3.14. *There exists an online algorithm, which, given online fractional bipartite matching* $\vec{x}$ *satisfying* $|x|_\infty \leqslant \varepsilon$*, outputs a random matching* $\mathcal{M}$ *which matches each edge* e *with probability*

$$x_e \cdot \left(1 - 11\sqrt[3]{\varepsilon \cdot \log(1/\varepsilon)}\right) \leqslant \Pr[e \in \mathcal{M}] \leqslant x_e.$$

We now outline our rounding scheme, which consists of phases, as follows. For each phase i, let U_i be the uncolored graph at start of phase i. (Initially, $U_1 = G$.) We compute an α-competitive fractional edge coloring in U_i online. Upon the algorithm's initialization, we sample each of the possible $\alpha \cdot n$ fractional matchings of this fractional coloring, i.i.d with probability p. We then round and color the sampled fractional matchings in an online fashion, as follows. Whenever a *sampled* fractional matching becomes non trivial, we assign it a new color. Whenever a new vertex v arrives, for each phase i in increasing order, we run the next step of MARKING for each of the sampled fractional matchings of phase i's fractional coloring, and color all newly-matched edges with the color assigned to the relevant fractional matching. Finally, we greedily color the remaining uncolored edges of v. Setting $p = o(1)$ (guaranteeing few re-colors) and also satisfying $\Delta \cdot p = \omega(\log n)$ (in order to have concentration up to $(1 \pm o(1))$ factors on number of colors used), this approach will use roughly $p \cdot \alpha \cdot \Delta(U_i)$ colors for the i^{th} phase, while decreasing the uncolored subgraph's maximum degree by roughly $p \cdot \Delta(U_i)$, or a $(1-p)$ factor. Thus, using $(1/p)\log(1/p)$ phases yield an uncolored subgraph of maximum degree $p \cdot \Delta$ (using $\alpha \cdot \Delta$ colors), which the greedy algorithm colors using $2p \cdot \Delta$ new colors. This implies Theorem 6.3.12.

6.3.6 Our Online Rounding Scheme

Our online rounding scheme, given an ε-bounded fractional edge-coloring algorithm $\mathcal{A}$ which is α competitive on graphs of maximum degree at least $2/\varepsilon$, for $\varepsilon = p^4/(12\log n)$, works as follows. Let $p := \sqrt[12]{24(\log n)/\Delta'}$. We use $P := \lceil (4/p)\log(1/p)\rceil$ many phases. For phase i, we sample in advance a subset $\mathcal{S}_i$ of all possible color indices, each taken into $\mathcal{S}_i$ with probability p. Let U_i be the subgraph of edges not colored before phase i. When online vertex v arrives, for each phase $i \in [P]$, we update a fractional coloring $x^{(i)}$ using Algorithm $\mathcal{A}$, based on v's arrival in U_i. For all sampled $j \in \mathcal{S}_i$ for which $x_j^{(i)}$ (the j^{th} fractional matching of $x^{(i)}$) is non trivial, we use a distinct color $c_{i,j}$ to color edges of a matching $M_{i,j}$ computed online by running MARKING on $x_j^{(i)}$. Finally, all remaining uncolored edges of v are greedily colored using new colors. This algorithm's pseudo-code is give in Algorithm 7.

Algorithm 7 Randomized Edge Coloring for Unknown Δ

```
Input: Online $n$-vertex bipartite graph $G(L,R,E)$ with $\Delta \geqslant \Delta' \geqslant c \cdot \log n$, for $c$ a constant
    TBD
    Parameter $p := \sqrt[12]{(24\log n)/\Delta'} (\leqslant 1/10)$
    An $\varepsilon$-bounded fractional online edge-coloring algorithm $\mathcal{A}$ which is $\alpha$ competitive on graphs
    $U$ with $\Delta(U) \geqslant 2/\varepsilon$, for $\varepsilon := (p^4/12\log n)$
Output: Integral $(\alpha + O(p)) \cdot \Delta$ edge coloring, w.h.p.
 1: for all $i$, set $\mathcal{S}_i \subseteq \lceil \alpha \cdot n \rceil$ to be such that each $j \in \lceil \alpha \cdot n \rceil$ is in $\mathcal{S}_i$ independently with
        probability $p$
 2: for all $i$, denote by $U_i$ the online subgraph of $G$ not colored during phases $1, 2, \ldots, i-1$
 3: for each arrival of a vertex $v \in R$ do
 4:     for phase $i = 1, 2, \ldots, \lceil (4/p)\log(1/p)\rceil$ do
 5:         $x^{(i)} \leftarrow$ output of Algorithm $\mathcal{A}$ on current $U_i$          ▷ run next step of $\mathcal{A}$
 6:         for $j \in \mathcal{S}_i$ with $x_j^{(i)} \neq \vec{0}$ do
 7:             if $c_{i,j}$ not set then
 8:                 set $c_{i,j}$ to next unassigned color index
 9:             $M_{i,j} \leftarrow$ output of MARKING on current $x_j^{(i)}$          ▷ run next step of MARKING
10:             if some $e \in M_{i,j}$ previously uncolored then
11:                 color $e$ using color $c_{i,j}$          ▷ note: $e \ni v$
12:     run greedy on uncolored edges of $v$, using colors not assigned during the phases
```

6.3.7 Analysis

We will study changes in the uncolored graph between subsequent phases and the colors used during the phases. For each i, let $\Delta_i := \Delta(U_i)$ be the maximum degree of the online graph not colored by phase $1, 2, \ldots, i-1$. In this section we will show that during each phase i, provided Δ_i is sufficiently large, Algorithm 7 uses some $\alpha \cdot \Delta_i \cdot p(1+O(p))$ new colors w.h.p., and obtain an uncolored subgraph U_{i+1} of maximum degree $\Delta_{i+1} = \Delta_i \cdot (1-p \pm O(p^2))$ w.h.p. This will imply a degree decrease at a rate of one per $\alpha + O(p)$ colors used. Repeating this for $\lceil (4/p)\log(1/p)\rceil$ phases, will therefore require $(\alpha+O(p))\Delta$ colors and yield a subgraph of maximum degree $p \cdot \Delta$, which we color greedily with $O(p)\Delta$ new colors, implying Theorem 6.3.12.

To upper bound the number of colors used in phase i, we note that the number of non-trivial (i.e., not identically zero) fractional matchings we round in each iteration is clearly a p-fraction of the (at most $\lceil \alpha \cdot \Delta_i \rceil$) non-trivial colors of $x^{(i)}$. Therefore, by standard Chernoff bounds (Lemma 2.4.10), if Δ_i is large enough, the number of colors in the phase is small, w.h.p.

Lemma 6.3.15. *If $\Delta_i \geqslant (6 \log n)/p^3$, then C_i, the number of colors used in phase i, satisfies*

$$\Pr\left[C_i \geqslant \alpha\Delta_i \cdot p \cdot (1+p)\right] \leqslant \frac{1}{n^2}.$$

Lemma 6.3.15 upper bounds the number of colors used in phase i by $\alpha\Delta_i \cdot p \cdot (1+p)$. Our main technical lemma, below, whose full proof is deferred to Section 6.4, asserts that these colors result in a decrease of roughly $\Delta_i \cdot p$ in the uncolored subgraph's maximum degree during the phase.

Lemma 6.3.16. *If $\Delta_i \geqslant (24 \log n)/p^4$, then*
1. $\Pr\left[\Delta_{i+1} \leqslant \Delta_i \cdot (1 - p - 4p^2)\right] \leqslant 3/n^3$.
2. $\Pr\left[\Delta_{i+1} \geqslant \Delta_i \cdot (1 - p + 7p^2)\right] \leqslant 6/n^2$.

Proof Sketch. Let v be a vertex of degree $d_i(v) \geqslant \Delta_i/2$ in U_i. By Lemma 6.3.14 and the ε-boundedness of the fractional algorithm $\mathcal{A}$ (and some simple calculations), each edge $e \in U_i$ is matched in $M_{i,j}$ ($j \in \mathcal{S}_i$) with probability $x^{(i)}_{e,j} \cdot (1 - O(p)) \leqslant \Pr[e \in M_{i,j}] \leqslant x^{(i)}_{e,j}$. That is, we match e in $M_{i,j}$ with probability close to its sampled "load" for this color. By Chernoff bounds, as we sample each color of $x^{(i)}$ with probability p, the sampled load on v's edges is $d_i(v) \cdot p(1 \pm O(p))$ w.h.p. So, by linearity and another Chernoff bound, the number of times v is matched during the i^{th} phase satisfies $M_v \leqslant d_i(v) \cdot p(1 + O(p))^2 \leqslant d_i(v) \cdot p(1 + O(p))$, and $M_v \geqslant d_i(v) \cdot p(1 - O(p))^3 \geqslant d_i(v) \cdot p(1 - O(p))$.

However, M_v also counts repeated matchings of edges of v, which do not contribute to v's degree decrease in the uncolored subgraph. We therefore want to bound R_v – the number of times a previously-colored edge of v is matched during the phase. By Chernoff's bound and ε-boundedness of the fractional algorithm, the load on each edge in the sampled colors $\mathcal{S}_i$, which in expectation is precisely p, is $O(p)$ w.h.p. So, intuitively, we would expect $R_v = \Theta(p) \cdot M_v$ w.h.p., implying $R_v = \Theta(d_i(v) \cdot p^2)$ w.h.p. Of course, as re-matches are not independent of matches, we cannot simply multiply these expressions this way. However, relying on the theory of negative association (see Section 2.4.1), the intuitive claim that $R_v = \Theta(d_i(v) \cdot p^2)$ w.h.p. can be formalized. We conclude that the degree decrease of vertex v in the uncolored graph during the i^{th} phase is $M_v - R_v = d_i(v) \cdot p \cdot (1 - \Theta(p))$ w.h.p. Taking union bound over all vertices v, the lemma follows. □

Theorem 6.3.12 now follows from Lemma 6.3.15 and Lemma 6.3.16. We sketch a proof of this theorem and defer its full proof to Section 6.4.

Proof of Theorem 6.3.12 (Sketch). Clearly, Algorithm 7 colors all edges of G, due to Line 12. By definition, all color classes computed are matchings. As we shall show, the number of colors used during the phases is at most $(\alpha + O(p)) \cdot \Delta$ w.h.p., and the greedy algorithm requires some

$O(p) \cdot \Delta$ colors w.h.p., implying our claimed result. We outline this proof using a stronger claim than Lemma 6.3.16.

Suppose instead of Lemma 6.3.16 we had that with high probability $\Delta_{i+1} = \Delta_i \cdot (1-p)$. Then, by induction we would have $\Delta_i = \Delta \cdot (1-p)^i$ and in particular for all $i \leqslant (1/p)\log(1/p)$ we would have $\Delta_i \geqslant \Delta \cdot p \geqslant \Delta' \cdot p$. Taking $p \geqslant \sqrt[5]{(24\log n)/\Delta'}$ would therefore imply that $\Delta_i \geqslant \Delta' \cdot p \geqslant (24\log n)/p^4$, which in turn would allow us to appeal to union bound to prove that $\Delta_i = \Delta \cdot (1-p)^i$ for all i, or in other words $\Delta_i - \Delta_{i+1} = \Delta_i \cdot p$, and that the number of colors used in each phase i is at most $C_i \leqslant \alpha \cdot \Delta_i \cdot p \cdot (1+p)$. Summing over all phases, since $\Delta_0 = \Delta$, this would imply that w.h.p., the number of colors used during the phases is

$$\sum_i C_i \leqslant \sum_i (\alpha + p(1+p)) \cdot (\Delta_i - \Delta_{i+1}) \leqslant (\alpha + p(1+p)) \cdot \Delta.$$

On the other hand, after $(1/p)\log(1/p)$ phases we would get a final uncolored subgraph of maximum degree $\Delta \cdot (1-p)^{(1/p)\log(1/p)} \approx \Delta \cdot p$ w.h.p., and so the greedy step of Line 12 would use at most $2\Delta \cdot p$ colors. Overall, Algorithm 7 therefore uses at most $(\alpha + O(p)) \cdot \Delta$ colors for $p = O(\sqrt[5]{(\log n)/\Delta'})$ and $\Delta \geqslant 24\log n$. Our more involved bounds are due to the slightly looser bounds for Δ_{i+1} in terms of Δ_i in Lemma 6.3.16. See full proof in Section 6.4 for details. $\square$

Applications to Known Δ: Algorithm 7 finds applications for *known* Δ, too. In particular, by Lemma 6.3.16 we find that if in each phase i we assign value $1/((1-p+7p^2)^i \cdot \Delta)$ for each edge-color pair, then we obtain a feasible coloring w.h.p., requiring $(1-p+7p^2)^i \cdot \Delta$ colors when the maximum degree is at least $(1-p-4p^2)^i \cdot \Delta$, w.h.p.; i.e., this is a $(1+O(p^2))$-competitive fractional algorithm for uncolored subgraph U_i. Replacing algorithm $\mathcal{A}$ in Algorithm 7 with this approach then yields, as in the proof of Theorem 6.3.12, an optimal, $(1+o(1))$-competitive randomized algorithm for known Δ. As we achieve better $o(1)$ terms in Section 6.2, we do not elaborate on this point here.

6.4 Omitted Proofs of Section 6.3.5

Here we provide the missing proofs of lemmas and theorem deferred from Section 6.3.5, restated here for ease of reference.

We start by bounding the number of colors used during each phase.

Lemma 6.3.15. *If $\Delta_i \geqslant (6\log n)/p^3$, then C_i, the number of colors used in phase i, satisfies*

$$\Pr\left[C_i \geqslant \alpha\Delta_i \cdot p \cdot (1+p)\right] \leqslant \frac{1}{n^2}.$$

Proof. As $\Delta_i \geqslant (6\log n)/p^3$, we have $\mathbb{E}[C_i] = \mathbb{E}[|\mathcal{S}_i|] \leqslant \alpha\Delta_i \cdot p \leqslant \alpha \cdot 6(\log n)/p^2$. Plugging

$\varepsilon = p$ into the upper multiplicative tail bound of Lemma 2.4.10, we get

$$\begin{aligned}\Pr[C \geqslant \alpha\Delta_i \cdot p(1+p)] &\leqslant \exp\left(-\frac{\alpha\Delta_i \cdot p(1+p)}{3}\right) \\ &\leqslant \exp\left(-\frac{((6\log n)/p^3)\cdot p^3}{3}\right) \\ &= 1/n^2.\end{aligned}$$

□

The main technical lemma of this section, bounding the maximum degree of the uncolored graph U_{i+1} in terms of its i^{th} phase counterpart, U_i, is as follows.

Lemma 6.3.16. *If $\Delta_i \geqslant (24\log n)/p^4$, then*
1. $\Pr\left[\Delta_{i+1} \leqslant \Delta_i \cdot (1-p-4p^2)\right] \leqslant 3/n^3$.
2. $\Pr\left[\Delta_{i+1} \geqslant \Delta_i \cdot (1-p+7p^2)\right] \leqslant 6/n^2$.

Before proving this lemma (in turn deferred to Section 6.4.1), we show how it implies our main theorem, restated below.

Theorem 6.3.12. *For all $\alpha \in [1,2]$ and $\varepsilon \leqslant 1$, if there exists an ε-bounded α-competitive fractional algorithm $\mathcal{A}$ for bipartite graphs with unknown maximum degree $\Delta \geqslant \Delta' \geqslant 2/\varepsilon$, then there exists a randomized integral algorithm $\mathcal{A}'$ which is $(\alpha + O(\sqrt[12]{(\log n)/\Delta'})$-competitive w.h.p on bipartite graphs of unknown maximum degree $\Delta \geqslant \Delta' \geqslant c\cdot\log n$ for some constant c.*

Proof. For our proof, we will require the following fact.

Fact 6.4.1. *All $p \in [0, 1/10]$ satisfy $(1-p-4p^2) \geqslant \exp(-2\cdot p)$ and $(1-p+7p^2) \leqslant \exp(-p/4)$.*

For $p = \sqrt[12]{(24\log n)/\Delta'} \leqslant 1/10$ to hold, we need $\Delta' \geqslant 24\cdot 10^{12}\cdot\log n$. That is, $c = 24\cdot 10^{12}$.

By Lemma 6.3.16 and Fact 6.4.1, $\Pr[\Delta_{i+1} \leqslant \Delta_i \cdot \exp(-2\cdot p)] \leqslant \Pr[\Delta_{i+1} \leqslant \Delta_i \cdot (1-p-4p^2)] \leqslant 3/n^2$, provided $\Delta_i \geqslant (24\log n)/p^4$. By our choice of $p = \sqrt[12]{(24\log n)/\Delta'}$, this implies that for all $i < \lceil (4/p)\log(1/p)\rceil$,

$$\Delta \cdot \exp(-2\cdot p)^i \geqslant \Delta \cdot p^8 \geqslant \Delta' \cdot p^8 = (24\log n)/p^4.$$

Consequently, if we let $A_i := [\bigwedge_i \Delta_i \geqslant (24\log n)/p^4]$ be an indicator for the event that Δ_i is large enough to appeal to Lemma 6.3.15 and Lemma 6.3.16 for phase i, then taking union bound

(Proposition 2.4.16) over all $j < i$, we have

$$\begin{aligned}\Pr[\overline{A_i}] &= \Pr\left[\Delta_i \leqslant (24\log n)/p^4\right]\\ &\leqslant \Pr\left[\bigvee_{j<i}(\Delta_j \leqslant \Delta \cdot \exp(-2\cdot p)^j)\right]\\ &\leqslant n \cdot 3/n^3 = 3/n^2.\end{aligned}$$

Now, by Lemma 6.3.16 and $p \leqslant 1/10$, we have

$$\Pr\left[\Delta_i - \Delta_{i+1} \leqslant p(1-7p)\cdot\Delta_i \mid A_i\right] = \Pr\left[\Delta_{i+1} \geqslant \Delta_i \cdot (1-p+7p^2) \mid A_i\right] \leqslant 6/n^2.$$

On the other hand, by Lemma 6.3.15, if we denote by C_i the number of colors used during the i^{th} phase, then the probability of any of the C_i being large is at most

$$\Pr\left[C_i \geqslant \alpha\Delta_i \cdot p(1+p) \mid A_i\right] \leqslant 1/n^2.$$

Now, by $\alpha \in [1,2]$ and $p \leqslant 1/10$, we find that $\alpha + 54p \leqslant \alpha(1+27p) \leqslant \alpha \cdot \frac{1+p}{1-7p}$. Therefore, if we let $B_i = \mathbb{1}[C_i \geqslant (\alpha+54p)\cdot(\Delta_i - \Delta_{i+1})]$ be the bad event that we use a significantly higher number of colors in phase i than the amount by which we decrease the maximum degree in the uncolored graph in that phase. Then, we have

$$\begin{aligned}\Pr[B_i] &\leqslant \Pr[C_i \geqslant (\alpha+54p)\cdot(\Delta_i - \Delta_{i+1}) \mid A_i] + \Pr[\overline{A_i}]\\ &\leqslant \Pr[C_i \geqslant \alpha\cdot\Delta_i \cdot p(1+p) \mid A_i]\\ &+ \Pr[\Delta_i - \Delta_{i+1} \leqslant p(1-7p)\cdot\Delta_i \mid A_i] + \Pr[\overline{A_i}]\\ &\leqslant 1/n^2 + 6/n^2 + 3/n^2 = 10/n^2.\end{aligned}$$

Therefore, by union bound, we have that with probability at least $1 - 10/n$, the number of colors used during the phases is at most

$$\sum_i (\alpha+54p)\cdot(\Delta_i - \Delta_{i+1}) \leqslant (\alpha+54p)\cdot\Delta.$$

Finally, we upper bound the number of colors used by the greedy step of Line 12, by upper bounding the uncolored subgraph's maximum degree before Line 12. We note that by Lemma 6.3.16 and Fact 6.4.1, we have $\Pr[\Delta_{i+1} \geqslant \Delta_i \cdot \exp(-p/4) \mid A] \leqslant \Pr[\Delta_{i+1} \geqslant \Delta_i \cdot (1-p+7p^2) \mid A] \leqslant 6/n^2$. Therefore, we find that the final uncolored subgraph U has maximum degree $\Delta(U) \leqslant \Delta \cdot p$, as

$$\begin{aligned}\Pr[\Delta(U) \geqslant \Delta\cdot p] &\leqslant \Pr[\Delta_{\lceil (4/p)\log(1/p)\rceil} \geqslant \Delta \cdot \exp(-p/4 \cdot \lceil (4/p)\log(1/p)\rceil)]\\ &\leqslant \Pr\left[\bigvee_i (\Delta_{i+1} \geqslant \Delta_i \cdot \exp(-p/4))\right]\\ &\leqslant \Pr\left[\bigvee_i (\Delta_{i+1} \geqslant \Delta_i \cdot \exp(-p/4)) \,\middle|\, A\right] + \Pr[\overline{A}]\\ &\leqslant n \cdot 6/n^2 + 3/n\\ &= 9/n.\end{aligned}$$

Consequently, the greedy step of Line 12 uses a further $2\Delta \cdot p$ colors, and so Algorithm 7 is an $(\alpha + 56p)$-competitive online edge coloring algorithm. □

6.4.1 Progress in degree decrease

In this section we will show that each phase i of Algorithm 7 with $\Delta_i \geqslant 24(\log n)/p^3$ decreases the maximum degree of the uncolored graph by a $1/(1-p\pm O(p^2))$ factor. That is, we will prove Lemma 6.3.16. As outlined in Section 6.3.5, our general approach will be to bound the number of times each near-maximum-degree vertex v in U_i is matched during the phase and the number of times it is matched without having an edge colored.

For the remainder of this section, we will need the following random variables. First, for any vertex v and index i, we let $d_i(v)$ denote v's degree in the uncolored subgraphs U_i. Moreover, for each edge e we let $L^{(i)}_{e,j} = x^{(i)}_j$ if $j \in \mathcal{S}_i$ and zero otherwise, and similarly $L^{(i)}_{v,j} := \sum_{e\ni v} L^{(i)}_{e,j}$. We refer to the above as the *load* of edge e and vertex v in color j of phase i. Finally, we denote by $\ell^{(i)}_e := \sum_j L^{(i)}_{e,j}$ and $\ell^{(i)}_v := \sum_j L^{(i)}_{v,j}$ the load of the edge e and vertex v in the sampled colors of phase i. Clearly, as each color index j is in $\mathcal{S}_i$ with probability p, and as each edge is fractionally matched exactly once, we have that $\mathbb{E}[\ell^{(i)}_e] = p$ and therefore $\mathbb{E}[\ell^{(i)}_v] = d_i(v)\cdot p$. The following lemma asserts that these variables are concentrated around their mean. In all notation, we omit i, which will be clear from context.

Lemma 6.4.2. *If $\Delta_i \geqslant (24\log n)/p^3$, then*

1. *for each edge e we have $\Pr[\ell_e \geqslant p(1+p)] \leqslant 1/n^4$, and*
2. *for each vertex v of degree $d_i(v) \geqslant \Delta_i/2$ in U_i we have $\Pr[|\ell_v - d_i(v)\cdot p| \geqslant d_i(v)\cdot p^2] \leqslant 2/n^3$.*

Proof. As noted above, $\mathbb{E}[\ell_e] = p$. Moreover, by the $(p^3/12\log n)$-boundedness of f we have that $\ell_e = \sum_j L_{e,j}$ is the sum of bounded independent variables $L_{e,j} \in [0, p^3/12\log n]$. So, by Chernoff bounds (Lemma 2.4.10) with $\varepsilon = p \in (0,1)$, we obtain

$$\begin{aligned}\Pr[\ell_e \geqslant p(1+p)] &= \Pr\left[\ell_e \geqslant \mathbb{E}[\ell_e]\cdot(1+p)\right] \\ &\leqslant \exp\left(-\frac{p\cdot p^2}{3p^3/(12\log n)}\right) \\ &= \exp\left(-4\log n\right) = 1/n^4.\end{aligned}$$

Similarly, as noted above, $\mathbb{E}[\ell_v] = p\cdot d_i(v)$. Moreover, as $x^{(i)}$ is a feasible fractional matching, we have $|L_{v,j}| \leqslant 1$ for all j. So, by Chernoff bounds (Lemma 2.4.10), with $\varepsilon = p \in (0,1)$, we obtain

$$\begin{aligned}\Pr[|\ell_v - \mathbb{E}[\ell_v]| \geqslant p^2\cdot d_i(v)] &= \Pr[|\sum_j L_{v,j} - \mathbb{E}[L_{v,j}]| \geqslant p\cdot\sum_j \mathbb{E}[L_{v,j}]] \\ &\leqslant 2\exp\left(-\frac{d_i(v)\cdot p\cdot p^2}{3}\right) \\ &\leqslant 2\exp\left(-\frac{\Delta_i\cdot p\cdot p^2}{6}\right) \\ &\leqslant 2\exp\left(-3\log n\right) \\ &\leqslant 2/n^3.\end{aligned}$$

□

We will now want to bound the number of times a vertex is matched during a phase. We will rely on Lemma 6.4.2 together with the following lemma.

Lemma 6.4.3. *Let $\vec{x}$ be a fractional matching with $\max_e x_e \leqslant p^4/(12\log n)$. Then for each edge e,* MARKING *run with input $\vec{x}$ outputs a matching $\mathcal{M}$ which matches each edge e with probability*
$$x_e \cdot (1-3p) \leqslant \Pr[e \in \mathcal{M}] \leqslant x_e$$

Proof. The upper bound on $\Pr[e \in \mathcal{M}]$ is true for all $\vec{x}$. For the lower bound, we have that by Lemma 6.3.14, as $p \in [0, 1/10]$ and as we may safely assume $n \geqslant 2$ (otherwise the problem is trivial), we have that the probability of e belonging to $\mathcal{M}$ is at least

$$\begin{aligned}
\Pr[e \in \mathcal{M}] &\geqslant x_e \cdot (1 - 11p\sqrt[3]{p \cdot \log(12\log n/p^3)/12\log n}) \\
&\geqslant x_e \cdot (1 - 11p\sqrt[3]{3p\log(1/p)/12\log n + p}) \\
&\geqslant x_e \cdot (1 - 11p\sqrt[3]{3(1/e)/12\log n + p}) && p \in [0,1] \\
&\geqslant x_e \cdot (1 - 11p\sqrt[3]{3/(e \cdot 12\log 2) + p}) && n \geqslant 2 \\
&\geqslant x_e \cdot (1 - 11p\sqrt[3]{3/(e \cdot 12\log 2) + 1/10}) && p \leqslant 1/10 \\
&\geqslant x_e \cdot (1-3p). && \square
\end{aligned}$$

Relying on Lemma 6.4.2.2 and Lemma 6.4.3, we obtain the following bounds on M_v, the number of times v is matched during the i^{th} phase.

Lemma 6.4.4. *If $\Delta_i \geqslant (24\log n)/p^4$, for each vertex v with degree at least $d_i(v) \geqslant \Delta_i/2$, then M_v, the number of times v is matched during the i^{th} phase, satisfies*
1. $\Pr[M_v \geqslant d_i(v) \cdot p(1+4p)] \leqslant 3/n^4$.
2. $\Pr[M_v \leqslant d_i(v) \cdot p(1-5p)] \leqslant 3/n^3$.

Proof. Let M_v^j be an indicator variable for the event that v is matched in $M_{i,j}$. For any instantiation of the variables $L_{e,j}$, Lemma 6.4.3 implies that each edge e is matched in $M_{i,j}$ with probability $L_{e,j} \cdot (1-3p) \leqslant \Pr[e \in M_{i,j}] \leqslant L_{e,j}$, and so by linearity we have $L_{v,j} \cdot (1-3p) \leqslant \Pr[M_v^j] \leqslant L_{v,j}$. In particular, if we let $A := [d_i(v) \cdot p(1-p) \leqslant \ell_v \leqslant d_i(v) \cdot p(1+p)]$, then, by linearity we have both $\mathbb{E}[M_v \mid A] \leqslant d_i(v) \cdot p(1+p)$ as well as $\mathbb{E}[M_v \mid A] \geqslant d_i(v) \cdot p(1-p)(1-3p) \geqslant d_i(v) \cdot p(1-4p)$. Now, clearly, $M_v = \sum_{j \in \mathcal{S}_i} M_v^j$ is the sum of binary random variables. Moreover, for any subset $\mathcal{S}_i$ sampled, these $\{M_v^j \mid j \in \mathcal{S}_i\}$ are independent, as all matchings $M_{i,j}$ for $j \in \mathcal{S}_i$ are computed using independent copies of MARKING. By Chernoff's upper tail bound (Lemma 2.4.10) with

$\varepsilon = 2p$, we thus obtain

$$\begin{aligned}
\Pr[M_v \geqslant d_i(v) \cdot p(1+4p) \mid A] &\leqslant \Pr[M_v \geqslant d_i(v) \cdot p(1+p)(1+2p) \mid A] \\
&\leqslant \Pr[M_v \geqslant \mathbb{E}[M_v \mid A] \cdot (1+2p) \mid A] \\
&\leqslant \exp\left(-\frac{\mathbb{E}[M_v \mid A] \cdot 4p^2}{3}\right) \\
&\leqslant \exp\left(-\frac{d_i(v) \cdot p(1-4p) \cdot 4p^2}{3}\right) \\
&\leqslant \exp\left(-\frac{(48 \log n)p^3(1-4p)}{3p^4}\right) \\
&\leqslant \exp(-4 \log n) && p \leqslant 1/5 \\
&\leqslant 1/n^4.
\end{aligned}$$

Therefore, we obtain the first claim, as

$$\Pr[M_v \geqslant d_i(v) \cdot p(1+4p)] \leqslant \Pr[M_v \geqslant d_i(v) \cdot p(1+4p) \mid A] + \Pr[\overline{A}] \leqslant 3/n^3.$$

Similarly, by Chernoff's lower tail bound (Lemma 2.4.10) with $\varepsilon = p$, we obtain

$$\begin{aligned}
\Pr[M_v \leqslant d_i(v) \cdot p(1-5p) \mid A] &\leqslant \Pr[M_v \leqslant d_i(v) \cdot p(1-p)(1-3p)(1-p) \mid A] \\
&\leqslant \Pr[M_v \leqslant \mathbb{E}[M_v \mid A] \cdot (1-p) \mid A] \\
&\leqslant \exp\left(-\frac{\mathbb{E}[M_v \mid A] \cdot p^2}{2}\right) \\
&\leqslant \exp\left(-\frac{d_i(v) \cdot p(1-p)(1-3p) \cdot p^2}{2}\right) \\
&\leqslant \exp\left(-\frac{12(\log n)p^3(1-p)(1-3p)}{2p^4}\right) \\
&\leqslant \exp(-3 \log n) \\
&\leqslant 1/n^3,
\end{aligned}$$

where the second to last inequality holds for all $p \leqslant 1/10$. From the above we obtain the second claim, as

$$\Pr[M_v \leqslant d_i(v) \cdot p(1-5p)] \leqslant \Pr[M_v \leqslant d_i(v) \cdot p(1-5p) \mid A] + \Pr[\overline{A}] \leqslant 3/n^3. \qquad \square$$

The above lemma asserts that the number of times a vertex v of high degree in U_i is matched during the i^{th} phase is $\Theta(d_i(v) \cdot p)$. The following lemma relies on the theory of Negative Association (NA, see Section 2.4.1) to show that all but $O(d_i(v) \cdot p^2)$ matches of v during this phase result in an edge of v being colored.

Lemma 6.4.5. *If $\Delta_i \geqslant (24 \log n)/p^3$, for each vertex v with degree at least $d_i(v) \geqslant \Delta_i/2$, the number of times v is matched along a previously colored edge, R_v, satisfies*

$$\Pr[R_v \geqslant 2d_i(v) \cdot p^2] \leqslant 2/n^2.$$

Proof. Fix the realizations of $L_{e,j}$ for all e, j. For any edge $e \ni v$, let $M_{e,j} := \mathbb{1}[e \in M_{i,j}]$ be an indicator for edge e being matched in iteration j of phase i. By the 0-1 rule, since at most one edge $e \ni v$ is in any matching, for each j the binary variables $\{M_{e,j} \mid e \ni v\}$ are NA. On the other hand, for $j \neq j'$ the joint distributions $\{M_{e,j} \mid e \ni v\}$ and $\{M_{e,j'} \mid e \ni v\}$ are independent. Thus, by closure of NA distributions under independent union (Proposition 2.4.4.1), the $\{M_{e,j} \mid j \in \mathcal{S}_i, e \ni v\}$ are NA. By closure of NA distributions under monotone increasing functions of disjoint variables (Proposition 2.4.4.2), if we let $R_e := \sum_j M_{e,j} \cdot \min\{1, \sum_{j'<j} M_{e,j'}\}$ denote the number of times e is matched and not colored, then these $\{R_e \mid e \ni v\}$ are NA. In this terminology, we have that $R_v = \sum_{e \ni v} R_e$ is the sum of NA variables. Moreover, as the $M_{e,j}$ are NA and as $\mathbb{E}[M_{e,j}] \leqslant L_{e,j}$ by Lemma 6.3.14, we have by the definition of NA variables (see (2.10)) that

$$\mathbb{E}\left[\sum_j M_{e,j} \cdot \sum_{j'<j} M_{e,j'}\right] \leqslant \sum_j \mathbb{E}\left[M_{e,j}\right] \cdot \mathbb{E}\left[\sum_{j'<j} M_{e,j'}\right] \leqslant \sum_j L_{e,j} \cdot \sum_{j'<j} L_{e,j'} \leqslant \ell_e \cdot \ell_e.$$

Let $A = \mathbb{1}\left[\forall e \ni v : \ \ell_e \leqslant p(1+p)\right]$ be an indicator for the high probability event that every edge $e \ni v$ has load at most $2p$ in the sampled matchings.

$$\mathbb{E}[R_e \mid A] \leqslant \mathbb{E}\left[\sum_j M_{e,j} \cdot \sum_{j'<j} M_{e,j'} \,\middle|\, A\right] \leqslant \mathbb{E}[\ell_e \mid A] \cdot \mathbb{E}[\ell_e \mid A] \leqslant p^2(1+p)^2.$$

Therefore, by linearity of expectation, $\mathbb{E}[R_v] = \sum_{e \ni v} \mathbb{E}[R_e] \leqslant d_i(v) \cdot p^2(1+p)^2$. Now, as $d_i(v) \geqslant \Delta_i/2 \geqslant 12(\log n)/p^3$ and as $R_v = \sum_e R_e$ is the sum of binary NA variables, we can upper bound R_v using the upper multiplicative Chernoff bound of Lemma 2.4.10 with $\varepsilon = \sqrt{p}$ to obtain

$$\begin{aligned}\Pr[R_v \geqslant d_i(v) \cdot p^2(1+p)^2(1+\sqrt{p}) \mid A] &\leqslant \exp\left(-\frac{d_i(v) \cdot p^2(1+p)^2 \cdot p}{3}\right) \\ &\leqslant \exp\left(-\frac{12 \log n}{3}\right) \\ &\leqslant \frac{1}{n^2}.\end{aligned}$$

Observing that for $p \leqslant 1/10$ we have $2 \leqslant (1+p)^2(1+\sqrt{p})$, we find that

$$\Pr[R_v \geqslant 2d_i(v) \cdot p^2 \mid A] \leqslant \Pr\left[R_v \geqslant d_i(v) \cdot p^2(1+p)^2(1+\sqrt{p}) \mid A\right] \leqslant 1/n^2.$$

Now, by Lemma 6.4.2.1 we have for every $e \ni v$ that $\Pr[\ell_e \geqslant p(1+p)] \leqslant 1/n^3$ and so by union bound we have $\Pr[\overline{A}] \leqslant n \cdot 1/n^3 = 1/n^2$. We therefore conclude that indeed

$$\Pr\left[R_v \geqslant 2 \cdot d_i(v) \cdot p^2\right] \leqslant \Pr\left[R_v \geqslant 2 \cdot d_i(v) \cdot p^2 \mid A\right] + \Pr\left[\overline{A}\right] \leqslant 2/n^2. \qquad \square$$

Lemma 6.3.16, restated below for ease of reference, follows from lemmas 6.4.4 and 6.4.5 and union bound of relevant subsets of vertices.

Lemma 6.3.16. *If $\Delta_i \geqslant (24 \log n)/p^4$, then*

1. $\Pr\left[\Delta_{i+1} \leqslant \Delta_i \cdot (1 - p - 4p^2)\right] \leqslant 3/n^3$.
2. $\Pr\left[\Delta_{i+1} \geqslant \Delta_i \cdot (1 - p + 7p^2)\right] \leqslant 6/n^2$.

Proof. For each vertex v, the decrease in v's degree in the uncolored subgraph during the i^{th} phase, denoted by $D_v := d_i(v) - d_{i+1}(v)$, is precisely the number of times v is matched and its matched edge is colored. That is, in the terminology of Lemma 6.4.4 and Lemma 6.4.5, $D_v = M_v - R_v$. So, by Lemma 6.4.4, every maximum degree vertex v in U_i (i.e. $d_i(v) = \Delta_i \geqslant \Delta_i/2$) satisfies

$$\begin{aligned}
\Pr[d_{i+1}(v) \leqslant \Delta_i \cdot (1 - p - 3p^2)] &= \Pr[d_{i+1}(v) \leqslant d_i(v) \cdot (1 - p - 3p^2)] \\
&= \Pr[d_i(v) - d_{i+1}(v) \geqslant d_i(v) \cdot p(1 + 3p)] \\
&= \Pr[D_v \geqslant d_i(v) \cdot p(1 + 3p)] \\
&\leqslant \Pr[M_v \geqslant d_i(v) \cdot p(1 + 3p)] \\
&\leqslant 3/n^4.
\end{aligned}$$

The first claim then follows by union bound over all maximum degree vertices v in U_i.

$$\Pr\left[\Delta_{i+1} \leqslant \Delta_i \cdot (1 - p - 3p^2)\right] \leqslant \sum_{v:\, d_i(v) = \Delta_i} \Pr\left[d_{i+1}(v) \leqslant \Delta_i \cdot (1 - p - 3p^2)\right] \leqslant 3/n^3.$$

Now, we let $\lambda := p(1 - 7p)$ and note that $(1 - \lambda) \cdot \Delta_i \geqslant \Delta_i/2$, since $p \leqslant 1/2$. All vertices v of degree $d_i(v) \leqslant (1 - \lambda) \cdot \Delta_i$ in U_i clearly have $d_{i+1}(v) \leqslant d_i(v) \leqslant (1 - \lambda) \cdot \Delta_i$. On the other hand, for every v with $d_i(v) \geqslant (1 - \lambda) \cdot \Delta_i \geqslant \Delta_i/2$, we have by lemmas 6.4.4 and 6.4.5 that

$$\begin{aligned}
\Pr[d_{i+1}(v) \geqslant (1 - \lambda) \cdot \Delta_i] &\leqslant \Pr[d_{i+1}(v) \geqslant (1 - \lambda) \cdot d_i(v)] \\
&= \Pr[d_i(v) - d_{i+1}(v) \leqslant d_i(v) \cdot \lambda] \\
&= \Pr[D_v \leqslant d_i(v) \cdot \lambda] \\
&= \Pr[D_v \leqslant d_i(v) \cdot p(1 - 7p)] \\
&\leqslant \Pr[M_v \leqslant d_i(v) \cdot p(1 - 5p)] + \Pr[R_v \geqslant d_i(v) \cdot p \cdot 2p] \\
&\leqslant 6/n^3.
\end{aligned}$$

The second claim then follows by union bound over all vertices v of degree $d_i(v) \geqslant (1-\lambda)\cdot\Delta_i$ in U_i, recalling that $\lambda = p(1 - 7p)$, since

$$\Pr[\Delta_{i+1} \geqslant (1 - \lambda) \cdot \Delta_i] \leqslant \sum_{v:\, d_i(v) \geqslant (1-\lambda)\cdot\Delta_i} \Pr[d_{i+1}(v) \geqslant (1 - \lambda) \cdot \Delta_i] \leqslant 6/n^2. \qquad \square$$

6.5 Omitted Proofs of Section 6.3.3

Here we provide the missing proofs of lemmas whose proof was deferred from Section 6.3.3, restated here for ease of reference.

Lemma 6.3.4. *For a color k critical at step T, for all $A < t \leqslant T$ such that $\ell^t(k) > \ell^{t-1}(k)$, we have*

$$\ell^t(i) - \ell^{t-1}(i) = \beta/\delta^t \qquad \forall k < i \leqslant \delta^t.$$

Proof. Suppose there exists $A < t \leqslant T$ such that $k+1 \leqslant \delta^t$ and $\ell^t(k+1) - \ell^{t-1}(k+1) < \beta/\delta^t$ and $\ell^t(k) - \ell^{t-1}(k) > 0$, then we can immediately derive that $\ell^t(k) = \ell^t(k+1)$, since k and $k+1$ are active at the end of the iteration. But by Observation 6.3.3 we know that $\ell^T(k) = \ell^T(k+1)$ – a contradiction. Finally, $\ell^t(i) - \ell^{t-1}(i) \geqslant \ell^t(k+1) - \ell^{t-1}(k+1)$ for all $k < i \leqslant \delta^t$ by Observation 6.3.3. □

Lemma 6.3.5. *If k is a critical color at step T, then $k > \delta^T \cdot (1 - 1/\beta)$.*

Proof. By Lemma 6.3.4, $\ell^T(k) > \ell^T(k+1)$ and $\ell^T(k) > \ell^{T-1}(k)$ imply $\ell^T(i) - \ell^{T-1}(i) = \beta/\delta^T$, for $k+1 \leqslant i \leqslant \delta^T$. Hence, if $k \leqslant \delta^T \cdot \left(1 - \frac{1}{\beta}\right)$, we would obtain

$$\begin{aligned}
\sum_{i=1}^{k}(\ell^T(i) - \ell^{T-1}(i)) &= 1 - \sum_{i=k+1}^{\delta^T}(\ell^T(i) - \ell^{T-1}(i)) \\
&= 1 - (\delta^T - k)\beta/\delta^T \\
&< 1 - (\beta/\delta^T) \cdot (\delta^T/\beta) \\
&= 0,
\end{aligned}$$

which would imply $\ell^T(k) = \ell^{T-1}(k)$ – contradicting the fact that k is critical. □

In order to lower bound V_k^2, we first prove the following two useful claims.

Claim 6.5.1. *If k is a critical color at step T, then for any $j > k$ and for any $\mathcal{S} \geqslant A$,*

$$\ell^T(j) - \ell^{\mathcal{S}}(j) = \sum_{\substack{\mathcal{S} < t \leqslant T \\ \delta^t \geqslant j}} \frac{\beta}{\delta^t}.$$

Proof. We prove that for any $t \geqslant A$ and $\delta^t \geqslant k$, then $\ell^t(k) > \ell^{t-1}(k)$. Assume not, then we have

$$1 = \sum_{i=1}^{\delta^t}(\ell^t(i) - \ell^{t-1}(i)) = \sum_{i=k+1}^{\delta^t}(\ell^t(i) - \ell^t(i-1)) \leqslant (\delta^t - k) \cdot \beta/\delta^t \leqslant (\delta^T - k) \cdot \beta/\delta^T < 1.$$

Where that last inequality is due to $k > (1-1/\beta)\delta^T$, by Lemma 6.3.5. Therefore, by Lemma 6.3.4, we have $\ell^t(j) - \ell^{t-1}(j) = \beta/\delta^t$ for $j \leqslant \delta^t$. Consequently,

$$\ell^T(j) - \ell^{\mathcal{S}}(j) = \sum_{t=\mathcal{S}+1}^{T} (\ell^t(j) - \ell^{t-1}(j)) = \sum_{t=\mathcal{S}+1}^{T} \mathbb{I}\{\delta^t \geqslant j\}(\ell^t(j) - \ell^{t-1}(j)) = \sum_{\substack{\mathcal{S}<t\leqslant T\\ \delta^t \geqslant j}} \frac{\beta}{\delta^t}. \qquad \square$$

Next, we bound the *total* load on the colors after a critical color k.

Claim 6.5.2. *If k is a critical color at step T, then for any $\mathcal{S} \geqslant A$*

$$\sum_{j=k+1}^{\delta^T} \left(\ell^T(j) - \ell^{\mathcal{S}}(j)\right) \geqslant \sum_{j=\mathcal{S}+1}^{\delta^T} \beta \cdot \frac{\delta^j - k}{\delta^j}.$$

Proof. By Claim 6.5.1, we have

$$\sum_{i=k+1}^{\delta^T} \left(\ell^T(i) - \ell^{\mathcal{S}}(i)\right) \geqslant \sum_{i=k+1}^{\delta^T} \sum_{\substack{\mathcal{S}+1\leqslant j\leqslant \delta^T\\ \delta^j \geqslant i}} \frac{\beta}{\delta^j} = \sum_{j=\mathcal{S}+1}^{\delta^T} \sum_{\delta^j \geqslant i \geqslant k} \frac{\beta}{\delta^j} = \sum_{j=k^*+1}^{\delta^T} \beta \cdot \frac{\delta^j - k}{\delta^j}. \qquad \square$$

We are now ready to prove the main lower bound volume lemma.

Lemma 6.3.6. *If k is a critical color at step T and $k^* \geqslant \max\{k, \delta^A\}$, then*

$$V_2^k \geqslant \sum_{j=k+1}^{\delta^T} \left(\ell^T(j) - \ell^{k^*}(j)\right) \geqslant \beta \cdot \left(\delta^T - k^* - k \log \frac{\delta^T}{k^*}\right).$$

Proof. Substituting $\mathcal{S}$ with k^* in Claim 6.5.2 (note that, $k^* \geqslant \delta^A \geqslant A$), we have

$$\begin{aligned}
\sum_{j=k+1}^{\delta^T} \left(\ell^T(j) - \ell^{k^*}(j)\right) &= \sum_{j=k^*+1}^{\delta^T} \beta \cdot \frac{\delta^j - k}{\delta^j} \\
&\geqslant \sum_{j=k^*+1}^{\delta^T} \beta \cdot \frac{j-k}{j} \\
&\geqslant \beta \cdot (\delta^T - k^*) - \beta \cdot k \log \frac{\delta^T}{k^*} \\
&= \beta \cdot \left(\delta^T - k^* - k \log \frac{\delta^T}{k^*}\right),
\end{aligned}$$

where the first inequality is since $\delta^j \geqslant j$. $\square$

Lemma 6.3.8. *If $k > \delta_u^{A_u}$ is a critical color at step T w.r.t. u, then $\ell_u^T(k) \leqslant \beta \log \frac{\beta}{\beta-1}$.*

Proof. As k is critical at step T, by Lemma 6.3.6, taking $k^* = k > \delta^A$, we have

$$V_2^k = \sum_{i=k+1}^{\delta^T} \ell^T(i) \geqslant \sum_{i=k+1}^{\delta^T} \left(\ell^T(i) - \ell^k(i)\right) \geqslant \beta \cdot \left(\delta^T - k - k \log \frac{\delta^T}{k}\right).$$

In addition, by Lemma 6.3.5, we have $k \geqslant \delta^T \cdot \left(1 - \frac{1}{\beta}\right)$. Thus, we find that indeed, by Equation (6.10)

$$\begin{aligned}\ell^T(k) &\leqslant \frac{\delta^T - V_2^k}{k} \\ &\leqslant \frac{\delta^T - \beta \cdot \left(\delta^T - k - k \log \frac{\delta^T}{k}\right)}{k} \\ &= (1-\beta)\frac{\delta^T}{k} + \beta + \beta \log \frac{\delta^T}{k} \\ &\leqslant \beta \log \frac{\beta}{\beta - 1}.\end{aligned}$$

□

Lemma 6.3.9. *If $k \leqslant \delta_u^{A_u}$ is a critical color at step T w.r.t. u, then $\ell_u^T(k) \leqslant \beta^2 - \beta + \beta \log \frac{1}{\beta-1}$.*

Proof. For ease of notation, in this lemma we will let $\Delta = \delta^T$. We will consider two cases and show the bound holds for both cases.

Case 1: $\boldsymbol{\delta^A/\beta \leqslant k \leqslant \delta^A}$: By Lemma 6.3.6 with $k^* = \delta^A \geqslant k$, we have

$$V_2^k \geqslant \beta \cdot \left(\Delta - \delta^A - k \log \frac{\delta^T}{\delta^A}\right).$$

As a consequence, by Equation (6.10), we have

$$\begin{aligned}
\ell^T(k) &\leqslant \frac{\Delta - V_2^k}{k} \\
&\leqslant \left(\Delta - \beta(\Delta - \delta^A) + \beta k \log\frac{\Delta}{\delta^A}\right)/k \\
&= (1-\beta)\frac{\Delta}{k} + \beta\frac{\delta^A}{k} + \beta\log\frac{\Delta}{\delta^A} \\
&= \frac{\delta^A}{k}((1-\beta)\frac{\Delta}{\delta^A} + \beta) + \beta\log\frac{\Delta}{\delta^A} \\
&\leqslant \beta((1-\beta)\frac{\Delta}{\delta^A} + \beta) + \beta\log\frac{\Delta}{\delta^A} \\
&\leqslant \beta^2 - \beta + \beta\log\frac{1}{\beta-1},
\end{aligned}$$

where the third inequality above holds because $\frac{\delta^A}{k} \leqslant \beta$ and $\frac{\Delta}{\delta^A} \leqslant \frac{\Delta}{k} \leqslant \beta/(\beta-1)$, by Lemma 6.3.5 and the last inequality holds because $\beta((1-\beta)\frac{\Delta}{\delta^A} + \beta) + \beta\log\frac{\Delta}{\delta^A}$ is maximized when $\frac{\Delta}{\delta^A} = 1/(\beta-1)$ (as can be verified by differentiating with respect to $x = \frac{\Delta}{\delta^A}$).

Case 2: $k \leqslant \delta^A/\beta$: Note that after the arrival of vertex u, the color load is at most β, by Observation 6.3.7. We may safely assume that $A \geqslant \beta k$, since we can always increase A to βk without increasing volume in V_2^k (which we aim to lower bound), by Observation 6.3.7.

$$\begin{aligned}
V_2^k &= \sum_{i=k+1}^{\Delta} \ell^\Delta(i) \\
&= \sum_{i=k+1}^{\Delta} \ell^A(i) + \sum_{i=k+1}^{\Delta} (\ell^{\delta^A}(i) - \ell^A(i)) + \sum_{i=k+1}^{\Delta} (\ell^\Delta(i) - \ell^{\delta^A}(i)) \\
&\geqslant (A - \beta k) + \sum_{j=A+1}^{\delta^A} \beta\cdot\frac{\delta^j - k}{\delta^j} + \beta\cdot\left(\Delta - \delta^A - k\log\frac{\Delta}{\delta^A}\right) \\
&\geqslant (A - \beta k) + (\delta^A - A)\cdot\beta\cdot\frac{\delta^A - k}{\delta^A} + \beta\cdot\left(\Delta - \delta^A - k\log\frac{\Delta}{\delta^A}\right) \qquad (6.11)\\
&\geqslant (\delta^A - \beta k)\cdot\beta\cdot\frac{\delta^A - k}{\delta^A} + \beta\cdot(\Delta - \delta^A) - \beta k\log\frac{\Delta}{\delta^A}.
\end{aligned}$$

The first inequality holds by Observation 6.3.7, Claim 6.5.1 and Lemma 6.3.6 with $k^* = \delta^A \geqslant \beta k \geqslant k$. The second inequality holds since for $j > A$, $\delta^j \geqslant \delta^A$. For the last inequality, substituting A with βk, a lower bound of A, will only decrease Equation (6.11), since the coefficient of A is non-negative; i.e. $1 - \beta + \beta\frac{k}{\delta^A} \geqslant 1 - \beta + \beta\frac{k}{\Delta} \geqslant 1 - \beta + \beta\cdot(1 - \frac{1}{\beta}) = 0$, where the last step follows by Lemma 6.3.5. Consequently, by Equation (6.10), we have that

$$\begin{aligned}
\ell^T(k) &\leqslant \frac{\Delta - V_2^k}{k} \\
&\leqslant \frac{\Delta - \left((\delta^A - \beta k)\cdot \beta \cdot \frac{\delta^A - k}{\delta^A} + \beta\cdot(\Delta - \delta^A) - \beta k \log\frac{\Delta}{\delta^A}\right)}{k} \\
&\leqslant (1-\beta)\frac{\Delta}{k} + \beta^2 + \beta - \beta^2\frac{k}{\delta^A} + \beta\log\frac{\Delta}{\delta^A} \\
&= (1-\beta)\frac{\Delta}{k} + \beta^2 + \beta - \beta^2\frac{k}{\delta^A} + \beta(\log\frac{\Delta}{k} + \log\frac{k}{\delta^A}) \\
&= \beta^2 + \beta + (\beta\log\frac{k}{\delta^A} - \beta^2\frac{k}{\delta^A}) + (\beta\log\frac{\Delta}{k} + (1-\beta)\frac{\Delta}{k}) \\
&\leqslant \beta^2 + \beta + (\beta\log\frac{1}{\beta} - \beta) + (\beta\log\frac{\beta}{\beta-1} - \beta) \\
&= \beta^2 - \beta + \beta\log\frac{1}{\beta-1}. \qquad \square
\end{aligned}$$

Finally, we will need the following simple inequalities for our analysis.

Fact 6.5.3. *For $\beta \in (1,2)$ we have $\beta \leqslant \beta^2 - \beta + \beta\log\frac{1}{\beta-1}$, as well as $\beta\log\frac{\beta}{\beta-1} \leqslant \beta^2 - \beta + \beta\log\frac{1}{\beta-1}$.*

Proof. For both inequalities, we rely on $x - 1 \geqslant \log(x)$ for all $x \geqslant 1$ to obtain the claimed inequalities. For the first, we have

$$\beta^2 - \beta + \beta\log\frac{1}{\beta-1} - \beta = \beta^2 - \beta + \beta\log\frac{1}{\beta-1} - \beta = \beta\left((\beta-1) - 1 - \log\left(\beta-1\right)\right) \geqslant 0.$$

For the second inequality, we have

$$\beta^2 - \beta + \beta\log\frac{1}{\beta-1} - \beta\log\frac{\beta}{\beta-1} = \beta(\beta - 1 - \log\beta) \geqslant 0. \qquad \square$$

6.6 Conclusion and Open Questions

In this chapter we presented optimal online edge coloring algorithms in bipartite graphs under one-sided vertex arrivals, both when the maximum degree is known and when it is not. This work suggests a few follow-up questions, most prominent of which is to obtain optimal online edge coloring algorithms under vertex arrivals, or even under edge arrivals. Bar-Noy et al. [25] suggested a candidate algorithm for edge arrivals with known Δ, though this algorithm seems challenging to analyze. Is their candidate algorithm $(1 + o(1))$ competitive? For unknown Δ, the problem seems much more challenging, even if one restricts oneself to fractional algorithms. Can one outperform the greedy algorithm for high-degree graphs with unknown maximum degree?

For vertex arrivals in general graphs we provided a better-than-greedy *fractional* algorithm. But can this algorithm be rounded without much loss? We note that our online rounding approach of Algorithm 7 works under vertex arrivals in general graphs too, though it requires an online dependent rounding scheme for fractional matching in general graphs generalizing the guarantees of our online dependent rounding scheme of Chapter 5. Such a tool would likely have applications to other online problems beyond edge coloring.

Part II

Online Algorithms: Beyond the Worst Case

Chapter 7

Online Ad Allocation: Structured Inputs

In this chapter we return to the online bipartite matching problem, and its extensions: online bipartite vertex-weighted matching and the budgeted ad allocation (or "AdWords") problem. In particular, in this chapter, based on [222] (joint work with Seffi Naor), we restrict our attention to structured inputs which are motivated by Internet advertising applications, which have motivated much work in this area in recent years.

7.1 Background

Internet advertising is ubiquitous. With over 120 billion dollars spent on Internet advertising in 2019 in the United States alone (see [236]), it has become, to a large extent, the driving economic force behind much of the content of the world wide web. How is this advertising space bought and sold? Most ads fall either under sponsored search or targeted advertising, both of which are sold in what constitute instances of the online ad allocation problem.

In online ad allocation, we are faced with the following problem: advertisers announce to an advertising platform (e.g. Yahoo, Google, Microsoft) what their advertising budgets are, and their bids for an ad to be displayed to every kind of user. The user "type" is determined, for example, by search terms searched, in the case of sponsored search, or user-demographics, in the case of targeted advertising. When a user visits a web-page with an ad slot managed by the ad platform, the latter needs to decide immediately and irrevocably which (if any) of the advertisers' ads to display to the user. The advertising platform's goal is to maximize its revenues, despite uncertainty concerning future page-views. This problem can be formulated as a generalization of online bipartite matching, with advertisers as the offline vertices and ad slots as online vertices. See Section 7.2 for a formal definition of this and other problems we consider.

The theoretical interest in online allocations can be traced back to 1990, when [179] considered the fundamental problem of bipartite maximum matching in an online setting. In their seminal paper, Karp et al. proved that randomized online algorithms cannot in general achieve competitive ratio above $1 - \frac{1}{e} \approx 0.632$, and presented the RANKING algorithm, which matches this lower bound and is thus optimal. See [49, 79, 86, 103, 134] for alternative analyses of this algorithm.

The online maximum matching problem was generalized, first by [173], and later by [6], who presented algorithms achieving optimal $1 - \frac{1}{e}$ competitive ratio for the b-matching and vertex-weighted matching problems, respectively. The AdWords problem, first proposed by [207], is the more general ad allocation problem, but subject to the realistic *small bid assumption*, i.e. assuming every advertiser i has budget B_i much larger than its bids b_{ij}. (This assumption is necessary to achieve non-trivial results. See Lemma 7.6.4). For this problem too the natural greedy algorithm has competitive ratio $\frac{1}{2}$. Mehta et al. gave an algorithm for this problem with competitive ratio $1 - \frac{1}{e}$. [53] achieved the same results using an online primal-dual approach. See [206] for an in-depth survey of prior art and techniques used to tackle these problems.

We will address the problems discussed above, but first, we start with motivation.

7.1.1 Motivation

As is to be expected of a problem for which a loss of $1/e \approx 36.7\%$ translates to billions of dollars in potential revenue lost yearly, researchers have studied weaker models than the adversarial model for the ad allocation problem, in the hope that these may permit better guarantees and help model real-world data and derive better algorithms for this data. (See 7.1.5.) In this chapter we revisit the stronger adversarial model, for graphs with structural characteristics met by many ad allocation instances arising from targeted advertising. Specifically, we assume advertisers are interested in a large number of ad slots (at least k), and that every ad slot is of interest to a relatively small number of advertisers (at most d). As with the small bid assumption $B_i \gg b_{ij}$ for the AdWords problem, assumption of the above structure is not only useful in order to obtain better bounds (as we will show), but also constitutes a reasonable assumption for targeted advertising, for the following twin reasons:

(I) Online side: advertisers typically target their advertising campaigns at *specific* segments of the population (e.g. young Californians who ski often); while these segments may be large in absolute terms, they are mostly small in relative terms (e.g., less than four percent of Californians ski often). Consequently, users tend to belong to relatively few segments. Coupled with the fact that the number of active campaigns at any given time is limited, this implies a restricted pool of ads that might be displayed to any particular user, justifying the small degree assumption for ad slots.

(II) Offline side: advertisers typically target *large* segments of the population (as in the example above), while not allocating a budget high enough to display ads to all users in a segment. Coupled with the fact that every page-view of a particular targeted user corresponds to a vertex in the graph, this implies the high degree assumption on the offline side, and more generally for the ad allocation problem, the assumption that $\sum_{i,j} b_{ij} \geqslant k \cdot B_i$ for some large k.

We call the graphs displaying these characteristics (k, d)-*bounded graphs*.

Definition 7.1.1 ((k, d)-bounded graphs)**.** *A bipartite graph* $G = (L, R, E)$ *is* (k, d)-bounded *if every left vertex* $i \in L$ *has degree* $d(i) \geqslant k$ *and every right vertex* $j \in R$ *has degree* $d(j) \leqslant d$*. For ad allocations, we replace* $d(i) \geqslant k$ *with the property* $\sum_j b_{ij} \geqslant k \cdot B_i$*.*

We concern ourselves with such graphs with k large and d small. For brevity's sake, as all graphs in this chapter will be bipartite, we refrain from stating the fact explicitly; likewise, we refer to (k, d)-bounded graphs as (k, d)-graphs henceforth. We recall that we adopt the convention that a lower bound indicates a negative (i.e., impossibility) result and an upper bound indicates a positive (i.e., algorithmic) result.

7.1.2 Our Results

By focusing on (k, d)-graphs, we justify the observed success of greedy algorithms "in the wild" beyond their theoretical guarantees (see Section 7.8 for a discussion of said success), and propose algorithms that are exponentially better, and provably optimal under these structural assumptions. Finally, we leverage our deterministic algorithms to prove simple randomized algorithms achieve the same bounds in expectation. Our results hold for the maximum matching, vertex-weighted matching and AdWords problems (with the exception of the matching lower bound for the latter). Table 7.1 delineates our results for these problems on $(k, d)-$graphs. We obtain similar results for the general ad allocation problem, even with large-ish bids (see Theorem 7.1.5.)

Table 7.1: Best results for general and (k, d)-graphs

Algorithms	General Graphs	$(k, d)-$Graphs
Greedy	$\frac{1}{2}$ (Tight) Folklore	$1 - \frac{d-1}{k+d-1}$ (Tight) This chapter
Deterministic	$\frac{1}{2}$ (Tight) Folklore	$1 - (1 - \frac{1}{d})^k$ (Tight) This chapter
Randomized	$1 - \frac{1}{e}$ (Tight)* [6, 53, 179, 207]	$1 - (1 - \frac{1}{d})^k$ This chapter

* can be achieved *deterministically* for AdWords.

We begin by explaining the empirical success of greedy algorithms for the above problems (i.e., algorithms matching an arriving ad slot to the most lucrative feasible neighbor), proving these algorithms' loss is proportional to the ratio of the maximal degree in the online side to the minimal degree in the offline side; i.e., their competitive ratio tends to *one* as this ratio tends to zero. We complement this upper bound with a family of examples for which these algorithms do no better.

Theorem 7.1.2. *Greedy algorithms achieve a competitive ratio of* $\frac{k}{k+d-1}$ *on* (k, d)*-bounded graphs. This analysis is tight for all* $k \geqslant d - 1$.

We improve on the above, designing deterministic algorithms with *exponentially* smaller loss. We prove this is optimal for deterministic algorithms.

Theorem 7.1.3. *There exist (new) deterministic online algorithms for the unweighted and vertex-weighted matching problems with competitive ratio* $1-(1-\frac{1}{d})^k > 1-(\frac{1}{e})^{k/d}$ *on* (k,d)*-bounded graphs. Moreover, these algorithms gain at least a* $1-(1-\frac{1}{d})^k$ *fraction of the total sum of weights. This is optimal whenever* $k \geqslant d$*.*

Corollary 7.1.4. *(Structural Corollary) For every bipartite graph* G *with the minimal degree of its left side at least* $\ln c$ *times larger than the maximal degree of its right side,* G *has a matching with at least a* $(1-\frac{1}{c})$*-fraction of* G*'s left side matched.*

In stating our bounds for general ad allocation, we follow the notation of [53] and denote the maximum bid-to-budget ratio by $R_{\max} = \max_{(i,j)\in E}\left\{\frac{b_{ij}}{B_i}\right\}$.

Theorem 7.1.5. *There exists a (new) deterministic algorithm which gains at least*

$$\left((1-R_{\max})\cdot\left(1-\left(1-\frac{1}{d}\right)^k\right)\right)\cdot\sum_{i\in L} B_i$$

total revenue for ad allocation on (k,d)*-graphs with* $k \geqslant d-1$*. Consequently, this algorithm has competitive ratio at least* $(1-R_{\max})\cdot(1-(1-\frac{1}{d})^k)$*.*

To contrast our results with the state-of-the-art, we note that the algorithms of [53, 79, 207] achieve competitive ratio $(1-R_{\max})\cdot\left(1-1/(1+R_{\max})^{1/R_{\max}}\right)$. This bound tends to $1-\frac{1}{e}$ from below as $R_{\max}$ tends to zero, but is far from this value for larger $R_{\max}$. Our algorithms fare better whenever $k \geqslant d$ even for large-ish $R_{\max}$. As stated in Section 7.1.1, we expect k to be significantly larger than d, but in order to emphasize the strength of our bound, let us consider only the case $d/k = R_{\max}$. Table 7.2 displays the resulting competitive ratios. Note that in this regime our algorithm is already better at $R_{\max} = \frac{1}{3}$ than prior algorithms are at the limit (i.e. when $R_{\max} \to 0$).

Table 7.2: Results for Ad Allocation with large-ish bids in (k,d)-graphs with $d/k = R_{\max}$

$R_{\max}$	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{5}$	$\frac{1}{6}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{100}$	$\to 0$
State-of-the-art	0.278	0.385	0.443	0.478	0.503	0.582	0.607	0.624	0.632
Our Work	0.432	0.633	0.736	0.795	0.831	0.938	0.969	0.99	1

Better still, our algorithms are robust to a few outlying advertisers increasing $R_{\max}$, as the $\sum_i B_i\cdot(1-R_{\max})$ term in Theorem 7.1.5's bound is rather $\left(\sum_i B_i - \max_{j\in N(i)} b_{ij}\right)$. This is the first such result in the adversarial setting. To the best of our knowledge only the algorithm of [78] for the iid model holds this desired property. Likewise, our algorithms are robust to few outlying advertisers making the input not (k, d)-bounded (alternatively, increasing k), as the following theorem asserts.

Theorem 7.1.6 (Outliers)**.** *If every advertiser i satisfies $\sum_j b_{ij} \geqslant k \cdot B_i$, except for a subset $S \subseteq L$ with total budget at most $\sum_{i \in S} B_i \leqslant \alpha \cdot \sum_{i \in L} B_i$, then the algorithms of theorems 7.1.3 and 7.1.5 gain revenue at least $(1-\alpha)$ times the bounds guaranteed by the above theorems. In particular, these algorithms achieve competitive ratio at least $(1-\alpha) \cdot (1-(1-\frac{1}{d})^k)$ and $(1-\alpha) \cdot ((1-R_{\max}) \cdot (1-(1-\frac{1}{d})^k))$, respectively.*

Finally, we prove that the naïve randomized algorithm, RANDOM, which matches every arriving ad slot to a feasible neighbor chosen uniformly at random, and in general has competitive ratio tending to $1/2$, attains the same bounds as our optimal deterministic algorithms in expectation, despite making no use of the input's structure.

Theorem 7.1.7. *Algorithm* RANDOM *matches the bounds of theorems 7.1.3, 7.1.5, and 7.1.6 in expectation.*

7.1.3 Techniques

As many previous ad allocation algorithms, our algorithms can be seen as bid-scaling algorithms. That is, matches are chosen greedily based on the bids b_{ij} of each advertiser i, times a scaling factor. However, contrary to previous algorithms [53, 79, 207] that scale bids according to $1 - e^{f(i)-1}$, where $f(i)$ is the fraction of i's budget spent so far, our algorithms essentially scale bids according to an exponential in u, the number of *unused opportunities* to spend as much as the current bid value b_{ij}; specifically, we scale by $\left(\frac{d}{d-1}\right)^u$. Other differences can be seen in our algorithms' primal-dual interpretation: we make no use of the ad slots' dual variables, leaving them at zero throughout (prior work increases these variables in order to guarantee dual feasibility); instead, our algorithms only update the dual variables of each arriving ad slot's neighbors. Interestingly, our online primal-dual algorithms do not guarantee dual feasibility throughout their execution, but only upon termination. To the best of our knowledge, ours are the first online primal-dual algorithms with this behavior.

The above approach works directly for vertex-weighted matching. To generalize our approach to ad allocations, we first consider an intermediary problem – equal-bids ad allocation – where every advertiser i bids the same bid b_i for all neighbors $j \in N(i)$. We reduce this problem in (k, d)-graphs to the vertex-weighted problem in (k, d)-graphs in an online manner. We then rewrite this reduction along with our vertex-weighted online algorithm as a single online primal-dual algorithm for the equal-bids problem. Guided by this algorithm we devise a primal-dual algorithm for general-bids ad allocation on (k, d)-graphs, using a bounded fraction of the advertisers' dual variables to guide our choice of matches and dual updates. This allows us to simulate the bid-scaling described above even when advertisers make different bids per ad slot.

Finally, our randomized results stem from our deterministic primal-dual algorithms, whose dual updates we use in our dual-fitting analysis of the randomized algorithms. Dual feasibility follows as it does for our algorithms. The dual costs are bounded in expectation by the primal cost times the required constant, conditioned over the random algorithm's previous choices. Taking total expectation over the possible previous choices yields the expected competitive ratio.

7.1.4 Intuition

Here we give a high-level outline of why one should expect to obtain better competitive guarantees on (k, d)-graphs than on more general graphs, and motivate our algorithms. Having k large implies each advertiser has many opportunities to exhaust her budget. On the other hand, having d (and $R_{\max}$) small implies each arriving ad slot "uses up" few of the opportunities of its neighboring advertisers. As a result, one would expect to have enough chances to spend much of each advertiser's budget. Our algorithms take this intuition one step further: guided by the observation that advertisers with many "missed" opportunities may have fewer remaining chances to spend their budget than other advertisers, we scale bids by a function of the number of missed opportunities. While the choice of this particular function may seem a little mysterious at first, it becomes clear once analyzed using the online primal-dual framework of [52].

7.1.5 Related Work

Several stochastic models have been studied for the problems we address. Most prominent among these are the random arrival order and i.i.d model with known/unknown distribution. Our algorithms beat all of these bounds *in the worst case* for sufficiently small d/k and $R_{\max}$, replacing stochastic assumptions by structural ones.

For the random order model a line of work beginning with [134] has shown the optimal competitive ratio for maximum matching lies in the range $(0.696, 0.823)$ [106, 176, 201, 203]. For the known distribution model [106] were the first to show the optimal competitive ratio is strictly greater than $1 - \frac{1}{e}$ and bounded away from 1. Subsequent work [21, 150, 170] showed the optimal competitive ratio for bipartite matching in this setting lies in the range $(0.706, 0.823)$, and $(0.729, 0.823)$ if the expected number of arrivals of each ad slot type is integral. For the vertex-weighted problem under the previously-mentioned integrality assumptions [150] and [170] showed an upper bound of 0.667 and 0.725, respectively. For the AdWords problem under the random order model, [77] give a $(1 - \varepsilon)$-competitive algorithm, assuming the online side's size is known in advance and no bid is higher than roughly $\varepsilon^3/|L|^2$ times the optimum value. [78] gave an algorithm in the unknown distribution model achieving asymptotically optimal competitive ratio of $1 - O(\sqrt{R_{\max}})$.

In a different vein, [202] considered the AdWords problem given black-box estimates of the input. They show how to obtain performance trading-off between the worst-case optimal and the black-box's performance on the given input. We require no such algorithm be available, but rather rely on domain-specific structure.

Closer to our work, [53] considered $(1, d)$-graphs for equal-bids ad allocation. We obtain more general results, and strictly better bounds for all $k > d$.

7.2 Problem Definitions

An instance of the *ad allocation* problem consists of a bipartite graph $G = (L, R, E)$. The left-hand L side corresponds to advertisers, and the right-hand side R to ad slots. Each advertiser $i \in L$ has some budget B_i and is willing to bid some value $b_{ij} \leqslant B_i$ for every neighboring ad slot $j \in N(i)$ (the bids of advertiser i need not be equal for all $j \in N(i)$). Each ad slot $j \in R$ can be allocated to (up to) one advertiser i, yielding a profit of b_{ij}. The bids for ad slots allocated to an advertiser i may not exceed i's budget, B_i. Figure 7.1 presents the ad allocation problem's LP relaxation and its dual.

Primal (Packing)		Dual (Covering)	
maximize	$\sum_{(i,j)\in E} b_{ij} \cdot x_{ij}$	minimize	$\sum_{i\in L} B_i \cdot z_i + \sum_{j\in R} y_j$
subject to:		subject to:	
$\forall j \in R$:	$\sum_{(i,j)\in E} x_{ij} \leqslant 1$	$\forall (i,j) \in E$:	$b_{ij} \cdot z_i + y_j \geqslant b_{ij}$
$\forall i \in L$:	$\sum_{(i,j)\in E} b_{ij} \cdot x_{ij} \leqslant B_i$	$\forall i \in L$:	$z_i \geqslant 0$
$\forall (i,j) \in E$:	$x_{ij} \geqslant 0$	$\forall j \in R$:	$y_j \geqslant 0$

Figure 7.1: The fractional ad allocation LP and the corresponding dual

An instance of the *online* ad allocation problem consists of an ad allocation instance; the advertisers given up-front, along with their budgets, and the ad slots arriving one-by-one, together with their edges and bids. An online ad allocation algorithm must, upon arrival of an ad slot j, determine to which advertiser (if any) to allocate the ad slot. Allocations are irrevocable, and so must be made to *feasible* advertisers, whose residual budget is sufficient to pay their actual bid.

We will consider several interesting special cases of the above problem throughout this chapter. These problems are both interesting in their own right (theoretically as well as practically), in addition to providing some insight towards achieving a solution to the general problem.

The *equal-bids* online ad allocation problem is the above problem with each advertiser i bidding the same value for all neighboring ad slots; i.e., $b_{ij} = b_i$ for all $j \in N(i)$.

The online *vertex-weighted matching* problem is the above problem with every advertiser i bidding all its budget for every neighboring ad slot; i.e., $b_{ij} = B_i$ for all $j \in N(i)$.

The online *maximum matching* problem is the above problem with all budgets and bids equal to 1; i.e., $b_{ij} = B_i = 1$ for all $j \in N(i)$.

7.3 Warm-up: Greediness in (k, d)-Graphs

In this section we show that the natural greedy algorithms for the problems considered, which in general graphs are only 1/2-competitive, achieve on (k, d)-graphs a competitive ratio tending to one as d/k tends to zero. We prove this result by applying dual-fitting, and prove our analysis is tight.

Algorithm GREEDY for the online ad allocation problem matches an ad slot $j \in R$ to a feasible neighbor i with highest bid b_{ij}. Our analysis relies on the dual-fitting formulation given in Algorithm 8 below.

Algorithm 8 AD ALLOCATION GREEDY (Dual-Fitting Formulation)

```
1: Init: set $z_i \leftarrow 0$ for all $i \in L$ and $y_j \leftarrow 0$ for all $j \in R$
2: for all $j \in R$ do
3:     if $j$ has a feasible neighbor then
4:         match $j$ to a feasible neighbor maximizing $b_{ij}$
5:         set $x_{ij} \leftarrow 1$
6:         set $z_i \leftarrow \min\{1, z_i + \frac{b_{ij}}{B_i}\}$
7:         set $z_{i'} \leftarrow \min\{1, z_{i'} + \frac{b_{i'j}}{k \cdot B_i}\}$ for every feasible neighbor of $j$, $i' \neq i$
8: for all $i \in L$ do
9:     if $i$'s residual budget is less than $R_{\max} \cdot B_i$ then
10:        set $z_i \leftarrow 1$
```

Theorem 7.3.1. *Algorithm* GREEDY *is* $\left(\frac{k}{k+d-1}\right)$*-competitive for the unweighted, vertex-weighted maximum matching and equal-bids ad allocation problems on* (k,d)*-graphs.*

Theorem 7.3.2. *Algorithm* GREEDY *is* $\frac{(1-R_{\max})\cdot k}{k+(d-1)\cdot(1-R_{\max})} > (1-R_{\max}) \cdot \frac{k}{k+d-1}$ *competitive for online ad allocation on* (k,d)*-graphs with* $k \geqslant 1$ *and* $R_{\max} = \max_{(i,j)\in E}\{b_{ij}/B_i\} < 1$.

Proof of Theorems 7.3.1 & 7.3.2. To prove these theorems, we prove the following claims:

(a) z, y form a feasible dual solution.
(b) for every ad slot $j \in R$ the changes to the primal and dual solutions' values due to j's arrival, ΔP and ΔD, satisfy $\Delta D / \Delta P \leqslant \frac{k+d-1}{k}$.
(c) for the vertex-weighted and unweighed matching problems and equal-bids problem lines 8-10 incur no dual cost.
(d) for the general ad allocation problem lines 8-10 cost the dual solution no more than $R_{\max}/(1-R_{\max})$ times the primal profit.

Before proving these claims, we show how they imply the above two theorems.

As x forms an integral feasible primal solution, claims (a,b,c) combined imply Theorem 7.3.1. Similarly, claims (a,b,d) imply Theorem 7.3.2, as claims (b) and (d) imply the ratio of the solutions' overall values is at least

$$\frac{P}{D} \geqslant \frac{P}{\frac{k+d-1}{k} \cdot P + \frac{R_{\max}}{1-R_{\max}} \cdot P} = \frac{(1-R_{\max}) \cdot k}{k + (d-1) \cdot (1-R_{\max})}.$$

Claim (a): For every advertiser $i \in L$, if over a $(1-R_{\max})$-fraction of i's budget is spent then z_i is set to one in Line 10. Otherwise, i is a feasible match of all of its neighbors j, each such j causing z_i to increase by at least $\frac{b_{ij}}{k \cdot B_i}$. As $\sum_j b_{ij} \geqslant k \cdot B_i$ then $z_i = 1$ by the algorithm's termination. Consequently, all dual inequalities are satisfied.

Claim (b): Consider an ad slot $j \in R$ with set F_j of feasible (unmatched) neighbors. If j is unmatched, then clearly $\Delta P = \Delta D = 0$. If j is matched, then by the choice of j's match i and the bound on $j's$ degree, $d(j) \leqslant d$, the primal value increases by $\Delta P = b_{ij}$ while the dual cost increases by at most $\Delta D = b_{ij} + \sum_{i' \in F_j \setminus \{i\}} b_{i'j}/k \leqslant b_{ij} \cdot (1 + \frac{d-1}{k})$.

Claim (c): For an advertiser $i \in L$ to have spent over $(1 - R_{\max})B_i$ for all but the general problem, it must and have z_i set to one. Thus lines 8-10 incur no dual cost.

Claim (d): For an advertiser $i \in L$ to be affected by lines 8–10, it must spend up to a $(1 - R_{\max})$-fraction of its budget. However, whenever i spends an f-fraction of its budget, the dual variable z_i increases by f in Line 6, and so the cost of increasing z_i in line 10 is at most $R_{\max} \cdot B_i$, while i garnered a primal profit of at least $(1 - R_{\max}) \cdot B_i$. The total dual cost of lines 8–10 is thus at most $\frac{R_{\max}}{1-R_{\max}} \cdot P$, for P the primal profit. □

7.3.1 Tight Examples for Algorithm GREEDY

We show that our analysis of algorithm GREEDY for the unweighted and vertex-weighted matching is tight whenever $k \geqslant d - 1$.[1]

Theorem 7.3.3. *For all $k \geqslant d - 1$ there exist $(k, d)-$graphs G with maximal matchings that are $\frac{k}{k+d-1}$-competitive in G.*

Proof. The tight example, along with a poor choice of matching, consists of $k+d-1$ advertisers and $k^2 + k$ ad slots. We denote by $M_L = \{i_1, i_2, \ldots, i_k\}$ and $U_L = \{i_{k+1}, i_{k+2}, \ldots, i_{k+d-1}\}$ the advertisers that will be matched and not matched, respectively. The first k ad slots by order of arrival, $j_1, j_2, \ldots, j_k$, each have degree exactly d, with the t-th ad slot j_t neighboring the t-th advertiser i_t, to which it is matched, as well as the $d - 1$ advertisers in U_L. (The first k ad slots and the advertisers in U_L form a copy of $K_{k,d-1}$. See Figure 7.2.) After these k arrivals, a further k^2 ad slots arrive, each with a single neighbor in M_L, each advertiser in M_L having k neighbors among these k^2 ad slots. The resulting graph is clearly a $(k, d)-$graph in which k advertisers are matched, though all $k + d - 1$ advertisers can be matched simultaneously, by matching the $d - 1$ advertisers in U_L to some $d - 1$ of the first $k \geqslant d - 1$ ad slots and each of the advertisers in M_L to one of their distinct $k(\geqslant 1)$ neighbors. □

For any $R_{\max}$ a unit fraction (i.e., the reciprocal of an integer), gluing $1/R_{\max}$ copies of the above tight example at the advertisers, with each advertiser having a budget of $1/R_{\max}$, yields an equal-bid ad allocation instance and greedy allocation for which the same $\frac{k}{k+d-1}$ performance holds, proving tightness of our analysis for equal-bid allocations. We now state a theorem implying our analysis' tightness for GREEDY in general ad allocations.

[1]For $k < d - 1$ the $\frac{k}{k+d-1}$ bound is strictly less than the $\frac{1}{2}$ bound obtained by all maximal matchings, and so the bound cannot be tight for $k < d - 1$. We therefore turn our attention to the case $k \geqslant d - 1$.

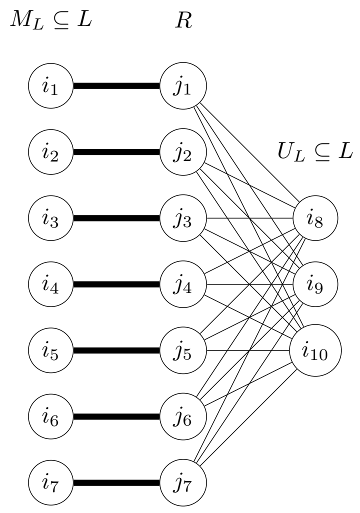


Figure 7.2: Tight Example for Greedy

Depicted are the graph and the matching (in **bold**) after the first k ad slots' arrivals, for $k = 7$ and $d = 4$.

Theorem 7.3.4. *For all $k \geqslant d - 1$ and $R_{\max} = \frac{1}{c}$ with $c \geqslant 2$ an integer, there exist (k, d) ad allocation instances for which algorithm* GREEDY *can achieve competitive ratio exactly* $\frac{(1-R_{\max})\cdot k}{k+(d-1)\cdot(1-R_{\max})}$.

Proof. Let $1 - R_{\max} = \frac{a}{b}$ for $0 < a < b$ and a and b integers. The hard instance will consist of $k \cdot b + (d-1) \cdot a = b \cdot (k + (d-1) \cdot (1 - R_{\max}))$ advertisers. Each advertiser has budget exactly one. We designate $k \cdot b$ advertisers to be the "lucky" advertisers, from which we will achieve revenue of $(1 - R_{\max})$ and the remaining $(d-1) \cdot a$ "unlucky" advertisers will garner no profit. The theorem will follow by constructing the instance such that all budgets can be exhausted simultaneously.

All edges have bids either $R_{\max}$ or some arbitrarily small positive ε. At first, each arriving ad slot will have d edges with bids $R_{\max}$, one to some lucky advertiser of lowest degree (to whom the ad slot is matched), and $(d-1)$ edges to some unlucky advertisers of lowest degree. After $\frac{a \cdot k \cdot b}{b-a}$ ad slots arrive (this value is integral, as is $\frac{1}{R_{\max}} = \frac{b}{b-a}$), the following holds

(i) every unlucky advertiser is unmatched and has degree exactly $k \cdot \frac{b}{b-a} = k \cdot \frac{1}{R_{\max}}$.

(ii) each of the lucky advertisers are matched to all of their neighbors, and have degree exactly $\frac{a}{b-a} = \frac{1}{R_{\max}} - 1$.

The remaining ad slots recreate the construction of Lemma 7.6.4, thus guaranteeing each of the lucky advertisers gain no more than $1 - R_{\max} + \varepsilon$. On the other hand all the lucky advertisers can exhaust their budgets without using any of the $R_{\max}$-bid edges of ad slots neighboring unlucky advertisers, which, as can be readily verified (using, e.g. Observation 7.6.1 repeatedly), allows both lucky and unlucky advertisers to exhaust their budgets simultaneously whenever $k \geqslant d - 1$. The described instance is (k, d) and the theorem follows. $\square$

The above bound holds for *any* $R_{\max} \leqslant \frac{1}{2}$, as the following theorem asserts.

Theorem 7.3.5. *For all $k \geqslant d-1$ and $R_{\max} \leqslant \frac{1}{2}$ there exist (k,d) ad allocation instances for which algorithm* GREEDY *can achieve competitive ratio exactly $\frac{(1-R_{\max})\cdot k}{k+(d-1)\cdot(1-R_{\max})}$.*

Proof (sketch). In order to generalize the above, we rely on the fact that every number $R_{\max}$ in the range $(0, \frac{1}{2}]$ can be written as a convex combination of two unit fractions, $\frac{1}{a}$ and $\frac{1}{b}$, with $R_{\max} \in [\frac{1}{a}, \frac{1}{b}]$. That is, $w_a \cdot \frac{1}{a} + w_b \cdot \frac{1}{b} = R_{\max}$ and $w_a + w_b = 1$. We glue $2n$ copies of the above construction at the advertisers, n of the copies with budget w_a/n (w_b/n) for each advertiser, and highest bid-to-budget ratio in the copy being $1/a$ (resp. $1/b$). In this case the overall budget from all copies is $n \cdot (w_a/n + w_b/n) = w_a + w_b = 1$, and for large enough n each bid is at most $w_b/(b \cdot n) < R_{\max}$. On the other hand, all unlucky advertisers are completely unmatched and garner no profit, and all lucky advertisers gain a total of $n \cdot (w_a/n + w_b/n - w_a/(a \cdot n) + w_b/(b \cdot n)) = 1 - (w_a/a + w_b/b) = 1 - R_{\max}$. As in Lemma 7.6.4, we can guarantee each such lucky advertiser yields at most ε additional revenue. □

7.4 Optimal Vertex-Weighted Matching on (k, d)-graphs

The previous section shows our analysis of GREEDY is tight, though for a particular(ly bad) input and instantiation of the algorithm. The family of tight examples suggests the following improved algorithm: match every arriving ad slot to an unmatched neighbor of highest degree. This algorithm, which we call HIGH-DEGREE, is given below. The intuition behind this algorithm, substantiated by the above examples, is that unmatched advertisers with higher degree may have fewer chances to be matched later. This approach fares better on the above examples (actually yielding an *optimal* solution), but can it do better than GREEDY on *all* $(k,d)-$graphs? We answer this question in the affirmative, proving a lower bound with *exponentially* smaller loss. In Section 7.6 we prove a matching lower bound, implying the algorithm's *optimality*.

Algorithm 9 HIGH-DEGREE

```
1: for all $j \in R$ do
2:     if $j$ has an unmatched neighbor then
3:         match $j$ to unmatched neighbor of highest degree
```

7.4.1 Analysis of HIGH-DEGREE

In this section we analyze algorithm HIGH-DEGREE, and prove the following bound on its competitive ratio.

Theorem 7.4.1. *Algorithm* HIGH-DEGREE *is $1-(1-\frac{1}{d})^k$ competitive for all $(k,d)-$graphs.*

A corollary of Theorem 7.4.1 is the first result for maximum online matching in regular graphs in the adversarial setting, beating the $1-\frac{1}{e}$ "barrier" *deterministically*.

Corollary 7.4.2. *On d-regular graphs algorithm* HIGH-DEGREE *is $1-(1-\frac{1}{d})^d$ competitive.*

Theorem 7.4.1 can be proven directly (see Section 7.4.2), but in order to set the groundwork for proofs of our more general results, we generalize this algorithm and rewrite it as a primal-dual algorithm. This is Algorithm 10, below. The constant C will be chosen during the analysis.

Algorithm 10 Vertex-Weighted HIGH-DEGREE (Primal-Dual Formulation)

```
1: Init: set $z_i \leftarrow 0$ for all $i \in L$ and $y_j \leftarrow 0$ for all $j \in R$
2: for all $j \in R$ do
3:     if $j$ has an unmatched neighbor $i$ then
4:         match $j$ to an unmatched neighbor $i$ maximizing $(z_i + C) \cdot b_{ij}$
5:         set $x_{ij} \leftarrow 1$
6:         set $z_i \leftarrow 1$
7:         set $z_{i'} \leftarrow \min\{1, z_{i'} \cdot \left(\frac{d}{d-1}\right) + \frac{1}{d-1} \cdot C\}$ for every feasible neighbor of $j$, $i' \neq i$
```

Theorem 7.4.3. *Algorithm 10 generalizes* HIGH-DEGREE *and is $1-\left(1-\frac{1}{d}\right)^k$ competitive. Moreover, it gains revenue at least $\left(1-\left(1-\frac{1}{d}\right)^k\right) \cdot \left(\sum_i B_i\right)$.*

Proof. We rely on the following observation, verifiable by induction: All unmatched advertisers i satisfy $z_i = C \cdot (\left(\frac{d}{d-1}\right)^{d(i)} - 1)$. Hence Algorithm 10 matches each ad slot j to an unmatched neighbor i maximizing $b_{ij} \cdot C \cdot \left(\frac{d}{d-1}\right)^{d(i)}$. For the unweighted problem, $b_{ij} = 1$. By monotonicity of exponentiation, picking such i is tantamount to picking an advertiser of highest degree. We proceed to bound the algorithm's gain.

Let $j \in R$ be some ad slot matched to i. The incurred change to the primal profit equals $\Delta P = b_{ij}$. By our choice of j's match, the change to the dual cost satisfies

$$\begin{aligned}\Delta D &= (1-z_i) \cdot b_{ij} + \textstyle\sum_{i' \in N(j)\setminus\{i\}} \left(\left(\frac{1}{d-1}\right) \cdot (z_{i'} + C) \cdot b_{i'j}\right) \\ &\leqslant (1-z_i) \cdot b_{ij} + (d-1) \cdot \left(\frac{1}{d-1}\right) \cdot (z_i + C) \cdot b_{ij} \\ &= (1+C) \cdot b_{ij}.\end{aligned}$$

Given dual feasibility, the above would imply a competitive ratio of $1/(1+C)$. Hence, we choose the minimal C ensuring $z_i = 1$ by the algorithm's end for all advertisers i (matched and unmatched alike). Recall all unmatched advertisers i satisfy $z_i = C \cdot ((\frac{d}{d-1})^{d(i)} - 1)$. As such i have degree at least k by the algorithm's end (but possibly no higher), the minimal C ensuring $z_i = 1$ is $C = 1/((\frac{d}{d-1})^k - 1)$. As the dual solution has $z_i = 1$ for all i by the algorithm's termination, the dual cost is exactly $D = \sum_{i \in L} B_i$. Consequently, the primal gain satisfies $P \geqslant \frac{1}{1+C} \cdot \left(\sum_i B_i\right)$. The theorem follows. ☐

The above algorithm implies structural Corollary 7.1.4 and the following corollary.

Corollary 7.4.4. *For (k,d)-graphs with $k \geqslant d \cdot \ln |L|$, by integrality of number of vertices matched,* HIGH-DEGREE *successfully matches* all *of L, obtaining a* maximum *matching.*

We can extend our analysis to handle the possible existence of *outlying* advertisers i, that do not satisfy $\sum_j b_{ij} \geqslant k \cdot B_i$, and so may not satisfy $z_i = 1$, ruining dual feasibility. Let $S \subseteq L$ be the set of outlying advertisers, and assume $\sum_{i\in S} B_i \leqslant \alpha \cdot \sum_{i\in L} B_i$. As $z_i = 1$ for all $i \notin S$, we have $D \geqslant (1-\alpha)\cdot \sum_{i\in L} B_i$, implying the following theorem.

Theorem 7.4.5 (Outliers)**.** *Let $S \subseteq L$ be the set of outlying advertisers, and α be a real number such that $\sum_{i\in S} B_i \leqslant \alpha \cdot \sum_{i\in L}$. Then Algorithm 10 gains at least $(1-\alpha)\cdot(1-\left(1-\frac{1}{d}\right)^k)\cdot\sum_i B_i$, and in particular is $(1-\alpha)\cdot(1-\left(1-\frac{1}{d}\right)^k)$-competitive.*

7.4.2 Potential-based Analysis of HIGH-DEGREE

In this subsection we present a potential-based proof of Theorem 7.4.1. We note that this proof can easily be extended to provide alternative proofs of theorems 7.4.3 and 7.4.5.

Theorem 7.4.6. *Algorithm* HIGH-DEGREE *achieves value at least $\left(1-(1-\frac{1}{d})^k\right)\cdot|L|$ for all $(k,d)-$graphs $G=(L,R,E)$, and it is therefore $\left(1-(1-\frac{1}{d})^k\right)$-competitive.*

Proof. Let $U_L \subseteq L$ denote the set of unmatched advertisers. Consider the following potential:

$$\Phi = \sum_{i\in U_L} \left(\frac{d}{d-1}\right)^{d(i)}.$$

Algorithm HIGH-DEGREE outputs a matching that effectively strives to greedily minimize Φ.[2] The initial and final values of the potential function hold $\Phi_{start} = |L|$ and $\Phi_{final} \geqslant (d/(d-1))^k \cdot |U_L|$, respectively. Denote by $\Delta\Phi_j$ the change to Φ incurred by the arrival of ad slot $j \in R$. Clearly, if j is unmatched we have $\Delta\Phi_j = 0$. On the other hand, if j is matched to a neighbor i, previously of degree $d(i)$, we find that i's matching results in Φ decreasing by $(d/(d-1))^{d(i)}$, and the degree of j's remaining unmatched neighbors increase each cause Φ to increase by at most $(d/(d-1))^{d(i)+1} - (d/(d-1))^{d(i)}$. Therefore, if j is matched to i we have

$$\begin{aligned}\Delta\Phi_j &\leqslant -\left(\tfrac{d}{d-1}\right)^{d(i)} + (d-1)\cdot\left(\left(\tfrac{d}{d-1}\right)^{d(i)}\cdot\left(\tfrac{d}{d-1}-1\right)\right)\\ &= -\left(\tfrac{d}{d-1}\right)^{d(i)} + \left(\tfrac{d}{d-1}\right)^{d(i)} = 0.\end{aligned}$$

[2]The sum of unmatched advertisers' degrees may seem like a more natural potential function to consider, but it turns out that it cannot be used to derive tight bounds. E.g., it does not yield a bound significantly better than $\frac{5}{8} = 0.625$ for $k = d$.

In other words $\Delta\Phi_j \leqslant 0$, irrespective of whether or not j is matched. By this fact and our bounds on the initial and final potential, we find that

$$\left(\frac{d}{d-1}\right)^k \cdot |U_L| \leqslant \Phi_{final} \leqslant \Phi_{start} = |L|.$$

The theorem follows. $\square$

7.5 Online Ad Allocation

In this section we solve the ad allocation problem. We consider first the equal-bids case, where each advertiser i offers the same bid for all its neighbors; i.e., $b_{ij} = b_i \ \forall j \in N(i)$. This will prove to be a useful stepping-stone towards a solution for general bids, in Section 7.5.1.

One way to solve equal-bids ad allocation is via an online reduction to vertex-weighted matching in (k,d)-graphs. As each advertiser i bids $\sum_{j\in N(i)} b_i \geqslant k \cdot B_i$ in total, we have $d(i) \geqslant k \cdot B_i/b_i$. Without loss of generality, B_i/b_i is integral. The reduction splits each i into B_i/b_i copies, each of value b_i and receiving up to k distinct edges of i, stopping if the copy is matched. The obtained graph G is (k,d)-bounded (perhaps after adding inconsequential neighbors to matched advertisers), and matchings in G induce allocations of same value for the ad allocation instance. As Algorithm 10 gains $1-(1-\frac{1}{d})^k$ of the sum of vertex weights, or equivalently the sum of budgets, applying it yields a $1-(1-\frac{1}{d})^k$ competitive solution to the original ad allocation instance.

We restate the above as a primal-dual algorithm for equal-bids ad allocation. (See Algorithm 11 below). In this algorithm, z_i^c serves the role of z_i in Algorithm 10 for i's "current copy" (hence the c in the notation), weighted to reflect the copy contributes b_i/B_i of i's budget. Intuitively, when i is matched we imagine its current copy is matched, and set z_i^c to b_i/B_i. Conversely, we ensure that once the copy has k edges $z_i^c = b_i/B_i$. Either way, once $z_i^c = b_i/B_i$, we add z_i^c to z_i and nullify z_i^c (moving to i's next copy, whose dual variable would be zero in Algorithm 10.) The number of copies of i guarantees dual feasibility and the choice of match and dual updates guarantee the desired bound.

Theorem 7.5.1. *Algorithm 11 with $C = 1/\left(\left(\frac{d}{d-1}\right)^k - 1\right)$ gains revenue $\left(1-(1-\frac{1}{d})^k\right) \cdot \sum_i B_i$, and is thus $\left(1-(1-\frac{1}{d})^k\right)$-competitive for the equal-bid problem on (k,d)-graphs.*

Proof. To bound the primal-dual ratio, we bound increases of $z_i^c \cdot B_i$, as all dual costs can be traced back to past increases of z_i^c. Consider some ad slot j matched to i. The primal gain is $\Delta P = b_i$, whereas the dual cost satisfies

$$\begin{aligned}\boldsymbol{\Delta D} &\leqslant (b_i/B_i - z_i^c)\cdot B_i + \textstyle\sum_{i'\in N(j)\setminus\{i\}} \left(\frac{1}{d-1}\right)\cdot(z_{i'}^c + C\cdot b_{i'}/B_{i'})\cdot B_{i'} \\ &\leqslant b_i - z_i^c \cdot B_i + (d-1)\cdot\left(\frac{1}{d-1}\right)\cdot(z_i^c\cdot B_i + C\cdot b_i) \boldsymbol{\leqslant (1+C)\cdot b_i}.\end{aligned}$$

As in Theorem 7.4.3's proof, $z_i^c = C\cdot\frac{b_i}{B_i}\left(\left(\frac{d}{d-1}\right)^{d^c(i)} - 1\right)$, where $d^c(i)$ is the degree of i's current copy, or equivalently, the number of i's edges since z_i^c was last nullified. Hence, by our choice

Algorithm 11 Equal-Bid Ad Allocation in (k,d)-graphs

```
1: Init: set $z_i \leftarrow 0$, $z_i^c \leftarrow 0$ for all $i \in L$ and $y_j \leftarrow 0$ for all $j \in R$
2: for all $j \in R$ do
3:     if $j$ has a feasible neighbor $i$ then
4:         match $j$ to feasible neighbor $i$ maximizing $z_i^c \cdot B_i + C \cdot b_i$
5:         set $x_{i,j} \leftarrow 1$
6:         set $z_i^c \leftarrow b_i/B_i$
7:         for all feasible neighbor of $j$, $i' \neq i$ do
8:             set $z_{i'}^c \leftarrow \min\{b_{i'}/B_{i'},\ z_{i'}^c \cdot \left(\frac{d}{d-1}\right) + \frac{1}{d-1} \cdot C \cdot b_{i'}/B_{i'}\}$
9:         for $i' \in N(j)$ with $z_{i'}^c = b_{i'}/B_{i'}$ do
10:            set $z_{i'} \leftarrow z_{i'} + z_{i'}^c$
11:            set $z_{i'}^c \leftarrow 0$
```

of C, after at most k i-edges, $z_i^c = \frac{b_i}{B_i}$ (whether or not i is matched), and z_i is increased by $\frac{b_i}{B_i}$. As $d(i) \geqslant k \cdot \frac{B_i}{b_i}$ by the end, $z_i \geqslant 1$ for all i. The theorem follows. $\square$

7.5.1 General Bids

A natural way to extend Algorithm 11 to general bids would be to replace for every ad slot j and every neighbor i (or i') all appearances of b_i (or $b_{i'}$) by b_{ij} (resp., $b_{i'j}$) in the choice of j's match and updates to z_i^c, $z_{i'}^c$ and z_i. Such dual updates would guarantee, similarly to our prior algorithms, that an advertiser i with budget B_i and rejected bids $b_{i0}, b_{i1}, \ldots, b_{it}$ since its last match (ordered chronologically) would have dual variable

$$z_i^c = \frac{1}{d-1} \cdot C \cdot \sum_{r=0}^{t} \frac{b_{ir}}{B_i} \cdot \left(\frac{d}{d-1}\right)^{t-r}. \tag{7.1}$$

Unfortunately, replacing b_i by b_{ij} in the updates for matched i could result in z_i arbitrarily small. Worse still, since previously-rejected bids may be greater than the current bid, setting z_i^c to $\frac{b_{ij}}{B_i}$ could even *decrease* z_i^c, complicating the task of bounding the primal-dual ratio. Algorithm 12 below sidesteps these issues by considering bounded fractions of z_i^c, and using the following notation, motivated by Equation (7.1), to represent variables z_i^c, and z_i^f (the f in the notation refers to a bounded fraction of z_i^c "used"). This notation's use will become apparent during the algorithm's analysis.

Definition 7.5.2. *Let $z = \frac{1}{d-1} \cdot C \cdot \sum_{r=0}^{t} b_r \cdot \left(\frac{d}{d-1}\right)^r$. We think of z as a number in base $\frac{d}{d-1}$, denoting it by $z = [b_t, \ldots, b_1, b_0]$, disregarding the $\frac{1}{d-1} \cdot C$ term for simplicity. Addition and subtraction of numbers in this notation is done place-wise, disallowing carries/borrows. In particular, if $z = [b_t, \ldots, b_1, b_0]$, then $z \cdot \frac{d}{d-1} + \frac{1}{d-1} \cdot C \cdot b = [b_t, \ldots, b_1, b_0, b]$. Comparisons involving numbers in this notation refer to their numerical value.*

The algorithm for the general bids setting is Algorithm 12, below. The algorithm's primal feasibility is trivial, as is its dual feasibility, due to linesline:feasibilityLoop-23. It remains to bound the ratio of the cost of the dual solution to the value of the primal solution.

Algorithm 12 Online Ad Allocation in (k, d)-graphs with general bids.

```
1: Init: set $z_i \leftarrow 0$ , $z_i^c \leftarrow 0$ for all $i \in L$ and $y_j \leftarrow 0$ for all $j \in R$
2: for all $j \in R$ do
3:     if $j$ has a feasible neighbor $i$ then
4:         for all feasible neighbors $i$ do
5:             let $z_i^c = [b_{k-1}, \ldots, b_1, b_0]$
6:             set $z_i^f \leftarrow [\min\{b_{k-1}, b_{ij}/B_i\}, \ldots, \min\{b_1, b_{ij}/B_i\}, \min\{b_0, b_{ij}/B_i\}]$
7:             set $z_i^c \leftarrow z_i^c - z_i^f$
8:         match $j$ to feasible neighbor $i$ maximizing $z_i^f \cdot B_i + C \cdot b_{ij}$
9:         set $x_{i,j} \leftarrow 1$
10:        set $z_i^f \leftarrow 0$
11:        set $z_i \leftarrow z_i + b_{ij}/B_i$
12:        for all feasible neighbor of $j$, $i' \neq i$ do
13:            set $z_{i'}^f \leftarrow z_{i'}^f \cdot \left(\frac{d}{d-1}\right) + \frac{1}{d-1} \cdot C \cdot b_{i'j}/B_{i'}$
14:            $z_{i'}^c \leftarrow z_{i'}^c + z_{i'}^f$
15:            if $z_{i'}^c = [b_k, b_{k-1}, \ldots, b_1, b_0]$ with $b_k \neq 0$ then
16:                set $z_{i'} \leftarrow z_{i'} + b_k \cdot \frac{1}{k}$
17:                set $z_{i'}^c \leftarrow [b_{k-1}, \ldots, b_1, b_0]$
18:            if $z_{i'}^c = [b_{k-1}, \ldots, b_1, b_0]$ with all digits $b_r \neq 0$ then
19:                let $b = \min\{b_r\}_{r=0}^{k-1}$
20:                set $z_{i'} \leftarrow z_{i'} + b$
21:                set $z_{i'}^c \leftarrow [b_{k-1} - b, \ldots, b_1 - b, b_0 - b]$
22: for all $i \in L$ do
23:     set $z_i \leftarrow \max\{1, z_i\}$
```

High-Level intuition:: The algorithm asserts three invariants. The first guarantees increases in z_i are "paid for" by increases in z_i^c, allowing us to focus on bounding changes to z_i^c. A second invariant guarantees every increase of z_i by some value b/B_i can be accredited to previous bids (or fractions thereof) of total value at most $k \cdot b/B_i$. As the graph is (k, d), if every bid of i of value b were to cause z_i to increase (by at least $b/(k \cdot B_i)$, by the above), then eventually $z_i \geqslant 1$. However, some bids may not incur an increase in z_i. The third and last invariant guarantees the total value of fractions of bids that do not cause z_i to increase is at most $k \cdot R_{\max}$, and so $z_i \geqslant (1 - R_{\max})$ before lines 22-23. Thus, the cost of rounding each z_i to one in these lines is at most $R_{\max}/(1 - R_{\max})$ of the previously-paid dual cost. The bound will follow. The following four lemmas formalize the above, allowing us to derive our sought-after bound.

Lemma 7.5.3. *Before every ad slot's arrival and before Line 22, every z_i^c is a number in the above numeral system satisfying the following three properties:*

(i) *z_i^c is a k-digit number; i.e., $z_i^c = [b_{k-1}, \dots, b_1, b_0]$.*

(ii) *z_i^c has at most $k-1$ non-null digits.*

(iii) *Each digit of z_i^c is no greater than $\max_j\{\frac{b_{ij}}{B_i}\}$.*

Proof. Properties (i) and (ii) are enforced explicitly by lines 15-17 and 18-21, respectively. Property (iii) follows by induction: When z_i^f is subtracted from z_i^c, every digit of z_i^c is either nullified, if it was smaller than b_{ij}/B_i, or decreased by b_{ij}/B_i. After z_i^f is updated and added to z_i^c, each digit of z_i^c is increased by at most b_{ij}/B_i. Thus each digit is no greater than its previous value and b_{ij}/B_i, both of which are at most $\max_j\{\frac{b_{ij}}{B_i}\}$. □

Lemma 7.5.4. *If $k \geqslant d-1$ and $C = 1/\big(\big(\frac{d}{d-1}\big)^k - 1\big)$, every increase in z_i by some b in lines 15-17 and 18-21 goes hand-in-hand with both*

(i) *a decrease of the same value or higher in z_i^c, and*

(ii) *a decrease of k times this value or less in the sum of digits of z_i^c.*

Proof. In lines 15-17, z_i is increased by b_k/k. On the other hand, we remove b_k, the k-th digit of z_i^c in this numeral system, resulting in a decrease of z_i^c by

$$\frac{1}{d-1} \cdot C \cdot b_k \cdot \left(\frac{d}{d-1}\right)^k \geqslant \frac{1}{k} \cdot b_k.$$

Thus, properties (i) and (ii) both hold for lines 15-17. In linesline:k-1-digit-loop-21, the value of z_i^c is decreased by $\frac{1}{d-1} \cdot C \cdot \sum_{r=0}^{k-1} b \cdot \big(\frac{d}{d-1}\big)^r = C \cdot \big(\big(\frac{d}{d-1}\big)^k - 1\big) \cdot b$, which is exactly b, by our choice of C. The decrease in the sum of digits of z_i^c on the other hand is exactly $k \cdot b$. □

Lemma 7.5.5. *Taking $C = 1/\big(\big(\frac{d}{d-1}\big)^k - 1\big)$ guarantees every increase in z_i by b_{ij}/B_i in Line 11 coincides with a decrease of at most b_{ij}/B_i in z_i^c. Moreover, $\Delta digit$, the decrease in sum of digits of z_i^c, satisfies $\Delta digit + b_{ij}/B_i \leqslant k \cdot b_{ij}/B_i$.*

Proof. In Line 11, z_i^f, which was subtracted from z_i^c, is nullified. Both bounds follow similarly to our proof of Lemma 7.5.4 relying on z_i^f being a k-digit number with at most $k-1$ non-null digits, by Lemma 7.5.3, and each digit of z_i^f being no greater than b_{ij}/B_i, by initialization of z_i^f. □

Lemma 7.5.6. *By Line 22 each i satisfies*

$$z_i \geqslant \frac{\sum_j b_{ij} - k \cdot \max_j\{b_{ij}\}}{k \cdot B_i} \geqslant 1 - \frac{\max_j b_{ij}}{B_i} \geqslant 1 - R_{\max}.$$

Proof. Throughout the algorithm, every edge (i,j) causes the sum of digits of z_i^c to increase by b_{ij}/B_i (again ignoring the $\frac{1}{d-1} \cdot C$ term), unless (i,j) are matched. Moreover, the sum of digits does not decrease due to carries. On the other hand, every increase in z_i by b coincides with a decrease in the sum of digits of z_i^c plus $\sum_{(i,j) \text{ matched}} b_{ij}/B_i$, of at most $k \cdot b$, by lemmas 7.5.4 and 7.5.5. Put otherwise, the increase in z_i is at least $1/k$ times the total sum of i's bids so far, minus the sum of digits of z_i^c. By Lemma 7.5.3, the sum of digits of z_i^c by Line 22 cannot exceed $k \cdot \max_j\{b_{ij}/B_i\}$. The lemma follows. □

Given the above we can now prove our main result.

Theorem 7.5.7. *On general-bid ad allocations on (k,d)-graphs with $k \geqslant d-1$ Algorithm 12 gains $\sum_i \left(B_i - \max_j b_{ij}\right) \cdot \left(1 - \left(1 - \frac{1}{d}\right)^k\right)$, and is thus $\left(1 - R_{\max}\right) \cdot \left(1 - \left(1 - \frac{1}{d}\right)^k\right)$-competitive.*

Proof. lemmas 7.5.4 and 7.5.5 imply increases in z_i can be traced back to a previous increase in z_i^c of the same value or higher. We therefore bound increases of $z_i^c \cdot B_i$ in order to bound the total dual cost. For each online $j \in R$, by our choice of match i, the change to the dual cost is at most $(1+C)$ times the change to the primal value, as in Algorithm 11. However, by Lemma 7.5.6, by Line 22 each i satisfies $z_i \geqslant (1 - \max_j b_{ij}/B_i)$. Consequently, we have that before Line 22 the primal value P and dual cost D satisfy

$$P \geqslant \frac{1}{1+C} \cdot D \geqslant \sum_i \left(B_i - \max_{j \in N(i)} b_{ij}\right) \cdot \left(1 - \left(1 - \frac{1}{d}\right)^k\right).$$

As the primal value is unaffected by linesline:feasibilityLoop-23, P above is our algorithm's gain. The competitive ratio follows from $OPT \leqslant \sum_i B_i$ and the definition of $R_{\max}$. □

Finally, we note that lemmas 7.5.3,7.5.4,7.5.5 and 7.5.6 hold for all advertisers i satisfying $\sum_j b_{ij} \geqslant k \cdot B_i$, irrespective of outliers who don't hold this property, implying the following.

Theorem 7.5.8 (Outliers)**.** *Let $S \subseteq L$ be the set of outlying advertisers (advertisers i with $\sum_j b_{ij} < k \cdot B_i$), and α be such that $\sum_{i \in S} B_i \leqslant \alpha \cdot \sum_{i \in L}$. Then Algorithm 10 gains at least*

$$(1-\alpha) \cdot \left(\left(1 - R_{\max}\right) \cdot \left(1 - \left(1 - \frac{1}{d}\right)^k\right)\right) \cdot \sum_i B_i,$$

and in particular it is $(1-\alpha) \cdot \left(\left(1 - R_{\max}\right) \cdot \left(1 - \left(1 - \frac{1}{d}\right)^k\right)\right)$-competitive.

7.6 Lower Bounds for Deterministic Algorithms

In this section we present lower bounds for deterministic algorithms. In particular, we present matching lower bounds for the unweighted matching problem, proving optimality of HIGH-DEGREE among deterministic algorithms.

7.6.1 Maximum Matching

In order to construct hard examples, we start by showing that the optimal matching in $(k, d)-$graphs matches *all* the advertisers whenever $k \geqslant d$.

Observation 7.6.1. *Every $(k, d)-$graph $G = (L, R, E)$ with $k \geqslant d$ has a matching matching all of L.*

Proof. By Hall's Theorem G has a matching with all of L matched if and only if every subset $A \subseteq L$ satisfies $|\Gamma(A)| \geqslant |A|$. But, as G is a $(k, d)-$graph we have

$$k \cdot |A| \leqslant |E[G[A]]| \leqslant d \cdot |\Gamma(A)|.$$

Consequently, we find that $|\Gamma(A)| \geqslant \frac{k}{d} \cdot |A| \geqslant |A|$, and the lemma follows. $\square$

Equipped with Observation 7.6.1 we may now prove this section's main result – a lower bound matching the upper bounds of Section 7.4, implying algorithm HIGH-DEGREE's optimality. To this end we cause HIGH-DEGREE to be effectively indistinguishable from any other algorithm.

Theorem 7.6.2. *For all $k \geqslant d$ no deterministic online algorithm for bipartite matching can achieve competitive ratio better than $1 - (1 - \frac{1}{d})^k$ on $(k, d)-$graphs.*

Proof. Let $\mathcal{A}$ be some online matching algorithm. The adversarial input consists of d^{k+1} advertisers, with the ad slots arriving in k phases, numbered 0 to $k-1$. During the i-th phase, which begins with $d^{k+1} \cdot (1 - \frac{1}{d})^i$ unmatched advertisers each of degree i, the arriving ad slots each have exactly d neighbors, all unmatched; every unmatched advertiser neighbors exactly one new ad slot per phase. Every phase causes unmatched advertisers to have their degree increase by one, and exactly a $(1 - \frac{1}{d})$-fraction of the advertisers unmatched at the phase's beginning remain unmatched. (If algorithm $\mathcal{A}$ does not match some ad slot to one of its d unmatched neighbors, we consider it matched to an arbitrary neighbor; this can only serve to improve $\mathcal{A}$'s performance.) After the k phases additional ad slots of degree exactly d arrive in order to increase the degree of the matched advertisers to k. The resulting graph is k-regular and d-regular on the offline and online sides respectively, and is thus a $(k, d)-$graph. Moreover, exactly $d^{k+1} \cdot (1 - \frac{1}{d})^k$ of the d^{k+1} advertisers are unmatched. However, by Observation 7.6.1 all d^{k+1} advertisers can be matched simultaneously. The theorem follows. $\square$

Recall from Chapter 6 that for Δ-regular graphs, randomized algorithms can achieve a competitive ratio tending to *one* as Δ increases. The following corollary of Theorem 7.6.2 implies that the same is not true of deterministic algorithms, and indeed the problem becomes *harder* as Δ increases, tending to an optimal competitive ratio of $1 - \frac{1}{e}$.

Corollary 7.6.3. *The bound of Theorem 7.6.2 holds for Δ-regular graphs with $\Delta^{\Delta+1} \leqslant n$, where $n = |L| = |R|$. In particular, for Δ-regular graphs with $\Delta = O\left(\frac{\log n}{\log \log n}\right)$, no deterministic algorithm has higher competitive ratio than the $1 - (1 - \frac{1}{\Delta})^{\Delta}$ achieved by algorithm* HIGH-DEGREE.

7.6.2 Lower Bound for Ad Allocation

In this subsection we prove a lower bound for deterministic ad allocation algorithms in (k, d)-graphs. We start by showing a simple weaker bound, useful in proving this section's main result.

Lemma 7.6.4. *For all ratio $R_{\max}$ no deterministic algorithm can achieve competitive ratio better than $(1 - R_{\max})$ for the ad allocation problem under the adversarial model. This bound holds even for (k, d)-graphs for all k and d.*

Proof. The hard input consists of disjoint stars with advertisers for internal vertices and ad slots for leaves. Every advertiser i has budget $B_i = 1$, with i's bids given by

$$b_{ij} = \begin{cases} R_{\max} & \text{if } i\text{'s remaining budget is less than } R_{\max} \\ \varepsilon & \text{else} \end{cases}$$

Given enough ad slots, an optimal allocation exhausts all advertisers' budgets, but every advertiser i gains at most $1 - R_{\max} + \varepsilon$, whether or not i has neighbors j with $b_{ij} = R_{\max}$. Summing over all advertisers, the lemma follows. $\square$

Using the above and extending Theorem 7.6.2's proof, we can now prove the following.

Theorem 7.6.5. *For all $k \geqslant d$ no deterministic online algorithm for ad allocation is better than $(1 - R_{\max}) \cdot \left(1 - (1 - \frac{1}{d})^{k/R_{\max}}\right)$-competitive on (k, d)-graphs with $R_{\max} \leqslant \frac{1}{2}$ a unit fraction.*

Proof. The offline side consists of $d^{k/R_{\max}}$ advertisers, each with a budget $B_i = 1$. For the first phase, all edges have bids $R_{\max}$. During $k/R_{\max}$ rounds ad slots arrive, each neighboring d distinct advertisers, and a $\frac{1}{d}$-fraction of the advertisers are matched. The next round is as the last, but restricted to the previously unmatched advertisers. There are $(1 - \frac{1}{d})^{k/R_{\max}}$ unmatched advertisers by this phase's termination; these advertisers now satisfy the offline side's constraints for (k, d)-graphs, and receive no more neighbors. All of these advertisers' potential profit is lost. For the matched advertisers we now apply the construction of Lemma 7.6.4 to guarantee that at most a $(1 - R_{\max})$-fraction of their potential profit in an optimal solution is gained, for a total gain of $(1 - R_{\max}) \cdot \left(1 - \left(1 - \frac{1}{d}\right)^{k/R_{\max}}\right) \cdot |L|$. Applying Observation 7.6.1 repeatedly we find that there exists an allocation with all advertisers unmatched by algorithm $\mathcal{A}$ matched $1/R_{\max}$ times (to neighbors for which they bid $R_{\max}$), and all advertisers matched by $\mathcal{A}$ also exhausting their budgets simultaneously. The theorem follows. □

7.7 Randomized Algorithms

By relying on the dual updates of Algorithm 10, we prove competitiveness of algorithm RANDOM, which matches every arriving ad slot to some feasible neighbor (i.e., a neighboring advertiser with non-exhausted budget) chosen uniformly at random.

Theorem 7.7.1. *Algorithm* RANDOM *achieves expected competitive ratio of* $1 - \left(1 - \frac{1}{d}\right)^k$ *for both unweighted and vertex-weighted matching problems.*

Proof. We maintain and update a dual solution as in our deterministic Algorithm 10, while choosing matches randomly. As observed in the proof of Theorem 7.4.3, such dual updates guarantee all unmatched advertisers $i \in L$ with current degree $d(i)$ satisfy

$$z_i = C \cdot \left(\left(\frac{d}{d-1}\right)^{d(i)} - 1\right).$$

Consequently, these dual update rules guarantee dual feasibility, provided $C = 1/((\frac{d}{d-1})^k - 1)$. We need only bound the expected ratio between the dual and primal solutions' values.

Consider some ad slot j matched to some i. We recall that in the vertex-weighted matching problem the bid b_{ij} is exactly $b_{ij} = B_i$. Therefore, given the current state (determined by the previous random choices), including the set $N_F(j)$ of j's unmatched (feasible) neighbors, j's match is chosen uniformly among $N_F(j)$ by RANDOM, and consequently

$$\mathbb{E}[\Delta P | state] = \frac{1}{|N_F(j)|} \cdot \sum_{i \in N_F(j)} B_i.$$

On the other hand, by the same argument

$$\begin{aligned}
\mathbb{E}[\Delta D | state] &= \tfrac{1}{|N_F(j)|} \cdot \sum_{i \in N_F(j)} \left((1 - z_i) \cdot B_i + \sum_{i' \in N_F(j) \setminus \{i\}} \left(\tfrac{1}{d-1} \cdot (z_{i'} + C) \cdot B_{i'}\right)\right) \\
&= \tfrac{1}{|N_F(j)|} \cdot \sum_{i \in N_F(j)} ((1 - z_i) \cdot B_i) + \tfrac{1}{|N_F(j)|} \cdot \sum_{i' \in N_F(j)} \left(\tfrac{|N_F(j)| - 1}{d-1} \cdot (z_{i'} + C) \cdot B_{i'}\right) \\
&\leqslant \tfrac{1}{|N_F(j)|} \cdot \sum_{i \in N_F(j)} B_i \cdot (1 + C).
\end{aligned}$$

With the last inequality following from $|N_F(j)| \leqslant |N(j)| \leqslant d$. Taking total expectation over the possible states, we obtain $\mathbb{E}[\Delta D] \leqslant (1+C) \cdot \mathbb{E}[\Delta P]$. The theorem follows. □

We note that Theorem 7.7.1 can also be proved using the potential-based proof of Section 7.4.2, observing that the expected potential change incurred by the processing of every online arrival is non-negative. In addition, in the same way that Theorem 7.4.3 is extended in Theorem 7.4.5, we can show that RANDOM is also robust to outliers. We omit the details for brevity. Finally, we show that RANDOM also performs well for the general online ad allocation problem.

Theorem 7.7.2. *Algorithm* RANDOM *achieves expected competitive ratios of* $1-\left(1-\frac{1}{d}\right)^k$ *and* $(1-R_{\max}) \cdot \left(1-\left(1-\frac{1}{d}\right)^k\right)$ *for the equal-bids and general-bids ad allocation problems.*

Proof (sketch). The proof resembles that of Theorem 7.7.1, relying on algorithms 11 and 12 respectively for the dual-fitting analysis. Dual feasibility is guaranteed by the dual updates. On the other hand, linearity of expectation implies the expected primal-dual ratio matches that of Algorithms 11 and 12 (for the latter, this requires showing lemmas 7.5.3–7.5.6 all hold in expectation). The claimed bounds follow. □

7.8 Conclusion and Open Questions

The study of online matching and ad allocation has seen a surge of interest, both theoretical and practical, ever since the influential work of [207] (see [206]). Several natural heuristics, most prominently the natural GREEDY algorithm, which in the worst case is only $1/2$-competitive, were observed to fare significantly better on real data (see e.g., [107] for results on the related Display Ads problem); the greedy algorithm was also shown to theoretically outperform its worst-case behavior under some stochastic assumptions (see [134]). This chapter attempts to give a theoretical explanation of the empirical success of simple heuristic algorithms for online ad allocation by considering structural assumptions regarding the inputs observed in practice, while eschewing stochastic input assumptions. Moreover, our work proposes better algorithms under such structural assumptions that could explain the above-mentioned empirical success. The chapter further raises several interesting follow-up questions.

Optimality for Adwords. We proved optimality of our algorithms among deterministic algorithms for the online maximum and vertex-weighted matching problems. However, for the general ad allocation problem our lower and upper bounds differ by a factor of $\left(1-\left(1-\frac{1}{d}\right)^k\right)/\left(1-\left(1-\frac{1}{d}\right)^{k/R_{\max}}\right)$. For small $R_{\max}$ (i.e., the AdWords problem), this discrepancy is large. Can better algorithms be obtained for this problem, or can the upper bounds be tightened (or both)?

Randomization. As we have shown, algorithm RANDOM matches the competitive ratio of our optimal deterministic algorithms in expectation. One may well wonder if randomization can allow us to improve on the optimal bounds achievable by deterministic algorithms. For the online matching problem, our results of Chapter 5 show that for d-regular graphs (a special case of (k, d)-graphs) randomization does indeed help, and allows for $1 - \tilde{O}(1/\sqrt{d})$-competitiveness for randomized algorithms (and this is optimal). On the other hand, by Corollary 7.6.3, our $1 - (1 - 1/d)^d$-competitive bounds are optimal for *deterministic* algorithms for this class. This proves that for this class of graphs, randomized algorithms outperform deterministic algorithms, which have competitive ratio tending to $1 - 1/e$ from above as d increases, while randomized algorithms have competitive ratio tending to *one* as d increases. On the negative side, we can show that no randomized algorithm can achieve competitive ratio better than $1 - e^{-\Theta(k/d)}$ for all problems considered in this chapter, contrasted with the $1 - e^{-k/d}$ competitive ratio achievable by deterministic algorithms (this partially answers our previous question concerning tighter bounds for AdWords). It would be interesting to see what exactly is the optimal competitive ratio achievable by randomized algorithms for this problem, given its practical importance.

Stochastic Models. An interesting direction would be to extend our exploration of (k, d)-graphs to stochastic models, in which it seems plausible that even better competitiveness guarantees should be achievable. More interestingly, can we show improved performance for somewhat more "robust" stochastic models? Note that the i.i.d input model is memoryless, and in particular the arrival of an ad slot due to a particular user's browsing does not increase said user's subsequent ad slots' arrival probability. Can we give bounds for less memoryless input models than the i.i.d model if we add structural assumptions about the input?

Chapter 8

Stochastic Online Metric Matching

In this chapter, based on [144] (joint with Anupam Gupta, Guru Guruganesh and Binghui Peng), we study the metric matching problem. In particular, we study this problem under stochastic arrivals.

8.1 Background and Contributions

We study the minimum-cost metric (perfect) matching problem under online i.i.d. arrivals. In this problem, we are given a fixed metric (S, d) with a server at each of the $n = |S|$ points. Then n requests arrive online, where each request is at a location that is drawn independently from a known probability distribution $\mathcal{D}$ over the points. Each such arriving request has to be matched immediately and irrevocably to a free server, whereupon it incurs a cost equal to distance of its location to this server. The goal is to minimize the total expected cost.

The minimization version of online matching was first considered in the standard adversarial setting by Khuller et al. [182] and Kalyanasundaram and Pruhs [172]; both papers showed $(2n-1)$-competitive deterministic algorithms, and proved that this was tight for, say, the star metric. After about a decade, a randomized algorithm with an $O(\log^3 n)$-competitiveness was given by Meyerson et al. [209]; this was improved to $O(\log^2 n)$ by Bansal et al. [23], which remains the best result known. (Recall that the maximization version of matching problems have been very widely studied, but they use mostly unrelated techniques.)

The competitive ratio model with adversarial online arrivals is often considered too pessimistic, since it assumes an all-powerful adversary. One model to level the playing field, and to make the model perhaps closer to practice, is to restrict the adversary's power. Two models have been popular here: the *random-order arrivals* (or *secretary*) model, and the *i.i.d.* model defined above. The random-order model is a *semi-random* model, in which the worst-case input is subjected to random perturbations. Specifically, the adversary chooses a *set* of requests, which are then presented to the algorithm in a uniformly random order. The min-cost online matching problem in this random-order model was studied by Raghvendra, who gave a tight $O(\log n)$-competitive algorithm [239]. The random-order model also captures the i.i.d. setting, so the natural goal is to get a better algorithm for the i.i.d. model. Indeed, our main result for the i.i.d. model gives exactly such a result:

Theorem 8.1.1 (Main Theorem). *There is an $O((\log\log\log n)^2)$-competitive algorithm for online minimum-cost metric perfect matching in the i.i.d. setting.*

Observe that the competitiveness here is better than the lower bounds of $\Omega(\log n)$ known for the worst-case and random-order models.

Matching on the Line and Trees. There has also been much interest in solving the problem for the line metric. However, getting better results for the line than for general metrics has been elusive: an $O(\log n)$-competitive *randomized* algorithm for line metrics (and for doubling metrics) was given by [140]. In the *deterministic* setting, recently Nayyar and Raghvendra [223] gave an $O(\log^2 n)$-competitive algorithm, whose competitive ratio was subsequently proven to be $O(\log n)$ by Raghvendra [240], improving on the $o(n)$-competitive algorithm of Antoniadis et al. [13]. To the best of our knowledge, nothing better is known for tree metrics than for general metrics in both the adversarial and the random-order models. Our second result for the i.i.d. model is a constant-competitive algorithm for tree metrics.

Theorem 8.1.2 (Algorithm for Trees). *There is a 9-competitive algorithm for online minimum-cost metric perfect matching on tree metrics in the i.i.d. setting.*

Max-Weight Perfect Matching. Recently, Chang et al. [59] presented a $1/2$-competitive algorithm for the *maximum*-weight perfect matching problem in the i.i.d. setting. We show that our algorithm is versatile, and that a small change to our algorithm gives us a maximization variant matching this factor of $1/2$. Our approach differs from that of [59], in that we match an arriving request based on the realization of free servers, while they do so based on the "expected realization". See Section 8.8 for details.

8.1.1 Our Techniques

Both theorems 8.1.1 and 8.1.2 are achieved by the same algorithm. The first observation guiding this algorithm is that we may assume that the distribution $\mathcal{D}$ of request locations is just the uniform distribution on the server locations. (In Section 8.5 we show how this assumption can be removed with a constant factor loss in the competitiveness.) Our algorithm is inspired by the following two complementary consequences of the uniformity of $\mathcal{D}$.

- Firstly, each of the $n-t+1$ free servers' locations at time t are equally likely to get a request in the future, and as such they should be left unmatched with equal probability. Put otherwise, we should match to them with equal probability of $1/(n-t+1)$. However, matching *any* arriving request to any free server with probability $1/(n-t+1)$ is easily shown to be a bad choice.
- So instead, we rely on the second observation: the t^{th} request is equally likely to arrive at each of the n server locations. This means we can couple the matching of free server locations with the location of the next request, to guarantee a marginal probability of $1/(n-t+1)$ for each free server to be matched at time t.

Indeed, the constraints that each location is matched at time t with probability $1/n$ (i.e., if it arrives) and each of the free servers are matched with marginal probability $1/(n-t+1)$ can be expressed as a *bipartite flow* instance, which guides the coupling used by the algorithm. Loosely speaking, our algorithm is fairly intuitive. It finds a min-cost fractional matching between the current open server locations and the expected arrivals, and uses that to match new requests. The challenge is to bound the competitive ratio—in contrast to previously used approaches (for the maximization version of the problem) it does not just try to match vertices using a fixed template of choices, but rather dynamically recomputes a template after each arrival.

A major advantage of this approach is that we understand the distribution of the open servers. We maintain the invariant that after t steps, the set of free servers form a uniform random $(n-t)$-subset of $[n]$—the randomness being over our choices, and over the randomness of the input. This allows us to relate the cost of the algorithm in the t^{th} step to the expected cost of this optimal flow between the original n points and a uniformly random subset of $(n-t)$ of these points. The latter expected cost is just a statistic based on the metric, and does not depend on our algorithm's past choices. For paths and trees, we bound this quantity explicitly by considering the variance across edge-cuts in the tree—this gives us the proof of Theorem 8.1.2.

Since general metrics do not have any usable cut structure, we need a different idea for Theorem 8.1.1. We show that tree-embedding results can be used either explicitly in the algorithm or just implicitly in the proof, but both give an $O(\log n)$ loss. To avoid this loss, we use a different balls-and-bins argument to improve our algorithm's competitiveness to $O((\log \log n))^2)$. In particular, we provide better bounds on our algorithm's per-step cost in terms of $\mathbb{E}[OPT]$ and the expected load of the k most loaded bins in a balls and bins process, corresponding to the number of requests in the k most frequently-requested servers. Specifically, we show that $\mathbb{E}[OPT]$ is bounded in terms of the expected *imbalance* between the number of requests and servers in these top k server locations. Coupling this latter uniform k-tuple with the uniform k-tuple of free servers left by our algorithm, we obtain our improved bounds on the per-step cost of our algorithm in terms of $\mathbb{E}[OPT]$ and these bins' load, from which we obtain our improved $O((\log \log n)^2)$ competitive ratio. Interestingly, combining both balls and bins and tree embedding bounds for the per-step cost of step k (appealing to different bounds for different ranges of k) gives us a further improvement: we prove that our algorithm is $O((\log \log \log n)^2)$ competitive.

8.1.2 Further Related Work

I.i.d. stochastic arrivals have been studied for various online problems, e.g., for Steiner tree/forest [125], set cover [138], and k-server [76]. Closer to the topic of this chapter, stochastic arrivals have been widely studied in the online matching literature, though so far mostly for maximization variants. Much of this work was motivated by applications to online advertising, for which the worst-case optimal $(1-{}^1\!/e)$-competitive ratios [6, 179, 207] seem particularly pessimistic, given the financial incentives involved and time-learned information about the distribution of requests. Consequently, many stochastic arrival models have been studied, and shown to admit better than $1-{}^1\!/e$ competitive guarantees. The stochastic models studied for online matching and related problems, in increasing order of attainable competitive ratios, include random order (e.g., [134, 176, 201]), unknown i.i.d.—where the request distribution is unknown—(e.g., [78, 213]), and known i.i.d. (e.g., [21, 50, 106]). Additional work has focused on interpolating between adversarial and stochastic input (e.g., [96, 202]). See Mehta's survey [206] and recent work [69, 120, 163–165, 222] for more details. The long line of work on online matching, both under adversarial and stochastic arrivals, have yielded a slew of algorithmic design ideas, which unfortunately do not seem to carry over to minimization problems, nor to perfect matching problems.

As mentioned above, the only prior work for stochastic online matching with minimization objectives was the random order arrival result of Raghvendra [239]. We are hopeful that our work will spur further research in online minimum-cost perfect matching under stochastic arrivals, and close the gap between our upper bounds and the (trivial) lower bounds for the problem.

8.2 Our Algorithm

In this section we present our main algorithm, together with some of its basic properties. In most of the chapter we assume that the distribution over request locations is uniform over the n servers' locations. We show in Section 8.5 that this assumption is WLOG: it increases the competitive ratio by at most a constant. In particular, we show the following.

Lemma 8.2.1. *Given an α-competitive algorithm* $\text{ALG}_{\mathcal{U}}$ *for the* uniform *distribution over server locations, $\mathcal{U}$, we can construct a $(2\alpha + 1)$-competitive algorithm* $\text{ALG}_{\mathcal{D}}$ *for any distribution $\mathcal{D}$.*

Focusing on the uniform distribution over server locations, our algorithm is loosely the following: in each round of the algorithm, we compute an optimal fractional matching between remaining free servers and remaining requests (in expectation). Now when a new request arrives, we just match the newly-arrived request according to this matching.

8.2.1 Notation

Our analysis will consider k-samples from the set $S = [n]$ both with and without replacement. We will set up the following notation to distinguish them:

- Let $\mathcal{I}_k$ be the distribution over k-sub-multisets of $S = [n]$ obtained by taking k i.i.d. samples from the uniform distribution over S. (E.g., $\mathcal{I}_n$ is the request set's distribution.)
- Let $\mathcal{U}_k$ be the distribution over k-subsets of S obtained by picking a uniformly random k-subset from $\binom{S}{k}$.

In other words, $\mathcal{I}_k$ is the distribution obtained by picking k elements from S uniformly *with* replacement, whereas $\mathcal{U}_k$ is *without* replacement.

For a sub-(multi)set $T \subseteq S$ of servers, let $M(T)$ denote the optimal fractional min-cost b-matching in the bipartite graph induced between T and the set of all locations S, with overall unit capacity on either side. That is, the capacity for each node in T is $1/|T|$ and the capacity for each node in S is $1/n$. So, if we denote by $d_{i,j}$ the distance between locations i and j, we let

$M(T)$ correspond to the following linear program.

$$\begin{aligned} M(T) := \min \sum_{i\in T, j\in S} & d_{i,j} \cdot x_{i,j} & & \qquad (M(\cdot)) \\ \text{s.t.} \sum_{j\in S} x_{i,j} &= \tfrac{1}{|T|} & \forall i \in T & \\ \sum_{i\in T} x_{i,j} &= \tfrac{1}{n} & \forall j \in S & \\ x &\geqslant 0 & & \end{aligned}$$

We emphasize that in the above LP, several servers in S (and likewise in T) may happen to be at the same point in the metric space, and hence there is a separate constraint for each such point j (and likewise i). Slightly abusing notation, we let $M(T)$ denote both the LP and its optimal value, when there is no scope for confusion.

8.2.2 Algorithm Description

The algorithm works as follows: at each time k, if $S_k \subseteq S$ is the current set of free servers, we compute the fractional assignment $M(S_k)$, and assign the next request randomly according to it. As argued above, since each free server location is equally likely to receive a request later (and therefore it is worth not matching it), it seems fair to leave each free server unmatched with equal probability. Put otherwise, it is only fair to match each of these servers with equal probability. Of course, matching any arriving request to a free server chosen uniformly at random can be a terrible strategy. In particular, it is easily shown to be $\Omega(\sqrt{n})$-competitive for n servers equally partitioned among a two-point metric. Therefore, to obtain good expected matching cost, we should bias servers' matching probability according to the arrived request, and in particular we should bias it according to $M(S_k)$. This intuition guides our algorithm FAIR-BIAS, and also inspires its name.

Algorithm 13 FAIR-BIAS

```
1: $S_n \leftarrow S$.                                   ▷ $S_k$ is the set of free servers, with $|S_k| = k$.
2: for time step $k = n, n-1, \cdots, 1$ do
3:     compute optimal fractional matching $M(S_k)$, denoted by $x^{S_k}$.
4:     for arrival of request $r_k = r$ do
5:         randomly choose server $s$ from $S_k$, where $s_i$ is chosen w/prob. $p_i = n \cdot x^{S_k}_{s_i,r}$.
6:         assign $r$ to $s$.
7:     $S_{k-1} \leftarrow S_k \setminus \{s\}$.
```

A crucial property of our algorithm is that the set S_k of free servers at each time k happens to be a uniformly random k-subset of S. Recall that FAIR-BIAS assigns each arriving request according to the assignment $M(S_k)$. This means that to analyze the algorithm, it suffices to relate the optimal assignment cost OPT to the optimal assignment costs for uniformly random subsets S_k, as follows.

Lemma 8.2.2. *(Structure Lemma) For each time k, the set S_k is a uniformly-drawn k-subset of S; i.e., $S_k \sim \mathcal{U}_k$. Consequently, the algorithm's cost is*

$$\mathbb{E}[ALG] = \sum_{k=1}^{n} \mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)].$$

Proof. The proof of the first claim is a simple induction from n down to 1. The base case of S_n is trivial. For any k-subset $T = \{s_1, \cdots, s_k\} \subseteq S$,

$$\begin{aligned}\Pr[S_k = T] &= \sum_{s \in S \setminus T} \Pr[S_{k+1} = T \cup \{s\}] \cdot \Pr[r_{k+1} \text{ assigns to } s \mid S_{k+1} = T \cup \{s\}] \\ &= (n-k) \cdot \frac{1}{\binom{n}{k+1}} \cdot \frac{1}{k+1} = \frac{1}{\binom{n}{k}},\end{aligned}$$

where the second equality follows from induction and the fact that

$$\Pr[r_{k+1} \text{ assigned to } s \mid S_{k+1} = T \cup \{s\}] = \sum_{r \in S} x_{s,r}^{S_{k+1}} = \frac{1}{k+1}.$$

To compute the algorithm's cost, we consider some set $S_k = T$ of k free servers. Since the request $r_k = r$ is chosen with probability $1/n$, following which we match it to some free server $s \in S_k$ with probability $n \cdot x_{s,r}^{S_k}$, we find that the next edge matched by the algorithm has expected cost

$$\mathbb{E}[d_{s,r_k} \mid S_k = T] = \sum_{r} \frac{1}{n} \cdot \sum_{s \in T} n \cdot x_{s,r}^{T} \cdot d_{s,r} = M(T).$$

Therefore, the expected cost of the algorithm is indeed

$$\begin{aligned}\mathbb{E}[ALG] = \sum_{k=1}^{n} \mathbb{E}[d_{s,r_k}] &= \sum_{k=1}^{n} \sum_{T \in \binom{S}{k}} \Pr_{S_k \sim \mathcal{U}_k}[S_k = T] \cdot \mathbb{E}[d_{s,r_k} \mid S_k = T] \\ &= \sum_{k=1}^{n} \sum_{T \in \binom{S}{k}} \Pr_{S_k \sim \mathcal{U}_k}[S_k = T] \cdot M(T) = \sum_{k=1}^{n} \mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)].\end{aligned}$$ □

The structure lemma implies that we may assume from now on that the set of free servers S_k is drawn from $\mathcal{U}_k$. In what follows, unless stated otherwise, we have $S_k \sim \mathcal{U}_k$. More importantly, Lemma 8.2.2 implies that to bound our algorithm's competitive ratio by α, it suffices to show that $\sum_k \mathbb{E}[M(S_k)] \leqslant \alpha \cdot \mathbb{E}[\text{OPT}]$. This is exactly the approach we use in the following sections.

8.3 Bounds for General Metrics

In Section 8.4 we will show that algorithm FAIR-BIAS is $O(1)$-competitive for line metrics (and more generally tree metrics), by relying on variance bounds of the number of matches across tree edges in OPT and $M(S_k)$, our algorithm's guiding LP. For general metrics, if we first embed the metric in a low-stretch tree metric [98] (blowing up the expected cost of $\mathbb{E}[\text{OPT}]$ by $O(\log n)$) and run algorithm FAIR-BIAS on the obtained metric, we immediately obtain an $O(\log n)$-competitive algorithm. In fact, explicitly embedding the input metric in a tree metric is not necessary in order to obtain this result using our algorithm. By relying on an *implicit* tree embedding, we obtain the following lemma (mirroring the variance-based bound underlying our result for tree metrics). This lemma's proof is deferred to Section 8.7.

Lemma 8.3.1. $\mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)] \leqslant \frac{O(\log n)}{\sqrt{nk}} \cdot \mathbb{E}[\text{OPT}]$.

Summing over all values of $k \in [n]$, we find that FAIR-BIAS is $O(\log n)$-competitive on general metrics. While this bound is no better than that of Raghvendra's t-net algorithm for random order arrival [239] (and therefore for i.i.d arrivals), the result will prove useful in our overall bound for our algorithm. In Sections 8.3.1 and 8.3.2, we use a different balls-and-bins argument to decrease our bounds on the algorithm's competitive ratio considerably, to $O((\log \log n))^2)$, by considering the imbalance between number of requests and servers in the top k most requested locations. (The former quantity corresponds to the load of the k most loaded bins in a balls and bins process – motivating our interest in this process.) Finally, in Section 8.3.3, we combine this improved bound with the one from Lemma 8.3.1, summing different bounds for different ranges of k, to prove our main result: an $O((\log \log \log n)^2)$ bound for our algorithm's competitive ratio.

8.3.1 Balls and Bins: The Poisson Paradigm

For our results, we need some technical facts about the classical balls-and-bins process.

The following standard lemma from [216, Theorem 5.10] allows us to use the Poisson distribution to approximate monotone functions on the bins. For $i \in [n]$, let X_i^m be a random variable denoting the number of balls that fall into the i^{th} bin, when we throw m balls into n bins. Let Y_i^m be independent draws from the Poisson distribution with mean m/n.

Lemma 8.3.2. *Let $f(x_1, \cdots, x_n)$ be a non-negative function such that $\mathbb{E}[f(X_1^m, \cdots, X_n^m)]$ is either monotonically increasing or decreasing with m, then*

$$\mathbb{E}[f(X_1^m, \cdots, X_n^m)] \leqslant 2 \cdot \mathbb{E}[f(Y_1^m, \cdots, Y_n^m)].$$

A classic result states that for $m = n$ balls, the maximum bin load is $\Theta(\log n / \log \log n)$ w.h.p. (see e.g., [216]). The following lemma is a partial generalization of this result. Its proof, which relies on the Poisson approximation of Lemma 8.3.2, is deferred to Section 8.7.

Lemma 8.3.3. *Let n balls be thrown into n bins, each ball thrown independently and uniformly at random. Let L_j be the load of the j^{th} heaviest bin, and $N_k := \sum_{j \leqslant k} L_j$ be the number of balls in the k most loaded bins. There exists a constant $C_0 > 0$ such that for any $k \leqslant C_0 n$,*

$$\mathbb{E}[N_k] \geqslant \Omega\left(k \cdot \frac{\log(n/k)}{\log\log(n/k)}\right).$$

In the next lemma, whose proof is likewise deferred to Section 8.7, we rely on a simple Chernoff bound to give a weaker lower bound for $\mathbb{E}[N_k]$ that holds for all $k \leqslant n/2$.

Lemma 8.3.4. *For sufficiently large n and any $k \leqslant n/2$, we have $\mathbb{E}[N_k] \geqslant 1.5k$.*

8.3.2 Relating Balls and Bins to Stochastic Metric Matching

We now bound the expected cost incurred by FAIR-BIAS at time k by appealing to the above balls-and-bins argument; this will give us our stronger bound of $O((\log\log n)^2)$. Specifically, we will derive another lower bound for $\mathbb{E}[\text{OPT}]$ in terms of $\mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)]$. In our bounds we will partition the probability space $\mathcal{I}_n$ (corresponding to n i.i.d. requests) into disjoint parts, based on $\mathbb{T}_k$, the top k most frequently requested locations (with ties broken uniformly at random). By symmetry, $\Pr[\mathbb{T}_k = T] = 1/\binom{n}{k}$ for all $T \in \binom{S}{k}$. By coupling $\mathbb{T}_k$ with $\mathcal{U}_k$, we will lower-bound $\mathbb{E}[OPT]$ by $\mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)]$ times $\mathbb{E}[N_k] - k$, the expected imbalance between number of requests and servers in $\mathbb{T}_k$. Here $\mathbb{E}[N_k]$ is the expected occupancy of the k most loaded bins in the balls and bins process discussed in Section 8.3.1.

To relate $\mathbb{E}[OPT \mid \mathbb{T}_k = S_k]$ to $M(S_k)$, we will bound both these quantities by the cost of a min-cost perfect b-matching between S_k and $S \setminus S_k$; i.e., each vertex v has some (possibly fractional) demand b_v which is the extent to which it must be matched. To this end, we need the following simple lemma, which asserts that for any min-cost metric b-matching instance, there exists an optimal solution which matches co-located servers and requests maximally. We defer the lemma's proof, which follows from a local change argument and triangle inequality, to Section 8.7.

Lemma 8.3.5. *Let $\mathcal{I}$ be a fractional min-cost bipartite metric b-matching instance, with demand ℓ_i and r_i for the servers and requests at location i. Then, there exists an optimal solution x for $\mathcal{I}$ with $x_{ii} = \min\{\ell_i, r_i\}$ for every point i in the metric.*

We are now ready to prove our main technical lemma, lower-bounding $\mathbb{E}[OPT \mid \mathbb{T}_k = S_k]$ in terms of $M(S_k)$ and the imbalance between number of requests of the k most requested locations, N_k, and the number of servers in those locations.

Lemma 8.3.6. *For all $k < n$ and $S_k \in \binom{S}{k}$, we have $\mathbb{E}[OPT \mid \mathbb{T}_k = S_k] \geqslant (\mathbb{E}[N_k] - k) \cdot M(S_k)$.*

Proof. Applying Lemma 8.3.5 to $M(S_k)$, we find that the optimal value of $M(S_k)$ is equal to that of a min-cost bipartite perfect b-matching instance with left vertices associated with S_k, each with demand $\frac{1}{k} - \frac{1}{n}$, and right vertices associated with $S \setminus S_k$, each with demand $\frac{1}{n}$.

We now turn to the meat of the proof – lower bounding $\mathbb{E}[OPT \mid \mathbb{T}_k = S_k]$. In particular, we will lower bound $\mathbb{E}[OPT \mid \mathbb{T}_k = S_k]$ by a min-cost bipartite perfect b-matching instance with left and right vertices as above (i.e., S_k and $S \setminus S_k$, respectively), but with uniform demands on both sides of at least $(\mathbb{E}[N_k] - k)/k$ and $(\mathbb{E}[N_k] - k)/(n-k)$, respectively. That is, the biregular min-cost bipartite b-matching whose cost C we showed lower bounds $M(S_k)$, but scaled by an $f \geqslant \frac{(\mathbb{E}[N_k]-k)}{k \cdot (1/k - 1/n)}$ factor. Before proving this lower bound on $\mathbb{E}[OPT \mid \mathbb{T}_k = S_k]$, we note that it implies our desired bound, as

$$\mathbb{E}[\mathrm{OPT} \mid \mathbb{T}_k = S_k] \geqslant \frac{(\mathbb{E}[N_k] - k)}{k \cdot (1/k - 1/n)} \cdot C > (\mathbb{E}[N_k] - k) \cdot C = (\mathbb{E}[N_k] - k) \cdot M(S_k).$$

It remains to lower bound $\mathbb{E}[OPT \mid \mathbb{T}_k = S_k]$ in terms of such a biregular b-matching instance.

For the remainder of this proof, for notational simplicity we denote by Ω the probability space induced by conditioning on the event $\mathbb{T}_k = S_k$. To lower bound $\mathbb{E}_\Omega[OPT]$, we will provide a fractional perfect matching $\vec{x}$ of the expected instance (in Ω), and show that $\mathbb{E}_\Omega[OPT] \geqslant \sum_{ij} d_{ij} \cdot x_{ij}$, while $\sum_{j \in S \setminus S_k} x_{ij} \geqslant (\mathbb{E}[N_k] - k)/k$ for all $i \in S_k$ and $\sum_{i \in S} x_{ij} \geqslant (\mathbb{E}[N_k] - k)/(n-k)$ for all $j \in S \setminus S_k$. Consequently, focusing on edges $(i,j) \in S_k \times (S \setminus S_k)$, we find that the min-cost biregular bipartite perfect b-matching above lower bounds $\sum_{i \in S_k, j \in S \setminus S_k} d_{ij} \cdot x_{ij} \leqslant \sum_{ij} d_{ij} \cdot x_{ij} \leqslant \mathbb{E}_\Omega[OPT]$. We now turn to producing an $\vec{x}$ satisfying our desired properties.

For any two locations $i, j \in S$, we let $(i,j) \in OPT$ indicate that a request in location i is served by the server in location j. Let $p_{ij} := \Pr_\Omega[(i,j) \in OPT]$. We will show how small modifications to $\vec{p}$ will yield a fractional perfect matching $\vec{x}$ as discussed in the previous paragraph. Let Y_i be the number of requests at server i. By Lemma 8.3.5, we know that $(i,i) \in OPT \iff Y_i \geqslant 1$. So, $p_{ii} = \Pr_\Omega[Y_i \geqslant 1]$. Consequently, if we let $\Delta_{in}(j) := \sum_{j' \in S \setminus \{j\}} p_{j'j}$ and $\Delta_{out}(j) := \sum_{j' \in S \setminus \{j\}} p_{jj'}$, we have by Lemma 8.3.5 that $\Delta_{in}(j) = \Pr[Y_i \geqslant 1]$ and $\Delta_{out}(i) = \mathbb{E}[(Y_i - 1)^+]$ for all $i \in S$. (As usual, $x^+ = \max\{x, 0\}$.) Consequently, $\Delta_{in}(j) = \Delta_{in}(j')$ and $\Delta_{out}(j) = \Delta_{out}(j')$ for all $j, j' \in S \setminus S_k$, as $[Y_j \mid \Omega]$ and $[Y'_j \mid \Omega]$ are identically distributed. Moreover, as $\sum_{j \in S \setminus S_k} (\Delta_{in}(j) - \Delta_{out}(j)) = N_k - k \geqslant 0$, we find that $\Delta_{in}(j) - \Delta_{out}(j) \geqslant 0$ for all $j \in S \setminus S_k$. Now, suppose $Y_i \geqslant 1$ for all $i \in S_k$ (conditioning on the complementary event is similar), we have by Lemma 8.3.5 that $p_{ji} = 0$ for all $i \in S_k$ and $j \in S \setminus \{i\}$. Moreover, by symmetry we have $\Delta_{out}(i) = (\mathbb{E}[N_k] - k)/k$ for all k locations $i \in S_k$. We now show how to obtain from $\vec{p}$ a fractional matching $\vec{x}$ between S_k and $S \setminus S_k$ of no greater cost than $\vec{p}$, such that $p_{jj'} = 0$ for all $j \neq j' \in S \setminus S_k$ and such that the values $\Delta_{in}(j) - \Delta_{out}(j)$ are unchanged for all $j \in S$. Consequently, all (simple) edges in the support of $\vec{x}$ go between S_k and $S \setminus S_k$, and $\Delta_{out}(i) = (\mathbb{E}[N_k] - k)/k$ for all $i \in S_k$ and $\Delta_{in}(j) = (\mathbb{E}[N_k] - k)/(n-k)$ for all $j \in S \setminus S_k$, yielding our desired lower bound on $\mathbb{E}_\Omega[OPT]$ in terms of a biregular bipartite b-matching instance.

We start by setting $\vec{x} \leftarrow \vec{p}$. While there exists a pair $j \neq j' \in S \setminus S_k$ with $x_{j'j} > 0$, we pick such a pair. As $\Delta_{in}(j) - \Delta_{out}(j) \geqslant 0$, there must also be some flow coming into j. We follow a sequence of edges $j_1 \leftarrow j_2 \leftarrow j_3 \leftarrow \dots$ with each $j_r \in S \setminus S_k$ and with $x_{j_r j_{r-1}} > 0$ until we either repeat some $j_r \in S\setminus$ or reach some j_r with $x_{ij_r}0$ for some $i \in S$. (Note that one such case must happen, as $\Delta_{in}(j) - \Delta_{out}(j) \geqslant 0$ for all $j \in S \setminus S_k$.) If we repeat a vertex, j_r, we only consider the sequence of nodes given by the obtained cycle, $j_1 \leftarrow j_2 \leftarrow j_3 \cdots \leftarrow j_r = j_1$. Let $\varepsilon = \min_r x_{j_r j_{r-1}}$ be the smallest $x_{jj'}$ in our trail. If we repeated a vertex, we found a cycle, and we decrease $x_{jj'}$ by ε for all consecutive j, j' in the cycle. If we found some $i \in S$ and $x_{ij_r} > 0$, we decrease all $x_{jj'}$ values along the path (including x_{ij_r}) by ε and increase x_{ij_1} by ε. In both cases, we only decrease the cost of $\vec{x}$ (either trivially, or by triangle inequality) and we do not change $\Delta_{in}(j) - \Delta_{out}(j)$ for any $j \in S$, while decreasing $\sum_{j \neq j' \in S \setminus S_k} x_{jj'}$. As the initial x-values are all rational, repeating the above terminates, with the above sum equal to zero, which implies a biregular fractional solution $\vec{x}$ as required. The lemma follows. □

Coupling the distribution of $\mathbb{T}_k$ and the set of k free servers, we obtain the following.

Lemma 8.3.7. $\mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)] \leqslant \mathbb{E}[\mathrm{OPT}]/(\mathbb{E}[N_k] - k)$.

Proof. Taking expectations over $S_k \sim \mathcal{U}_k$, we obtain our claimed bound.

$$\begin{aligned}
\mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)] &= \sum_{S_k \in \binom{S}{k}} \frac{1}{\binom{n}{k}} \cdot M(S_k) && \text{defn. of } \mathcal{U}_k \\
&\leqslant \sum_{S_k \in \binom{S}{k}} \frac{1}{\binom{n}{k}} \frac{1}{(\mathbb{E}[N_k] - k)} \cdot \mathbb{E}[\mathrm{OPT} \mid \mathbb{T}_k = S_k] && \text{Lemma 8.3.6} \\
&= \frac{1}{(\mathbb{E}[N_k] - k)} \cdot \mathbb{E}[\mathrm{OPT}]. && \Pr[\mathbb{T}_k = S_k] = \frac{1}{\binom{n}{k}}.
\end{aligned}$$
□

Plugging in the lower bounds of Lemmas 8.3.3 and 8.3.4 for the top k most loaded bins' loads, $\mathbb{E}[N_k]$, we obtain the following bounds on FAIR-BIAS's per-step cost in terms of $\mathbb{E}[OPT]$.

Lemma 8.3.8. *For C_0 a constant as in Lemma 8.3.3, there exists a constant C such that*

$$\mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)] \leqslant \begin{cases} C \cdot \frac{\log\log(n/k)}{k \log(n/k)} \cdot \mathbb{E}[\mathrm{OPT}] & \text{if } k < C_0 n \\ \frac{2}{k} \cdot \mathbb{E}[\mathrm{OPT}] & \text{if } C_0 n \leqslant k \leqslant n/2. \end{cases}$$

The following lemma allows us to leverage Lemma 8.3.8, since it allows us to focus on $\mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)]$ for $k \leqslant n/2$. Its proof relies on our characterization of $M(S_k)$ in terms of a balanced b-matching instance between S_k and $S \setminus S_k$ as in the proof of Lemma 8.3.6, which implies that $M(S_k) \leqslant M(S_{n-k})$ for all $k \leqslant n/2$. Its proof is deferred to Section 8.7.

Lemma 8.3.9. $\sum_{k=1}^{n} \mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)] \leqslant 2 \cdot \sum_{k=1}^{n/2} \mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)]$.

Using our upper bound on $\mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)]$ of Lemma 8.3.8 and summing the two ranges of $k \leqslant n/2$ in Lemma 8.3.9 we find that FAIR-BIAS is $O((\log\log n)^2)$ competitive. We do not elaborate on this here, as we obtain an even better bound in the following section.

8.3.3 Our Main Result

We are now ready to prove our main result, by combining our per-step cost bounds given by our balls and bins argument (Lemma 8.3.8) and our implicit tree embedding argument (Lemma 8.3.1).

Theorem 8.3.10. *Algorithm* FAIR-BIAS *is $O((\log\log\log n)^2)$-competitive for the online bipartite metric matching problem under i.i.d arrivals on general metrics.*

Proof. By the structure lemma (Lemma 8.2.2) and Lemma 8.3.9, we have that

$$\mathbb{E}[ALG] = \sum_{k=1}^{n} \mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)] \leqslant 2 \cdot \sum_{k=1}^{n/2} \mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)]. \tag{8.1}$$

We use the three bounds from Lemma 8.3.1 and Lemma 8.3.8 for different ranges of k to bound the above sum. Specifically, by relying on Lemma 8.3.1 for $k \leqslant n/\log^2 n$, we have that

$$\begin{aligned}\sum_{k=1}^{n/\log^2 n} \mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)] &\leqslant \sum_{k=1}^{n/\log^2 n} \frac{O(\log n)}{\sqrt{nk}} \cdot \mathbb{E}[OPT] \\ &\leqslant O\left(\sqrt{\frac{n}{\log^2 n}} \cdot \frac{\log n \cdot \mathbb{E}[OPT]}{\sqrt{n}}\right) = O(1) \cdot \mathbb{E}[OPT].\end{aligned}$$

Next, by the first bound of Lemma 8.3.8 applied to $k \in [n/\log^2 n, C_0 n]$, we have that

$$\begin{aligned}\sum_{k=n/\log^2 n}^{C_0 n} \mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)] &\leqslant \sum_{k=n/\log^2 n}^{C_0 n} \frac{O(\log\log(n/k))}{k \cdot \log(n/k)} \cdot \mathbb{E}[OPT] \\ &\leqslant O\left(-(\log\log(n/k))^2 \Big|_{n/\log^2 n}^{C_0 n}\right) \cdot \mathbb{E}[OPT] \\ &= O((\log\log\log n)^2) \cdot \mathbb{E}[OPT].\end{aligned}$$

Finally, by the second bound of Lemma 8.3.8 applied to $k \geqslant C_0 n$, we have that

$$\sum_{k=C_0 n}^{n/2} \mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)] \leqslant \sum_{C_0 n}^{n/2} \frac{2}{k} \cdot \mathbb{E}[OPT] \leqslant O\left(\log\left(\frac{n/2}{C_0 n}\right)\right) \cdot \mathbb{E}[OPT] = O(1) \cdot \mathbb{E}[OPT].$$

Combining all three bounds with Equation (8.1), the theorem follows. $\square$

8.4 A Simple $O(1)$ Bound for Tree Metrics

In this section we show the power of the structure lemma, by analyzing FAIR-BIAS on tree metrics. Recall that a *tree metric* is defined by shortest-path distances in a tree $T = (V, E)$, with edge lengths d_e. By adding zero-length edges, we may assume that the tree has n leaves, and that servers are on the leaves of the tree. For any edge e in the tree, deleting this edge creates two components $T_1(e)$ and $T_2(e)$; denote by $T_1(e)$ the component with fewer servers/leaves. Let n_e denote the number of leaves on this smaller side, $T_1(e)$. Hence $n_e \leqslant n/2$ for all edges e.

We now lower bound $\mathbb{E}[\mathrm{OPT}]$, by considering the mean average deviation of the number of requests which arrive in $T_1(e)$ for each edge e.

Lemma 8.4.1. *The expected optimal matching cost in a tree metric on $n \geqslant 2$ vertices is at least $\mathbb{E}[\mathrm{OPT}] \geqslant \frac{1}{2} \cdot \sum_{e \in T} d_e \cdot \sqrt{n_e}$.*

Proof. Let X_e denote the number of requests that arrive in the component with fewer leaves, $T_1(e)$. Every matching will match at least $|X_e - n_e| = |X_e - \mathbb{E}[X_e]|$ requests across the edge e (with the equality due to the uniform IID arrivals). Summing over all edges and taking expectations, we find that

$$\mathbb{E}[\mathrm{OPT}] \geqslant \sum_e d_e \cdot \mathbb{E}\big[|X_e - n_e|\big] = \sum_e d_e \cdot \mathbb{E}\big[|X_e - \mathbb{E}[X_e]|\big]. \tag{8.2}$$

It remains to lower bound $\mathbb{E}[|X_e - \mathbb{E}[X_e]|]$, the mean average deviation of X_e. Observe that $X_e \sim \mathrm{Bin}(n, n_e/n)$, with $n_e \in [1, n-1]$. The following probabilistic bound appears in [31, Theorem 1]:

Claim 8.4.2. *Let $Y \sim Bin(n, p)$, with $n \geqslant 2$ and $p \in [1/n, 1 - 1/n]$. Then, we have both*

$$\mathbb{E}|Y - \mathbb{E}Y| \geqslant \mathrm{std}(Y)/\sqrt{2},$$

(Note that convexity implies that $\mathbb{E}|Y - \mathbb{E}Y| \leqslant \mathrm{std}(Y)$ holds for all distributions, so this is a partial converse.) Applying Claim 8.4.2 to our case, where $p = n_e/n \in [1/n, 1 - 1/n]$,

$$\mathbb{E}[|X_e - \mathbb{E}X_e|] \geqslant \mathrm{std}(X_e)/\sqrt{2} = \sqrt{n_e(1 - n_e/n)/2} \geqslant \sqrt{n_e/4},$$

where the second inequality follows from $n_e \leqslant n/2$. Combined with (8.2), the lemma follows. $\square$

To upper bound $\mathbb{E}[M(S_k)]$, we again consider the mean average deviation of the number of requests in $T_1(e)$, but this time when drawing k *i.i.d.* samples. First, we need to bound the cost of $M(S_k)$ for a set S_k resulting from k draws *without replacement* by the cost for a multiset obtained by taking k i.i.d. draws *with replacement*.

Lemma 8.4.3. *(Replacement Lemma) For all S and $k \in [|S|]$, we have*

$$\mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)] \leqslant \mathbb{E}_{S_k \sim \mathcal{I}_k}[M(S_k)].$$

We defer the proof of this lemma to Section 8.6, where we prove a more general statement regarding stochastic convex optimization with constraints and coefficients determined by elements of a set chosen uniformly with and without replacement. Armed with this lemma, it suffices to bound $\mathbb{E}_{S_k \sim \mathcal{I}_k}[M(S_k)]$ from above, which we do in the following.

Lemma 8.4.4. $\mathbb{E}_{S_k \sim \mathcal{I}_k}[M(S_k)] \leqslant \sum_{e \in T} d_e \cdot \sqrt{n_e/(kn)}$.

Proof. Fix some edge e and let $T_1(e)$ be its smaller subtree, containing $n_e \leqslant n/2$ leaves. Let $X_e \sim \mathrm{Bin}(k, n_e/n)$ be the random variable denoting the number of servers in $T_1(e)$ chosen in k i.i.d samples from S. For any given realization of S_k (and therefore of X_e) the fractional solution to $M(S_k)$ utilizes edges between the different subtrees of e by exactly $|X_e/k - n_e/n|$. Since this is a tree metric, we have

$$M(S_k) = \sum_{e \in T} d_e \cdot \left| \frac{X_e}{k} - \frac{n_e}{n} \right| = \sum_{e \in T} d_e \cdot \frac{1}{k} \cdot \left| X_e - \frac{k}{n} \cdot n_e \right| = \sum_{e \in T} d_e \cdot \frac{1}{k} \cdot |X_e - \mathbb{E}[X_e]|.$$

Taking expectations over S_k, and using the fact that the mean average deviation is always upper bounded by the standard deviation (by Jensen's inequality), we find that indeed

$$\begin{aligned}\mathbb{E}_{S_k \sim \mathcal{I}_k}[M(S_k)] &= \sum_{e \in T} d_e \cdot \frac{1}{k} \cdot \mathbb{E}[|X_e - \mathbb{E}[X_e]|] \leqslant \sum_{e \in T} d_e \cdot \frac{1}{k} \cdot \mathrm{std}(X_e) \\ &= \sum_{e \in T} d_e \cdot \frac{1}{k} \cdot \sqrt{k \cdot \frac{n_e}{n}\left(1 - \frac{n_e}{n}\right)} \leqslant \sum_{e \in T} d_e \cdot \sqrt{\frac{n_e}{k \cdot n}}.\end{aligned}$$

□

Combining the replacement lemma (Lemma 8.4.3) with Lemmas 8.4.4 and 8.4.1, we obtain the following upper bound on $\mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)]$ in terms of $\mathbb{E}[OPT]$.

Lemma 8.4.5. $\mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)] \leqslant 2 \cdot \frac{\mathbb{E}[OPT]}{\sqrt{nk}}$.

We can now prove our simple result for tree metrics.

Theorem 8.4.6. *(Tree Bound) Algorithm* FAIR-BIAS *is 4-competitive on tree metrics with $n \geqslant 2$ nodes, if the requests are drawn from the uniform distribution.*

Proof. We have by the structural lemma (Lemma 8.2.2) and Lemma 8.4.5 that

$$\begin{aligned}\mathbb{E}[\mathrm{ALG}] = \sum_{k=1}^{n} \mathbb{E}[M(S_k)] &\leqslant \sum_{k=1}^{n} 2 \cdot \frac{\mathbb{E}[OPT]}{\sqrt{nk}} \\ &\leqslant 2 \cdot \frac{\mathbb{E}[OPT]}{\sqrt{n}} \cdot \left(1 + \int_{x=1}^{n} \frac{1}{\sqrt{x}} dx\right) \leqslant 4 \cdot \mathbb{E}[OPT]. \qquad \square\end{aligned}$$

The above bound holds for all $n \geqslant 2$ (for $n = 1$ any algorithm is trivially 1 competitive). For n large, however, our proof yields an improved asymptotic bound of $\sqrt{2} \cdot e + o(1) \approx (3.845 + o(1))$, by relying on the asymptotic counterpart of Claim 8.4.2 in [31, Corollary 2], $\mathbb{E}|Y - \mathbb{E}Y| \geqslant \mathrm{std}(Y)/(e/2 + o(1))$. Combining Theorem 8.4.6 with our transshipment argument (Lemma 8.2.1), we obtain a 9-competitive algorithm under any i.i.d. distribution on tree metrics on $n \geqslant 2$ nodes, and even better than 9-competitive algorithms for large enough n.

8.5 Distribution over Server Locations (Transshipment Argument)

In this section, we show that the assumption that the requests are drawn from $\mathcal{U}$, the uniform distribution over server locations, is without loss of generality.

Lemma 8.2.1. *Given an α-competitive algorithm $\mathrm{ALG}_{\mathcal{U}}$ for the* uniform *distribution over server locations, $\mathcal{U}$, we can construct a $(2\alpha + 1)$-competitive algorithm $\mathrm{ALG}_{\mathcal{D}}$ for any distribution $\mathcal{D}$.*

Proof. As before, we identify the set of servers S with the n points on the metric and let $r_1, \dots, r_n$ be the requests that arrive according to the distribution $\mathcal{D}$. Define $p_i := \Pr_{r \sim \mathcal{D}}[r = i]$.

Consider the linear program defined by the transshipment problem between the distribution $\mathcal{D}$ to the uniform distribution on the servers S.

$$\begin{aligned} LP := \min \sum_{i,j} d_{i,j} \cdot x_{i,j} & \\ \text{s.t.} \ \sum_{j} x_{i,j} = p_i & \qquad \forall i \in \text{metric} \\ \sum_{i} x_{i,j} = \frac{1}{n} & \qquad \forall j \in S \\ x_{i,j} \geqslant 0 & \end{aligned}$$

Let $M = n \cdot LP$. Given a request sequence $\{r_1, \dots, r_n\}$ drawn from $\mathcal{D}$, we create a coupled sequence $\{\tilde{r}_1, \dots, \tilde{r}_n\}$ by moving an arrived request r_k at server location j to location i in the metric with probability $x_{i,j}/p_i$ Each server location $j \in S$ appears with probability $\sum_i x_{i,j} = \frac{1}{n}$ and hence the sequence $\{\tilde{r}_1, \dots, \tilde{r}_n\}$ is distributed according to the uniform distribution $\mathcal{U}$. After this move, it matches the request according to $\mathrm{ALG}_{\mathcal{U}}$.

We bound this algorithm's cost as follows. First, the probability of a given request being moved from some location i to j is precisely $p_i \cdot x_{i,j}/p_i = x_{i,j}$. Summing up over all i, j, the expected movement cost for all n time steps is precisely $M = n \cdot LP$. Secondly, the expected cost of matching from $\tilde{r}_i$ is precisely $\mathbb{E}[\mathrm{ALG}_{\mathcal{U}}]$. By the triangle inequality, we can bound the total cost by the sum of the initial costs and the matching costs according to $\mathrm{ALG}_{\mathcal{U}}$, yielding the relation

$$\mathbb{E}[\mathrm{ALG}_{\mathcal{D}}] \leqslant \mathbb{E}[\mathrm{ALG}_{\mathcal{U}}] + M. \tag{8.3}$$

We use the same coupling as above, but in the other direction to relate $\mathrm{OPT}_{\mathcal{U}}$ to M. In particular, given a request sequence $\{r_1, \ldots, r_n\}$ drawn from $\mathcal{U}$, we create a coupled sequence $\{\tilde{r}_1, \ldots, \tilde{r}_n\}$ by moving an arrived request r_k at server location j to location i in the metric with probability $n \cdot x_{i,j}$. Now $\Pr[\tilde{r}_k = i] = \frac{1}{n} \cdot \sum_j n \cdot x_{i,j} = \sum_j x_{i,j} = p_i$. That is, the resulting distribution is $\mathcal{D}$. One way to bound the optimal solution for distribution $\mathcal{U}$ is to match request r_k to the match of $\tilde{r}_k$. As before, the expected movement cost to locations $\{\tilde{r}_1, \ldots, \tilde{r}_n\}$ is M, and by triangle inequality, we find that

$$\mathbb{E}[\mathrm{OPT}_{\mathcal{U}}] \leqslant \mathbb{E}[\mathrm{OPT}_{\mathcal{D}}] + M. \tag{8.4}$$

We now bound $\mathbb{E}[\mathrm{OPT}_{\mathcal{D}}]$ in terms of M. Each location i in the metric has an expected np_i appearances, who must therefore be matched an expected np_i many times. Each server, on the other hand, is matched precisely once in expectation. Therefore, the probabilities $p_{i,j}$ of an arrival at location i being matched to a server at location j constitute a feasible solution to $n \cdot LP$, and so must have $\sum_{i,j} d_{i,j} \cdot p_{i,j} \geqslant n \cdot LP = M$. Therefore, $\mathbb{E}[\mathrm{OPT}_{\mathcal{D}}]$ satisfies

$$\mathbb{E}[\mathrm{OPT}_{\mathcal{D}}] \geqslant M. \tag{8.5}$$

Combining equations (8.3), (8.4) and (8.5) with $\mathrm{ALG}_{\mathcal{U}}$'s α-competitiveness, we obtain our desired result.

$$\begin{aligned}
\mathbb{E}[\mathrm{ALG}_{\mathcal{D}}] &\leqslant \mathbb{E}[\mathrm{ALG}_{\mathcal{U}}] + M && \text{Equation (8.3)}\\
&\leqslant \alpha \cdot \mathbb{E}[\mathrm{OPT}_{\mathcal{U}}] + M && \mathrm{ALG}_{\mathcal{U}} \text{ is } \alpha\text{-comp.}\\
&\leqslant \alpha \cdot (\mathbb{E}[\mathrm{OPT}_{\mathcal{D}}] + M) + M && \text{Equation (8.4)}\\
&\leqslant (2\alpha + 1) \cdot \mathbb{E}[\mathrm{OPT}_{\mathcal{D}}]. && \text{Equation (8.5)}
\end{aligned}$$ □

8.6 Stochastic Convex Optimization, with and without Replacement

In Lemma 8.4.3 we claimed that the expected cost of the linear program $M(S_k)$ for S_k chosen at random from the k-subsets of S is lower than its counterpart when S_k is obtained from k i.i.d draws from S. More succinctly, we claimed that $\mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)] \leqslant \mathbb{E}_{S_k \sim \mathcal{I}_k}[M(S_k)]$. In this section we prove a more general claim for any linear program (and more generally, any convex program), implying the above. Let S be some n-element set, and for any multiset T with all its elements taken from S, let $P(T)$ be the following convex program.

$$\begin{aligned} P(T) := \min & f(x, \chi_T) \\ \text{s.t. } & g_i(x, \chi_T) \leqslant 0 \qquad \forall i \in [m] \\ & h_j(x, \chi_T) = 0 \qquad \forall j \in [\ell] \end{aligned} \tag{$P(\cdot)$}$$

Here $f(x, \chi_T)$ and all $g_i(x, \chi_T)$ are convex functions and $h_j(x, \chi_T)$ are affine in their arguments x and χ_T, and χ_T is the incidence vector of the multiset T. (That is, for any $s \in S$, we let $\chi_T(s)$ denote the number of appearances of s in T.) Note that $M(T)$ defined in Section 8.2.1 is a linear program of the above form. As such, the following lemma generalizes – and implies – Lemma 8.4.3.

Lemma 8.6.1. *For any convex program P as above, we have*

$$\mathbb{E}_{S_k \sim \mathcal{U}_k}[P(S_k)] \leqslant \mathbb{E}_{S_k \sim \mathcal{I}_k}[P(S_k)].$$

Proof. Our proof relies on a coupling argument, starting with a refined partition of the probability space of $S_k \sim \mathcal{I}_k$. This space is partitioned into equiprobable events A_M for each ordered multiset M of size k supported in S, corresponding to M being sampled. For each ordered multiset M, we denote by $M) := \{s \in S \mid s \in M\}$ the set of elements in M. Next, we denote by $\text{SUP}(M) := \{T \in \binom{S}{k} \mid T \supseteq M)\}$ the family of k-sets which contain M's elements (i.e., supersets of M's support). We will wish to "equally partition" the event A_M among the k-tuples in $\text{SUP}(M)$. To this end, when M is sampled from $\mathcal{I}_k$, we roll a $|\text{SUP}(M)|$-sided die labeled by the members of $\text{SUP}(M)$. For any k-set $T \in \text{SUP}(M)$, we denote by $A_{M,T}$ the event that M was sampled from $\mathcal{I}_k$ and the die-roll came out T, and for any k-tuple $T \in \binom{S}{k}$, we let $A_T := \bigcup_M A_{M,T}$. It is easy to verify that by symmetry we have $Pr[A_T] = 1/\binom{|S|}{k}$ for every $T \in \binom{S}{k}$.

We now wish to couple the above refinement of the probability space of $\mathcal{I}_k$ and the optimal solution to $P(S_k)$ with their counterpart under $\mathcal{U}_k$. We will need the following claim.

Claim 8.6.2. *For all k-set $T \in \binom{S}{k}$ and element $s \in T$, we have $\mathbb{E}_{S_k \sim \mathcal{I}_k}[\chi_{S_k}(s) \mid A_T] = 1$.*

Proof. By definition, each non-empty $A_{M,T} \subseteq A_T$ satisfies $\mathbb{E}_{S_k \sim \mathcal{I}_k}[\sum_{s \in T} \chi_{S_k}(s) \mid A_{M,T}] = k$, since any ordered multiset M of size k with $\text{SUP}(M) \ni T$ has all its elements in T. Therefore, taking total expectation over M with $\text{SUP}(M) \ni T$, we get $\mathbb{E}_{S_k \sim \mathcal{I}_k}[\sum_{s \in T} \chi_{S_k}(s) \mid A_T] = k$. Therefore, by symmetry, we find that indeed each of the k elements $s \in T$ has $\mathbb{E}_{S_k \sim \mathcal{I}_k}[\chi_{S_k}(s) \mid A_T] = 1$. □

Now, consider some k-set $T \in \binom{S}{k}$. For any ordered multiset of k elements M such that $SUP(M) \ni T$, denote by $x^M \in \arg\min P(M)$ a solution of $P(M)$ of minimum cost. By definition, for each $i \in [m]$ we have that $g_i(x^M, \chi_M) \leqslant 0$ and for each $j \in [\ell]$ we have that

$h_j(x^M, \chi_M) = 0$. Therefore, if we let $y^T := \mathbb{E}_{M\sim\mathcal{I}_k}[x^M \mid A_T]$ be the "average" optimal solution for $P(M)$ over all M with $SUP(M) \ni T$, then by Jensen's inequality and convexity of g_i, we have that

$$\begin{aligned} 0 &\geqslant \mathbb{E}_{M\sim\mathcal{I}_k}[g_i(x^M, \chi_M) \mid A_T] && \text{linearity} \\ &\geqslant g_i(\mathbb{E}_{M\sim\mathcal{I}_k}[x^M \mid A_T], \mathbb{E}_{M\sim\mathcal{I}_k}[\chi_M \mid A_T]) && \text{Jensen's Ineq.} \\ &= g_i(y^T, \chi_T). && \text{Claim 8.6.2} \end{aligned}$$

Similarly, we have that $h_j(y^T, \chi_T) = \mathbb{E}_{M\sim\mathcal{I}_k}[h_j(x^M, \chi_M) \mid A_T] = 0$ for all $j \in [\ell]$, as h_j is affine. We conclude that y^T is a feasible solution to $P(T)$, and therefore $f(y^T, \chi_T) \geqslant P(T)$. Again appealing to Jensen's inequality, recalling that $y^T = \mathbb{E}_{M\sim\mathcal{I}_k}[x^M \mid A_T]$ and that $\mathbb{E}_{M\sim\mathcal{I}_k}[\chi_M \mid A_T] = \chi_T$ by Claim 8.6.2, we find that

$$\mathbb{E}_{M\sim\mathcal{I}_k}[f(x^M, \chi_M) \mid A_T] \geqslant f(y^T, \chi_T) \geqslant P(T).$$

The lemma follows by total expectation over M, relying on $\Pr[A_T] = 1/\binom{|S|}{k}$ for each $T \in \binom{S}{k}$.

$$\begin{aligned} \mathbb{E}_{M\sim\mathcal{I}_k}[P(M)] &= \sum_{T\in\binom{S}{k}} \mathbb{E}_{M\sim\mathcal{I}_k}[P(M) \mid A_T] \cdot \Pr[A_T] \\ &\geqslant \sum_{T\in\binom{S}{k}} P(T) \cdot \Pr[A_T] = \mathbb{E}_{T\sim\mathcal{U}_k}[P(T)]. \end{aligned}$$ □

8.7 Deferred Proofs of Section 8.3

In this section we provide the proofs deferred from Section 8.3.

8.7.1 Implicit Tree Embedding

In Section 8.4, we proved that algorithm FAIR-BIAS is $O(1)$-competitive on tree metrics. Therefore, as noted in Section 8.3, using tree embeddings and applying algorithm FAIR-BIAS to the points according to distances in the obtained tree embedding yields an $O(\log n)$-competitive algorithm for general metrics. Here we present an upper bound on FAIR-BIAS's expected per-arrival cost which implies the same competitive bound, by relying on an *implicit* tree embedding.

Lemma 8.3.1. $\mathbb{E}_{S_k\sim\mathcal{U}_k}[M(S_k)] \leqslant \frac{O(\log n)}{\sqrt{nk}} \cdot \mathbb{E}[\mathrm{OPT}]$.

Proof. For our proof we rely on low-stretch tree embeddings [98]. Given an n-point metric with distances $d_{i,j}$, this embedding is a distribution $\mathcal{D}$ over tree metrics T over the same point set, with tree distances $d^T_{i,j}$ satisfying the following for any two points i, j in the metric.

$$d_{i,j} \leqslant d^T_{i,j}. \tag{8.6}$$

$$\mathbb{E}_{T\sim\mathcal{D}}[d^T_{i,j}] \leqslant O(\log n) \cdot d_{i,j}. \tag{8.7}$$

For such a tree metric T, let $M^T(S)$ denote $M(S)$ with the distances $d_{i,j}$ replaced by $d^T_{i,j}$. (As before, we also let this denote the optimum value of this program.) By (8.6) we immediately have that $M(S) \leqslant M^T(S)$ for any set S, as any solution $\vec{x}$ to $M^T(S)$ is feasible for $M(S)$ and has lower cost for this latter metric, $\sum_{i,j} x_{i,j} \cdot d_{i,j} \leqslant \sum_{i,j} x_{i,j} \cdot d^T_{i,j}$. Consequently, we have

$$M(S) \leqslant \mathbb{E}_{T\sim\mathcal{D}}[M^T(S)]. \tag{8.8}$$

Next, we denote by OPT^T the optimum cost of the min-cost perfect matching of the requests to servers for distances $d^T_{i,j}$. By Lemma 8.4.5 we have that for a tree metric T

$$\mathbb{E}_{S_k\sim\mathcal{U}_k}[M^T(S_k)] \leqslant \frac{4\cdot\mathbb{E}[OPT^T]}{\sqrt{nk}}. \tag{8.9}$$

Finally, for any realization of requests, the minimum-cost matching of requests to servers under $d_{i,j}$ has expected cost (over the choice of T) at most $O(\log n)$ times higher under $d^T_{i,j}$, by (8.7). Therefore, by a coupling argument we get the following bound on $\mathbb{E}_{T\sim\mathcal{D}}\mathbb{E}[OPT^T]$ in terms of $\mathbb{E}[OPT]$.

$$\mathbb{E}_{T\sim\mathcal{D}}[OPT^T] \leqslant O(\log n)\cdot\mathbb{E}[OPT]. \tag{8.10}$$

Combining Equations (8.8), (8.9) and (8.10), we obtain our desired bound.

$$\mathbb{E}_{S_k\sim\mathcal{U}_k}[M(S_k)] \leqslant \mathbb{E}_{T\sim\mathcal{D}}\mathbb{E}_{S_k\sim\mathcal{U}_k}[M^T(S_k)] \leqslant \frac{4\cdot\mathbb{E}_{T\sim\mathcal{D}}\mathbb{E}[OPT^T]}{\sqrt{nk}} \leqslant \frac{O(\log n)\cdot\mathbb{E}[OPT]}{\sqrt{nk}}. \square$$

8.7.2 Load of k Most Loaded Bins

Here we prove our lower bounds on the sum of loads of the k most loaded bins in a balls and bins process with n balls and bins.

Lemma 8.3.3. *Let n balls be thrown into n bins, each ball thrown independently and uniformly at random. Let L_j be the load of the j^{th} heaviest bin, and $N_k := \sum_{j\leqslant k} L_j$ be the number of balls in the k most loaded bins. There exists a constant $C_0 > 0$ such that for any $k \leqslant C_0 n$,*

$$\mathbb{E}[N_k] \geqslant \Omega\left(k\cdot\frac{\log(n/k)}{\log\log(n/k)}\right).$$

Proof. Let $t = \frac{\log(n/k)}{\log\log(n/k)}$, and define

$$f(x_1,\cdots,x_n) = \begin{cases} 1 & \text{if the } k^{th} \text{ largest number in } x_1,\cdots,x_n \text{ is less than } t/2 \\ 0 & \text{otherwise} \end{cases}.$$

Clearly, the function $f(x_1,\cdots,x_n)$ satisfies the condition in Lemma 8.3.2, i.e., $f(x_1,\cdots,x_n)$ is nonnegative and $\mathbb{E}[f(X_1^m,\cdots,X_n^m)]$ is monotonically decreasing with m. Since we have an equal number of balls and bins, we consider the case $m = n$. We abbreviate X_i^n to X_i and Y_i^n to Y_i. Let M_k be the k^{th} largest number among $Y_1, \cdots, Y_n$. Applying Lemma 8.3.2,

$$\Pr[L_k < t/2] = \mathbb{E}[f(X_1, \cdots, X_n)] \leqslant 2 \cdot \mathbb{E}[f(Y_1, \cdots, Y_n)] = 2 \cdot \Pr[M_k < t/2].$$

Define the indicator variable $Z_i := \mathbf{1}_{(Y_i \geqslant t/2)}$, and observe that $\Pr[M_k < t/2] = \Pr[\sum_i Z_i < k]$. We bound the latter via a Chernoff bound, so we need a lower bound on $\mathbb{E}[\sum_i Z_i]$.

$$\mathbb{E}[\sum_i Z_i] = n \cdot \Pr[Y_i \geqslant t/2] \geqslant n \cdot \Pr[Y_i = t/2] \overset{(a)}{=} \frac{n}{e(t/2)!} \overset{(b)}{\geqslant} \frac{4n}{t!} \overset{(c)}{\geqslant} 4k. \tag{8.11}$$

The equality (a) uses the definition of the Poisson distribution, the inequality (b) uses that $t! \geqslant 4e(t/2)!$ for sufficiently large t. For inequality (c), we know $t! \leqslant \sqrt{t}/e\,(t/e)^t$ from Stirling's approximation, and so when n/k is sufficiently large, plugging in $t = \frac{\log(n/k)}{\log\log(n/k)}$ gives

$$\log(t!) \leqslant (t + 1/2)\log t - t - 1 \leqslant t \log t \leqslant \log(n/k).$$

Putting things together, and using a Chernoff bound, we get

$$\Pr[L_k < t/2] \leqslant 2 \cdot \Pr[M_k < t/2] = 2 \cdot \Pr[\sum_i Z_i < k] \leqslant 2e^{-\frac{(3/4)^2 \cdot 4k}{2}} \leqslant 2e^{-k}.$$

The lemma then follows directly, as

$$\mathbb{E}[N_k] \geqslant \mathbb{E}[N_k \mid L_k \geqslant t/2] \cdot \Pr[L_k \geqslant t/2] \geqslant k \cdot (t/2) \cdot (1 - 2e^{-k}) = \Omega\Big(\frac{k \cdot \log(n/k)}{\log\log(n/k)}\Big). \square$$

The following simple lemma states that in the min cost perfect matching, we can always match requests and servers in the same location as much as possible. That is, $x_{ii} = \frac{1}{n}$ for every requested location i.

Lemma 8.3.4. *For sufficiently large n and any $k \leqslant n/2$, we have $\mathbb{E}[N_k] \geqslant 1.5k$.*

Proof. In expectation, there are $n(1 - 1/n)^n \sim n/e$ empty bins, thus on average one would expect $1/(1 - 1/e) > 1.5$ balls in each non-empty bin. To make this intuition formal, let $t = (1 - 1/e + 0.01)n$ and define

$$f(x_1, \cdots, x_n) = \begin{cases} 1 & \text{if more than } t \text{ of } x_1, \cdots, x_n \text{ are greater than } 0 \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to verify that the function $f(x_1, \cdots, x_n)$ is non-negative and $\mathbb{E}[f(X_1^m, \cdots, X_n^m)]$ is monotonically increasing in m. Define the variable $Z_i := \mathbf{1}_{(Y_i > 0)}$; then $Z_i \sim \text{Bernoulli}(1 - 1/e)$. Lemma 8.3.2 and a Chernoff bound now give that for sufficiently large n,

$$\mathbb{E}[f(X_1, \cdots, X_n)] \leqslant 2 \cdot \mathbb{E}[f(Y_1, \cdots, Y_n)] = 2 \cdot \Pr\left[\sum_i Z_i > tn\right] \leqslant 2e^{-\frac{0.01^2 \cdot (1-1/e)n}{2}} < 0.01.$$

Hence

$$\mathbb{E}[N_t] \geqslant \mathbb{E}[N_t \mid f(X_1, \cdots, X_n) = 0] \cdot \Pr[f(X_1, \cdots, X_n) = 0] \geqslant n \cdot (1 - 0.01) = 0.99n.$$

Finally, for $k \leqslant n/2 (\leqslant t)$, we have that indeed $\frac{\mathbb{E}[N_k]}{k} \geqslant \frac{\mathbb{E}[N_t]}{t} \geqslant \frac{0.99n}{(1 - 1/e + 0.01)n} \geqslant \frac{3}{2}$. $\square$

8.7.3 Further Deferred Proofs

Lemma 8.3.5. *Let $\mathcal{I}$ be a fractional min-cost bipartite metric b-matching instance, with demand ℓ_i and r_i for the servers and requests at location i. Then, there exists an optimal solution x for $\mathcal{I}$ with $x_{ii} = \min\{\ell_i, r_i\}$ for every point i in the metric.*

Proof. Fix an optimal solution x^* of $\mathcal{I}$ of maximum $\sum_i x^*_{ii}$ among optimal solutions of $\mathcal{I}$. Suppose for contradiction that there exists some $i \in S_k$ such that $x^*_{ii} < \min\{\ell_i, r_i\}$. WLOG $\ell_i \leqslant r_i$ and so there exists some locations j, j' such that $x^*_{ij} > 0$ and $x^*_{j'i} > 0$. Let $\varepsilon = \min\{x^*_{ij}, x^*_{j'i}\}$. Consider the solution $\tilde{x}$ obtained from x^* by increasing x^*_{ii} and $x^*_{j'j}$ by ε and decreasing x^*_{ij} and $x^*_{j'i}$ by ε. This $\tilde{x}$ is a feasible solution to $\mathcal{I}$ (as sums of the form $\sum_i x_{ij}$ and $\sum_j x_{ij}$ are unchanged and $\tilde{x} \geqslant 0$). Moreover, we find that

$$\begin{aligned}\sum_{ij} d_{ij} \cdot \tilde{x}_{ij} &= \left(\sum_{ij} d_{ij} \cdot x^*_{ij}\right) + \varepsilon \cdot (d_{ii} + d_{jj'} - d_{ij} - d_{ij'}) \\ &= OPT(\mathcal{I}) + \varepsilon \cdot (d_{jj'} - d_{ij} - d_{ij'}) \leqslant OPT(\mathcal{I}),\end{aligned}$$

by triangle inequality. That is, $\tilde{x}$ is an optimal solution to $\mathcal{I}$ with a higher $\sum_i x_{ii}$ than x^*, contradicting our assumption. The lemma follows. $\square$

Lemma 8.3.9. $\sum_{k=1}^{n} \mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)] \leqslant 2 \cdot \sum_{k=1}^{n/2} \mathbb{E}_{S_k \sim \mathcal{U}_k}[M(S_k)]$.

Proof. As noted in the proof of Lemma 8.3.7, by Lemma 8.3.5, the optimal value of $M(S_k)$ is equal to that of a min-cost bipartite perfect b-matching instance with left vertices associated with S_k with demand $\frac{1}{k} - \frac{1}{n}$ and right vertices associated with $S \setminus S_k$ with demand $\frac{1}{n}$. Similarly, $M(S \setminus S_k)$ is equal to the same, but with each $i \in S_k$ having demand $\frac{1}{n}$ and each $i \in S \setminus S_k$ having demand $\frac{1}{n-k} - \frac{1}{n}$. That is, these programs are just scaled versions of each other, and we we have that for any $k \leqslant n/2$,

$$M(S_k) = \frac{1/k - 1/n}{1/n} \cdot M(S \backslash S_k) = \left(\frac{n}{k} - 1\right) \cdot M(S \backslash S_k) \geqslant M(S \backslash S_k).$$

Consequently, taking expectation over S_k (equivalently, over $S \setminus S_k$), we find that for any $k \leqslant n/2$, we have $E_{S_k \sim \mathcal{U}_k}[M(S_k)] \geqslant E_{S_{n-k} \sim \mathcal{U}_{n-k}}[M(S_{n-k})]$. The lemma follows. $\square$

8.8 Max Weight Perfect Matching under i.i.d Arrivals

Here we prove that, with a small modification, FAIR-BIAS achieves the optimal competitive ratio, i.e $1/2$, in the max weight perfect matching problem introduced in [59]. Here, rather than compute a minimum cost perfect matching, we are tasked with computing a maximum weight perfect matching, where the weights need not correspond to a metric. Since we are now in a maximization problem and we are no longer in a metric space, we will not make the assumption that the distribution of all requests is uniform among all servers. Moreover, we make the following modification to our algorithm: in each round of FAIR-BIAS, instead of finding a min cost perfect matching, we would find the max weight perfect matching. Correspondingly, we change the notation for $M(T)$: instead of being a min cost perfect b-matching induced by the set of free servers T and requests R, now $M(T)$ refers to the *max* weight perfect b-matching between the set of free servers T and requests R. More formally, we have

$$\begin{aligned} M(T) := \max & \sum_{i\in T, j\in R} w_{i,j}\cdot x_{i,j} \\ \text{s.t.} & \sum_{j\in T} x_{i,j} = \frac{1}{|T|} \qquad \forall i\in T \\ & \sum_{i\in R} x_{i,j} = p_i \qquad \forall j \in R \\ & x_{i,j} \geqslant 0. \end{aligned} \tag{8.12}$$

Generalizing FAIR-BIAS, if S_k is the realized set of free servers and x^{S_k} an optimal solution to $M(S_k)$, then upon arrival of a request at location i (which happens with probability p_i), we randomly pick a server s to match this request to, chosen with probability $x^{S_k}_{i,s}/p_i$.

Difference compared to [59]. We note that Chang et al. [59] used a similar LP to $M(T)$. Essentially, they used $M(S)$, the program obtained by considering *all* servers (and not just free ones). Following [106, 150], they refer to this as the optimum of the "expected graph". Their algorithm picks a preferred server among all servers with probability $x^{S_k}_{r,s}/p_i$. If this server is already matched, in order to output a perfect matching they randomly (i.e., uniformly) pick an alternative server to match to. Our algorithm does not need to fall back on a second random choice, as it only picks a server among free servers. As we shall see, our algorithm's analysis follows rather directly from our analysis of FAIR-BIAS for the minimization variant.

A key observation is that the structure lemma (Lemma 8.2.2) still holds for our maximization variant of FAIR-BIAS. We restate it here.

Claim 8.8.1. *(Structure Lemma, Restated) For each time k, the set S_k is a uniformly-drawn k-subset of S; i.e., $S_k \sim \mathcal{U}_k$. Consequently, the weight of the algorithm's output matching is*

$$\mathbb{E}[ALG] = \sum_{k=1}^{n} \mathbb{E}_{S_k\sim\mathcal{U}_k}[M(S_k)].$$

Claim 8.8.1 holds due to the same argument in Lemma 8.2.2. Notice that all we needed in the proof of Lemma 8.2.2 is that upon arrival of a request $r_k = i$ when there are k free servers S_k we match $r_k = i$ to a any free server s with probability $x^{S_k}_{i,s}/p_i$, and so we use edge (i,s) with probability precisely $x^{S_k}_{i,s}$. This implies that each free server $s \in S_k$ is matched with probability precisely $\frac{1}{k}$ and that the expected weight of the edge matched is precisely $\sum_{i\in S, j\in S_k} w_{i,j}\cdot x^{S_k}_{i,j}$.

Next, we note that $\mathbb{E}[OPT]$ can be upper bounded in terms of $M(S)$.

Claim 8.8.2. $\mathbb{E}[\text{OPT}] \leqslant n \cdot M(S)$.

The proof is exactly the same as Equation (8.5). See also [59, Lemma 1].

Now we can prove that the maximization variant of FAIR-BIAS is $1/2$ competitive for the max weight perfect matching problem in the i.i.d model.

Theorem 8.8.3. *The max-weight variant of* FAIR-BIAS *is $1/2$ competitive.*

Proof. Letting $x^{S_k} \in \arg\max M(S_k)$ for every S_k, we have the following bound

$$\begin{aligned}
\mathbb{E}_{S_k\sim\mathcal{U}_k}[M(S_k)] &= \sum_{S_k} \frac{1}{\binom{n}{k}} \sum_{i\in S_k, j\in R} w_{i,j}\cdot x^{S_k}_{i,j} && \text{def. of } x^{S_k}\\
&\geqslant \sum_{S_k} \frac{1}{\binom{n}{k}} \sum_{i\in S_k, j\in R} w_{i,j}\cdot x^{S}_{i,j} && \text{def. of } x^{S_k} \text{ and } M(S_k)\\
&= \sum_{i\in S} \Pr_{S_k\sim\mathcal{U}_k}[i\in S_k]\cdot \sum_{j\in R} w_{i,j}\cdot x^{S}_{i,j}\\
&= \frac{k}{n}\cdot M(S) && \text{def. of } M(S)\\
&\geqslant \frac{k}{n^2}\cdot \mathbb{E}[\text{OPT}]. && \textit{Claim } 8.8.2
\end{aligned}$$

Summing these up, by the structure lemma (Claim 8.8.1) we have

$$\mathbb{E}[\text{ALG}] = \sum_{i=1}^{n} \mathbb{E}_{S_k\sim\mathcal{U}_k}[M(S_k)] \geqslant \sum_{k=1}^{n} \frac{k}{n^2}\cdot \mathbb{E}[\text{OPT}] \geqslant \frac{1}{2}\cdot \mathbb{E}[\text{OPT}]. \qquad \square$$

8.9 Conclusion and Open Questions

In this chapter, we presented algorithm FAIR-BIAS and proved that it is $O((\log\log\log n)^2)$-competitive for general metrics, and 9-competitive for tree metrics. Perhaps the first question is whether our algorithm (or indeed any algorithm) is $O(1)$ competitive for (known or unknown) i.i.d arrivals for general metrics. Indeed, we do not know of any instances where Algorithm FAIR-BIAS's performance is worse than $O(1)$ competitive. However, it is not clear how to extend our proofs to establish an $O(1)$ competitive ratio.

Another question is the relationship between the known and unknown i.i.d. models and the random order model. The optimal competitive ratios for the various arrival models for online problems can be sorted as follows (see e.g. [206, Theorem 2.1])

$$C.R.(Adversarial) \geqslant C.R.(Random\,Order) \geqslant C.R.(Unknown\,IID) \geqslant C.R.(Known\,IID).$$

For the online metric matching problem the best bounds known for the above are, respectively, $O(\log^2 n)$ [23], $\Theta(\log n)$, $O(\log n)$ (both [239]), and $O((\log\log\log n)^2)$ (this chapter). Given the lower bound of [239], the main result of this chapter implies that one or both of the inequalities in the chain

$$C.R.(Random\ Order) \geqslant C.R.(Unknown\ IID) \geqslant C.R.(Known\ IID)$$

is strict (and asymptotically so). It would be interesting to see which of these inequalities is strict, by either presenting a $o(\log n)$-competitive algorithm for unknown i.i.d or a $\omega((\log\log\log n)^2)$ lower bound for this model. For the line metric, we give the first constant-competitive algorithm for this well-studied metric under any non-trivial arrivals. Extending this result, and more generally understanding the exact relationships between these arrival models for this simple metric may prove useful in understanding the relationships between the different stochastic arrival models more broadly. Moreover, it would be interesting to study these questions for other combinatorial optimization problems with online stochastic arrivals.

Chapter 9

Random-Order Online Edge Coloring

We now return to the online edge-coloring problem, fist discussed in Chapter 6. Nearly thirty years ago, Bar-Noy, Motwani, and Naor [25] conjectured that an online $(1 + o(1))\Delta$-edge-coloring algorithm exists for n-node graphs of maximum degree $\Delta = \omega(\log n)$. This conjecture remains open in general. In Chapter 6 we proved this conjecture for bipartite graphs under *one-sided vertex arrivals*. Progress was made on this conjecture by [5] and [22] under random-order edge arrivals, though here this conjecture remained unanswered. In this chapter, based on joint work with Sayan Bhattacharya and Fabrizio Grandoni, we resolve this conjecture under random-order edge arrivals.

9.1 Background

Edge coloring is the problem of assigning one of k colors to all edges of a simple graph, so that no two incident edges have the same color. The objective is to minimize the number of colors, k. The edge coloring problem goes back to the 19th century and studies of the four-color theorem [234, 255]. In 1916, König [184], in what many consider to be the birth of matching theory, proved that any bipartite graph of maximum degree Δ is colorable using Δ colors. (Clearly, no fewer colors suffice.) Nearly half a century later, Vizing [261] proved that any general graph is $(\Delta + 1)$-edge-colorable. Vizing's proof is algorithmic, yielding such a coloring in polynomial time. This is likely optimal, as it is NP-hard to determine if a general graph is Δ-edge-colorable [160]. Algorithms for the edge coloring problem have been studied in several different models of computation, including offline, online, distributed, parallel, and dynamic models (see, e.g., [60, 70, 72, 82, 177, 219, 254] and references therein.) In this chapter, we study the edge coloring problem in online settings.

Online edge coloring: Here, an adversary picks an n-node graph G of maximum degree Δ (the algorithm knows n and Δ, but not G), and then reveals the edges of G one at a time. Immediately after the arrival of an edge, the algorithm must irrevocably assign a color to it, with the objective of minimizing the final number of colors used. This problem was first studied nearly thirty years ago, by Bar-Noy, Motwani, and Naor [25]. They showed that the greedy algorithm, which returns a proper $(2\Delta - 1)$-edge coloring, is worst-case optimal among online algorithms. This might seem to be the end of the story for this line of research. However, as pointed out by Bar-Noy et al.

[25], their lower bound only holds for bounded-degree graphs, with some $\Delta = O(\text{poly}\log n)$. This led them to conjecture that online $(1+o(1))\Delta$-edge-coloring is possible for graphs with $\Delta = \omega(\text{poly}\log n)$. This conjecture remains wide open, though in Chapter 6 we resolved it for one-sided vertex arrivals in bipartite graphs. In this chapter we address this problem under (the incomparable) random-order edge arrivals.

Random-order online edge coloring: Here, an adversarially-chosen graph has its edges revealed to the algorithm in *uniformly random order*. Such random-order arrivals, which capture numerous stochastic arrival models, have been widely studied for many online problems. (See, e.g., [176, 181, 189, 201, 208] and the survey by Gupta and Singla [141] and references therein.) In the context of edge coloring, this model was studied by [5, 22]. Aggarwal et al. [5] were the first to show that high Δ suffices for near-ideal coloring in this model, giving a $(1+o(1))\Delta$-edge-coloring algorithm for *multigraphs* with $\Delta = \omega(n^2)$. Bahmani et al. [22] then breached the greedy $2\Delta - 1$ barrier for simple graphs with polylogarithmic Δ, giving a 1.26Δ-edge coloring algorithm for $\Delta = \omega(\log n)$. This leads to the following natural open question: can one obtain "the best of both worlds" w.r.t. [5, 22]? That is, can one obtain a $(1+o(1))\Delta$-edge-coloring for graphs of maximum degree $\Delta = \omega(\log n)$ whose edges are presented in random order? Put another way, is the Bar-Noy et al. conjecture true for random-order edge arrivals? We answer this question in the affirmative.

Theorem 9.1.1. *For some absolute constant $\gamma \in (0,1)$, there exists an online algorithm that, when given a graph G of maximum degree $\Delta = \omega(\log n)$, whose edges are presented in random order, computes a proper $\left(\Delta + O\left(\Delta^\gamma \cdot \log^{1-\gamma} n\right)\right) = (1+o(1))\Delta$-edge-coloring of G w.h.p.*

We complement this upper bound with a lower bound showing that, for some $\Delta = O(\sqrt{\log n})$, not only is it impossible to guarantee a $(1+o(1))\Delta$-edge-coloring under random-order arrivals, but it is even impossible to use any fewer than $2\Delta - 1$ colors. (See Section 9.4.)

We note that previous random-order online edge coloring algorithms [5, 22] required the graph to be Δ-regular. This assumption is without loss of generality in an offline setting, but it is unclear whether the same holds in the random-order online model. Our algorithm from Theorem 9.1.1, however, works on any graph (including non-regular ones): this is discussed in Section 9.3.1.

Remark. In the same joint work with Sayan Bhattacharya and Fabrizio Grandoni which this chapter is based on, we presented improved algorithms for dynamic settings, based on the same underlying basic algorithm. In particular, we gave dynamic $(1+\varepsilon)\Delta$-edge-coloring algorithms with constant *recourse*, or number of changes per edge arrival/departure, improving on previous $\text{poly}\log n$-recourse algorithms due to Duan et al. [82]. However, to keep this chapter consistent with the remainder of this part of the thesis, we do not elaborate on this result.

9.1.1 Our Techniques

Underlying our result is an algorithmic approach inspired by the Rödl Nibble Method [11], as applied to distributed edge coloring by Dubhashi et al. [84]. This method and its variants have since found further uses in distributed settings [60, 91]. To the best of our knowledge, the results described here are the first to export this method to online settings.

We analyze our basic algorithm, which is a variant of [84], in an offline model. We then show how to implement this algorithm in online and dynamic settings, from which we obtain our results. We now outline this basic algorithm, and the ideas needed to implement it in the models we study. For simplicity, we focus on Δ-regular graphs in this section.

The Basic Algorithm: The Nibble Method in the framework of edge coloring was first used in [84] in the distributed model. Let us sketch how their algorithm would work in the offline setting. The algorithm consists of multiple rounds. In each round, each vertex v selects a random ε fraction of its incident uncolored edges. Each sampled edge e chooses a tentative color u.a.r. among the colors in $[\Delta]$ not yet taken by incident edges (*palette* of e). We then assign the tentative color $c(e)$ to sampled edges e for which no incident edge e' picked the same tentative color $c(e)$, else we mark e as *failed*, and leave e uncolored. It turns out that each sampled edge fails at each round with probability $O(\varepsilon)$. Crucially, picking ε appropriately results in a number of important parameters (degrees in the uncolored subgraph, palette sizes, etc') behaving in a predictable manner, and being sharply concentrated around their mean, w.h.p. In particular, this results in the uncolored subgraph's maximum degree decreasing w.h.p. at a rate of roughly $1-\varepsilon$ per application of this subroutine, or *round*. Consequently, some $t_\varepsilon = O(\log(1/\varepsilon)/\varepsilon)$ rounds leave an uncolored subgraph of maximum degree $\Delta' = \text{poly}(\varepsilon)\Delta$ w.h.p., which can then be greedily colored using a further $2\Delta' = \text{poly}(\varepsilon)\Delta$ colors. This approach therefore yields a proper $(1 + \text{poly}(\varepsilon))\Delta$ edge coloring.

In part inspired by [60], we consider a slight modification of the above algorithm which is more convenient for our goals. In more detail, we make the following changes:
(1) We do not attempt to re-color an edge e which fails in a given round in future rounds, instead leaving e to be colored greedily in the final stage. Intuitively, ignoring these edges still results in a low-degree uncolored graph after t_ε rounds, since few edges incident to each vertex fail.
(2) Whenever an edge e picks a tentative color c, we remove c from the palettes of its incident edges even if e fails. Intuitively, this does not decrease the palette sizes much, again, since few edges incident to each vertex fail.
(3) We sample each edge independently with probability ε in each round.

Our modifications bring two main advantages. First, the analysis can be substantially simplified: rather than using a specialized concentration inequality of Grable [137], we mostly use Hoeffding bounds for negatively-associated variables (see Section 2.4.1). This allows us to provide a relatively concise, but complete analysis for sub-constant values of ε and for non-regular graphs. Second, and importantly for us, it is easier to adapt the modified algorithm to the online setting that we study.

Random-Order Online Implementation: To obtain our results for random-order arrivals, we first observe that our edge-centric sampling of modification (3) allows us to use the randomness of edge arrivals to "sample edges for us". More formally, we implement the independent edge-sampling part of each round by considering an appropriate binomially-distributed prefix of the remaining edges (relying on our knowledge of the number of edges of the Δ-regular graph, $m = \frac{n\Delta}{2}$). This results in each remaining edge of the graph being sampled independently with probability ε.

For each round, we have each edge of the round sample a tentative color u.a.r. from its palette. In this online setting, however, we cannot always tell when an edge arrives whether it picked the same tentative color as its incident edges of the same round (since some of these arrive *later*). We therefore assign the tentative color $c(e)$ to sampled edges e for which no *previous* incident edge e' picked the same tentative color $c(e)$, else we mark e as *failed*. Modification (2) in the basic algorithm implies that this change still results in a feasible (partial) coloring. On the other hand, the uncolored subgraph "after" the rounds in this algorithm clearly has lower maximum degree than its counterpart in the basic algorithm, and so greedily coloring this subgraph requires fewer colors than the same stage of the basic algorithm. Finally, modification (1) of our basic algorithm, whereby we do not attempt to re-color a failed edge in "future rounds" (which would require knowledge of future arrivals), implies that we can greedily color every failed edge before the next edge arrives. So, by the analysis of our basic algorithm, we obtain Theorem 9.1.1 for Δ-regular graphs. In Section 9.3 we build on this approach to obtain our full result, for general graphs.

9.2 The Basic Algorithm

In this section, we describe our basic algorithm for near-regular graphs, and state the key theorem needed for its analysis. We defer a more detailed analysis to Section 9.6.

The input to the algorithm is a graph $G = (V, E)$ with $|V| = n$ nodes, where the degree of each node lies in the interval $[(1-\varepsilon^2)\Delta, (1+\varepsilon^2)\Delta]$. The parameter ε satisfies the following condition:

$$1/10^4 \geqslant \varepsilon \geqslant 10 \cdot (\ln n/\Delta)^{1/6}. \tag{9.1}$$

Note that such ε exist if $\Delta = \Omega(\log n)$ is large enough. The algorithm runs in two *phases*, as follows.

Phase One. In phase one, the algorithm properly colors a subset of edges of G using $(1+\varepsilon^2)\Delta$ colors, while leaving an uncolored subgraph of small maximum degree. This phase consists of $t_\varepsilon - 1$ *rounds* $\{1, \ldots, t_\varepsilon - 1\}$, for

$$t_\varepsilon := \lfloor \ln(1/\varepsilon)/(2K\varepsilon) \rfloor, \text{ and } K = 48. \tag{9.2}$$

Each round $i \in [t_\varepsilon - 1]$ operates on a subgraph $G_i := (V, E_i)$ of the input graph (with $E_1 = E$), identifies a subset of edges $S_i \subseteq E_i$, picks a *tentative* color $c(e) \in [(1+\varepsilon^2)\Delta] \cup \{\text{null}\}$ for each edge $e \in S_i$, and returns the remaining set of edges $E_{i+1} = E_i \setminus S_i$ for the next round. Thus, we have: $E = E_1 \supseteq E_2 \supseteq \cdots \supseteq E_{t_\varepsilon}$.

We now introduce a couple of notations that will be useful in subsequent discussions. (a) For all $i \in [t_\varepsilon - 1]$ and $v \in V$, let $P_i(v) := \{\chi \in [(1+\varepsilon^2)\Delta] : \chi \neq c(u,v) \text{ for all } (u,v) \in \bigcup_{j<i} S_j\}$ denote the *palette* of the node v for round i. A color $\chi \in [(1+\varepsilon^2)\Delta]$ belongs to $P_i(v)$ iff no edge incident on v has tentatively picked the color χ in previous rounds $j < i$. (b) Similarly, for all $i \in [t_\varepsilon - 1]$ and $(u,v) \in E_i$, let $P_i(u,v) := P_i(u) \cap P_i(v)$ denote the *palette* of the edge (u,v) for round i.

We now describe each such round i. First, we sample each edge $e \in E_i$ *independently* with probability ε. Let $S_i \subseteq E_i$ be the set of sampled edges. Next, each edge $e \in S_i$ with

$P_i(e) \neq \emptyset$ tentatively picks a color $c(e)$ from its palette $P_i(e)$ uniformly and independently at random. We say that an edge $e \in S_i$ *failed* in round i iff either (a) $P_i(e) = \emptyset$ (in this case we set $c(e) :=$ null), or (b) among the edges $N(e) \subseteq E$ that are adjacent to e, there is some edge $e' \in S_i$ that tentatively picked the same color (i.e., $c(e) = c(e')$). Let $F_i \subseteq S_i$ denote the set of failed edges in round i. The remaining sampled edges $e \in S_i \setminus F_i$ are called *successful* in round i. Each such edge $e \in S_i \setminus F_i$ is *assigned* the color $c(e)$ it tentatively picked in round i. Before terminating the current round, we set $E_{i+1} := E_i \setminus S_i$ and $G_{i+1} := (V, E_{i+1})$. We remark that the color tentatively sampled by a failed edge e cannot be used by the edges incident to e in subsequent rounds. This will prove useful both for our analysis and when implementing this algorithm in other models in subsequent sections.

Phase Two. Finally, in phase two, we greedily color all edges that were not successful in phase one. That is, letting $G_F := (V, \cup_i F_i)$ be the subgraph consisting of all the edges that failed in phase one, and $G_{t_\varepsilon} := (V, E_{t_\varepsilon})$ be the subgraph consisting of all the edges that were never sampled in phase one, we color the edges of $G_{t_\varepsilon} \cup G_F$ greedily, using a new palette of $2\Delta(G_{t_\varepsilon} \cup G_F) - 1$ colors. Here $\Delta(H)$ denotes the maximum degree in any graph H.

Algorithm 14 The Basic Algorithm

```
1:  $E_1 \leftarrow E$ and $G_1 \leftarrow (V, E_1)$
2:  for $i = 1, 2, \ldots, t_\varepsilon - 1$ do
3:      $S_i \leftarrow \emptyset$
4:      for each $e \in E_i$ independently do
5:          with probability $\varepsilon$, add $e$ to $S_i$
6:          $P_i(e) \leftarrow [(1+\varepsilon^2)\Delta] \setminus \{c(e') \mid e' \in N(e) \cap \bigcup_{j<i} S_j\}$.
7:          If $P_i(e) \neq \emptyset$, sample $c(e) \sim_R P_i(e)$, else set $c(e) \leftarrow$ null      ▷ tentative coloring of $e$
8:      let $F_i \leftarrow \{e \in S_i \mid c(e) \in \{\text{null}\} \cup \{c(e') \mid e' \in N(e) \cap S_i\}\}$      ▷ the set of failed edges
9:      color each edge $e \in S_i \setminus F_i$ using color $c(e)$
10:     $E_{i+1} \leftarrow E_i \setminus S_i$ and $G_{i+1} \leftarrow (V, E_{i+1})$.
11: let $G_F := (V, \bigcup_i F_i)$ denote the subgraph of $G$ consisting of failed edges.
12: color $G_{t_\varepsilon} \cup G_F$ greedily using colors $(1+\varepsilon^2)\Delta + 1, \ldots, (1+\varepsilon^2)\Delta + 2\Delta(G_{t_\varepsilon} \cup G_F) - 1$.
```

The algorithm's pseudocode is given in Algorithm 14. We now turn to discussing its analysis.

Observation 9.2.1. *Algorithm 14 outputs a proper* $((1+\varepsilon^2)\Delta + 2\Delta(G_{t_\varepsilon} \cup G_F) - 1)$*-edge-coloring of the input graph* $G = (V, E)$.

Proof. First observe that the algorithm computes a valid partial coloring in Phase One. Indeed, any $e \in S_i \setminus F_i$ selects a color $c(e) \in P_i(e) \subseteq [(1+\varepsilon^2)\Delta]$, and the definition of $P_i(e)$ and F_i guarantees that no other edge $e' \in N(e)$ in any round of Phase One can be colored with $c(e)$. The claim follows by observing that in Phase One we use only colors from $[(1+\varepsilon^2)\Delta]$, while in Phase Two the greedy algorithm uses a disjoint set of at most $2\Delta(G_{t_\varepsilon} \cup G_F) - 1$ extra colors. □

The key property of the basic algorithm is captured in the following theorem.

Theorem 9.2.2. $\Delta(G_{t_\varepsilon} \cup G_F) = O\left(\varepsilon^{1/(3K)}\Delta\right)$ *w.h.p.*

Corollary 9.2.3. *The basic algorithm* $\left(\Delta + O\left(\varepsilon^{1/(3K)}\Delta\right)\right)$ *edge colors* G*, w.h.p.*

Proof. Follows from Theorem 9.2.2 and Observation 9.2.1. □

In some sense, the arguments behind the proof of Theorem 9.2.2 were already apparent in the work of Dubhashi et al. [84]. Consequently, we defer a complete and self-contained proof of this theorem to Section 9.6. For now, we turn to exploring implications of this theorem and Algorithm 14 to online edge coloring.

9.3 Random-Order Online Algorithm

In this section we present algorithms which (essentially) implement Algorithm 14 in the random-order online model. We start with a warm-up case, where the input graph is near-regular, and we know the value of m (the number of edges in the final graph).

9.3.1 Warm-up: Near-Regular Graphs with Known m

One subroutine we rely on is the ability to use the stream's randomness to simulate independent sampling of edges. For completeness, we provide a proof of the following simple fact in Section 9.5.

Fact 9.3.1. *Consider a universe* U *of* n *elements. Let* $U_k \subseteq U$ *denote the first* k *elements in a random-order stream of* U*. Then, for* $X \sim Bin(n, p)$ *a binomial random variable with parameters* n *and* p*, the random set* U_X *contains every element in* U *independently with probability* p*.*

Using Fact 9.3.1, we simulate (a variant of) Algorithm 14 with parameter ε under random-order edge arrivals in a graph $G = (V, E)$ with m edges and n nodes, where the degree of each node lies in the interval $(1 \pm \varepsilon^2)\Delta$, and $\Delta = \omega(\log n)$. The algorithm knows n, Δ and m (but not G).

Warm-up Algorithm: Set $\varepsilon := 10 \cdot (\ln n/\Delta)^{1/6}$ (see (9.1)). For round $i = 1, \dots, t_\varepsilon - 1$, sample an independent random variable $X_i \sim Bin(m - \sum_{j<i} X_j, \varepsilon)$, and let S_i be the set of edges in G whose positions in the random-order stream lie in the interval $(\sum_{j<i} X_j, \sum_{j\leqslant i} X_j]$. As with Algorithm 14, each edge $e \in S_i$, upon its arrival, samples a tentative color

$$c(e) \sim_R P_i(e) := [(1 + \varepsilon^2)\Delta] \setminus \{c(e') \mid e' \in N(e) \cap S_j,\ j < i\},$$

where we set $c(e) \leftarrow null$ if $P_i(e) = \emptyset$. Unlike in Algorithm 14, in this online setting the algorithm cannot know whether the color $c(e)$ conflicts with the tentative color of a neighboring edges $e' \in N(e) \cap S_i$ that arrives in the same round i, but *after* e in the stream. Hence, we color each edge $e \in S_i$ with its tentative color $c(e)$, unless $c(e) = null$ or some *previously-arrived* neighboring edge $e' \in N(e) \cap S_i$ also picked color $c(e') = c(e)$. In the latter case, we instead color e greedily with the first available color $j > (1+\varepsilon^2)\Delta$. We let F'_i be the edges in S_i which are colored greedily.

As we show, this online algorithm inherits the performance of the basic Algorithm 14.

Theorem 9.3.2. *For some absolute constant $\gamma \in (0,1)$, the warm-up algorithm described above yields a proper $\left(\Delta + O\left(\Delta^\gamma \cdot \log^{1-\gamma} n\right)\right) = (1+o(1))\Delta$-edge coloring of G w.h.p.*

Proof. This algorithm outputs a valid edge coloring, as it colors every edge (due to the greedy stage) and never assigns an edge a color used by an incident edge. It remains to bound its performance.

For any $i \geqslant 0$, Let E_i be the set of edges whose positions in the random-order stream lie in the interval $(\sum_{j<i} X_j, m]$. By Fact 9.3.1, the set of edges S_i is a random subset of E_i which contains each edge in E_i independently with probability ε. A simple induction on i shows that the sets S_i and E_i share the same distributions as their counterparts in Algorithm 14. Next, denote by $F_i \supseteq F'_i$ the set of edges $e \in S_i$ for which $c(e) \in \{null\} \cup \{c(e') \mid e' \in N(e) \cap S_i\}$. Since each edge $e \in S_i$ picks a color uniformly at random from the set of colors not picked by any of its neighboring edges in previous rounds (including the edges in F_j for all $j < i$), a simple induction on i shows that the random variables F_i and $c(e)$ in this algorithm are distributed exactly as their counterparts in Algorithm 14. Consequently, the upper bounds on $\Delta(\bigcup_i F_i) \geqslant \Delta(\bigcup_i F'_i)$ and $\Delta(G_{t_\varepsilon})$ of Algorithm 14 hold for this online algorithm as well. Therefore, the greedy (online) algorithm colors the uncolored edges in $G_{t_\varepsilon} \cup G_F$ using at most $2 \cdot \Delta(G_{t_\varepsilon} \cup G_F) - 1 = O(\varepsilon^{1/(3K)}\Delta)$ colors w.h.p., by Theorem 9.2.2 and our choice of $\varepsilon = 10 \cdot (\ln n/\Delta)^{1/6}$, as in (9.1). As we use $(1+\varepsilon^2)\Delta$ distinct colors for all other edges, this online algorithm uses $\Delta + O\left(\varepsilon^{1/(3K)}\Delta\right)$ colors overall w.h.p. Since $\Delta = \omega(\log n)$ and K is an absolute constant (see (9.2)), the theorem follows from our choice of ε. $\square$

Assuming near-regularity, and known m. The assumption of near-regularity used by the above algorithm is common in the literature. Indeed, all prior random-order online edge-coloring algorithms assume perfect regularity [5, 22]. As pointed out in those papers, this assumption is without loss of generality in the offline model, where we can add dummy edges to make the graph regular. In a random-order online setting, this is problematic, however, as these dummy edges should be interspersed among the real edges to create a regular graph *presented in random order*. This last point seems impossible without prior knowledge of vertices' final degrees, and the number of edges, m, which we assume prior knowledge of. In the next section we show how to remove the assumption of near-regularity, as well as knowledge of m, while retaining the asymptotic performance of Theorem 9.3.2.

9.3.2 General Graphs

We now present and analyze our random-order online edge coloring algorithm for general graphs $G = (V, E)$ with n nodes, m edges and maximum degree $\Delta = \omega(\log n)$. The algorithm knows n, Δ; but does *not* know m or G. Let $e_1, \dots, e_m$ be the random stream of edges, $G^{(k)}$ be the subgraph induced by $e_1, \dots, e_k$, and $d^{(k)}(v)$ be the degree of node v in $G^{(k)}$. Our key insight is to observe the first few edges in the input stream until we are able to infer (approximately) the value of m and the degree of each node in G.

In more detail, our algorithm consists of 3 main steps. In Step (I), we observe the first T edges until some node v reaches the degree $d^{(T)}(v) = \varepsilon\Delta$ (or we reach the end of the stream). This first set of edges is colored greedily using the first available color. Let Δ_1 be the largest color used in Step (I). The following technical lemma follows from a standard application of Chernoff bounds over sums of negatively-associated variables (proof in Section 9.5).

Lemma 9.3.3. *Let $\varepsilon \leqslant \frac{1}{2}$, and let $\alpha > 0$ be a constant, and assume $\Delta \geqslant \frac{24(\alpha+3)\ln n}{\varepsilon^8}$. Then, with probability at least $1 - O(n^{-\alpha})$, the following properties hold:*

1. *$T = \varepsilon \cdot m(1 \pm \varepsilon^2)$.*
2. *$d^{(T)}(v) = \varepsilon \cdot d(v) \pm 2\varepsilon^3\Delta$ for every node v.*
3. *Let $m' := T/(\varepsilon(1+\varepsilon^2))$. Conditioned on $m' \leqslant m$, every node v has $d(v) - d^{(m')}(v) \leqslant 2\varepsilon^2\Delta$.*

Henceforth, we assume that all the high-probability events in Lemma 9.3.3 actually occur (otherwise the algorithm fails). In Step (II), we color the next $m' - T$ edges $R := \{e_{T+1}, \dots, e_{m'}\}$ using colors larger than Δ_1, as described below.

Let $G_R = (V, R)$ denote the subgraph of G induced by the edges in R. Before processing the $(T+1)^{th}$ update e_{T+1}, we virtually expand G_R by adding *dummy nodes* W and *dummy edges* D in the following manner. For each node $v \in V$, create Δ dummy nodes $v_1, v_2, \dots, v_\Delta$ which form a Δ-clique via dummy edges, and add extra dummy edges from v to $\max\{0, \Delta - (1/\varepsilon - 1) \cdot d^{(T)}(v)\}$ of these dummy nodes $\{v_1, \dots, v_\Delta\}$. Let H be the resulting graph. Note that at this point we only know the dummy edges in H, as the edges in G_R will arrive in future.

In Step (II), we run the warm-up online algorithm $\mathcal{A}$ (from Section 9.3.1) with parameter 2ε on H, where the edges of H are presented to $\mathcal{A}$ in *random order*. More precisely, initializing $j = T + 1$, $D' = D$ and $R' = R$, we perform the following operations for $|R| + |D|$ iterations.

- With probability $|D'|/(|R'| + |D'|)$, we sample a random edge e_d from D', and feed the edge e_d to the online algorithm $\mathcal{A}$, and set $D' \leftarrow D' \backslash \{e_d\}$ before going to the next iteration.
- With remaining probability, we feed the edge e_j to $\mathcal{A}$ and color e_j with the color $\chi(e_j) + \Delta_1$, where $\chi(e)$ is the color chosen by $\mathcal{A}$ for e_j, and set $R' \leftarrow R' \setminus \{e_j\}$ and increase j by one.

Let Δ_2 be the largest color chosen in Step (II).

Finally, in Step (III), we color the remaining edges $e_{m'+1}, \dots, e_m$ greedily with the first available color $j > \Delta_2$. Let Δ_3 be the largest color used at the end of Step (III).

We next analyze the above algorithm (assuming the occurrence of the high probability events from Lemma 9.3.3). Obviously this algorithm computes a feasible coloring. By definition, Step (I) uses $\Delta_1 \leqslant 2\varepsilon\Delta$ colors. Analogously, by Item 3 of Lemma 9.3.3, the number of colors used in

Step (III) is at most $\Delta_3 - \Delta_2 = O(\varepsilon^2\Delta)$. It remains to upper bound the number of colors $\Delta_2 - \Delta_1$ used in the second step. To this end, we note that Lemma 9.3.3 implies that H is near-regular. More precisely, we have the following bound, whose proof is deferred to Section 9.5.

Lemma 9.3.4. *The graph H satisfies $d_H(v) = \Delta(1 \pm 4\varepsilon^2)$ for all $v \in V(H)$, w.h.p.*

It is easy to see that Step (II) implements the warm-up algorithm on H, as the edges of H are fed to this algorithm in a uniform random order. Thus, by Theorem 9.3.2, w.h.p. the number of colors used in Step (II) is at most $\Delta_2 - \Delta_1 \leqslant \Delta + O\left(\Delta^\gamma \cdot \log^{1-\gamma} n\right)$ for a constant $\gamma \in (0,1)$. By choosing ε small enough so that $\varepsilon\Delta \leqslant \Delta^\gamma \cdot \log^{1-\gamma} n$, we immediately obtain our main result.

Theorem 9.1.1. *For some absolute constant $\gamma \in (0,1)$, there exists an online algorithm that, when given a graph G of maximum degree $\Delta = \omega(\log n)$, whose edges are presented in random order, computes a proper $\left(\Delta + O\left(\Delta^\gamma \cdot \log^{1-\gamma} n\right)\right) = (1 + o(1))\Delta$-edge-coloring of G w.h.p.*

9.4 A Lower Bound

Bar-Noy et al. [25] gave a simple lower bound for edge coloring under adversarial arrivals. Specifically, they showed a family of graphs $\mathcal{F}$ with maximum degree $\Delta = O(\sqrt{\log n})$ for which any randomized online algorithm $\mathcal{A}$ colors some graphs in $\mathcal{F}$ with $2\Delta - 1$ colors with constant probability. Extending these ideas slightly, we show that the same holds even if the arrival order is randomized.

Lemma 9.4.1. *There exists a distribution over n-node graphs $\mathcal{G}$ of maximum degree $\Delta = \Omega(\sqrt{\log n})$, for which any online edge coloring algorithm $\mathcal{A}$ must, with constant probability, use $2\Delta - 1$ colors on a graph $G \sim \mathcal{G}$ presented in random order.*

Proof. Consider a star on $\Delta - 1$ leaves. If Algorithm $\mathcal{A}$ uses $2\Delta - 2$ or fewer colors, then it may color any such star's edges with $\binom{2\Delta-2}{\Delta-1}$ possible subsets of colors. If Δ such stars' edges are colored using the same subset $S \subseteq [2\Delta - 2]$ of $\Delta - 1$ colors some node v neighbors the roots of these Δ stars, then the algorithm fails, as it is forced to use Δ colors outside of S for the edges of v, for a total of $2\Delta - 1$ distinct colors. We show a random graph $\mathcal{G}$ for which this bad event happens with constant probability, even when the edges are presented in random order.

Our graph consists of independent copies of the following random graph, $\mathcal{H}$. The graph $\mathcal{H}$ contains $\beta := 2\Delta \cdot \binom{2\Delta-2}{\Delta-1} \cdot \binom{2\Delta-1}{\Delta} \leqslant 4^{O(\Delta)}$ stars with $\Delta - 1$ leaves, and one node v which neighbors the centers of Δ randomly-chosen such stars. For any star, the probability that all $\Delta - 1$ edges of the star arrive before any of the Δ edges of v arrive is $\frac{(\Delta-1)!\Delta!}{(2\Delta-1)!} = 1/\binom{2\Delta-1}{\Delta}$. Therefore, by linearity of expectation, if we denote by X the fraction of stars in $\mathcal{H}$ whose edges arrive before all edges of v, we have that $\mu := \mathbb{E}[X] = 1/\binom{2\Delta-1}{\Delta}$. By Markov's inequality applied to the non-negative variable $Y := 1 - X$, whose expectation is $\mathbb{E}[Y] = 1 - \mu$, we have that

$$\Pr\left[X \leqslant \frac{\mu}{2}\right] = \Pr\left[Y \geqslant 1 - \frac{\mu}{2}\right] \leqslant \frac{1-\mu}{1-\frac{\mu}{2}} = 1 - \frac{\mu}{2-\mu} \leqslant 1 - \frac{\mu}{2}. \tag{9.3}$$

Now, if $X \geqslant \frac{\mu}{2} = 1/(2 \cdot \binom{2\Delta-1}{\Delta})$, then at least $\beta \cdot \frac{\mu}{2} = \Delta \cdot \binom{2\Delta-2}{\Delta-1}$ of the stars of $H \sim \mathcal{H}$ have all their edges arrive before any edge of v arrives. By pigeonhole principle, some Δ of these stars are colored with a common set of $\Delta - 1$ colors, $S \subset [2\Delta - 1]$. If v neighbors the roots of Δ such stars whose edges are colored with the colors in S, then Algorithm $\mathcal{A}$ fails, as it must color the graph using $2\Delta - 1$ colors, as argued above. Therefore, conditioned on $X \geqslant \frac{\mu}{2}$, Algorithm $\mathcal{A}$ fails when coloring $H \sim \mathcal{H}$ with probability at least

$$\Pr\left[\mathcal{A} \text{ fails on } H \sim \mathcal{H} \,\middle|\, X \geqslant \frac{\mu}{2}\right] \geqslant 1/\binom{\Delta \cdot \binom{2\Delta-2}{\Delta-1}}{\Delta} \geqslant 4^{-O(\Delta^2)}. \tag{9.4}$$

Consequently, combining Equation (9.3) and Equation (9.4), and using $\mu = 1/\binom{2\Delta-1}{\Delta} = 4^{-O(\Delta)}$, we find that the unconditional probability of Algorithm $\mathcal{A}$ not failing due to H is at most

$$\begin{aligned}
\Pr[\mathcal{A} \text{ does not fail on } H \sim \mathcal{H}] &\leqslant 1 - \Pr\left[\mathcal{A} \text{ fails on } H \sim \mathcal{H} \,\middle|\, X \geqslant \frac{\mu}{2}\right] \cdot \Pr\left[X \geqslant \frac{\mu}{2}\right] \\
&\leqslant 1 - 4^{-O(\Delta^2)} \cdot \frac{\mu}{2} \\
&= 1 - 4^{-O(\Delta^2)}.
\end{aligned}$$

As stated above, the random graph $\mathcal{G}$ we consider consists of some γ independent copies of $\mathcal{H}$. For independent copies of $\mathcal{H}$, the above upper bound on the probability of $\mathcal{A}$ not failing on a copy $H \sim \mathcal{H}$ holds independently of other copies' realization and coloring by $\mathcal{A}$. Therefore, letting $\mathcal{G}$ consist of some sufficiently large $\gamma := 4^{\Theta(\Delta^2)}$ independent copies of $\mathcal{H}$, we have that

$$\Pr[\mathcal{A} \text{ does not fail on } G] \leqslant \left(1 - 4^{-O(\Delta^2)}\right)^{\gamma} \leqslant \frac{1}{e}.$$

Therefore, Algorithm $\mathcal{A}$ fails on G with constant probability. The lemma follows by noting that G consists of some $n = \gamma \cdot (\beta + 1) = 4^{\Theta(\Delta^2)}$ nodes, and therefore $\Delta = \Omega(\sqrt{\log n})$. $\square$

9.5 Deferred Proofs from Section 9.3

In this section we give the full proofs deferred from Section 9.3, restated below for ease of reference.

We start by proving Fact 9.3.1, which intuitively implies that we can use the stream's random order to sample edges independently.

Fact 9.3.1. *Consider a universe U of n elements. Let $U_k \subseteq U$ denote the first k elements in a random-order stream of U. Then, for $X \sim Bin(n, p)$ a binomial random variable with parameters n and p, the random set U_X contains every element in U independently with probability p.*

Proof. For any given subset $S \subseteq U$ of size $|S| = k$, we have $U_k = S$ precisely when $X = k$ and the elements of S are the first k elements in the stream. As X is independent of the stream's randomness, this gives us:

$$\Pr[U_k = S] = \Pr[X = k] \cdot \Pr[S \text{ is a prefix of the stream}] = \Pr[X = k]/\binom{n}{k} = p^k(1-p)^{n-k}.$$

This is precisely the probability of getting a specific set S, when each of the n elements in U is sampled independently with probability p. □

Next, we prove Lemma 9.3.3, which asserts that the edges until time T where some vertex reaches degree $\varepsilon \cdot \Delta$ gives sharp estimates of natural graph parameters, such as the number of edges, and every vertex's degree, w.h.p.

Lemma 9.3.3. *Let $\varepsilon \leqslant \frac{1}{2}$, and let $\alpha > 0$ be a constant, and assume $\Delta \geqslant \frac{24(\alpha+3)\ln n}{\varepsilon^8}$. Then, with probability at least $1 - O(n^{-\alpha})$, the following properties hold:*

1. $T = \varepsilon \cdot m(1 \pm \varepsilon^2)$.
2. $d^{(T)}(v) = \varepsilon \cdot d(v) \pm 2\varepsilon^3\Delta$ *for every node* v.
3. *Let* $m' := T/(\varepsilon(1+\varepsilon^2))$. *Conditioned on* $m' \leqslant m$, *every node* v *has* $d(v) - d^{(m')}(v) \leqslant 2\varepsilon^2\Delta$.

Proof. The proof relies on several applications of Chernoff bound and union bound, as follows. Fix some vertex v, and let X_i be an indicator variable for the i-th edge in the stream containing v. Clearly, we have that $\mathbb{E}[X_i] = \frac{d(v)}{m}$, and so by linearity of expectation

$$\mathbb{E}[d^{(k)}(v)] = k \cdot \frac{d(v)}{m}. \tag{9.5}$$

On the other hand, the joint distribution $(X_1, \ldots, X_m)$ is a permutation distribution, and so it is NA. We may therefore apply Chernoff bounds to sums of such variables, such as $d^{(k)}(v) = \sum_{i=1}^{k} X_i$. We will make use of this to prove that the three properties hold w.h.p.

We begin by proving Property 1. Fix any vertex v of degree $d(v) = \Delta$. For any $k \geqslant \varepsilon \cdot m(1+\varepsilon^2)$, we have by Equation (9.5) that $\mathbb{E}[d^{(k)}(v)] \geqslant \varepsilon \cdot \Delta(1+\varepsilon^2)$. Consequently, by Chernoff bound, we have

$$\begin{aligned}
\Pr\left[d^{(k)}(v) \leqslant \varepsilon \cdot \Delta\right] &= \Pr\left[d^{(k)}(v) \leqslant \frac{1}{1+\varepsilon^2} \cdot \varepsilon \cdot \Delta(1+\varepsilon^2)\right] \\
&\leqslant \Pr\left[d^{(k)}(v) \leqslant (1-\varepsilon^2) \cdot \varepsilon \cdot \Delta(1+\varepsilon^2)\right] & 1 - x \leqslant \frac{1}{1+x} \ \forall x > -1 \\
&\leqslant \exp\left(\frac{-\varepsilon^4 \cdot \varepsilon \cdot \Delta(1+\varepsilon^2)}{3}\right) & \text{Lemma 2.4.10} \\
&\leqslant n^{-\alpha}. & \Delta \geqslant \frac{3\alpha \log n}{\varepsilon^5}
\end{aligned}$$

Therefore, $\Pr[T \geqslant \varepsilon \cdot m(1+\varepsilon^2)] \leqslant n^{-\alpha}$.

Now, fix a vertex v. For any $k \leqslant \varepsilon \cdot m(1-\varepsilon^2)$, we have by Equation (9.5) that $\mathbb{E}[d^{(k)}(v)] \leqslant \varepsilon \cdot d(v)(1-\varepsilon^2) \leqslant \varepsilon \cdot \Delta(1-\varepsilon^2)$. Consequently, by Chernoff bound, we have

$$\begin{aligned}
\Pr\left[d^{(k)}(v) \geqslant \varepsilon \cdot \Delta\right] &= \Pr\left[d^{(k)}(v) \geqslant \frac{1}{1-\varepsilon^2} \cdot \varepsilon \cdot \Delta(1-\varepsilon^2)\right] \\
&\leqslant \Pr\left[d^{(k)}(v) \geqslant (1+\varepsilon^2) \cdot \varepsilon \cdot \Delta(1-\varepsilon^2)\right] && 1+x \leqslant \frac{1}{1-x} \ \forall x<1 \\
&\leqslant \exp\left(\frac{-\varepsilon^4 \cdot \varepsilon \cdot \Delta(1+\varepsilon^2)}{3}\right) && \text{Lemma 2.4.10} \\
&\leqslant n^{-(\alpha+1)}. && \Delta \geqslant \frac{6(\alpha+1)\log n}{\varepsilon^5}
\end{aligned}$$

Taking union bound over all vertices, we find that $\Pr[T \leqslant \varepsilon \cdot m(1-\varepsilon^2)] \leqslant n^{-\alpha}$, which together with the above, implies that Property 1 holds with probability at least $1-2n^{-\alpha}$.

We next prove that Properties 2 and 3 hold w.h.p. Consider the following property for a generic $k \leqslant m$:

$$d^{(k)}(v) = \varepsilon \cdot d(v) \pm 2\varepsilon^3\Delta. \tag{9.6}$$

Let $B^{(k)}$ denote the bad event that some vertex v *fails* to satisfy Equation (9.6) for this k. We will show that the probability of any event $B^{(k)}$ to happen for $k \in [\varepsilon m(1-\varepsilon^2), m]$ is at most $2n^{-\alpha}$. Observe that Property 2 and Property 3 hold if events $B^{(T)}$ and $B^{(m')}$, resp., do *not* happen. Assuming Property 1, both T and m' fall in the range $[\varepsilon m(1-\varepsilon^2), m]$. It then follows that the 3 properties simultaneously hold with probability at least $1-4n^{-\alpha}$.

Equation (9.6) trivially holds for vertices v with $d(v) \leqslant \varepsilon^3\Delta$, for which $0 \leqslant d^{(k)}(v) \leqslant d(v) \leqslant \varepsilon^3\Delta$. On the other hand, for vertices v with $d(v) \geqslant \varepsilon^3\Delta$ and k as above, we have by Equation (9.5) that $\mathbb{E}[d^{(k)}(v)] = \varepsilon \cdot \Delta(1 \pm \varepsilon^2)$. Consequently, by Chernoff bound, we have that $\Pr\left[d^{(k)}(v) \geqslant \varepsilon \cdot d(v) + 2\varepsilon^3\Delta\right]$ is at most

$$\begin{aligned}
&\Pr\left[d^{(k)}(v) \geqslant (1+2\varepsilon^2) \cdot \varepsilon \cdot d(v)\right] && d(v) \leqslant \Delta \\
\leqslant &\Pr\left[d^{(k)}(v) \geqslant (1+\varepsilon^2/2) \cdot \varepsilon \cdot d(v)(1+\varepsilon^2)\right] && 1+2\varepsilon^2 \geqslant (1+\varepsilon^2)(1+\varepsilon^2/2) \\
\leqslant &\exp\left(\frac{-\varepsilon^4/4 \cdot \varepsilon \cdot d(v)(1+\varepsilon^2)}{3}\right) && \text{Lemma 2.4.10} \\
\leqslant &\exp\left(\frac{-\varepsilon^4/4 \cdot \varepsilon \cdot \varepsilon^3\Delta(1+\varepsilon^2)}{3}\right) && d(v) \geqslant \varepsilon^3\Delta \\
\leqslant &n^{-(\alpha+3)}. && \Delta \geqslant \frac{12(\alpha+3)\ln n}{\varepsilon^8}
\end{aligned}$$

Similarly, we have that $\Pr\left[d^{(k)}(v) \leqslant \varepsilon \cdot d(v) - 2\varepsilon^3\Delta\right]$ is at most

$$\begin{aligned}
&\Pr\left[d^{(k)}(v) \leqslant (1-2\varepsilon^2)\cdot\varepsilon\cdot d(v)\right] && d(v) \leqslant \Delta \\
\leqslant &\Pr\left[d^{(k)}(v) \geqslant (1-\varepsilon^2/2)\cdot\varepsilon\cdot d(v)(1-\varepsilon^2)\right] && 1-2\varepsilon^2 \leqslant (1-\varepsilon^2)(1-\varepsilon^2/2) \\
\leqslant &\exp\left(\frac{-\varepsilon^4/4\cdot\varepsilon\cdot d(v)(1-\varepsilon^2)}{3}\right) && \text{Lemma 2.4.10} \\
\leqslant &\exp\left(\frac{-\varepsilon^4/4\cdot\varepsilon\cdot\varepsilon^3\Delta(1-\varepsilon^2)}{3}\right) && d(v) \geqslant \varepsilon^3\Delta \\
\leqslant &n^{-(\alpha+3)}. && \Delta \geqslant \frac{24(\alpha+3)\ln n}{\varepsilon^8}
\end{aligned}$$

By union bound, we have that

$$\Pr[B^{(k)}] \leqslant \sum_v \Pr[d^{(k)}(v) \neq \varepsilon\cdot d(v) \pm 2\varepsilon^3\Delta] \leqslant 2n^{-(\alpha+2)}.$$

The claim follows by union bound over the values $k \in [\varepsilon m(1-\varepsilon^2), m]$. □

Finally, we show that adding Δ dummy vertices per vertex v in G forming a Δ-clique, and $(\Delta - (1/\varepsilon - 1)\cdot d^{(T)}(v))^+$ dummy edges from v to some of its dummy nodes, to the edges in the time range $(T, m' - T]$, yields a near-regular graph H.

Lemma 9.3.4. *The graph H satisfies $d_H(v) = \Delta(1 \pm 4\varepsilon^2)$ for all $v \in V(H)$, w.h.p.*

Proof. By Lemma 9.3.3 (Item 2), the number of dummy edges of v is $\Delta - (1/\varepsilon - 1)\cdot d^{(T)}(v)$, as this number is non-negative for every vertex $v \in V$ w.h.p., since

$$\Delta - (1/\varepsilon - 1)\cdot d^{(T)}(v) = \Delta - (1-\varepsilon)\cdot(d(v) \pm 2\varepsilon^2\Delta) \geqslant \Delta(1-(1-\varepsilon)-2\varepsilon^2) = \Delta(\varepsilon - 2\varepsilon^2) \geqslant 0.$$

Each dummy node v_i belongs to a single clique of size Δ, and possibly has another dummy edge to a single real node v, and therefore it has degree $d_H(v) \in \{\Delta - 1, \Delta\}$. As for real vertices v, again appealing to Lemma 9.3.3 (Item 2), combined with v having at most $\varepsilon^2\Delta$ edges in the range $(m', m]$ by Lemma 9.3.3 (Item 3), we find that the degree of v in H satisfies

$$\begin{aligned}
d_H(v) &= d(v) - d^{(T)}(v) + \Delta - (1/\varepsilon - 1)\cdot d^{(T)}(v) \pm 2\varepsilon^2\Delta \\
&= d(v) + \Delta - (d(v) \pm 2\varepsilon^2\Delta) \pm 2\varepsilon^2\Delta \\
&= \Delta(1 \pm 4\varepsilon^2).
\end{aligned}$$

□

9.6 Analysis of the Basic Algorithm

In this section, we analyze our *basic algorithm* (Algorithm 14) from Section 9.2. Before proceeding any further, the reader will find it useful to review Section 9.2. The analysis of Algorithm 14 boils down to proving that the uncolored subgraph after phase one, consisting of all the edges that either failed or were not sampled in the first place, has bounded degree. This is (re-)stated in Theorem 9.2.2 below.

Theorem 9.2.2. $\Delta(G_{t_\varepsilon} \cup G_F) = O\left(\varepsilon^{1/(3K)}\Delta\right)$ *w.h.p.*

The rest of this section (Section 9.6) is dedicated to the proof of Theorem 9.2.2, and it is organized as follows. In Section 9.6.1, we give a brief, high-level and *informal* overview of the proof of Theorem 9.2.2. We start the formal proof of Theorem 9.2.2 in Section 9.6.2, which defines some key random variables and events that will be extensively used in our analysis. In Section 9.6.3, we show how Theorem 9.2.2 follows from a sequence of lemmas. The remainder of Section 9.6 is devoted to the proofs of these individual lemmas from Section 9.6.3, starting with a review of some useful concentration inequalities we will need, in Section 11.3.

Throughout Section 9.6, we use the notation $a \pm b$ to denote the interval $[a-b, a+b]$. Thus, whenever we write $x = a \pm b$ in this section, it means that $x \in [a-b, a+b]$. Similarly, whenever we write $a \pm b = a' \pm b'$, it means that $[a-b, a+b] \subseteq [a'-b', a'+b']$.

9.6.1 An informal overview of the proof of Theorem 9.2.2

For all $i \in [t_\varepsilon - 1], v \in V$, let $N_i(v) := \{(u,v) \in E_i\}$ denote the edges of v that are present in $G_i = (V, E_i)$. The next lemma helps us bound the maximum degree in the subgraph G_{t_ε}.

Lemma 9.6.1. $|N_i(v)| \approx (1-\varepsilon)^{(i-1)} \cdot (1 \pm \varepsilon^2)\Delta$ *for all* $i \in [t_\varepsilon], v \in V$, *w.h.p.*

Proof. (Sketch) The statement clearly holds for $i = 1$. The proof follows from induction on i. Condition on all the random choices made by the algorithm during rounds $\{1, \ldots, i-1\}$. Fix any node $v \in V$, and suppose that $|N_i(v)| \approx (1-\varepsilon)^{(i-1)} \cdot (1 \pm \varepsilon^2)\Delta$ for some $i \in [t_\varepsilon - 2]$. Each edge $e \in N_i(v)$ belongs to $N_{i+1}(v)$ independently with probability $1 - \varepsilon$. From linearity of expectation, we derive that $\mathbb{E}[|N_{i+1}(v)|] = (1-\varepsilon) \cdot |N_i(v)|$. Now, a standard Hoeffding bound gives us: $|N_{i+1}(v)| = (1-\varepsilon) \cdot |N_i(v)| \pm \Theta(\sqrt{\Delta \ln n})$ w.h.p. From (9.1), (9.2) and the inductive hypothesis, we get: $\sqrt{\Delta \ln n} \ll \varepsilon^2 \cdot \Delta \ll \varepsilon \cdot |N_i(v)|$. This implies that $|N_{i+1}(v)| \approx (1-\varepsilon)^i \cdot |N_i(v)| \approx (1-\varepsilon)^i \cdot (1 \pm \varepsilon^2)\Delta$ w.h.p. □

The proof of Lemma 9.6.1 gives us a glimpse as to why we need a lower bound on ε: Whenever we take a concentration bound during the analysis, we will end up with an additive error term of the form $\sqrt{\Delta \ln n}$. We will like this additive error term to get subsumed within $\varepsilon^\beta \Delta$ for some large constant $\beta > 1$. Specifically, we will like to have $\varepsilon^\beta \Delta \gg \sqrt{\Delta \ln n}$, which implies that $\varepsilon \gg \left(\frac{\ln n}{\Delta}\right)^{1/(2\beta)}$. Henceforth, to convey the main ideas, we will often gloss over this issue in this section.

Corollary 9.6.2. $\Delta(G_{t_\varepsilon}) = O\left(\varepsilon^{1/(3K)}\Delta\right)$, *w.h.p.*

Proof. (Sketch) Fix any node $v \in V$. Since $t_\varepsilon \approx (\ln(1/\varepsilon))/(2K\varepsilon)$, Lemma 9.6.1 gives us: $|N_{t_\varepsilon}(v)| \approx (1-\varepsilon)^{(t_\varepsilon - 1)} \cdot \Delta \approx \exp(\varepsilon \cdot (t_\varepsilon - 1)) \cdot \Delta \approx \varepsilon^{1/(2K)}\Delta$ w.h.p. In other words, every node in G_{t_ε} has degree at most $\varepsilon\Delta$ w.h.p. This implies the corollary. $\square$

It now remains to upper bound the maximum degree of any node in the subgraph G_F. Before proceeding any further, we need to introduce the following notation. For all $i \in [t_\varepsilon - 1]$, $v \in V$ and $c \in [(1+\varepsilon^2)\Delta]$, let $N_{i,c}(v) = \{(u,v) \in N_i(v) : c \in P_i(u)\}$ denote the set of edges in $N_i(v)$ whose other endpoints have the color c in their palettes for round i. We refer to the quantity $|N_{i,c}(v)|$ as the *c-degree* of the node v for round i. The main challenge will be to bound the c-degrees of the nodes and the palette sizes for the edges in each round, as captured in the lemma below.

Lemma 9.6.3. *The following guarantees hold for all rounds $i \in [t_\varepsilon - 1]$, w.h.p.*

- *(a) $|P_i(e)| \approx (1-\varepsilon)^{2(i-1)} \cdot (1 \pm \varepsilon^2)\Delta$ for all edges $e \in E_i$.*
- *(b) $|N_{i,c}(v)| \approx (1-\varepsilon)^{2(i-1)} \cdot (1 \pm \Delta^2)\Delta$ for all nodes $v \in V$ and colors $c \in [\Delta]$.*

Lemma 9.6.3 is proved via an induction on i. We skip the rather technical proof of this lemma in this overview section. Instead, here we only explain how this lemma is used to give an upper bound on the maximum degree in G_F. The following lemma will be very useful towards this end.

Lemma 9.6.4. *With high probability, for all $i \in [t_\varepsilon - 1], v \in V$,*

$$|N_i(v) \cap F_i| \approx 2\varepsilon \cdot |N_i(v) \cap S_i| \pm \Theta(\sqrt{\Delta \ln n}).$$

Proof. (Sketch) Fix any round $i \in [t_\varepsilon - 1]$ and any node $v \in V$. Since each edge $e \in E_i$ is sampled in S_i independently with probability ε, standard concentration bounds imply that:

$$|N_{i,c}(x) \cap S_i| \approx \varepsilon \cdot |N_{i,c}(x)| \text{ for all colors } c \in [(1+\varepsilon^2)\Delta], \text{ w.h.p.} \tag{9.7}$$

Condition on all the random choices made by the algorithm during rounds $\{1, \ldots, i-1\}$, as well as the random choices which determine the set S_i. Suppose that these random choices we are conditioning upon satisfy (9.7) and Lemma 9.6.3 for round i (which anyway occur w.h.p.).

Fix any edge $(u,v) \in N_i(v) \cap S_i$. The probability that (u,v) belongs to the set F_i, conditioned on it having tentatively picked any color $c \in P_i(u,v)$ in round i, is given by:

$$\begin{aligned}
\Pr[(u,v) \in F_i \mid c(u,v) = c] &= 1 - \prod_{e' \in (N_{i,c}(u) \cap S_i)} \left(1 - \frac{1}{|P_i(e')|}\right) \cdot \prod_{e' \in (N_{i,c}(v) \cap S_i)} \left(1 - \frac{1}{|P_i(e')|}\right) \\
&\approx 1 - \left(1 - \frac{1}{(1-\varepsilon)^{2(i-1)} \cdot (1 \pm \varepsilon^2)\Delta}\right)^{\varepsilon \cdot |N_{i,c}(u)| + \varepsilon \cdot |N_{i,c}(v)|} \\
&\approx 1 - \left(1 - \frac{1}{(1-\varepsilon)^{2(i-1)} \cdot (1 \pm \varepsilon^2)\Delta}\right)^{2\varepsilon \cdot (1-\varepsilon)^{2(i-1)} \cdot (1 \pm \varepsilon^2)\Delta} \\
&\approx 1 - \exp(-2\varepsilon) \approx 2\varepsilon.
\end{aligned}$$

Since the above derivation holds for all colors $c \in P_i(u,v)$, we infer that $\Pr[(u,v) \in F_i] \approx 2\varepsilon$. Now, by linearity of expectation, we get: $\mathbb{E}[|N_i(v) \cap F_i|] = \sum_{e \in N_i(v) \cap S_i} \Pr[(u,v) \in F_i] \approx 2\varepsilon \cdot |N_i(v) \cap S_i|$. With some extra effort (see Section 9.6.9), we can show that the value of $|N_i(v) \cap F_i|$ is tightly concentrated around $\pm\Theta(\sqrt{\Delta \ln n})$ of its expectation. □

Corollary 9.6.5. $\Delta(G_F) = O(\varepsilon\Delta)$, *w.h.p.*

Proof. Consider any node $v \in V$. The degree of v in G_F is given by $\deg_v(G_F) := \sum_{i=1}^{t_\varepsilon - 1} |N_i(v) \cap F_i|$. Hence, w.h.p., Lemma 9.6.4 gives us the following bound on $\deg_v(G_F)$.

$$\begin{aligned}\deg_v(G_F) &\approx \sum_{i=1}^{t_\varepsilon-1}\left(2\varepsilon \cdot |N_i(v) \cap S_i| \pm \Theta(\sqrt{\Delta \ln n})\right) \approx 2\varepsilon \cdot \sum_{i=1}^{t_\varepsilon} |N_i(v) \cap S_i| \pm \Theta(t_\varepsilon \cdot \sqrt{\Delta \ln n}) \\ &\leqslant 2\varepsilon \cdot \deg_v(G) \pm \Theta(t_\varepsilon \cdot \sqrt{\Delta \ln n}) \leqslant 2\varepsilon \cdot (1+\varepsilon^2)\Delta + \Theta(t_\varepsilon \cdot \sqrt{\Delta \ln n}) = \Theta(\varepsilon\Delta).\end{aligned}$$

In the above derivation, the last step follows from (9.1) and (9.2). Since every node in G_F has degree at most $O(\varepsilon\Delta)$ w.h.p., we conclude that $\Delta(G_F) = O(\varepsilon\Delta)$ w.h.p. □

Theorem 9.2.2 follows from Corollary 9.6.2 and Corollary 9.6.5.

9.6.2 Key random variables and events

This section defines some random variables and events that will be extensively used in our analysis.

Random variables: We will need to deal with the following random variables for each $i \in [t_\varepsilon]$.

- $P_i(v) \subseteq [(1+\varepsilon^2)\Delta]$: A color $c \in [(1+\varepsilon^2)\Delta]$ belongs to the set $P_i(v)$ iff there is no edge $(u,v) \in S_j$ which picked the color c at some earlier round $j < i$. We refer to the set $P_i(v)$ as the *palette* of the node $v \in V$ for round i.
- $P_i(e) \subseteq [(1+\varepsilon^2)\Delta]$: A color $c \in [(1+\varepsilon^2)\Delta]$ belongs to the set $P_i(e)$ iff $c \in P_i(u) \cap P_i(v)$, where $e = (u,v) \in E$. We refer to the set $P_i(e)$ as the *palette* of the edge $e \in E$ for round i.
- $N_i(v)$: The set of neighboring edges of $v \in V$ in G_i, that is, $N_i(v) = \{(u,v) \in E : (u,v) \in E_i\}$.
- $N_{i,c}(v)$: This is the set of neighboring edges of the node $v \in V$ in G_i whose other endpoints have the color c in their palettes for round i, that is, $N_{i,c}(v) = \{(u,v) \in N_i(v) : c \in P_i(u)\}$.

Error parameters: While analyzing the basic algorithm, we need to keep track of the amount by which the random variables $|P_i(e)|$ and $|N_{i,c}(v)|$ can deviate from their expected values. The magnitude of these deviations will be captured by the error-parameters $\{\gamma_i\}_{i \in [t_\varepsilon]}$, where:

$$\gamma_1 = K\varepsilon^2. \tag{9.8}$$

$$\gamma_{i+1} = (1+K\varepsilon)\gamma_i + K\varepsilon^2 \text{ for all } i \in [t_\varepsilon - 1]. \tag{9.9}$$

Corollary 9.6.6. *We have $\gamma_i \leqslant \varepsilon^{1/2}$ for all $i \in [t_\varepsilon]$.*

Proof. From (9.8) and (9.9), we derive that:

$$\begin{aligned}
\gamma_{t_\varepsilon} =& K\varepsilon^2 \cdot \sum_{i=0}^{t_{\varepsilon-1}} (1+K\varepsilon)^i \\
\leqslant& (K\varepsilon^2) \cdot \frac{(1+K\varepsilon)^{t_\varepsilon}}{(K\varepsilon)} \\
\leqslant& \varepsilon \cdot \exp(K\varepsilon t_\varepsilon) \\
\leqslant& \varepsilon \cdot \exp((1/2) \cdot \ln(1/\varepsilon)) && \text{(by (9.2))} \\
=& \varepsilon^{1/2}.
\end{aligned}$$

The corollary now follows from the observation that $\gamma_i \leqslant \gamma_{t_\varepsilon}$ for all $i \in [t_\varepsilon]$. □

Random events: We will track the three events $\mathcal{E}_i, \mathcal{C}_i$ and $\mathcal{B}_i$, defined below, for each $i \in [t_\varepsilon - 1]$.

- Event $\mathcal{E}_i$ occurs if $|P_i(e)| = (1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta$ for all $e \in E_i$.
- Event $\mathcal{C}_i$ occurs if $|N_{i,c}(v)| = (1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta$ for all $v \in V$ and $c \in [(1+\varepsilon^2)\Delta]$.
- Event $\mathcal{B}_i$ occurs if the following conditions hold for all nodes $v \in V$ and colors $c \in [(1+\varepsilon^2)\Delta]$:

$$|S_i \cap N_i(v)| = (\varepsilon \pm \varepsilon^2) \cdot |N_i(v)|, \ \text{ and } \ |S_i \cap N_{i,c}(v)| = (\varepsilon \pm \varepsilon^2) \cdot |N_{i,c}(v)|.$$

Remark:: Since the degree of every node in G is $(1 \pm \varepsilon^2)\Delta$, we have $\Pr[\mathcal{E}_1] = \Pr[\mathcal{C}_1] = 1$.

Random bits used by our algorithm: While proving Theorem 9.2.2, we will often need to condition upon certain critical events. It will be easier to follow the proof if we view these conditionings via the prism of a classification of random bits used by the algorithm, as described below.

During any given round $i \in [t_\varepsilon - 1]$, there are two distinct tasks for which the algorithm uses randomness: (a) To determine the set of sampled edges S_i, and (b) to pick a color $c(e)$ for each sampled edge $e \in S_i$. We let $r_i^{\text{(edges)}}$ and $r_i^{\text{(colors)}}$ respectively denote the random bits used by the algorithm for task (a) and task (b). The random bits $r_i^{\text{(edges)}}$ and $r_i^{\text{(colors)}}$ are mutually independent of each other, and they are also independent of all the random bits used in the previous rounds $j < i$. We let $r_i = r_i^{\text{(edges)}} \cup r_i^{\text{(colors)}}$ denote all the random bits used by the algorithm in round i. Furthermore, we let $r_{<i} = \bigcup_{j=1}^{i-1} r_j$ denote the set of all random bits used by the algorithms in rounds $\{1, \ldots, i-1\}$. Note that the random bits $r_{<i}$ completely determine the occurrences of the following events: $\{\mathcal{E}_j, \mathcal{C}_j, \mathcal{B}_j\}_{j<i}$ and $\{\mathcal{E}_i, \mathcal{C}_i\}$. On the other hand, the occurrence of the event $\mathcal{B}_i$ is completely determined by the random bits $r_{<i} \cup r_i^{\text{(edges)}}$.

9.6.3 Proof of Theorem 9.2.2

The main challenge is to show that all the key events defined in Section 9.6.2 occur w.h.p. This is summarized in Corollary 9.6.10, which in turn follows from Lemma 9.6.7, Lemma 9.6.8 and Lemma 9.6.9. The proofs of these three crucial lemmas appear in Section 9.6.5, Section 9.6.6 and Section 9.6.8, respectively.

Lemma 9.6.7. *Consider any round $i \in [t_\varepsilon - 1]$, and fix any instantiation of the bits $r_{<i}$ which ensure the occurrence of the event $\mathcal{E}_i \cap \mathcal{C}_i$. Then we have:* $\Pr[\mathcal{B}_i \mid r_{<i}] \geqslant 1 - 1/n^{1500}$.

Lemma 9.6.8. *Consider any round $i \in [t_\varepsilon - 2]$, and fix any instantiation of the bits $r_{<i} \cup r_i^{(edges)}$ which ensure the occurrence of the event $\mathcal{E}_i \cap \mathcal{C}_i \cap \mathcal{B}_i$. Then we have:* $\Pr\left[\mathcal{E}_{i+1} \mid r_{<i} \cup r_i^{(edges)}\right] \geqslant 1 - 1/n^{1500}$.

Lemma 9.6.9. *Consider any round $i \in [t_\varepsilon - 2]$, and fix any instantiation of the bits $r_{<i} \cup r_i^{(edges)}$ which ensure the occurrence of the event $\mathcal{E}_i \cap \mathcal{C}_i \cap \mathcal{B}_i$. Then we have:* $\Pr\left[\mathcal{C}_{i+1} \mid r_{<i} \cup r_i^{(edges)}\right] \geqslant 1 - 1/n^{500}$.

Corollary 9.6.10. *We have* $\Pr\left[\bigcap_{i=1}^{t_\varepsilon - 1} (\mathcal{E}_i \cap \mathcal{C}_i \cap \mathcal{B}_i)\right] \geqslant 1 - 1/n^{400}$.

Proof. First, recall that $\Pr[\mathcal{E}_1] = \Pr[\mathcal{C}_1] = 1$, and hence Lemma 9.6.7 gives us:

$$\Pr[\mathcal{E}_1 \cap \mathcal{C}_1 \cap \mathcal{B}_1] \geqslant 1 - 1/n^{1500}. \tag{9.10}$$

Next, applying a union bound over Lemma 9.6.8 and Lemma 9.6.9, we get:

$$\Pr[\mathcal{E}_{i+1} \cap \mathcal{C}_{i+1} \mid \mathcal{E}_i \cap \mathcal{C}_i \cap \mathcal{B}_i] \geqslant 1 - 2/n^{500} \text{ for all rounds } i \in [t_\varepsilon - 2]. \tag{9.11}$$

Accordingly, from (9.11) and Lemma 9.6.7 we infer that:

$$\begin{aligned}
\Pr[\mathcal{E}_{i+1} \cap \mathcal{C}_{i+1} \cap \mathcal{B}_{i+1} \mid \mathcal{E}_i \cap \mathcal{C}_i \cap \mathcal{B}_i] &= \Pr[\mathcal{E}_{i+1} \cap \mathcal{C}_{i+1} \mid \mathcal{E}_i \cap \mathcal{C}_i \cap \mathcal{B}_i] \cdot \Pr[\mathcal{B}_{i+1} \mid \mathcal{E}_{i+1} \cap \mathcal{C}_{i+1}] \\
&\geqslant \left(1 - 2/n^{500}\right) \cdot \left(1 - 1/n^{1500}\right) \text{ for all } i \in [t_\varepsilon - 2]
\end{aligned} \tag{9.12}$$

Now, from (9.10) and (9.12) we derive that:

$$\begin{aligned}
\Pr\left[\bigcap_{i=1}^{t_\varepsilon - 1} (\mathcal{E}_i \cap \mathcal{C}_i \cap \mathcal{B}_i)\right] &= \Pr[\mathcal{E}_1 \cap \mathcal{C}_1 \cap \mathcal{B}_1] \cdot \prod_{i=1}^{t_\varepsilon - 2} \Pr[\mathcal{E}_{i+1} \cap \mathcal{C}_{i+1} \cap \mathcal{B}_{i+1} \mid \mathcal{E}_i \cap \mathcal{C}_i \cap \mathcal{B}_i] \\
&\geqslant \left(1 - 1/n^{1500}\right) \cdot \left(1 - 2/n^{500}\right)^{t_\varepsilon} \cdot \left(1 - 1/n^{1500}\right)^{t_\varepsilon} \\
&\geqslant \left(1 - 1/n^{1500}\right) \cdot \left(1 - 2t_\varepsilon/n^{500}\right) \cdot \left(1 - t_\varepsilon/n^{1500}\right) \\
&\geqslant 1 - 1/n^{1500} - 2t_\varepsilon/n^{500} - t_\varepsilon/n^{1500} \geqslant 1 - 1/n^{400}.
\end{aligned}$$

In the derivation above, the last inequality holds since $t_\varepsilon \leqslant n$ and $n \geqslant 2$. $\square$

In order to prove Theorem 9.2.2, we need to upper bound the maximum degree of any node in the subgraph $G_{t_\varepsilon} \cup G_F$. Accordingly, Corollary 9.6.11 upper bounds the maximum degree in the subgraph G_{t_ε}, whereas Corollary 9.6.13 (which follows from Lemma 9.6.12), upper bounds the maximum degree in the subgraph G_F. Section 9.6.9 contains the proof of Lemma 9.6.12.

Corollary 9.6.11. $\Delta(G_{t_\varepsilon}) = O\left(\varepsilon^{1/(3K)} \cdot \Delta\right)$ *with probability at least* $1 - 1/n^{400}$.

Proof. Define the event $\mathcal{B} := \bigcap_{i \in [t_\varepsilon - 1]} \mathcal{B}_i$. From Corollary 9.6.10, we infer that $\Pr[\mathcal{B}] \geqslant 1 - 1/n^{400}$. Henceforth, we condition on the event $\mathcal{B}$.

Consider any node $v \in V$. Conditioned on the event $\mathcal{B}$, we have $|S_i \cap N_i(v)| \geqslant (\varepsilon - \varepsilon^2) \cdot |N_i(v)|$ for each round $i \in [t_\varepsilon - 1]$. Since $|N_{i+1}(v)| = |N_i(v)| - |N_i(v) \cap S_i|$, we infer that:

$$|N_{i+1}(v)| \leqslant (1 - \varepsilon + \varepsilon^2) \cdot |N_i(v)| \text{ for all } i \in [t_\varepsilon - 1], \text{ conditioned on the event } \mathcal{B}. \tag{9.13}$$

Since $|N_1(v)| \leqslant (1 + \varepsilon^2)\Delta$, it is now easy to derive from (9.13) that:

$$\begin{aligned}
|N_{t_\varepsilon}(v)| &\leqslant (1 - \varepsilon + \varepsilon^2)^{t_\varepsilon} \cdot (1 + \varepsilon^2) \cdot \Delta \\
&\leqslant \exp(-\varepsilon(1-\varepsilon)t_\varepsilon) \cdot (1 + \varepsilon^2) \cdot \Delta \\
&\leqslant \exp\left(-\varepsilon(1-\varepsilon) \cdot \left(\frac{\ln(1/\varepsilon)}{2K\varepsilon} - 1\right)\right) \cdot (1 + \varepsilon^2) \cdot \Delta && \text{(by (9.1) and (9.2))} \\
&\leqslant \exp\left(-\frac{(1-\varepsilon)\ln(1/\varepsilon)}{2K} + 2\varepsilon\right) \cdot (1 + \varepsilon^2) \cdot \Delta && \text{(by (9.1))} \\
&\leqslant \varepsilon^{(1-\varepsilon)/(2K)} \cdot (1 + 4\varepsilon) \cdot (1 + \varepsilon^2) \cdot \Delta && \text{(by (9.1))} \\
&\leqslant O(\varepsilon^{1/(3K)} \cdot \Delta) \text{ for all } v \in V, \text{ conditioned on the event } \mathcal{B}. && \text{(by (9.1) and (9.2))}
\end{aligned}$$

The degree of a node $v \in V$ in G_{t_ε} exactly equals $|N_{t_\varepsilon}(v)|$. Hence, we conclude that $\Delta(G_{t_\varepsilon}) = O\left(\varepsilon^{1/(3K)}\Delta\right)$, conditioned on the event $\mathcal{B}$. The corollary follows since $\Pr[\mathcal{B}] \geqslant 1 - 1/n^{400}$. $\square$

Lemma 9.6.12. *Consider any round* $i \in [t_\varepsilon - 1]$, *and fix any instantiation of the bits* $r_{<i} \cup r_i^{(edges)}$ *which ensure the occurrence of the event* $\mathcal{E}_i \cap \mathcal{C}_i \cap \mathcal{B}_i$. *Then* $\Pr\left[\Delta(G_{F_i}) \leqslant 9\varepsilon^2\Delta \mid r_{<i} \cup r_i^{(edges)}\right] \geqslant 1 - 1/n^{300}$, *where* $G_{F_i} = (V, F_i)$ *denotes the subgraph of* G *consisting of the failed edges in round* i.

Corollary 9.6.13. $\Delta(G_F) = O(\varepsilon \ln(1/\varepsilon) \cdot \Delta)$ *with probability at least* $1 - 1/n^{200}$.

Proof. From Corollary 9.6.10 and Lemma 9.6.12, we infer that:

$$\begin{aligned}
\Pr\left[\Delta(G_{F_i}) = O(\varepsilon^2 \cdot \Delta)\right] &= \Pr\left[\Delta(G_{F_i}) = O(\varepsilon^2 \cdot \Delta) \mid \mathcal{E}_i \cap \mathcal{C}_i \cap \mathcal{B}_i\right] \cdot \Pr[\mathcal{E}_i \cap \mathcal{C}_i \cap \mathcal{B}_i] \\
&\geqslant \left(1 - 1/n^{300}\right) \cdot \left(1 - 1/n^{400}\right) \\
&\geqslant 1 - 1/n^{300} - 1/n^{400} \\
&\geqslant 1 - 1/n^{250} \text{ for each round } i \in [t_\varepsilon - 1]. \qquad (9.14)
\end{aligned}$$

As $\Delta(G_F) \leqslant \sum_{i=1}^{t_\varepsilon - 1} \Delta(G_{F_i})$ and $t_\varepsilon \leqslant n$, the corollary follows from (9.2) and a union bound over (9.14). □

Theorem 9.2.2 now follows by Corollary 9.6.11 and Corollary 9.6.13.

9.6.4 A couple of important technical claims

Here, we prove two technical claims that will be used multiple times in the subsequent sections.

Claim 9.6.14. *Fix any round $i \in [t_\varepsilon - 1]$ and condition on the event $\mathcal{E}_i \cap \mathcal{C}_i$. Then we have:*

$$\begin{aligned}
(\varepsilon^2/2) \cdot |N_{i,c}(v)| &\geqslant 50\sqrt{\Delta \ln n} \textit{ for all nodes } v \in V \textit{ and colors } c \in [(1+\varepsilon^2)\Delta]. \qquad (9.15) \\
(\varepsilon^2/2) \cdot |N_i(v)| &\geqslant 50\sqrt{\Delta \ln n} \textit{ for all nodes } v \in V. \qquad (9.16) \\
(\varepsilon^2/2) \cdot |P_i(e)| &\geqslant 50\sqrt{\Delta \ln n} \textit{ for all edges } e \in E_i. \qquad (9.17)
\end{aligned}$$

Proof. Fix any node $v \in V$ and any color $c \in [(1+\varepsilon^2)\Delta]$. Conditioned on the event $\mathcal{E}_i \cap \mathcal{C}_i$, we get:

$$\begin{aligned}
(\varepsilon^2/2) \cdot |N_{i,c}(v)| &\geqslant (\varepsilon^2/2) \cdot (1-\varepsilon)^{2(i-1)} \cdot (1-\gamma_i) \cdot \Delta \\
&\geqslant (\varepsilon^2/2) \cdot (1-\varepsilon)^{2(t_\varepsilon - 1)} \cdot (1-\varepsilon^{1/2}) \cdot \Delta && \text{(by Corollary 9.6.6)} \\
&\geqslant (\varepsilon^2/2) \cdot \exp(-4\varepsilon(t_\varepsilon - 1)) \cdot (\Delta/2) && \text{(by (9.1))} \\
&\geqslant (\varepsilon^2/2) \cdot \exp\left(-(2/K) \cdot \ln(1/\varepsilon)\right) \cdot (\Delta/2) && \text{(by (9.2))} \\
&= (\varepsilon^2/2) \cdot \varepsilon^{2/K} \cdot (\Delta/2) \\
&\geqslant \varepsilon^3 \cdot (\Delta/4) && \text{(by (9.1) and (9.2))} \\
&\geqslant 50\sqrt{\Delta \ln n} && \text{(by (9.1))}
\end{aligned}$$

Applying the same line of reasoning, one can derive that $(\varepsilon^2/2) \cdot |P_i(e)| \geqslant 50\sqrt{\Delta \ln n}$ for all $e \in E_i$. Finally, (9.16) follows from (9.15) and the observation that $N_i(v) \supseteq N_{i,c}(v)$. □

Claim 9.6.15. $\left(1 - \frac{1}{(1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i)\Delta}\right)^{(\varepsilon \pm \varepsilon^2) \cdot (1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i)\Delta} = 1 - \varepsilon \pm (4\varepsilon\gamma_i + \varepsilon^2).$

Proof. Let $M = \left(1 - \frac{1}{(1-\varepsilon)^{2(i-1)}\cdot(1\pm\gamma_i)\Delta}\right)^{(\varepsilon\pm\varepsilon^2)\cdot(1-\varepsilon)^{2(i-1)}\cdot(1\pm\gamma_i)\Delta}$. We first upper bound M as follows.

$$\begin{aligned}
M &\leqslant \left(1 - \frac{1}{(1-\varepsilon)^{2(i-1)}\cdot(1+\gamma_i)\Delta}\right)^{(\varepsilon-\varepsilon^2)\cdot(1-\varepsilon)^{2(i-1)}\cdot(1-\gamma_i)\Delta} \\
&\leqslant \exp\left(-(\varepsilon-\varepsilon^2)\cdot(1-\gamma_i)\cdot(1+\gamma_i)^{-1}\right) \\
&\leqslant \exp\left(-(\varepsilon-\varepsilon^2)\cdot(1-2\gamma_i)\right) && \text{(by (9.1) and Corollary 9.6.6)} \\
&\leqslant \exp\left(-(\varepsilon-3\varepsilon\gamma_i)\right) && \text{(by (9.1) and Corollary 9.6.6)} \\
&\leqslant 1-(\varepsilon-3\varepsilon\gamma_i)+(1/2)\cdot(\varepsilon-3\varepsilon\gamma_i)^2 \\
&\leqslant 1-\varepsilon+(4\varepsilon\gamma_i+\varepsilon^2). && \text{(by (9.1) and Corollary 9.6.6)}
\end{aligned}$$

Next, we lower bound M as follows.

$$\begin{aligned}
M &\geqslant \left(1 - \frac{1}{(1-\varepsilon)^{2(i-1)}\cdot(1-\gamma_i)\Delta}\right)^{(\varepsilon+\varepsilon^2)\cdot(1-\varepsilon)^{2(i-1)}\cdot(1+\gamma_i)\Delta} \\
&\geqslant \exp\left(-\frac{(\varepsilon+\varepsilon^2)(1+\gamma_i)}{(1-\gamma_i)}\right)\cdot\left(1-\frac{(\varepsilon+\varepsilon^2)(1+\gamma_i)}{(1-\varepsilon)^{2(i-1)}\cdot(1-\gamma_i)^2\Delta}\right) \\
&\geqslant \exp\left(-\frac{(\varepsilon+\varepsilon^2)(1+\gamma_i)}{(1-\gamma_i)}\right)\cdot\left(1-\frac{8}{(1-\varepsilon)^{2t_\varepsilon}\Delta}\right) && \text{(by (9.1) and Corollary 9.6.6)} \\
&\geqslant \exp\left(-(\varepsilon+2\varepsilon\gamma_i)\right)\cdot\left(1-\frac{8}{(1-\varepsilon)^{2t_\varepsilon}\Delta}\right) && \text{(by (9.1) and Corollary 9.6.6)} \\
&\geqslant \exp\left(-(\varepsilon+2\varepsilon\gamma_i)\right)\cdot\left(1-\frac{8\cdot\exp(4\varepsilon t_\varepsilon)}{\Delta}\right) && \text{(by (9.1))} \\
&\geqslant \exp\left(-(\varepsilon+2\varepsilon\gamma_i)\right)\cdot\left(1-\frac{8}{\Delta}\cdot\exp\left(4\varepsilon+\frac{2\cdot\ln(1/\varepsilon)}{K}\right)\right) && \text{(by (9.2))} \\
&\geqslant \exp\left(-(\varepsilon+2\varepsilon\gamma_i)\right)\cdot\left(1-\frac{8}{\Delta}\cdot(1+8\varepsilon)\cdot\frac{1}{\varepsilon^{2/K}}\right) && \text{(by (9.1))} \\
&\geqslant \exp\left(-(\varepsilon+2\varepsilon\gamma_i)\right)\cdot(1-\varepsilon^2) && \text{(by (9.1) and (9.2))} \\
&\geqslant (1-\varepsilon-2\varepsilon\gamma_i)\cdot(1-\varepsilon^2) \\
&\geqslant 1-\varepsilon-(4\varepsilon\gamma_i+\varepsilon^2). && \text{(by (9.1) and Corollary 9.6.6)}
\end{aligned}$$

The second inequality holds since $\left(1-\frac{\lambda}{x}\right)^x \geqslant \exp(-\lambda)\cdot\left(1-\frac{\lambda^2}{x}\right)$ for all $0<\lambda<x$. $\square$

9.6.5 Proof of Lemma 9.6.7

For any node $v \in V$ and any color $c \in [(1+\varepsilon^2)\Delta]$, let $\mathcal{B}_i(v,c)$ denote the event which occurs iff:

$$|S_i \cap N_i(v)| = (\varepsilon\pm\varepsilon^2)\cdot|N_i(v)| \text{ and } |S_i \cap N_{i,c}(v)| = (\varepsilon\pm\varepsilon^2)\cdot|N_{i,c}(v)|.$$

We first focus on bounding $\Pr[\mathcal{B}_i(v,c) \mid r_{<i}]$ for a given node $v \in V$ and color $c \in [(1+\varepsilon^2)\Delta]$. We start by observing that $N_{i,c}(v) \subseteq N_i(v)$. For each edge $e \in N_i(v)$, consider an indicator

random variable $X_e \in \{0,1\}$ that is set to one iff the edge e is sampled in round i. Thus, we have $|S_i \cap N_i(v)| = \sum_{e \in N_i(v)} X_e$ and $|S_i \cap N_{i,c}(v)| = \sum_{e \in N_{i,c}(v)} X_e$. We also have $\mathbb{E}[X_e \mid r_{<i}] = \Pr[X_e = 1 \mid r_{<i}] = \varepsilon$ for all edges $e \in N_i(v)$. Hence, applying linearity of expectation, we get:

$$\begin{aligned}\mathbb{E}[|S_i \cap N_i(v)| \mid r_{<i}] &= \sum_{e \in N_i(v)} \mathbb{E}[X_e \mid r_{<i}] = \varepsilon \cdot |N_i(v)|.\\ \mathbb{E}[|S_i \cap N_{i,c}(v)| \mid r_{<i}] &= \sum_{e \in N_{i,c}(v)} \mathbb{E}[X_e \mid r_{<i}] = \varepsilon \cdot |N_{i,c}(v)|.\end{aligned}$$

The random variables $\{X_e\}$ are mutually independent (even after conditioning on $r_{<i}$). Since $|N_{i,c}(v)| \leqslant |N_i(v)| \leqslant (1+\varepsilon^2)\Delta$ and each X_e is a $0/1$ random variable, Lemma 2.4.11 gives us:

$$\Pr\left[|S_i \cap N_i(v)| = \varepsilon \cdot |N_i(v)| \pm 50\sqrt{\Delta \ln n} \,\middle|\, r_{<i}\right] \geqslant 1 - 1/n^{2000}. \tag{9.18}$$

$$\Pr\left[|S_i \cap N_{i,c}(v)| = \varepsilon \cdot |N_{i,c}(v)| \pm 50\sqrt{\Delta \ln n} \,\middle|\, r_{<i}\right] \geqslant 1 - 1/n^{2000}. \tag{9.19}$$

From (9.1) and Claim 9.6.14, we infer that:

$$\varepsilon \cdot |N_i(v)| \pm 50\sqrt{\Delta \ln n} = (\varepsilon \pm \varepsilon^2) \cdot |N_i(v)|. \tag{9.20}$$

$$\varepsilon \cdot |N_{i,c}(v)| \pm 50\sqrt{\Delta \ln n} = (\varepsilon \pm \varepsilon^2) \cdot |N_{i,c}(v)|. \tag{9.21}$$

From (9.18) and (9.20), we get:

$$\Pr\left[|S_i \cap N_i(v)| = (\varepsilon \pm \varepsilon^2) \cdot |N_i(v)| \,\middle|\, r_{<i}\right] \geqslant 1 - 1/n^{2000}. \tag{9.22}$$

Similarly, from (9.19) and (9.21), we get:

$$\Pr\left[|S_i \cap N_{i,c}(v)| = (\varepsilon \pm \varepsilon^2) \cdot |N_{i,c}(v)| \,\middle|\, r_{<i}\right] \geqslant 1 - 1/n^{2000}. \tag{9.23}$$

Applying a union bound over (9.22) and (9.23), we get: $\Pr[\mathcal{B}_i(v,c) \mid r_{<i}] \geqslant 1 - 1/n^{1900}$. Since $\mathcal{B}_i = \bigcap_{v,c} \mathcal{B}_i(v,c)$, the lemma follows from one last union bound over all $v \in V$ and $c \in [(1+\varepsilon^2)\Delta]$.

9.6.6 Proof of Lemma 9.6.8

Throughout this section, we fix the bits $r_{<i} \cup r_i^{(\text{edges})}$ which ensure the occurrence of the event $\mathcal{E}_i \cap \mathcal{C}_i \cap \mathcal{B}_i$. To ease notations, henceforth we refrain from repeatedly stating that we are conditioning on the bits $r_{<i} \cup r_i^{(\text{edges})}$. However, the reader should keep in mind that we are relying upon this conditioning for the rest of Section 9.6.6.

We first bound the expected value of $|P_{i+1}(e)|$ for any given edge $e \in E_{i+1}$. (Note that the subset of edges E_{i+1} is completely determined by the bits $r_{<i} \cup r_i^{(\text{edges})}$.) Next, we show that w.h.p. the value of $|P_{i+1}(e)|$ does not deviate too far away from its expectation. Finally, we take a union bound over all the edges $e \in E_{i+1}$ to argue that the event $\mathcal{E}_{i+1}$ occurs w.h.p.

Calculating the expected value of $|P_{i+1}(e)|$ for a given edge $e = (u,v) \in E_{i+1}$:

Since $e \in E_{i+1}$, we have $e \notin S_i$. In particular, this implies that the sets $(N_{i,c}(u) \cap S_i)$ and $(N_{i,c}(v) \cap S_i)$ are mutually disjoint. This observation will be useful in subsequent derivations.

Consider any color $c \in P_i(e)$. For any endpoint $x \in \{u, v\}$ of the edge e, let $\Gamma_{x,c}$ be the event that at least one edge $e' \in (N_{i,c}(x) \cap S_i)$ picks the color c in round i. Note that $\Pr[c \notin P_{i+1}(e)] = \Pr[\Gamma_{u,c} \cup \Gamma_{v,c}]$. Since the sets $(N_{i,c}(u) \cap S_i)$ and $(N_{i,c}(v) \cap S_i)$ are mutually disjoint, the events $\Gamma_{u,c}$ and $\Gamma_{v,c}$ are mutually independent. Hence, from the inclusion-exclusion principle we infer that:

$$\Pr[c \notin P_{i+1}(e)] = \Pr[\Gamma_{u,c}] + \Pr[\Gamma_{v,c}] - \Pr[\Gamma_{u,c}] \cdot \Pr[\Gamma_{v,c}]. \tag{9.24}$$

We now focus on estimating the value of $\Pr[\Gamma_{x,c}]$ for a given node $x \in \{u, v\}$. Recall that the bits $r_{<i} \cup r_i^{(\text{edges})}$ we condition upon ensure the occurrence of the event $\mathcal{E}_i \cap \mathcal{C}_i \cap \mathcal{B}_i$. Hence, we have:

$$|P_i(e')| = (1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta \text{ for all edges } e' \in E_i. \tag{9.25}$$
$$|N_{i,c}(x)| = (1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta. \tag{9.26}$$
$$|N_{i,c}(x) \cap S_i| = (\varepsilon \pm \varepsilon^2) \cdot (1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta. \tag{9.27}$$

Since the event $\Gamma_{x,c}$ occurs iff some edge $e' \in N_{i,c}(x) \cap S_i$ picks color c in round i, we infer that:

$$\begin{aligned}
\Pr[\Gamma_{x,c}] &= 1 - \prod_{e' \in N_{i,c}(x) \cap S_i} \left(1 - \frac{1}{|P_i(e')|}\right) \\
&= 1 - \left(1 - \frac{1}{(1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta}\right)^{(\varepsilon \pm \varepsilon^2) \cdot (1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta} \\
&= \varepsilon \pm (4\varepsilon\gamma_i + \varepsilon^2).
\end{aligned} \tag{9.28}$$

In the derivation above, the second step follows from (9.25) and (9.27), whereas the last step follows from Claim 9.6.15. From (9.24) and (9.28), we next infer that:

$$\Pr[c \notin P_{i+1}(e)] = 2\left(\varepsilon \pm (4\varepsilon\gamma_i + \varepsilon^2)\right) - \left(\varepsilon \pm (4\varepsilon\gamma_i + \varepsilon^2)\right)^2 \text{ for every color } c \in P_i(e).$$

Equivalently, for every color $c \in P_i(e)$, we have:

$$\Pr[c \in P_{i+1}(e)] = 1 - \Pr[c \notin P_{i+1}(e)] = \left(1 - \varepsilon \pm (4\varepsilon\gamma_i + \varepsilon^2)\right)^2.$$

Applying linearity of expectation, we now get:

$$\mathbb{E}[|P_{i+1}(e)|] = \sum_{c \in P_i(e)} \Pr[c \in P_{i+1}(e)] = \left(1 - \varepsilon \pm (4\varepsilon\gamma_i + \varepsilon^2)\right)^2 \cdot |P_i(e)|. \tag{9.29}$$

Deriving a concentration bound on $|P_{i+1}(e)|$ for a given edge $e = (u, v) \in E_{i+1}$:

For each color $c \in P_i(e)$, let $X_c \in \{0, 1\}$ be an indicator random variable that is set to one iff $c \in P_{i+1}(e)$. Clearly, we have: $|P_{i+1}(e)| = \sum_{c \in P_i(e)} X_c$. We will now show that the random variables $\{X_c\}_{c \in P_i(e)}$ are negatively associated, and then apply Hoeffding bound.

Claim 9.6.16. *The random variables $\{X_c\}_{c \in P_i(e)}$ are negatively associated.*

Proof. For each color $c \in P_i(e)$ and each edge $e' \in (N_i(u) \cap S_i) \cup (N_i(v)) \cap S_i)$, define an indicator random variable $X_{c,e'} \in \{0,1\}$ that is set to one iff the edge e' picks color c in round i. Since each edge picks at most one color in round i, Proposition 2.4.2 implies that for each edge $e' \in (N_i(u) \cap S_i) \cup (N_i(v) \cap S_i)$, the random variables $\{X_{c,e'}\}_c$ are negatively associated. Next, note that the color picked by any edge $e' \in S_i$ in round i is independent of the color picked by a different edge $e'' \in S_i \setminus \{e'\}$ in round i. Hence, part (1) of Proposition 2.4.4 implies that the random variables $\{X_{c,e'}\}_{c,e'}$ are also negatively associated. Finally, note that $X_c = 1 - \max_{e' \in (N_i(u) \cap S_i) \cup (N_i(v) \cap S_i)} \{X_{c,e'}\}$ for all colors $c \in P_i(e)$. Accordingly, part (2) of Proposition 2.4.4 implies that the random variables $\{X_c\}_{c \in P_i(e)}$ are negatively associated. This concludes the proof of the claim. □

Claim 9.6.17. *We have:* $\Pr\left[|P_{i+1}(e)| = \mathbb{E}\left[|P_{i+1}(e)|\right] \pm 50\sqrt{\Delta \ln n}\right] \geqslant 1 - 1/n^{2000}$.

Proof. Note that $|P_{i+1}(e)| = \sum_{c \in P_i(e)} X_c$, where $|P_i(e)| \leqslant (1+\varepsilon^2)\Delta$ and each X_c is a $0/1$ random variable. Since the random variables $\{X_c\}$ are negatively associated according to Claim 9.6.16, from Lemma 2.4.11 we now infer that $\Pr\left[|P_{i+1}(e)| = \mathbb{E}\left[|P_{i+1}(e)|\right] \pm 50\sqrt{\Delta \ln n}\right] \geqslant 1 - 1/n^{2000}$. □

Corollary 9.6.18. *We have:* $\Pr\left[|P_{i+1}(e)| = (1-\varepsilon)^{2i} \cdot (1 \pm \gamma_{i+1}) \cdot \Delta\right] \geqslant 1 - 1/n^{2000}$.

Proof. Consider any $M = \mathbb{E}\left[|P_{i+1}(e)|\right] \pm 50\sqrt{\Delta \ln n}$. Observe that:

$$\begin{aligned}
M &= \left(1 - \varepsilon \pm (4\varepsilon\gamma_i + \varepsilon^2)\right)^2 \cdot |P_i(e)| \pm 50\sqrt{\Delta \ln n} && \text{(by (9.29))}\\
&= \left(\left(1 - \varepsilon \pm (4\varepsilon\gamma_i + \varepsilon^2)\right)^2 \pm \varepsilon^2\right) \cdot |P_i(e)| && \text{(by Claim 9.6.14)}\\
&= \left((1-\varepsilon)^2 + (4\varepsilon\gamma_i + \varepsilon^2)^2 \pm 2(4\varepsilon\gamma_i + \varepsilon^2) \pm \varepsilon^2\right) \cdot |P_i(e)| \\
&= \left((1-\varepsilon)^2 \pm (12\varepsilon\gamma_i + 12\varepsilon^2)\right) \cdot |P_i(e)| && \text{(by (9.1) and Corollary 9.6.6)}\\
&= (1-\varepsilon)^2 \cdot \left(1 \pm (24\varepsilon\gamma_i + 24\varepsilon^2)\right) \cdot (1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta && \text{(by (9.1) and (9.25))}\\
&= (1-\varepsilon)^{2i} \cdot \left(1 \pm \left((1 + 48\varepsilon)\gamma_i + 48\varepsilon^2\right)\right) \cdot \Delta && \text{(by (9.1) and Corollary 9.6.6)}\\
&= (1-\varepsilon)^{2i} \cdot (1 \pm \gamma_{i+1}) \cdot \Delta. && \text{(by (9.2) and (9.9))}
\end{aligned}$$

To summarize, we have derived that if $|P_{i+1}(e)| = \mathbb{E}\left[|P_{i+1}(e)|\right] \pm 50\sqrt{\Delta \ln n}$, then it must be the case that $|P_{i+1}(e)| = (1-\varepsilon)^{2i} \cdot (1 \pm \gamma_{i+1}) \cdot \Delta$. The corollary now follows from Claim 9.6.17. □

Wrapping up the proof of Lemma 9.6.8:

Lemma 9.6.8 follows from Corollary 9.6.18 and a union bound over all the edges $e \in E_{i+1}$.

Before proceeding to prove the remaining lemmas needed to complete the proof of Theorem 9.2.2, we describe a few concentration inequalities for Lipschitz functions of independent variables, which we will use for the proofs of these remaining lemmas, Lemma 9.6.9 and Lemma 9.6.12.

9.6.7 Useful concentration inequalities

In the following section we present our analysis of Algorithm 14. As mentioned previously, most of this will rely on Chernoff-Hoeffding bounds for NA variables. Here we describe some additional concentration inequalities we will rely on, based on the *method of bounded differences*, described below.

Definition 9.6.19. *[85] Let $A_1, \ldots, A_n$ be sets and $f : A_1 \times \cdots \times A_n \to \mathbf{R}$ be a real-valued function. The function f satisfies the* Lipschitz property *with constants $\{d_i\}, i \in [n]$, if $|f(\mathbf{a}) - f(\mathbf{a}')| \leqslant d_i$ whenever $\mathbf{a}$ and $\mathbf{a}'$ differ only in the i^{th} co-ordinate, for all $i \in [n]$.*

Lemma 9.6.20. *[85] Let $f(X_1, \ldots, X_n)$ be a function of n independent random variables $X_1, \ldots, X_n$ satisfying the Lipschitz property with $\{d_i \mid i \in [n]\}$. Then, for all $t > 0$,*

$$\begin{aligned}\Pr[f \geqslant \mathbb{E}[f] + t] &\leqslant \exp\left(-\frac{2t^2}{\sum_{i=1}^n d_i^2}\right), \\ \Pr[f \leqslant \mathbb{E}[f] - t] &\leqslant \exp\left(-\frac{2t^2}{\sum_{i=1}^n d_i^2}\right).\end{aligned}$$

A more refined bound will prove useful when considering functions of bounded variance.

Lemma 9.6.21. *Let $f(X_1, \ldots, X_n)$ be a function of n independent $0/1$ random variables $X_1, \ldots, X_n$ that satisfy the Lipschitz property with $\{d_i\}, i \in [n]$. For each $i \in [n]$, let $X_{-i} \in \{0,1\}^{n-1}$ denote the values taken by every other random variable $\{X_j\}, j \in [n] \setminus \{i\}$. Furthermore, suppose that $\mathrm{Var}[f \mid X_{-i}] \leqslant \lambda_i$ for all $i \in [n]$ and all $X_{-i} \in \{0,1\}^{n-1}$. Let $\lambda := \sum_{i=1}^n \lambda_i$, and $d := \max_{i \in [n]}\{d_i\}$. Then, for all $t > 0$,*

$$\begin{aligned}\Pr[f \geqslant \mathbb{E}[f] + t] &\leqslant \exp\left(-\frac{t^2}{2\lambda + (2/3) \cdot td}\right), \\ \Pr[f \leqslant \mathbb{E}[f] - t] &\leqslant \exp\left(-\frac{t^2}{2\lambda + (2/3) \cdot td}\right).\end{aligned}$$

Proof. The lemma follows from the method of bounded variances, as explained in Chapter 8.1 of [85] (in particular, the lemma follows from equation (8.5) in this chapter). □

9.6.8 Proof of Lemma 9.6.9

Throughout this section, we fix the bits $r_{<i} \cup r_i^{\text{(edges)}}$ which ensure the occurrence of the event $\mathcal{E}_i \cap \mathcal{C}_i \cap \mathcal{B}_i$. To ease notations, henceforth we refrain from repeatedly stating that we are conditioning on the bits $r_{<i} \cup r_i^{\text{(edges)}}$. However, the reader should keep in mind that we are relying upon this conditioning for the rest of Section 9.6.8.

We first bound the expected value of $|N_{i+1,c}(v)|$ for a given $(c, v) \in [(1 + \varepsilon^2)\Delta] \times V$. Next, we show that w.h.p. the value of $|N_{i+1,c}(v)|$ does not deviate too far away from its expectation. Finally, we take a union bound over all $(c, v) \in [(1 + \varepsilon^2)\Delta] \times V$ to argue that $\mathcal{C}_{i+1}$ occurs w.h.p.

Calculating the expected value of $|N_{i+1,c}(v)|$ for a given $(c, v) \in [(1 + \varepsilon^2)\Delta] \times V$:
First, note that $N_{i+1,c}(v) \subseteq N_{i,c}(v) \setminus S_i$. Consider any edge $e' = (u, v) \in N_{i,c}(v) \setminus S_i$, and let $\Gamma_{e'}$ be the event that the edge e' belongs to the set $N_{i+1,c}(v)$. Our immediate goal is to calculate the value of $\Pr[\Gamma_{e'}]$. Towards this end, we first recall that the bits $r_{<i} \cup r_i^{\text{(edges)}}$ we are conditioning upon ensure the occurrence of the event $\mathcal{E}_i \cap \mathcal{C}_i \cap \mathcal{B}_i$. Hence, we have:

$$
\begin{aligned}
|P_i(e)| &= (1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta \text{ for all edges } e \in E_i. && (9.30)\\
|N_{i,c}(x)| &= (1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta \text{ for all nodes } x \in V. && (9.31)\\
|N_{i,c}(x) \cap S_i| &= (\varepsilon \pm \varepsilon^2) \cdot (1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta \text{ for all nodes } x \in V. && (9.32)
\end{aligned}
$$

The event $\Gamma_{e'}$ occurs iff no edge $(u, w) \in N_{i,c}(u) \cap S_i$ picks the color c in round i. Hence, from (9.30), (9.32) and Claim 9.6.15, we now derive that:

$$
\begin{aligned}
\Pr[\Gamma_{e'}] &= \prod_{w \in N_{i,c}(u) \cap S_i} \left(1 - \frac{1}{|P_i(u, w)|}\right)\\
&= \left(1 - \frac{1}{(1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta}\right)^{(\varepsilon \pm \varepsilon^2) \cdot (1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta}\\
&= 1 - \varepsilon \pm (4\varepsilon\gamma_i + \varepsilon^2) \qquad (9.33)
\end{aligned}
$$

Since we have already conditioned on the event $\mathcal{B}_i$, we get:

$$
|N_{i,c}(v) \setminus S_i| = |N_{i,c}(v)| - |N_{i,c}(v) \cap S_i| = (1 - \varepsilon \pm \varepsilon^2) \cdot |N_{i,c}(v)|. \qquad (9.34)
$$

From (9.33) and (9.34), together with linearity of expectation, we now derive that:

$$
\begin{aligned}
\mathbb{E}\left[|N_{i+1,c}(v)|\right] &= \sum_{e' \in N_{i,c}(v) \setminus S_i} \Pr[\Gamma_{e'}]\\
&= (1 - \varepsilon \pm (4\varepsilon\gamma_i + \varepsilon^2)) \cdot |N_{i,c}(v) \setminus S_i|\\
&= (1 - \varepsilon \pm (4\varepsilon\gamma_i + \varepsilon^2)) \cdot (1 - \varepsilon \pm \varepsilon^2) \cdot |N_{i,c}(v)|\\
&= \left(1 - \varepsilon \pm (4\varepsilon\gamma_i + \varepsilon^2)\right)^2 \cdot |N_{i,c}(v)|. \qquad (9.35)
\end{aligned}
$$

Deriving a concentration bound on $|N_{i+1,c}(v)|$ for a given $(c, v) \in [(1 \pm \varepsilon^2)\Delta] \times V$:
We first identify the sampled edges in round i that are responsible for determining which edges from $N_{i,c}(v) \setminus S_i$ will end up being included in $N_{i+1,c}(v)$. Towards this end, we define $T_i(v) :=$

$\bigcup_{u \in V : (u,v) \in N_{i,c}(v) \setminus S_i} N_{i,c}(u) \cap S_i$. Observe that if an edge $(u,u') \in T_i(v)$, where $(u,v) \in N_{i,c}(v) \setminus S_i$, picks the color c in round i, then $(u,v) \notin N_{i+1,c}(v)$. Conversely, if an edge $(u,v) \in N_{i,c}(v) \setminus S_i$ ends up not being part of $N_{i+1,c}(v)$, then some edge $(u,u') \in T_i(v)$ must pick the color c in round i. For each edge $e \in T_i(v)$, define an indicator random variable $X_e \in \{0,1\}$ that is set to one iff the edge e picks color c in round i. Clearly, the random variables $\{X_e\}$ are mutually independent. Since $|N_{i+1,c}(v)|$ is completely determined by the random variables $\{X_e\}, e \in T_i(v)$, we write $|N_{i+1,c}(v)| := f(X)$, where $X \in \{0,1\}^{|T_i(v)|}$ follows the joint distribution of the random variables $\{X_e\}$. We now prove a concentration bound on $f(X)$.

Claim 9.6.22. *The function f satisfies the Lipschitz property (Definition 9.6.19) with $d_e = 2$, $e \in T_i(v)$. Furthermore, for each edge $e \in T_i(v)$, let $X_{-\{e\}} \in \{0,1\}^{|T_i(v)|-1}$ denote the values of all the remaining variables $\{X_{e'}\}_{e' \in T_i(v) \setminus \{e\}}$. Then for all $e \in T_i(v)$ and $X_{-\{e\}} \in \{0,1\}^{|T_i(v)|-1}$, we have:*

$$var\left[f(X) \,\middle|\, X_{-\{e\}}\right] \leqslant \lambda_e, \text{ where } \lambda_e := \frac{8}{(1-\varepsilon)^{2(i-1)}\Delta}.$$

Proof. Throughout the proof, we fix any edge $e = (u,u') \in T_i(v)$ and the color picked by every other edge $e' \in T_i(v) \setminus \{e\}$ in round i, which determine the value of $X_{-\{e\}}$. Let $Z_e = \{(v,w) \in N_{i,c}(v) \setminus S_i : w \in \{u,u'\}\}$ denote the set of edges in $N_{i,c}(v) \setminus S_i$ that are adjacent to the edge e.

When we are trying to figure out which edges from $N_{i+1,c}(v) \setminus S_i$ will end up being part of $N_{i+1,c}(v)$, observe that the color picked by e can change the *fate* of only the edges in Z_e. Indeed, if the edge e picks the color c, then the edges in Z_e will *not* be included in $N_{i+1,c}(v)$. In contrast, if the edge e picks some color $c' \in P_i(e) \setminus \{c\}$, then the edges in Z_e can potentially be included in $N_{i+1,c}(v)$. (In this event, whether or not an edge in Z_e is actually included in $N_{i+1,c}(v)$ will depend on $X_{-\{e\}}$.) The fate of every other edge $e'' \in (N_{i,c}(v) \setminus S_i) \setminus Z_e$ is completely determined by $X_{-\{e\}}$.

Since $|Z_e| \leqslant 2$, the function f satisfies the Lipschitz property with $d_e = 2$, $e \in T_i(v)$. Furthermore, since the edge e picks the color $c \in P_i(e)$ with probability $1/|P_i(e)|$, we conclude that:

$$\begin{aligned}
\text{Var}\left(f(X) \,\middle|\, X_{-\{e\}}\right) &\leqslant \frac{2^2}{|P_i(e)|} \\
&= \frac{4}{(1-\varepsilon)^{2(i-1)}(1 \pm \gamma_i)\Delta} && \text{(by (9.30))} \\
&\leqslant \frac{8}{(1-\varepsilon)^{2(i-1)}\Delta}. && \text{(by (9.1) and Corollary 9.6.6)}
\end{aligned}$$

□

Claim 9.6.23. *We have:* $\Pr\left[|N_{i+1,c}(v)| = \mathbb{E}\left[|N_{i+1,c}(v)|\right] \pm 50\sqrt{\Delta \ln n}\right] \geqslant 1 - 1/n^{600}$.

Proof. We derive an upper bound on the size of the set $T_i(v)$.

$$\begin{aligned}
|T_i(v)| \leqslant & \sum_{u \in V:(u,v)\in N_{i,c}(v)\setminus S_i} |N_{i,c}(u) \cap S_i| \\
=& |N_{i,c}(v) \setminus S_i| \cdot (\varepsilon \pm \varepsilon^2) \cdot (1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta && \text{(by (9.32))} \\
\leqslant & |N_{i,c}(v) \setminus S_i| \cdot 4\varepsilon \cdot (1-\varepsilon)^{2(i-1)} \cdot \Delta && \text{(by (9.1))} \\
\leqslant & 8\varepsilon \cdot (1-\varepsilon)^{2(i-1)} \cdot \Delta^2. && (9.36)
\end{aligned}$$

From (9.1), (9.36) and Claim 9.6.22, we infer that:

$$\sum_{e \in T_i(v)} \lambda_e \leqslant 64\varepsilon \cdot \Delta \leqslant \Delta. \tag{9.37}$$

Recall that $|N_{i+1,c}(v)| = f(X)$, where X is drawn from the joint distribution of mutually independent random variables $\{X_e\}_{e \in T_i(v)}$. Hence, from (9.37), Claim 9.6.22 and Lemma 9.6.21, we get:

$$\begin{aligned}
\Pr\left[|N_{i+1,c}(v)| = \mathbb{E}\left[|N_{i,c}(v)|\right] \pm 50\sqrt{\Delta \ln n}\right] &\geqslant 1 - 2 \cdot \exp\left(-\frac{2500 \cdot \Delta \log n}{2\Delta + (2/3) \cdot 100 \cdot \sqrt{\Delta \log n}}\right) \\
&\geqslant 1 - 2 \cdot \exp\left(-\frac{2500 \cdot \Delta \log n}{4\Delta}\right) \quad \text{(by (9.1))} \\
&\geqslant 1 - 1/n^{600}.
\end{aligned}$$

This concludes the proof of the claim. □

Corollary 9.6.24. *We have:* $\Pr\left[|N_{i+1,c}(v)| = (1-\varepsilon)^{2i} \cdot (1 \pm \gamma_{i+1}) \cdot \Delta\right] \geqslant 1 - 1/n^{600}$.

Proof. Consider any $M = \mathbb{E}\left[|N_{i+1,c}(v)|\right] \pm 50\sqrt{\Delta \ln n}$. Observe that:

$$\begin{aligned}
M =& \left(1 - \varepsilon \pm (4\varepsilon\gamma_i + \varepsilon^2)\right)^2 \cdot |N_{i,c}(v)| \pm 50\sqrt{\Delta \ln n} && \text{(by (9.35))} \\
=& \left(\left(1 - \varepsilon \pm (4\varepsilon\gamma_i + \varepsilon^2)\right)^2 \pm \varepsilon^2\right) \cdot |N_{i,c}(v)| && \text{(by Claim 9.6.14)} \\
=& \left((1-\varepsilon)^2 + (4\varepsilon\gamma_i + \varepsilon^2)^2 \pm 2(4\varepsilon\gamma_i + \varepsilon^2) \pm \varepsilon^2\right) \cdot |N_{i,c}(v)| \\
=& \left((1-\varepsilon)^2 \pm (12\varepsilon\gamma_i + 12\varepsilon^2)\right) \cdot |N_{i,c}(v)| && \text{(by (9.1) and Corollary 9.6.6)} \\
=& (1-\varepsilon)^2 \cdot \left(1 \pm (24\varepsilon\gamma_i + 24\varepsilon^2)\right) \cdot (1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta && \text{(by (9.1) and (9.31))} \\
=& (1-\varepsilon)^{2i} \cdot \left(1 \pm \left((1 + 48\varepsilon)\gamma_i + 48\varepsilon^2\right)\right) \cdot \Delta && \text{(by (9.1) and Corollary 9.6.6)} \\
=& (1-\varepsilon)^{2i} \cdot (1 \pm \gamma_{i+1}) \cdot \Delta. && \text{(by (9.2) and (9.9))}
\end{aligned}$$

To summarize, we have derived that if $|N_{i+1,c}(v)| = \mathbb{E}\left[|N_{i+1,c}(v)|\right] \pm 50\sqrt{\Delta \ln n}$, then it must be the case that $|N_{i+1,c}(v)| = (1-\varepsilon)^{2i} \cdot (1 \pm \gamma_{i+1}) \cdot \Delta$. The corollary now follows from Claim 9.6.23. □

Wrapping up the proof of Lemma 9.6.9:

Lemma 9.6.9 follows from Corollary 9.6.24 and a union bound over all pairs $(c, v) \in [(1 + \varepsilon^2)\Delta] \times V$.

9.6.9 Proof of Lemma 9.6.12

Recall the discussion on the random bits $r_{<i}, r_i^{\text{(edges)}}$ and $r_i^{\text{(colors)}}$ from Section 9.6.2. We will prove the lemma stated below. Lemma 9.6.12 follows from Lemma 9.6.25 and a union bound over all $v \in V$.

Lemma 9.6.25. *Fix any instantiation of the bits $r_{<i} \cup r_i^{(edges)}$ which ensure the occurrence of the event $\mathcal{E}_i \cap \mathcal{C}_i \cap \mathcal{B}_i$. Fix any node $v \in V$, and let $F_i(v) = N_i(v) \cap F_i$ denote the set of failed edges in round i that are incident on v. Then we have:* $\Pr\left[|F_i(v)| \leqslant 9\varepsilon^2\Delta \,\middle|\, r_{<i} \cup r_i^{(edges)}\right] \geqslant 1 - 1/n^{305}$.

The rest of Section 9.6.9 is devoted to the proof of Lemma 9.6.25. We fix the bits $r_{<i} \cup r_i^{\text{(edges)}}$ which ensure the occurrence of the event $\mathcal{E}_i \cap \mathcal{C}_i \cap \mathcal{B}_i$. To ease notations, henceforth we refrain from repeatedly stating that we are conditioning on the bits $r_{<i} \cup r_i^{\text{(edges)}}$. However, the reader should keep in mind that we are relying upon this conditioning for the rest of Section 9.6.9.

A classification of failed edges:

Let $F_i^{(1)}(v) = \{(u, v) \in F_i(v) : c(v', v) = c(u, v) \text{ for some } (v', v) \in N_i(v) \cap S_i\}$ denote the set of edges (u, v) that fails in round i because of the following reason: Some other edge incident on v picks the same color as (u, v) in round i. Let $F_i^{(2)}(v) = F_i(v) \setminus F_i^{(1)}(v)$ denote the set of remaining edges incident on v that fails in round i. An edge $(u, v) \in N_i(v) \cap S_i$ belongs to the set $F_i^{(2)}(v)$ iff no other edge $e_v \in N_i(v) \cap S_i$ picks the same color as (u, v) in round i, *and* at least one edge $e_u \in N_i(u) \cap S_i$ picks the same color as (u, v) in round i. We say that a failed edge $e \in F_i(v)$ is of type-(1) iff $e \in F_i^{(1)}(v)$ and it is of type-(2) iff $e \in F_i^{(2)}(v)$. We will separately prove concentration bounds on the number of failed type-(1) and type-(2) edges incident on v. As $|F_i(v)| = |F_i^{(1)}(v)| + |F_i^{(2)}(v)|$, this will lead to the desired concentration bound on $|F_i(v)|$.

Deriving a concentration bound on $|F_i^{(1)}(v)|$:

Claim 9.6.26 bounds the expected value of $|F_i^{(1)}(v)|$. Claim 9.6.27 shows that w.h.p. $|F_i^{(1)}(v)|$ does not deviate too far away from its expectation. Claim 9.6.27 follows from Claim 9.6.26 and Claim 9.6.27.

Claim 9.6.26. *We have* $\mathbb{E}\left[|F_i^{(1)}(v)|\right] \leqslant 4\varepsilon^2\Delta$.

Proof. Note that $F_i^{(1)}(v) \subseteq N_i(v) \cap S_i$. Consider any edge $e = (u, v) \in N_i(v) \cap S_i$. Our immediate goal is to bound the probability that this edge e does *not* belong to $F_i^{(1)}(v)$, conditioned on it picking a given color $c \in P_i(e)$ in round i. Towards this end, we first recall that the bits $r_{<i} \cup r_i^{(\text{edges})}$ we are conditioning upon ensure the occurrence of the event $\mathcal{E}_i \cap \mathcal{C}_i \cap \mathcal{B}_i$. Hence, we have:

$$\begin{aligned}
|P_i(e')| &= (1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta \text{ for all edges } e' \in E_i. && (9.38)\\
|N_{i,c}(x)| &= (1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta \text{ for all nodes } x \in V. && (9.39)\\
|N_{i,c}(x) \cap S_i| &= (\varepsilon \pm \varepsilon^2) \cdot (1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta \text{ for all nodes } x \in V. && (9.40)
\end{aligned}$$

Conditioned on the edge e picking the color $c \in P_i(e)$, it does not belong to $F_i^{(1)}(v)$ iff none of the edges $e' \in (N_{i,c}(v) \cap S_i) \setminus \{e\}$ picks the same color c in round i. Hence, we derive that:

$$\begin{aligned}
\Pr\left[e \notin F_i^{(1)}(v) \,\middle|\, c(e) = c\right] &= \prod_{e' \in (N_{i,c}(v) \cap S_i) \setminus \{e\}} \left(1 - \frac{1}{|P_i(e')|}\right)\\
&= \left(1 - \frac{1}{(1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta}\right)^{|N_{i,c}(v) \cap S_i| - 1}\\
&\geqslant \left(1 - \frac{1}{(1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta}\right)^{|N_{i,c}(v) \cap S_i|}\\
&= \left(1 - \frac{1}{(1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta}\right)^{(\varepsilon \pm \varepsilon^2) \cdot (1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta}\\
&\geqslant 1 - \varepsilon - (4\varepsilon\gamma_i + \varepsilon^2)\\
&\geqslant 1 - 2\varepsilon. && (9.41)
\end{aligned}$$

In the derivation above, the second step follows from (9.38), the fourth step follows from (9.40), the fifth step follows from Claim 9.6.15, and the last step follows from (9.1) and Corollary 9.6.6. Since (9.41) holds for every color $c \in P_i(e)$, we conclude that:

$$\Pr\left[e \notin F_i^{(1)}(v)\right] \geqslant 1 - 2\varepsilon \text{ for every edge } e \in N_i(v) \cap S_i. \qquad (9.42)$$

Now, applying linearity of expectation, we get:

$$\begin{aligned}
\mathbb{E}\left[|F_i^{(1)}(v)|\right] &= \sum_{e \in N_i(v) \cap S_i} \Pr\left[e \in F_i^{(1)}(v)\right]\\
&\leqslant |N_i(v) \cap S_i| \cdot (2\varepsilon) && \text{(by (9.42))}\\
&\leqslant (2\varepsilon\Delta) \cdot (2\varepsilon) && \text{(by (9.1), (9.40) and Corollary 9.6.6)}\\
&= 4\varepsilon^2\Delta.
\end{aligned}$$

This concludes the proof of the claim. □

Claim 9.6.27. *We have:* $\Pr\left[|F_i^{(1)}(v)| \leqslant \mathbb{E}\left[|F_i^{(1)}(v)|\right] + 50\sqrt{\Delta \ln n}\right] \geqslant 1 - 1/n^{310}$.

Proof. For each edge $e \in N_i(v) \cap S_i$, define a random variable $X_e \in P_i(e)$ whose value indicates the color picked by the edge e in round i. The quantity $|F_i^{(1)}(v)|$ is a function of the random variables $\{X_e\}, e \in N_i(v) \cap S_i$, and the random variables $\{X_e\}$ themselves are mutually independent.

We claim that the function $|F_i^{(1)}(v)|$ satisfies the Lipschitz property (see Definition 9.6.19) with constants $d_e = 4$, $e \in N_i(v) \cap S_i$. To see why the claim holds, consider any given edge $e \in N_i(v) \cap S_i$ and fix the colors picked by every other edge $e' \in (N_i(v) \cap S_i) \setminus \{e\}$ in round i. Fix any two distinct colors $c_1, c_2 \in P_i(e)$. Let n_{c_1} and n_{c_2} respectively denote the number of edges $e' \in (N_i(v) \cap S_i) \setminus \{e\}$ that have picked color c_1 and color c_2 in round i. Now, consider the following two scenarios:

- (1) The edge e picks the color $c_1 \in P_i(e)$ in round i.
- (2) The edge e picks the color $c_2 \in P_i(e) \setminus \{c_1\}$ in round i.

As we switch from scenario (1) to scenario (2), the number of type-(1) failed edges in $N_i(v) \cap S_i$ that pick color c_2 increases by $\phi(n_{c_2} + 1) - \phi(n_{c_2})$; where $\phi(y) = y$ if $y \geqslant 2$, and $\phi(y) = 0$ otherwise. Similarly, the number of type-(1) failed edges in $N_i(v) \cap S_i$ that pick color c_1 decreases by $\phi(n_{c_1} + 1) - \phi(n_{c_1})$. In contrast, the number of type-(1) failed edges in $N_i(v) \cap S_i$ that pick any color $c \notin [\Delta] \setminus \{c_1, c_2\}$ remains unchanged. Thus, as we switch from scenario (1) to scenario (2), the absolute value of the change in $|F_i^{(1)}(v)|$ is given by:

$$\begin{aligned}
&|\{\phi(n_{c_2} + 1) - \phi(n_{c_2})\} - \{\phi(n_{c_1} + 1) - \phi(n_{c_1})\}| \\
\leqslant &|\phi(n_{c_2} + 1) - \phi(n_{c_2})| + |\phi(n_{c_1} + 1) - \phi(n_{c_1})| \\
\leqslant &2 + 2 = 4.
\end{aligned}$$

We therefore conclude that $|F_i^{(1)}(v)|$ is a function of mutually independent random variables $\{X_e\}$ that satisfy the Lipschitz property (see Definition 9.6.19) with constants $d_e = 4$, $e \in N_i(v) \cap S_i$. Since $\sum_{e \in N_i(v) \cap S_i} d_e^2 = 16 \cdot |N_i(v) \cap S_i| \leqslant 16\Delta$, applying Lemma 9.6.20 we get:

$$\Pr\left[|F_i^{(1)}(v)| \leqslant \mathbb{E}\left[|F_i^{(1)}(v)|\right] + 50\sqrt{\Delta \ln n}\right] \geqslant 1 - 1/n^{310}.$$

This concludes the proof of the claim. □

Corollary 9.6.28. *We have:* $\Pr\left[|F_i^{(1)}(v)| \leqslant 4\varepsilon^2\Delta + 50\sqrt{\Delta \ln n}\right] \geqslant 1 - 1/n^{310}$.

Proof. Follows from Claim 9.6.26 and Claim 9.6.27. □

Deriving a concentration bound on $|F_i^{(2)}(v)|$:

While analyzing this quantity, for technical reasons we first condition upon the colors picked by all the edges incident on v that are sampled in round i. After this conditioning, we bound the expected value of $|F_i^{(2)}(v)|$ in Claim 9.6.29, and in Claim 9.6.30 we show that w.h.p. $|F_i^{(2)}(v)|$ does not deviate too far away from its expectation. Corollary 9.6.31 follows from Claim 9.6.29 and Claim 9.6.30.

Claim 9.6.29. *Fix any color $c^*(e) \in P_i(e)$ for every edge $e \in N_i(v) \cap S_i$. Let Γ^* be the event which occurs iff every edge $e \in N_i(v) \cap S_i$ picks the color $c^*(e)$ in round i. Then we have:*

$$\mathbb{E}\left[|F_i^{(2)}(v)| \;\middle|\; \Gamma^*\right] \leqslant 4\varepsilon^2\Delta.$$

Proof. The proof is analogous to the proof of Claim 9.6.26. Nevertheless, for the sake of completeness, we reproduce the same chain of reasoning in its entirety.

The event Γ^* completely determines the set $F_i^{(1)}(v)$. Furthermore, we have $F_i^{(2)}(v) \subseteq (N_i(v) \cap S_i) \setminus F_i^{(1)}(v)$. Consider any edge $e = (u, v) \in (N_i(v) \cap S_i) \setminus F_i^{(1)}(v)$, which picks the color $c^*(e)$ in round i. Our immediate goal is to bound the probability that this edge e does *not* belong to the set $F_i^{(2)}(v)$. Towards this end, we first recall that the bits $r_{<i} \cup r_i^{(\text{edges})}$ we have already conditioned upon ensure the occurrence of the event $\mathcal{E}_i \cap \mathcal{C}_i \cap \mathcal{B}_i$. Hence, we have:

$$\begin{aligned}
|P_i(e')| &= (1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta \text{ for all edges } e' \in E_i. && (9.43)\\
|N_{i,c^*(e)}(x)| &= (1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta \text{ for all nodes } x \in V. && (9.44)\\
|N_{i,c^*(e)}(x) \cap S_i| &= (\varepsilon \pm \varepsilon^2) \cdot (1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta \text{ for all nodes } x \in V. && (9.45)
\end{aligned}$$

The edge $e = (u, v)$ does not belong to the set $F_i^{(2)}(v)$ iff no edge $e' \in (N_{i,c^*(e)}(u) \cap S_i) \setminus \{e\}$ picks the color $c(e') = c^*(e)$ in round i. Hence, we derive that:

$$\begin{aligned}
\Pr\left[e \notin F_i^{(2)}(v) \;\middle|\; \Gamma^*\right] &= \prod_{e' \in (N_{i,c^*(e)}(u) \cap S_i) \setminus \{e\}} \left(1 - \frac{1}{|P_i(e')|}\right)\\
&= \left(1 - \frac{1}{(1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta}\right)^{|N_{i,c^*(e)}(u) \cap S_i| - 1}\\
&\geqslant \left(1 - \frac{1}{(1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta}\right)^{|N_{i,c^*(e)}(u) \cap S_i|}\\
&= \left(1 - \frac{1}{(1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta}\right)^{(\varepsilon \pm \varepsilon^2) \cdot (1-\varepsilon)^{2(i-1)} \cdot (1 \pm \gamma_i) \cdot \Delta}\\
&\geqslant 1 - \varepsilon - (4\varepsilon\gamma_i + \varepsilon^2)\\
&\geqslant 1 - 2\varepsilon. && (9.46)
\end{aligned}$$

In the derivation above, the second step follows from (9.43), the fourth step follows from (9.45), the fifth step follows from Claim 9.6.15, the last step follows from (9.1) and Corollary 9.6.6. Thus, we have:

$$\Pr\left[e \notin F_i^{(2)}(v) \;\middle|\; \Gamma^*\right] \geqslant 1 - 2\varepsilon \text{ for every edge } e \in (N_i(v) \cap S_i) \setminus F_i^{(1)}(v). \tag{9.47}$$

Now, applying linearity of expectation, we get:

$$\begin{aligned}
\mathbb{E}\left[|F_i^{(2)}(v)| \,\middle|\, \Gamma^*\right] &= \sum_{e \in (N_i(v) \cap S_i) \setminus F_i^{(1)}(v)} \Pr\left[e \in F_i^{(2)}(v) \,\middle|\, \Gamma^*\right] \\
&\leqslant |N_i(v) \cap S_i| \cdot (2\varepsilon) && \text{(by (9.47))} \\
&\leqslant (2\varepsilon\Delta) \cdot (2\varepsilon) && \text{(by (9.1), (9.45) and Corollary 9.6.6)} \\
&= 4\varepsilon^2\Delta.
\end{aligned}$$

This concludes the proof of the claim. □

Claim 9.6.30. *Fix any color $c^*(e) \in P_i(e)$ for every edge $e \in N_i(v) \cap S_i$. Let Γ^* be the event which occurs iff every edge $e \in N_i(v) \cap S_i$ picks the color $c^*(e)$ in round i. Then we have:*

$$\Pr\left[|F_i^{(2)}(v)| \leqslant \mathbb{E}\left[|F_i^{(2)}(v)|\right] + 50\sqrt{\Delta \ln n} \,\middle|\, \Gamma^*\right] \geqslant 1 - 1/n^{2000}.$$

Proof. Let $W_i(v) = (N_i(u) \cap S_i) \setminus F_i^{(1)}(v)$ denote the set of edges incident on v that get sampled in round i and do *not* end up being type-(1) failures under the event Γ^*. By definition, all the edges in $W_i(v)$ receive distinct colors under the event Γ^*, and we have $F_i^{(2)}(v) \subseteq W_i(v)$. Let $Z_i(v) = \{(u, u') \in S_i \setminus N_i(v) : \text{either } u \in W_i(v) \text{ or } u' \in W_i(v)\}$ denote the set of edges sampled in round i that are not themselves incident on v, but are neighbors of at least one edge in $W_i(v)$. Note that the sets $W_i(v)$ and $Z_i(v)$, along with the colors picked by the edges in $W_i(v)$, are completely determined by the bits $r_{<i} \cup r_i^{(\text{edges})}$ and the event Γ^* we are conditioning upon. On the other hand, the colors picked by the edges in $Z_i(v)$ are yet to be determined. In particular, each edge in $Z_i(v)$ will pick a color uniformly at random from the set $P_i(e)$, independently of the other edges, and these colors will uniquely determine the set $F_i^{(2)}(v)$. We can think of the edges $e \in Z_i(v)$ picking their colors in round i as a "balls and bins" process, as described below.

There is a ball for each edge $e \in Z_i(v)$, a bin for each color $c \in [(1 + \varepsilon^2)\Delta]$, and an additional dummy bin $\perp$. Consider any edge $e \in Z_i(v)$, and let $W_i(v, e) \subseteq W_i(v)$ be the set of edges $e' \in W_i(v)$ that share an endpoint with e. Note that $|W_i(v, e)| \in \{1, 2\}$. Suppose that the edge e picks a color $c \in P_i(e)$ in round i. Then the ball for e gets thrown into the bin for c iff some edge $e' \in W_i(v, e)$ picked the same color c under the event Γ^*; otherwise the ball for e goes to the dummy bin.

Let $\chi_i(v) = \bigcup_{e \in W_i(v)} \{c^*(e)\}$ denote the set of colors picked by the edges $e \in W_i(v)$ in round i, under the event Γ^*. For each color $c \in \chi_i(v)$, define an indicator random variable $Y_c \in \{0, 1\}$ that is set to one iff the bin for the color c is nonempty (has at least one ball in it) at the end of the balls and bins process described above. As each edge $e \in Z_i(v)$ picks a color independently of the other edges in $Z_i(v)$, the balls get thrown into the bins independently of each other. Hence, Corollary 2.4.6 implies that the random variables $\{Y_c\}, c \in \chi_i(v)$, are negatively associated. Since $|\chi_i(v)| \leqslant (1 + \varepsilon^2)\Delta$, from Lemma 2.4.11 we get:

$$\Pr\left[\sum_{c \in \chi_i(v)} Y_c \leqslant \mathbb{E}\left[\sum_{c \in \chi_i(v)} Y_c\right] + 50\sqrt{\Delta \ln n} \,\middle|\, \Gamma^*\right] \geqslant 1 - 1/n^{2000}. \tag{9.48}$$

Recall that no two edges in $W_i(v)$ pick the same color under the event Γ^*. Accordingly, there is a natural bijective mapping $g : \chi_i(v) \to W_i(v)$, where $g(c)$ is the unique edge in $W_i(v)$ that picked the color $c \in \chi_i(v)$ under the event Γ^*. For each color $c \in \chi_i(v)$, we have $Y_c = 1$ iff $g(c) \in F_i^{(2)}(v)$. Since $F_i^{(2)}(v) \subseteq W_i(v)$, we infer that $\sum_{c\in\chi_i(v)} Y_c = |F_i^{(2)}(v)|$. The claim now follows from (9.48). □

Corollary 9.6.31. *We have:* $\Pr\left[|F_i^{(2)}(v)| \leqslant 4\varepsilon^2\Delta + 50\sqrt{\Delta \ln n}\right] \geqslant 1 - 1/n^{2000}$.

Proof. Fix any color $c^*(e) \in P_i(e)$ for all $e \in N_i(v) \cap S_i$. Let Γ^* be the event which occurs iff every edge $e \in N_i(v) \cap S_i$ picks the color $c^*(e)$ in round i. Claim 9.6.29 and Claim 9.6.30 imply that:

$$\Pr\left[|F_i^{(2)}(v)| \leqslant 4\varepsilon^2\Delta + 50\sqrt{\Delta \ln n} \;\middle|\; \Gamma^*\right] \geqslant 1 - 1/n^{2000}. \tag{9.49}$$

Since the bound in (9.49) holds for every possible Γ^*, the corollary follows. □

Wrapping up the proof of Lemma 9.6.25:
Applying a union bound over Corollary 9.6.28 and Corollary 9.6.31, we get:

$$\Pr\left[|F_i^{(1)}(v)| + |F_i^{(2)}(v)| \leqslant 8\varepsilon^2\Delta + 100\sqrt{\Delta \ln n}\right] \geqslant 1 - 1/n^{305}.$$

Now, Lemma 9.6.25 follows from the following two observations: (a) $|F_i(v)| = |F_i^{(1)}(v)| + |F_i^{(2)}(v)|$, and (b) $100\sqrt{\Delta \ln n} \leqslant \varepsilon^2\Delta$ according to (9.1).

9.7 Conclusion and Open Questions

In this chapter we presented a $(1 + o(1))\Delta$-edge-coloring online algorithm for graphs with $\Delta = \omega(\log n)$ under random-order edge arrivals. Thus, we resolve the conjecture of Bar-Noy et al. [25] for this model. We conclude with a few interesting research directions.

Adversarial Online Arrivals: The most natural question is whether the Bar-Noy et al. conjecture holds in the strictest, adversarial edge-arrival model. This question still seems out of reach. One algorithmic approach which suggests itself is to extend the ideas of Chapter 6. This would require some form of online dependent rounding for fractional matching under edge arrivals, generalizing our work of Chapter 5. Alternatively, it is not implausible that the Bar-Noy et al. conjecture is false under adversarial edge arrivals, despite being true for vertex arrivals. Such a refutation of this conjecture would mirror a similar separation between these arrival models which we presented for online matching, in Chapter 3 and Chapter 4.

Knowledge of Δ: Our algorithms of this chapter assume knowledge of the maximum degree Δ. This assumption is common to all prior best algorithms in the random-order online model [5, 22]. Recall that in Chapter 6 we showed that under adversarial vertex arrivals, not knowing Δ results in a strictly harder problem, for which no better than $\frac{e}{e-1}\Delta$-edge-coloring algorithm exists, for any (unknown) Δ. Is the same separation between known and unknown Δ true for random-order edge arrivals?

Part III

Dynamic and Streaming Algorithms

Chapter 10

Dynamic Matching Versus Adaptive Adversaries

In this chapter we present our work on dynamic matching algorithms against adaptive adversaries, which appeared previously in [266]. This work generalizes and extends joint work with Moab Arar, Shiri Chechik, Sarel Cohen and Cliff Stein [14].

10.1 Background

The field of dynamic graph algorithms studies the maintenance of solutions to graph-theoretic problems subject to graph updates, such as edge additions and removals. For any such dynamic problem, a trivial approach is to recompute a solution from scratch following each update, using a static algorithm. Fortunately, significant improvements over this naïve polynomial-time approach are often possible, and many fundamental problems admit *polylogarithmic* update time algorithms. Notable examples include minimum spanning tree and connectivity [158, 159, 175, 258] and spanners [29, 39, 116]. Many such efficient dynamic algorithms rely on randomization and the assumption of a weak, *oblivious* adversary, i.e., an adversary which cannot decide its updates adaptively based on the algorithm's output. As recently pointed out by Nanongkai and Saranurak [220],

> It is a fundamental question whether the true source of power of randomized dynamic algorithms is the randomness itself or in fact the oblivious adversary assumption.

In this chapter, we address this question for the heavily-studied dynamic matching problem. For this problem, the assumption of an oblivious adversary is known to allow for constant-approximate worst-case polylogarithmic update time algorithms [14, 39, 62]. In contrast, all deterministic algorithms with worst-case time guarantees have *polynomial* update time [37, 48, 145, 224, 233]. The main advantage of deterministic algorithms over their randomized counterparts is their robustness to *adaptive* adversaries; i.e., their guarantees even hold for update sequences chosen adaptively. Before outlining our results, we discuss some implications of the oblivious adversary assumption, which motivate the study of algorithms which are robust to adaptive adversaries.

Static implications. As Mądry [217] observed, randomized dynamic algorithms' assumption of an oblivious adversary renders them unsuitable for use as a black box for many static applications. For example, [115, 124] show how to approximate multicommodity flows by repeatedly routing flow along approximate shortest paths, where edges' lengths are determined by their current congestion. These shortest path computations can be sped up by a dynamic shortest path algorithm, provided it works against an *adaptive* adversary (since edge lengths are determined by prior queries' outputs). This application has motivated much work on faster *deterministic* dynamic shortest path algorithms [33, 35, 36, 148, 157], as well as a growing interest in faster randomized dynamic algorithms which work against adaptive adversaries [67, 68, 147].

Dynamic implications. The oblivious adversary assumption can also make a dynamic algorithm $\mathcal{A}$ unsuitable for use by other *dynamic* algorithms, even ones which themselves assume an oblivious adversary! For example, for dynamic algorithms that use several copies of $\mathcal{A}$ whose inputs depend on each other's output, the different copies may act as adaptive adversaries for one another, if the behavior of copy i affects that of copy j, which in turn affects that of copy i. (See [220].)

Faster algorithms that are robust to adaptive adversaries thus have the potential to speed up both static and dynamic algorithms. This motivated Nanogkai et al. [221], who studied dynamic MST, to ask whether there exist algorithms against adaptive adversaries for other well-studied dynamic graph problems, with similar guarantees to those known against oblivious adversaries.

In this chapter we answer this question affirmatively for the dynamic matching problem, for which we give the first randomized algorithms that are robust to adaptive adversaries (and outperform known deterministic algorithms).

10.1.1 Our Contributions

Our main contribution is a framework for dynamically rounding fractional matchings against adaptive adversaries. That is, we develop a method which given a dynamically-changing fractional matching (i.e., a point $\vec{x}$ in the fractional matching polytope, $\mathcal{P} := \{\vec{x} \in \mathbb{R}^m_{\geqslant 0} \mid \sum_{e \ni v} x_e \leqslant 1 \ \forall v \in V\}$), outputs a matching M of size roughly equal to the value of the fractional matching, $\sum_e x_e$. This framework allows us to obtain dynamic matching algorithms robust to adaptive adversaries, including adversaries that see the algorithms' **entire state** after each update.

Key to our framework is a novel matching sparsification scheme, i.e., a method for computing a sparse subgraph which approximately preserves the maximum matching size. We elaborate on our sparsification scheme and dynamic rounding framework and their analyses in later sections. For now, we discuss some of the dynamic matching algorithms we obtain from applying our framework to various known dynamic fractional matching algorithms.

Our first result (applying our framework to [45]) is a $(2+\varepsilon)$-approximate matching algorithm with worst-case polylogarithmic update time against an adaptive adversary.

Theorem 10.1.1. *For every $\varepsilon \in (0, 1/2)$, there exists a (Las Vegas) randomized $(2+\varepsilon)$-approximate algorithm with update time $\text{poly}(\log n, 1/\varepsilon)$ w.h.p. against an* adaptive *adversary.*

All algorithms prior to this work either assume an oblivious adversary or have *polynomial* worst-case update time, for any approximation ratio.

Our second result (applying our framework to [41]) yields amortized *constant-time* algorithms matching Theorem 10.1.1's approximation ratio, also against an adaptive adversary.

Theorem 10.1.2. *For every $\varepsilon \in (0, 1/2)$, there exists a randomized $(2+\varepsilon)$-approximate dynamic matching algorithm with $\text{poly}(1/\varepsilon)$ amortized update time whose approximation and update time guarantees hold in expectation against an* adaptive *adversary.*

No constant-time algorithms against adaptive adversaries were known before this work, for *any* approximation ratio. A corollary of Theorem 10.1.2, obtained by amplification, is the first algorithm against adaptive adversaries with logarithmic amortized update time and $O(1)$-approximation w.h.p.

Finally, our framework also lends itself to better-than-two approximation. In particular, plugging in the fractional matching algorithm of [43] into our framework yields $(2-\delta)$-approximate algorithms with *arbitrarily-small* polynomial update time against adaptive adversaries in bipartite graphs.

Theorem 10.1.3. *For all constant $k \geqslant 10$, there exists a $\beta_k \in (1, 2)$, and a β_k-approximate dynamic bipartite matching algorithm with expected update time $O(n^{1/k})$ against* adaptive *adversaries.*

Similar results were recently achieved for general graphs, assuming an oblivious adversary [30]. All other $(2-\delta)$-approximate algorithms are deterministic (and so do not need this assumption), but have $\Omega(\sqrt[4]{m})$ update time.

As a warm-up to our randomized rounding framework, we present a family of deterministic algorithms with arbitrarily-small polynomial worst-case update time, yielding the following time-approximation trade-off.

Theorem 10.1.4. *For any $K > 1$, there exists a deterministic $O(K)$-approximate matching algorithm with worst-case $\tilde{O}(n^{1/K})$ update time.*

This family of algorithms includes the first deterministic constant-approximate algorithms with $o(\sqrt[4]{m})$ worst-case update time. It also includes the first deterministic $o(\log n)$-approximate algorithm with worst-case polylog update time. No deterministic algorithms with worst-case polylog update time were known for any *sublinear* $n^{1-\varepsilon}$ approximation ratio.

Weighted Matching. Our dynamic matching algorithms imply dynamic maximum *weight* matching (MWM) algorithms with roughly twice the approximation ratio, with only a logarithmic slowdown, by standard reductions (see [14, 253]). Since our matching algorithms work against adaptive adversaries, we can apply these reductions as a black box, and need not worry about the inner workings of these reductions. As an added bonus, the obtained MWM algorithms work against adaptive adversaries (the first such randomized algorithms), since their constituent subroutines do.

10.1.2 Techniques

In this section we outline our sparsification scheme and framework for dynamic matching against adaptive adversaries. Specifically, we show how to use edge colorings—partitions of the edges into (few) matchings—to quickly round fractional matchings dynamically against adaptive adversaries. Before detailing these, we explain why the work of Gupta and Peng [145] motivates the study of dynamic matching sparsification.

In [145], Gupta and Peng present a $(1+\varepsilon)$-approximate $O(\sqrt{m}/\varepsilon^2)$-time algorithm, using a sparsifier and what they call the "stability" of the matching problem, which lends itself to lazy re-computation, as follows. Suppose we compute a matching M of size at least $1/C$ times $\mu(G)$, the maximum matching size in G. Then, regardless of the updates in the following period of $\varepsilon \cdot \mu(G)$ steps, the edges of M not deleted during the period remain a $C(1+O(\varepsilon))$-approximate matching in the dynamic graph, since both the size of M and $\mu(G)$ can at most change by $\varepsilon \cdot \mu(G)$ during such a period. So, for example, using a static $O(m/\varepsilon)$-time $(1+\varepsilon)$-approximate matching algorithm [210] every $\varepsilon \cdot \mu(G)$ updates yields a $(1+O(\varepsilon))$-approximate dynamic matching algorithm with amortized update time $O_\varepsilon(m/\mu(G))$. To obtain better update times from this observation, Gupta and Peng apply this idea to a sparsifier of size $S = O(\min\{m, \mu(G)^2\})$ which contains a maximum matching of G and which they show how to maintain in $O(\sqrt{m})$ update time, using the algorithm of [224]. From this they obtain a $(1+O(\varepsilon))$-approximate matching algorithm with update time $O(\sqrt{m}) + (S/\varepsilon)/(\varepsilon \cdot \mu(G)) = O(\sqrt{m}/\varepsilon^2)$. We note that this lazy re-computation approach would even allow for *polylogarithmic-time* dynamic matching algorithms with approximation ratio $C+O(\varepsilon)$, provided we could compute C-approximate matching sparsifiers of (optimal) size $S = \tilde{O}_\varepsilon(\mu(G))$, in time $\tilde{O}_\varepsilon(\mu(G))$. (We note that any sparsifier containing a constant-approximate matching must have size $\Omega(\mu(G))$.)

In this work we show how to use edge colorings to sample such size-optimal matching sparsifiers in optimal time. For simplicity, we describe our approach in terms of the subroutines needed to prove Theorem 10.1.1, deferring discussions of extensions to future sections.

Suppose we run the dynamic fractional matching algorithm of [45], maintaining a constant-approximate fractional matching $\vec{x}$ in deterministic worst-case polylog time. Also, for some $\varepsilon > 0$, we dynamically partition G's edges into $O(\log n)$ subgraphs G_i, for $i = 1, 2, \ldots, O(\log_{1+\varepsilon}(n))$, where G_i is the subgraph induced by edges of x-value $x_e \in ((1+\varepsilon)^{-i}, (1+\varepsilon)^{-i+1}]$. By the fractional matching constraint ($\sum_{e \ni v} x_e \leqslant 1 \; \forall v \in V$) and since $x_e \geqslant (1+\varepsilon)^{-i}$ for all edges $e \in E(G_i)$, the maximum degree of any G_i is at most $\Delta(G_i) \leqslant (1+\varepsilon)^i$. We can therefore edge-color each G_i with $2(1+\varepsilon)^i (\geqslant 2\Delta(G_i))$ colors in deterministic worst-case $O(\log n)$ time per update in G_i, using [46]; i.e., logarithmic time per each of the $\operatorname{poly}\log n$ many changes which algorithm $\mathcal{A}$ makes to $\vec{x}$ per update. Thus, edge coloring steps take worst-case $\operatorname{poly}\log n$

time per update. A simple averaging argument shows that the largest color in these different G_i is an $O(\log n)$-approximate matching, which can be maintained efficiently. Extending this idea further yields Theorem 10.1.4 (see Section 10.3). So, picking a singe color yields a fairly good approximation/time tradeoff. As we show, randomly combining a few colors yields space- and time-optimal constant-approximate matching sparsifiers.

To introduce our random sparsification scheme, we start by considering sampling of a single color M among the $2(1+\varepsilon)^i$ colors of the coloring of subgraph G_i. For each edge $e \in G_i$, since $x_e \approx (1+\varepsilon)^{-i}$, when sampling a random color M among these $2(1+\varepsilon)^i$ colors, we sample the unique color containing e with probability proportional to x_e. Specifically, we have

$$\Pr[e \in M] = \frac{1}{2(1+\varepsilon)^i} \approx \frac{x_e}{2}.$$

Our approach will be to sample $\min\left\{2(1+\varepsilon)^i, \frac{2\log n}{\varepsilon^2}\right\}$ colors without replacement in G_i, yielding a subgraph H of G which contains each edge e with probability roughly

$$p_e := \min\left\{1,\, x_e \cdot \frac{\log n}{\varepsilon^2}\right\}. \tag{10.1}$$

As shown by Arar et al. [14], sampling a subgraph H with each edge $e \in E[G]$ belonging to H *independently* with probability p_e as above, with $\vec{x}$ taken to be the $(2+\varepsilon)$-approximate fractional matching output by [45], yields a $(2+\varepsilon)$-approximate matching sparsifier.[1] Sampling H in this independent manner, however, requires $\Omega(m)$ time, and so is hopelessly slow against an adaptive adversary, who can erase H in $\tilde{O}(\mu(G))$ time, therefore forcing an update time of $\tilde{\Omega}(m/\mu(G))$. We prove that sampling H in our above *dependent* manner yields as good a matching sparsifier as does independent sampling, while allowing for $\tilde{O}(1)$ update time.

To bound the approximation ratio of our (dependent) sampling-based sparsifiers, we appeal to the theory of *negative association* (see Section 2.4.1). In particular, we rely on sampling without replacement being a negatively-associated joint distribution. This implies sharp concentration of weighted degrees of vertices in H, which forms the core of our analysis of the approximation ratio of this sparsification scheme. In particular, we show that our matching sparsification yields sparsifiers with approximation ratio essentially equaling that of any "input" fractional matching in bipartite graphs, as well as a $(2+\varepsilon)$-approximate sparsifiers in general graphs, using the fractional matchings of [41, 45].

Finally, to derive fast dynamic algorithms from this sparsification scheme, we note that our matching sparsifier H is the union of only $\text{poly}\log n$ many matchings, and thus has size $\tilde{O}(\mu(G))$. Moreover, sampling this sparsifier requires only $\text{poly}\log n$ random choices, followed by writing H. Therefore, H can be sampled in $\tilde{O}(\mu(G))$ time (given the edge colorings, which we maintain dynamically). The space- and time-optimality of our sparsification scheme implies that we can maintain a matching with approximation ratio essentially equal to that of the obtained sparsifier, in worst-case $\text{poly}\log n$ update time. In particular, we can re-sample such a sparsifier, and compute a $(1+\varepsilon)$-approximate matching in it, in $\tilde{O}_\varepsilon(\mu(G))$ time, after every period of

[1]A simpler argument implying H contains a $(2+\varepsilon)$-*fractional* matching with respect to G only implies a $(3+\varepsilon)$-approximation. This is due to the $\frac{3}{2}$ integrality gap of the fractional matching polytope, and in particular the fact that fractional matchings may be $\frac{3}{2}$ times larger than the largest matching in a graph (see Chapter 2).

$\varepsilon \cdot \mu(G)$ steps. This results in an $\tilde{O}_\varepsilon(\mu(G))/(\varepsilon \cdot \mu(G)) = \tilde{O}_\varepsilon(1)$ amortized time per update (which is easily de-amortized). Crucially for our use, during such periods, $\mu(G)$ and $\mu(H)$ do not change by much, as argued before. In particular, during such short periods of few updates, an adaptive adversary—even one which sees **the entire state** of the algorithm after each update—cannot increase the approximation ratio by more than a $1 + O(\varepsilon)$ factor compared to the approximation quality of the sparsifier. This yields a $(2 + \varepsilon)$-approximate dynamic matching algorithm with worst-case polylogarithmic update time against adaptive adversaries, proving Theorem 10.1.1. Generalizing this further, we design a framework for dynamically rounding fractional matchings against adaptive adversaries, underlying all our algorithms of theorems 10.1.1, 10.1.2 and 10.1.3.

10.1.3 Related Work

Here we discuss the dynamic matching literature in more depth, contrasting it with the results obtained from our dynamic rounding framework.

In 2007, Sankowski [244] presented an $O(n^{1.495})$ update time algorithm for maintaining the *value* (size) of a maximum matching, recently improved to $O(n^{1.407})$ [260]. These algorithms, while faster than the naïve $O(m)$ time algorithm for sufficiently dense graphs, are far from the gold standard for data structures – polylog update time. Several works show that this is inevitable, however, as polylog update time (exact) maximum matching is *impossible*, assuming several widely-held conjectures, including the strong exponential time hypothesis and the 3-sum conjecture [2, 3, 75, 156, 188]. A natural question is then whether polylog update time suffices to maintain an *approximate* maximum matching.

Polylog-time algorithms: In a seminal paper, Onak and Rubinfeld [228] presented the first polylog-time algorithm for constant-approximate matching. Baswana et al. [28] improved this with an $O(\log n)$-time maximal (and thus 2-approximate) matching algorithm. Some years later Bhattacharya et al. [43] presented a deterministic $(2 + \varepsilon)$-approximate matching algorithm with amortized $\text{poly}(\log n, 1/\varepsilon)$ update time. Solomon [251] then gave a randomized maximal matching algorithm with *constant* amortized time. Recently, several randomized $(2 + \varepsilon)$-approximate/maximal matching algorithms with worst-case polylog time were developed, with either the approximation ratio or the update time holding w.h.p. [14, 39, 62].[2] All prior randomized algorithms assume an oblivious adversary, and obtaining the same guarantees against an adaptive adversary remained open. Another line of work studied the dynamic maintenance of large *fractional* matchings in polylog update time, thus maintaining a good approximation of the maximum matching's *value* (though not a large matching) [41, 44, 45, 47, 143]. The best current bounds for this problem are deterministic $(2 + \varepsilon)$-approximate fractional matching algorithms with $\text{poly}(\log n, 1/\varepsilon)$ worst-case and $\text{poly}(1/\varepsilon)$ amortized update times [41, 45]. Our randomized algorithms of Theorems 10.1.1 and 10.1.2 match these bounds, for *integral* matching, against adaptive adversaries.

[2]The algorithm of [39] even maintains a 2-approximate matching if one allows for (implicitly) changing the entire matching between updates. Otherwise, the framework of [250] allows to decrease the number of changes to the matching per update, at the cost of increasing the approximation ratio slightly to $2+\varepsilon$. Our algorithms, as stated, similarly require implicitly changing the entire matching between algorithms. For applications where this is not desirable, we can similarly apply the framework of [250] to our algorithms, while keeping the same approximation ratio and asymptotic update time, and keeping the number of changes to the matching per update to be constant.

Polytime algorithms: Many sub-linear time dynamic matching algorithms were developed over the years. The first is due to Ivkovic and Lloyd [168], who showed how to maintain maximal matchings in $O((m+n)^{1/\sqrt{2}})$ amortized update time. More recent work includes $(1+\varepsilon)$-approximate algorithms with $O(\sqrt{m}/\varepsilon^2)$ worst-case update time [145, 233] (the former building on a maximal $O(\sqrt{m})$-time algorithm of [224]), and $(2+\varepsilon)$-approximate algorithms with worst-case $O(\min\{\sqrt[3]{m}, \sqrt{n}\}/\operatorname{poly}(\varepsilon))$ update time [48]. The fastest known algorithm with worst-case update time is a $(\frac{3}{2}+\varepsilon)$-approximate $O(\sqrt[4]{m}/\operatorname{poly}(\varepsilon))$-time algorithm for bipartite graphs [37] (similar amortized bounds are known for general graphs [38]). In contrast, we obtain algorithms with *arbitrarily-small* polynomial update time, yielding a constant approximation deterministically (Theorem 10.1.4), and even better-than-2 approximation in bipartite graphs against adaptive adversaries (Theorem 10.1.3). This latter bound was previously only known for dynamic *fractional* matching [43], and nearly matches a recent $O(\Delta^{\varepsilon})$-time algorithm for general graphs, which assumes an oblivious adversary [30].

Matching sparsifiers: Sparsification is a commonly-used algorithmic technique. In the area of dynamic graph algorithms it goes back more than twenty years [94]. For the matching problem in various computational models, multiple sparsifiers were developed [15, 16, 37, 38, 48, 132, 145, 191, 233, 252]. Unfortunately for dynamic settings, all these sparsifiers are either polynomially larger than $\mu(G)$, the maximum matching size in G, or were not known to be maintainable in $n^{o(1)}$ time against adaptive adversaries. In this chapter we show how to efficiently maintain a generalization of matching kernels of [48] of size $\tilde{O}(\mu(G))$, efficiently, against adaptive adversaries.

10.2 Preliminaries

In a fully-dynamic setting, the input is a dynamic graph G, initially empty, on a set of n fixed vertices V, subject to edge *updates* (additions and removals). A dynamic algorithm has *worst-case* update time $f(n)$ if it requires $f(n)$ time for each update. It is said to have *amortized* update time $f(n)$ if it requires $O(t \cdot f(n))$ time for any sequence of t updates. If we assume an oblivious adversary, these time bounds need only hold for sequences chosen before the algorithm's run. An α-approximate matching algorithm $\mathcal{A}$ maintains a matching M of size at least $|M| \geqslant \frac{1}{\alpha} \cdot \mu(G)$. If $\mathcal{A}$ is deterministic, $|M| \geqslant \frac{1}{\alpha} \cdot \mu(G)$ holds for any sequence of updates. If $\mathcal{A}$ is randomized, this bound on M's size can hold in expectation or w.h.p., though here one must be more careful about the sequence of updates. The strongest guarantees for randomized algorithms are those which hold for sequences generated by an adaptive adversary.

Dynamic Edge Coloring: An important ingredient in our matching algorithms are algorithms for the "complementary" problem of edge coloring, i.e., the problem of covering the graph's edge-set with few matchings (colors). Vizing's theorem [261] asserts that $\Delta+1$ colors suffice to edge color any graph of maximum degree Δ. (Clearly, at least Δ colors are needed.) In dynamic graphs, a deterministic $(2\Delta-1)$-edge-coloring algorithm with $O(\log n)$ worst-case update time is known [46]. Also, a 3Δ-edge-coloring can be trivially maintained in $O(1)$ expected update time against an adaptive adversary, by picking random colors for each new edge (u, v) until an available color is picked. Dynamic algorithms using fewer colors are known, though they are slower [82]. Moreover, as the number of colors $\gamma\Delta$ used only affects our update times by a factor of γ (and does not affect our approximation ratio), the above simple 2Δ- and 3Δ-edge-coloring algorithms will suffice for our needs.

10.3 Warm Up: Deterministic Algorithms

We start by discussing our deterministic matching algorithms obtained by generalizing the discussion in Section 10.1.1.

First, we note that the $(2\Delta - 1)$-edge-coloring algorithm of [46] works for multigraphs.

Lemma 10.3.1. *For any dynamic* multigraph *G with maximum degree Δ, there exists a deterministic $(2\Delta - 1)$-edge-coloring algorithm with worst-case update time $O(\log \Delta)$.*

Broadly, the algorithm of [46] relies on binary search, relying on the following simple observation. For $(2\Delta - 1)$ colors, if we add an edge (u, v), then the total number of colors used by u and v for all their (at most $\Delta - 1$) edges other than (u, v), even counting repetitions, is at most $2\Delta - 2$. That is, fewer than the number of colors in the entire palette, $[2\Delta - 1]$. Consequently, either the range $\{1, 2, \ldots, \Delta\}$ or $\{\Delta + 1, \Delta + 2, \ldots, 2\Delta - 1\}$ has a smaller number of colors used by u and v (again, counting repetitions). This argument continues to hold recursively in this range in which u and v have used fewer colors than available. With the appropriate data structures, this observation is easily implemented to support $O(\log \Delta)$ worst-case update time for both edge insertions and deletions (see [46] for details). As the underlying binary-search argument above did not rely on simplicity of the graph, this algorithm also works for multigraphs.

We now show how to use this simple edge-coloring algorithm in conjunction with dynamic fractional matching algorithms to obtain a family of deterministic algorithms allowing to trade off approximation ratio for worst-case update time.

Theorem 10.1.4. *For any $K > 1$, there exists a deterministic $O(K)$-approximate matching algorithm with worst-case $\tilde{O}(n^{1/K})$ update time.*

Proof. We maintain in the background a 2.5-approximate fractional matching $\vec{x}$ using a deterministic algorithm with worst-case polylogarithmic update time, such as that of [45] run with $\varepsilon = 0.5$. Letting $\mathcal{R} := n^{1/K}$, we define $O(K)$ multigraphs whose union contains all edges in G. Specifically, for each $i = 1, 2, \ldots, 2\log_{\mathcal{R}}(2n)$ we let G_i be a multigraph whose edges are the edges of G of x-value $x_e \in [\mathcal{R}^{-i}, \mathcal{R}^{-i+1}]$, with each such edge e having $\lceil x_e/\mathcal{R}^{-i} \rceil$ parallel copies in G_i. So, for example, an edge with x-value of $\mathcal{R}^{-i}$ will have a single parallel copy in G_i, and an edge wit x-value of $\mathcal{R}^{-i+1}$ will have $\lceil \mathcal{R} \rceil \leqslant n^{1/K} + 1$ parallel copies in G_i. By the fractional matching constraint ($\sum_{e \ni v} x_e \leqslant 1 \; \forall v \in V$), the maximum degree in each graph G_i is at most $\Delta(G_i) \leqslant \mathcal{R}^i$. Therefore, using the edge coloring algorithm of [46] we can maintain a $2\Delta(G_i) - 1 \leqslant 2 \cdot \mathcal{R}^i$ edge coloring in each G_i deterministically in worst-case $O(\log n)$ time per edge update in G_i. Since for any edge e a change to x_e causes at most $\lceil \mathcal{R} \rceil$ parallel copies of

e to be added to or removed from multigraphs G_i, we find that each x-value changes performed by the fractional matching algorithm require $O(\mathcal{R} \cdot \log n)$ worst-case time. As the fractional algorithm has polylogarithmic update time (and therefore at most that many x-value changes per update), the overall update time of these subroutines is therefore at most $\tilde{O}(\mathcal{R}) = \tilde{O}(n^{1/K})$. Our algorithm simply maintains as its matching the largest color class in any of these multigraphs. It remains to bound the approximation ratio of this approach.

First, we note that all edges not in any G_i, i.e., of x-value at most $\mathcal{R}^{-\log_{\mathcal{R}}(2n)} = 1/(4n^2)$, contribute at most $\sum_{e:x_e \leqslant \varepsilon^2/n^2} x_e \leqslant 1/4$ to $\sum_e x_e$. So, as $\vec{x}$ is a 2.5-approximate fractional matching, we have that

$$\sum_{e \in \bigcup G_i} x_e \geqslant \frac{1}{2.5} \cdot \mu(G) - \frac{1}{4} \geqslant \frac{1}{O(1)} \cdot \mu(G),$$

where as before, $\mu(G) \geqslant 1$ is the maximum matching size in G. (Note that if $\mu(G) = 0$ any algorithm is trivially 1-approximate.) Therefore, as $\mathcal{R} = n^{1/K}$ at least one of these $2\log_{\mathcal{R}}(2n) = O(K)$ multigraphs G_i must have total x-value at least

$$\sum_{e \in G_i} x_e \geqslant \frac{1}{O(K)} \cdot \frac{1}{O(1)} \cdot \mu(G) = \frac{1}{O(K)} \cdot \mu(G).$$

But, as this multigraph G_i has at least $|E(G_i)| = \sum_{e \in G_i} \lceil x_e / \mathcal{R}^{-i+1} \rceil \geqslant \sum_{e \in G_i} x_e \cdot \mathcal{R}^{i-1}$ edges, one of the $2\Delta(G_i) - 1 \leqslant 2\mathcal{R}^{i+1}$ colors (matchings) in G_i must have size at least

$$\frac{|E(G_i)|}{2\Delta(G_i) - 1} \geqslant \frac{\sum_{e \in G_i} x_e \cdot \mathcal{R}^{i-1}}{2\mathcal{R}^i} \geqslant \frac{\sum_{e \in G_i} x_e}{4} \geqslant \frac{1}{4} \cdot \frac{1}{O(K)} \cdot \mu(G) = \frac{1}{O(K)} \cdot \mu(G).$$

As this algorithm's matching is the largest color class in all the edge colorings of all the different G_i, it is $O(K)$ approximate, as claimed. □

Corollary 10.3.2. *There exists a deterministic $O\left(\frac{\log n}{\log \log n}\right)$-approximate matching algorithm with worst-case* poly $\log n$ *update time.*

Remark 1: We note that the algorithm of Theorem 10.1.4 requires $O(m \cdot n^{1/K})$ space to store the multigraphs G_i and their relevant data structures, since each edge e in a graph G_i may have x-value precisely $\mathcal{R}^{-i+1}$, which means we represent this edge using $O(\mathcal{R}) = O(n^{1/K})$ parallel edges in G_i. It would be interesting to see if its approximation to worst-case update time tradeoff can be matched by a deterministic algorithm requiring $\tilde{O}(m)$ space.

Remark 2: We note that the matching maintained by our deterministic algorithms can change completely between updates. For applications where this is undesirable, combining this algorithm with a recent framework of Solomon and Solomon [250] yields a dynamic matching M' of roughly the same size while only changing $O(1/\varepsilon)$ edges of M' per update.

10.4 Edge-Color and Sparsify

In this section we present our edge-coloring-based matching sparsification scheme, and useful properties of this sparsifier, necessary to bound its quality. We then show how to implement this scheme in a dynamic setting against an adaptive adversary with $(1-\varepsilon)$ loss in the approximation ratio. We start by defining our sparsification scheme in a static setting.

10.4.1 The Sparsification Scheme

Our edge-coloring-based sparsification scheme receives a fractional matching $\vec{x}$ as an input, as well as parameters $\varepsilon \in (0,1), d \geqslant 1$ and integer $\gamma \geqslant 1$. It assumes access to a $\gamma\Delta$-edge-coloring algorithm for graphs of maximum degree Δ. For some logarithmic number of indices $i = 1, 2, \ldots, 3\log_{1+\varepsilon}(n/\varepsilon) = O(\log(n/\varepsilon)/\varepsilon)$, our algorithm considers subgraphs G_i induced by edges with x-value in the range $((1+\varepsilon)^{-i}, (1+\varepsilon)^{-i+1}]$, and $\gamma\Delta(G_i) \leqslant \gamma(1+\varepsilon)^i$-edge-colors each such subgraph G_i. It then samples at most γd colors without replacement in each such G_i. The output matching sparsifier H is the union of all these sampled colors. The algorithm's pseudocode is given in Algorithm 15.

Algorithm 15 Edge-Color and Sparsify

```
1: for $i \in \{1, 2, \ldots, \lceil 2\log_{1+\varepsilon}(n/\varepsilon)\rceil\}$ do
2:     let $E_i := \{e \mid x_e \in ((1+\varepsilon)^{-i}, (1+\varepsilon)^{-i+1}]\}$.
3:     compute a $\gamma\lceil(1+\varepsilon)^i\rceil$-edge-coloring $\chi_i$ of $G_i := G[E_i]$.        ▷ Note: $\Delta(G_i) < (1+\varepsilon)^i$
4:     Let $S_i$ be a sample of $\min\{\gamma\lceil d(1+\varepsilon)\rceil, \gamma\lceil(1+\varepsilon)^i\rceil\}$ colors without replacement in $\chi_i$.
5: return $H := (V, \bigcup_i \bigcup_{M \in S_i} M)$.
```

We note that H is the union of few matchings in G, all of size at most $\mu(G)$, by definition, and so H is sparse.

Observation 10.4.1. *The size of H output by Algorithm 15 is at most*

$$|E(H)| = O\left(\frac{\log(n/\varepsilon)}{\varepsilon} \cdot \gamma \cdot d \cdot \mu(G)\right).$$

Remark: The choice of $2\log_{1+\varepsilon}(n/\varepsilon)$ ranges implies that the total x-value of edges not in these ranges (for which $x_e \leqslant \varepsilon^2/n^2$) is at most ε^2. Thus the fractional matching $\vec{x}'$ supported by these G_i has the same approximation ratio as $\vec{x}$, up to $o(\varepsilon)$ terms. Likewise, $\vec{x}'$ preserves the guarantees of fractional matchings $\vec{x}$ studied in Section 10.5.2.

10.4.2 Basic Properties of Algorithm 15

In Section 10.5 we show that running Algorithm 15 on a goof approximate fractional matching $\vec{x}$ yields a subgraph H which is a good matching sparsifier, in the sense that it contains a matching of size $\mu(H) \geqslant \frac{1}{c} \cdot \mu(G)$ for some small c. We refer to this c as the *approximation ratio* of H. Our analysis of the approximation of H relies crucially on the following lemmas of this section.

Throughout our analysis we will focus on the run of Algorithm 15 on some fractional matching $\vec{x}$ with some parameters d, γ and ε, and denote by H the output of this algorithm. For each edge $e \in E$, we let $X_e := \mathbb{1}[e \in H]$ be an indicator random variable for the event that e belongs to this random subgraph H. We first prove that the probability of this event occurring nearly matches p_e given by Equation (10.1) with $\frac{\log n}{\varepsilon^2}$ replaced by d. Indeed, the choice of numbers of colors sampled in each G_i was precisely made with this goal in mind. The proof of the corresponding lemma below, which follows by simple calculation, is deferred to Section 10.6.

Lemma 10.4.2. *If $d \geqslant \frac{1}{\varepsilon}$ and $\gamma \geqslant 1$, then for every edge $e \in E$,*

$$\min\{1, x_e \cdot d\}/(1+\varepsilon)^2 \leqslant \Pr[e \in H] \leqslant \min\{1, x_e \cdot d\} \cdot (1+\varepsilon).$$

Moreover, if $x_e > \frac{1}{d}$, then $\Pr[e \in H] = 1$.

Crucially for our analysis, which bounds weighted vertex degrees, the variables X_e for edges of any vertex are NA.

Lemma 10.4.3. *For any vertex v, the variables $\{X_e \mid e \ni v\}$ are NA.*

To prove this lemma, we rely on the following proposition.

Proposition 10.4.4. *Let $e_1, \ldots, e_n$ be some n elements. For each $i \in [k]$, let X_i be an indicator for element e_i being sampled in a sample of $k \leqslant n$ random elements without replacement from $e_1, \ldots, e_n$. Then $X_1, \ldots, X_n$ are NA.*

Proof. Randomly sampling k elements from $e_1, \ldots, e_n$ without replacement is equivalent to the vector $(X_1, \ldots, X_n)$ taking on all permutations of a $0-1$ vector with k ones, equiprobabily. So, the proposition follows from NA of permutation distributions [171]. □

Proof of Lemma 10.4.3. For all G_i, add a dummy edge to v for each color not used by (non-dummy) edges of v in G_i. Randomly sampling $k = \min\{\lceil \gamma d \rceil, \lceil \gamma \cdot (1+\varepsilon)^i \rceil\}$ colors in the coloring without replacement induces a random sample without replacement of the (dummy and non-dummy) edges of v in G_i. By Proposition 10.4.4, the variables $\{X_e \mid e \ni v, \text{non-dummy } e \in G_i\}$ are NA (since subsets of NA variables are themselves NA). The sampling of colors in the different G_i is independent, and so by closure of NA under independent union, the variables $\{X_e \mid e \ni v\}$ are indeed NA. □

The negative correlation implied by negative association of the variables $\{X_e \mid e \ni v\}$ also implies that conditioning on a given edge $e' \ni v$ being sampled into H only decreases the probability of any other edge $e \ni v$ being sampled into H. So, from lemma 10.4.2 and 10.4.3 we obtain the following.

Corollary 10.4.5. *For any vertex v and edges $e, e' \ni v$,*

$$\Pr[X_e \mid X_{e'}] \leqslant \Pr[X_e] \leqslant \min\{1, x_e \cdot d\} \cdot (1+\varepsilon).$$

Finally, we will need to argue that the negative association of edges incident on any vertex v holds even after conditioning on some edge $e' \ni v$ appearing in H.

Lemma 10.4.6. *For any vertex v and edge $e' \ni v$, the variables $\{[X_e \mid X_{e'}] \mid e \ni v\}$ are NA.*

The proof of Lemma 10.4.6 is essentially the same as Lemma 10.4.3's, noting that if e' is in H, then the unique matching containing e' in the edge coloring of $G_i \ni e'$ must be sampled. Thus, the remaining colors sampled in G_i also constitute a random sample without replacement, albeit a smaller sample from a smaller population (both smaller by one than their unconditional counterparts).

10.4.3 The Dynamic Rounding Framework

Here we present our framework for dynamically rounding fractional matchings.

Key to this framework is Observation 10.4.1, which implies that we can sample H using Algorithm 15 and compute a $(1+\varepsilon)$-approximate matching in H in $O_\varepsilon(\mu(G))$ time. This allows us to (nearly) attain the approximation ratio of this subgraph H dynamically, against an adaptive adversary.

Theorem 10.4.7. *Let $\gamma \geqslant 1$, $d \geqslant 1$ and $\varepsilon > 0$. Let $\mathcal{A}_f$ be a constant-approximate dynamic fractional matching algorithm with update time $T_f(n,m)$. Let $\alpha = \alpha(d, \varepsilon, \gamma, \mathcal{A}_f)$ be the approximation ratio of the subgraph H output by Algorithm 15 with parameters d, ε and γ when run on the fractional matching of $\mathcal{A}_f$. Let $\mathcal{A}_c$ be a dynamic $\gamma\Delta$-edge-coloring algorithm with update time $T_c(n,m)$. If the guarantees of $\mathcal{A}_f$ and $\mathcal{A}_c$ hold against an adaptive adversary, then there exists an $\alpha(1+O(\varepsilon))$-approximate dynamic matching algorithm $\mathcal{A}$ against an* **adaptive** *adversary, with update time*

$$O\left(T_f(n,m) \cdot T_c(n,m) + \log(n/\varepsilon) \cdot \gamma \cdot d/\varepsilon^3\right).$$

Moreover, if $\mathcal{A}_f$ and $\mathcal{A}_c$ have worst-case update times, so does $\mathcal{A}$, and if the approximation ratio given by H is w.h.p., then so is the approximation ratio of $\mathcal{A}$.

This theorem relies on the following simple intermediary lemma, which follows directly from the sparsity of a graphs sampled from $\mathcal{H}$, and known static $O(m/\varepsilon)$-time $(1+\varepsilon)$-approximate matching algorithms [161, 210].

Lemma 10.4.8. *Let $\vec{x}$ be a fractional matching in some graph G. Let $\mathcal{H}$ be the distribution over subgraph H of G obtained by running Algorithm 15 on $\vec{x}$ with parameters d, ε and γ. Then, if the edge colorings of Algorithm 15 based on $\vec{x}$ and the above parameters are given, we can sample a graph $H \sim \mathcal{H}$, and compute a $(1+\varepsilon)$-approximate matching in H, in time*

$$O\left(\frac{\log(n/\varepsilon)}{\varepsilon^2} \cdot \gamma \cdot d \cdot \mu(G)\right).$$

Our algorithm of Theorem 10.4.7 will appeal to Lemma 10.4.8 periodically, "spreading" its across epochs of length $O(\lceil \varepsilon \cdot \mu(G) \rceil)$, as follows.

Proof of Theorem 10.4.7. Algorithm $\mathcal{A}$ runs Algorithm $\mathcal{A}_f$ with which it maintains a fractional matching $\vec{x}$. In addition, it runs $\mathcal{A}_c$ to maintain a $\lceil \gamma(1+\varepsilon)^i \rceil$-edge-colorings in each subgraph $G_i := G[\{e \mid x_e \in (1+\varepsilon)^{-i}, (1+\varepsilon)^{-i+1}\}]$, for all $i = 1, 2, \ldots, 2\log_{1+\varepsilon}(n/\varepsilon) = O\left(\frac{\log(n/\varepsilon)}{\varepsilon}\right)$. Maintaining this fractional matching and the different subgraphs' edge colorings appropriately require at most $O(T_f(n,m) \cdot T_c(n,m))$ time per update: $T_c(n,m)$ time for each of the at most $T_f(n,m)$ edge value changes $\mathcal{A}_f$ makes to the fractional matching $\vec{x}$ per update, as well as $T_f(n,m)$ time to update $\vec{x}$ and $\sum_e x_e$.

By Lemma 10.4.8, the above edge colorings allow us to sample a subgraph H obtained by running Algorithm 15 on $G^{(t)}$, as well as a $(1+\varepsilon)$-approximate matching in H, in time $O\left(\frac{\log(n/\varepsilon)}{\varepsilon^2} \cdot \gamma \cdot d \cdot \mu(G)\right)$. We perform such computations periodically. In particular, we divide time into *epochs* of different lengths (number of updates), starting the first epoch at time zero. Denoting by $G^{(t)}$ and $x^{(t)}$ the graph G and fractional matching $\vec{x}$ at the beginning of epoch t, we spread the work of computing a matching during each epoch, as follows.

If $|x^{(t)}|_1 \leqslant \frac{1}{\varepsilon}$, then epoch t has length one. We sample $H^{(t)} \subseteq G^{(t)}$ and compute a $(1+\varepsilon)$-approximate matching $M^{(t)}$ in $H^{(t)}$ as our matching for epoch t. By Lemma 10.4.8, this takes time

$$O\left(\frac{\log(n/\varepsilon)}{\varepsilon^2} \cdot \gamma \cdot d \cdot \mu(G^{(t)})\right) = O\left(\frac{\log(n/\varepsilon)}{\varepsilon^3} \cdot \gamma \cdot d\right),$$

which is within our claimed time bounds. Moreover, our matching at this point is $\alpha(1+\varepsilon)$-approximate in $G^{(t)}$, as desired.

For an epoch with $|x^{(t)}| > \frac{1}{\varepsilon}$, which we term *long*, we compute $H^{(t)}$ and a $(1+\varepsilon)$-approximate matching $M^{(t)}$ in $H^{(t)}$, but spread this work over the length of the epoch, which we take to be $\lceil \varepsilon \cdot |x^{(t)}|_1 \rceil$. In particular, we use the non-deleted edges of $M^{(t)}$ as our matching for queries during epoch $t+1$. Ignoring the cost of maintaining additional information needed to sample $H^{(t)}$ and $M^{(t)}$ during phase t, these steps increase the update time by

$$\frac{O\left(\frac{\log(n/\varepsilon)}{\varepsilon^2} \cdot \gamma \cdot d \cdot \mu(G^{(t)})\right)}{\lceil \varepsilon \cdot |x^{(t)}|_1 \rceil} = O\left(\frac{\log(n/\varepsilon)}{\varepsilon^3} \cdot \gamma \cdot d\right),$$

since $x^{(t)}$ is a constant-approximate fractional matching, and therefore $|x^{(t)}|_1 \geqslant \Omega(\mu(G^{(t)}))$. Now, in order to perform these operations efficiently during the epoch, we need to maintain the edge colorings *at the beginning of the epoch*. This, however, is easily done by maintaining a mapping (using arrays and lists) from colors in each subgraph to a list of edges added/removed from this color during the epoch. This allows us to maintain $\vec{x}$ and the colorings induced by it, as well as maintain the colorings at the beginning of the epoch, at a constant overhead in the time to update $\vec{x}$ and the colorings, as well as the time to sample $H^{(t)}$. Finally, if space is a concern,[3] the list of updates from epoch t can be removed during epoch $t+1$ at only a constant overhead, due to epochs t and $t+1$ having the same asymptotic length, as we now prove.

To show that if epoch t is long then epoch $t+1$ has the same asymptotic length as epoch t, we note that a long epoch t has length $\lceil \varepsilon \cdot |x^{(t)}|_1 \rceil \leqslant \lceil \frac{3\varepsilon}{2} \cdot \mu(G^{(t)}) \rceil = O(\varepsilon \cdot \mu(G^{(t)}))$, by the integrality gap of the fractional matching polytope. Therefore, the maximum matchings in $G^{(t)}$ and $G^{(t+1)}$ have similar size. In particular, since $|\mu(G^{(t+1)}) - \mu(G^{(t)})| \leqslant O(\varepsilon \cdot \mu(G^{(t)}))$, we have

$$\mu(G^{(t)}) \cdot (1 - O(\varepsilon)) \leqslant \mu(G^{(t+1)}) \leqslant \mu(G^{(t)}) \cdot (1 + O(\varepsilon)). \tag{10.2}$$

On the other hand, since the fractional matchings $x^{(t)}$ and $x^{(t+1)}$ are constant-approximate in $G^{(t)}$ and $G^{(t+1)}$, respectively, then if either epoch t or $t+1$ is long, then both epochs have length $\Theta(\varepsilon \cdot \mu(G^{(t)})) = \Theta(\varepsilon \cdot \mu(G^{(t+1)}))$. We conclude that our algorithm runs within the claimed time bounds. It remains to analyze its approximation ratio for long epochs.

Recall that for a long epoch t, we use the non-deleted edges of some $(1+\varepsilon)$-approximate matching $M^{(t-1)}$ in $H^{(t-1)}$ as our matching during epoch t. (Note that we have finished computing $M^{(t-1)}$ by the beginning of epoch t.) By assumption we have that $\mu(H^{(t-1)}) \geqslant \frac{1}{\alpha} \cdot \mu(G^{(t-1)})$ at the beginning of the epoch. Denote by $M \subseteq M^{(t-1)}$ the non-deleted edges of $M^{(t-1)}$ at some time point in epoch t. As M contains all edges of $M^{(t-1)}$ (which is a $(1+\varepsilon)$-approximate matching in $H^{(t-1)}$), except the edges of $M^{(t-1)}$ removed during epochs $t-1$ and t (of which there are at most $\lceil \varepsilon \cdot |x^{(t-1)}|_1 \rceil + \lceil \varepsilon \cdot |x^{(t)}|_1 \rceil$), we find that the size of M during any point in epoch t is at least

$$\begin{aligned}
&|M^{(t-1)}| - \lceil \varepsilon \cdot |x^{(t-1)}|_1 \rceil - \lceil \varepsilon \cdot |x^{(t)}|_1 \rceil \\
\geqslant & \frac{1}{1+\varepsilon} \cdot \mu(H^{(t-1)}) - \lceil \varepsilon \cdot |x^{(t-1)}|_1 \rceil - \lceil \varepsilon \cdot |x^{(t)}|_1 \rceil \\
\geqslant & \frac{1}{\alpha(1+\varepsilon)} \cdot \mu(G^{(t-1)}) - \left\lceil \frac{3\varepsilon}{2} \cdot \mu(G^{(t-1)}) \right\rceil - \left\lceil \frac{3\varepsilon}{2} \cdot \mu(G^{(t)}) \right\rceil \\
\geqslant & \frac{1}{\alpha(1+O(\varepsilon))} \cdot \mu(G^{(t)}),
\end{aligned}$$

where the third inequality follows from $|x^{(t)}|_1 \leqslant \frac{3}{2} \cdot \mu(G^{(t)})$ for all t, by the aforementioned integrality gap, and the ultimate inequality follows from consecutive epochs' maximum matchings' cardinalities being similar, by Equation (10.2). Therefore, our algorithm is indeed $\alpha(1+O(\varepsilon))$ approximate. □

Remark. A $\log(n/\varepsilon)/\varepsilon$ factor in the above running time is due to the size of $H^{(t)}$ being $|E(H^{(t)})| = O(d \cdot \gamma \cdot \log(n/\varepsilon) \cdot \mu(G)/\varepsilon)$ and the number of subgraphs $G_i^{(t)}$ based on which we sample $H^{(t)}$ being $O(\log(n/\varepsilon)/\varepsilon)$. For some of the fractional matchings we apply our framework

[3] And why wouldn't it be?

to, the sparsifier $H^{(t)}$ has a smaller size of $|E(H^{(t)})| = O(\gamma \cdot d \cdot \mu(G))$, and we only need to sample colors from $O(\gamma \cdot d \cdot \mu(G))$ edge colorings to sample this subgraph. For these fractional matchings the update time of the above algorithm therefore becomes $T_f(n, m) \cdot T_c(n, m) + O(\gamma \cdot d/\varepsilon^2)$.

Theorem 10.4.7 allows us to obtain essentially the same approximation ratio as that of H computed by Algorithm 15 in a static setting, but dynamically, and against an adaptive adversary. The crux of our analysis will therefore be to bound the approximation ratio of H, which we now turn to.

10.5 Analysis of Sparsifiers

In order to analyze the approximation ratio of the subgraph H output by Algorithm 15 (i.e., the ratio $\mu(G)/\mu(H)$), we take two approaches, yielding different (incomparable) guarantees. One natural approach, which we take in Section 10.5.1, shows that Algorithm 15 run on an α-approximate fractional matching outputs a subgraph H which itself contains a fractional matching which is α-approximate in G. For bipartite graphs this implies H contains an α-approximate *integral* matching. For general graphs, however, this only implies the existence of a $\frac{3\alpha}{2}$-approximate integral matching in H, due to the integrality gap of the fractional matching polytope in general graphs. Our second approach, which we take in Section 10.5.2, does not suffer this deterioration in the approximation ratio compared to the fractional matching, for a particular (well-studied) class of fractional matchings.

10.5.1 Fractional Matching Sparsifiers

The approach we apply in this section to analyze Algorithm 15 consists of showing that the subgraph H obtained by running Algorithm 15 on a fractional matching $\vec{x}$ with appropriate choices of d and ε supports a fractional matching $\vec{y}$ with $\mathbb{E}[\sum_e y_e] \geqslant \sum_e x_e(1-O(\varepsilon))$. That is, we prove H is a near-lossless *fractional* matching sparsifier.

Lemma 10.5.1. *(Algorithm 15 Yields Fractional Matching Sparsifiers) Let $\varepsilon \in (0, 1/2)$ and $d \geqslant \frac{4\log(2/\varepsilon)}{\varepsilon^2}$. If H is a subgraph of G output by Algorithm 15 when run on a fractional matching $\vec{x}$ with parameters ε and d as above, then H supports a fractional matching $\vec{y}$ of expected value at least*

$$\mathbb{E}\left[\sum_e y_e\right] \geqslant \sum_e x_e(1-6\varepsilon).$$

Proof. We consider the intermediate assignment of values to edges in H, letting $z_e = \frac{x_e(1-3\varepsilon)}{\min\{1, x_e \cdot d\}} \cdot X_e$. Therefore, by our choice of $\vec{z}$ and by Lemma 10.4.2, each edge e has

$$\mathbb{E}[z_e] = \mathbb{E}[z_e \mid X_e] \cdot \Pr[X_e] \geqslant \frac{x_e(1-3\varepsilon)}{(1+\varepsilon)^2} \geqslant x_e(1-5\varepsilon). \tag{10.3}$$

We now define a random fractional matching $\vec{y}$ such that $\mathbb{E}[y_e] \geqslant \mathbb{E}[z_e \cdot X_e] \cdot (1 - O(\varepsilon)) \geqslant x_e(1 - O(\varepsilon))$, which implies the lemma, by linearity of expectation. In particular, we consider the trivially-feasible fractional matching $\vec{y}$ given by

$$y_e = \begin{cases} 0 & x_e < 1/d \text{ and } \max_{v \in e}(\sum_{e' \ni v} z_{e'}) > 1 \\ z_e & \text{else.} \end{cases}$$

For edges e with $x_e \geqslant \frac{1}{d}$, we always have $y_e = z_e$, so trivially $\mathbb{E}[y_e] = \mathbb{E}[z_e]$. Now, fix an edge $e' = (u, v)$ with $x_{e'} < \frac{1}{d}$. On the one hand, $z_{e'} < \frac{1}{d} < \varepsilon$. On the other hand, by Corollary 10.4.5 and Lemma 10.4.2, any edge $e \ni v$ with $e \neq e'$ has $\Pr[X_e \mid X_{e'}] \leqslant \Pr[X_e] \leqslant \min\{1, x_e \cdot d\} \cdot (1 + \varepsilon)$, and so $\mathbb{E}[z_e \mid X_{e'}] \leqslant x_e(1 - 3\varepsilon)(1 + \varepsilon) \leqslant x_e(1 - 2\varepsilon)$. Consequently,

$$\mathbb{E}\left[\sum_{e \ni v} z_e \,\middle|\, X_{e'}\right] \leqslant \varepsilon + \sum_{e \ni v, e \neq e'} x_e \cdot (1 + \varepsilon) \leqslant 1 - \varepsilon, \tag{10.4}$$

where the last inequality follows from the fractional matching constraint, $\sum_{e \ni v} x_e \leqslant 1$. We now upper bound the probability that this expression deviates so far above its expectation that $\vec{z}$ violates the fractional matching constraint of an endpoint v of e'.

By Lemma 10.4.6, the variables $\{[X_e \mid X_{e'}] \mid e \ni v\}$ are NA. So, by closure of NA under scaling by positive constants, the variables $\{[z_e \mid X_{e'}] \mid e \ni v\}$ are similarly NA. By Lemma 10.4.2, any edge e with $x_e > 1/d$ has $\Pr[X_e] = 1$, and so $\Pr[X_e \mid X_{e'}] = 1$. Thus, the variance of $[z_e \mid X_{e'}]$ is zero. On the other hand, if $x_e \leqslant 1/d$, then $[z_e \mid X_{e'}]$ is a Bernoulli variable scaled by $\frac{1-3\varepsilon}{d}$, with success probability at most $\Pr[X_e \mid X_{e'}] \leqslant \min\{1, x_e \cdot d\} \cdot (1+\varepsilon) = x_e \cdot (1 + \varepsilon)$. Therefore, the variance of this variable is at most

$$\mathrm{Var}([z_e \mid X_{e'}]) \leqslant \left(\frac{1 - 3\varepsilon}{d}\right)^2 \cdot x_e \cdot d \cdot (1 + \varepsilon) \leqslant \frac{x_e}{d}.$$

Summing over all edges $e \ni v$, we have that

$$\mathrm{Var}\left(\sum_{e \ni v} [z_e \mid X_{e'}]\right) \leqslant \sum_{e \ni v} \frac{x_e}{d} \leqslant \frac{1}{d}.$$

Recall that $\mathbb{E}[\sum_{e \ni v} z_e \mid X_{e'}] \leqslant 1 - \varepsilon$, by (10.4). So, for v to have its fractional matching constraint violated by $\vec{z}$ (conditioned on $X_{e'}$), the sum $\sum_{e \ni v} [z_e \mid X_{e'}]$ must deviate from its expectation by at least ε, which in particular requires that the sum of the non-constant variables $[z_e \mid X_{e'}]$ (i.e., for edges $e \ni v$ with $x_e \leqslant \frac{1}{d}$) must deviate from its expectation by ε. So, applying Bernstein's Inequality (Lemma 2.4.12) to the NA variables $\{[z_e \mid X_{e'}] \mid e \ni v,\, x_e \leqslant \frac{1}{d}\}$, each of which has absolute value at most $\frac{1-3\varepsilon}{d} \leqslant \frac{1}{d}$ by definition, we find that the probability that $\vec{z}$

violates the fractional matching constraint of v, conditioned on $X_{e'}$, is at most

$$\begin{aligned}\Pr\left[\sum_{e\ni v} z_e \geqslant 1 \,\middle|\, X_{e'}\right] &\leqslant \Pr\left[\sum_{e\ni v,\, x_e\leqslant \frac{1}{d}} [z_e \mid X_{e'}] \geqslant \sum_{e\ni v,\, x_e\leqslant \frac{1}{d}} [z_e \mid X_{e'}] + \varepsilon\right] \\ &\leqslant \exp\left(-\frac{\varepsilon^2}{2\cdot(1/d+\varepsilon/3d)}\right) \\ &\leqslant \exp\left(-\frac{\varepsilon^2}{4/d}\right),\end{aligned}$$

which is at most $\varepsilon/2$ by our choice of $d \geqslant \frac{4\log(2/\varepsilon)}{\varepsilon^2}$.

By union bound, the probability that $y_e \neq z_e$ (due to $\vec{z}$ violating the fractional matching constraint of an endpoint of e), conditioned on e being sampled, is at most ε. That is, $\Pr[y_e = z_e \mid X_e] \geqslant 1-\varepsilon$. Combined with (10.3), this yields

$$\mathbb{E}[y_e] = \frac{x_e(1-3\varepsilon)}{\min\{1, x_e\cdot d\}}\cdot \Pr[y_e = z_e \mid X_e]\cdot \Pr[X_e] \geqslant (1-\varepsilon)\cdot \mathbb{E}[z_e] \geqslant x_e(1-6\varepsilon).$$

We conclude that the random subgraph H contains a fractional matching of expected value at least $1-6\varepsilon$ times the value of the fractional matching $\vec{x}$ in G. □

It is well known that the integrality gap of the fractional matching polytope is one in bipartite graphs and $\frac{3}{2}$ in general graphs. Therefore, if H admits a fractional matching of value at least $\alpha\cdot\mu(G)$, then H contains an integral matching of value at least $\frac{1}{\alpha}\cdot\mu(G)$ or $\frac{2}{3\alpha}\cdot\mu(G)$ if G is bipartite or general, respectively. Consequently, Lemma 10.5.1 implies the following.

Lemma 10.5.2. *For any $\varepsilon\in(0,1/2)$, Algorithm 15 run with an α-approximate fractional matching and $d \geqslant \frac{4\log(2/\varepsilon)}{\varepsilon^2}$ has approximation ratio $\frac{\alpha}{1-6\varepsilon}$ $(\frac{3\alpha}{2(1-6\varepsilon)})$ in bipartite (general) graphs.*

Plugging the better-than-two approximate fractional matching algorithm of [43] into our dynamic matching framework, we thus obtain the first $(2-\delta)$-approximate algorithms with arbitrarily-small polynomial update time against adaptive adversaries in bipartite graphs, as stated in Theorem 10.1.3.

Remark. We note that in our proof of Lemma 10.5.1 we proved a stronger guarantee, namely that each edge e is assigned in expectation a y-value of at least $\mathbb{E}[y_e] \geqslant x_e(1-6\varepsilon)$. This implies that Lemma 10.5.1 extends to rounding fractional *weighted* matchings, which may prove useful in designing dynamic MWM algorithms.

10.5.2 Integral Matching Sparsifiers

Here we show how to avoid the multiplicative factor of $\frac{3}{2}$ implied by the integrality gap when sparsifying using (particularly well-structured) fractional matchings $\vec{x}$. To prove this improved approximation ratio we generalize the notion of *kernels*, introduced in [48] and later used by

[14, 43]. In particular, we extend this definition to allow for *distributions* over subgraphs, as follows.

Definition 10.5.3. *(Kernels) A (c,d,ε)-*kernel *of a graph G is a (random) subgraph $\mathcal{H}$ of G satisfying:*

1. *For each vertex $v \in V$, the degree of v in $\mathcal{H}$ is at most $d_{\mathcal{H}}(v) \leqslant d$ always.*
2. *For each edge $e \in E$ with $\Pr[e \notin \mathcal{H}] > \varepsilon$, we have $\mathbb{E}[\max_{v\in e} d_{\mathcal{H}}(v) \mid e \notin \mathcal{H}] \geqslant d/c$.*

If $\mathcal{H}$ is a deterministic distribution, we say $\mathcal{H}$ is a deterministic kernel.

Such a graph is clearly sparse, containing at most $O(nd)$ edges. (Crucially for our needs, the kernels we compute even have size $|E(H)| = \tilde{O}(\mu(G))$.) As shown in [14], deterministic $(c,d,0)$-kernels have approximation ratio $2c(1+1/d)$. Generalizing this proof, we show that a randomized (c,d,ε)-kernel has approximation ratio $2c(1+1/d)$ in expectation. The key difference is that now rather than comparing $\mu(G)$ to the value of some fractional matching in $H \sim \mathcal{H}$, we compare $\mu(G)$ to some (randomized) fractional matching's *expected* value.

Lemma 10.5.4. *Let $\mathcal{H}$ be a (c,d,ε)-kernel of G for $c \geqslant \frac{1}{1-\varepsilon}$. Then*

$$\mathbb{E}[\mu(\mathcal{H})] \geqslant \frac{1}{2c(1+1/d)} \cdot \mu(G).$$

Proof. Let M^* be some maximum matching in G (i.e., $|M^*| = \mu(G)$). For any realization H of $\mathcal{H}$, consider the following fractional matching:

$$f^H_{u,v} := \begin{cases} \frac{1}{d} & (u,v) \in H \setminus M^* \\ \max\{1 - \frac{d_h(u)+d_H(v)-2}{d}, 0\} & (u,v) \in H \cap M^*. \end{cases}$$

This is a feasible fractional matching due to the degree bound of H and the fractional values assigned to edges of a vertex v incident on an edge $e \in H \cap M^*$ being at most $\frac{d_H(v)-1}{d} + \frac{d-d_H(v)+1}{d} = 1$. We start by showing that this fractional matching has high expected value, $\mathbb{E}_{H\sim\mathcal{H}}[\sum_e f^H_e]$.

To lower bound the above expected value, we consider the variables $y^H_v := \sum_{e\ni v} f^H_e$. By the handshake lemma, $\sum_{u,v} f^H_{u,v} = \frac{1}{2}\sum_v y^H_v$. Now, consider some edge $e=(u,v) \in M^*$. For any realization H of $\mathcal{H}$ with $e \in M^* \cap H$, we have $y^H_u + y^H_v \geqslant 1 (\geqslant \frac{1}{c})$ by construction. Therefore if $\Pr[e \notin \mathcal{H}] \leqslant \varepsilon$, we have $\mathbb{E}[y^H_u + y^H_v] \geqslant 1-\varepsilon \geqslant \frac{1}{c}$ (by our choice of $c \geqslant \frac{1}{1-\varepsilon}$). On the other hand, if $e \in M^* \setminus H$, then we have $y^H_u + y^H_v \geqslant \max_{v\in e} y^H_v \geqslant \max_{v\in e} d_H(v)/d$. But by the second property of (c,d,ε)-kernels we have that if $\Pr[e \notin \mathcal{H}] > \varepsilon$, then $\mathbb{E}_{H\sim\mathcal{H}}[\max_{v\in e} d_H(v) \mid e \notin H] \geqslant d/c$. Consequently, for each edge $e = (u,v) \in M^*$ with $\Pr[e \notin \mathcal{H}] > \varepsilon$ we have that

$$\mathbb{E}_{H\sim\mathcal{H}}\left[y^H_u + y^H_v\right] \geqslant \frac{1}{c} \cdot \Pr[e \in H] + \frac{d}{c} \cdot \frac{1}{d} \cdot \Pr[e \notin H] = \frac{1}{c}.$$

Now, as each vertex v neighbors at most one edge of the (optimal) matching M^*, we obtain

$$\mathbb{E}_{H\sim\mathcal{H}}\left[\sum_e f_e^H\right] = \frac{1}{2}\cdot\mathbb{E}_{H\sim\mathcal{H}}\left[\sum_v y_v^H\right] \geqslant \frac{1}{2c}\cdot|M^*| = \frac{1}{2c}\cdot\mu(G). \tag{10.5}$$

So, $\mathcal{H}$ contains a large fractional matching in expectation.

To show that $\mathcal{H}$ contains a large *integral* matching in expectation, we again consider a realization H of $\mathcal{H}$, and now construct a multigraph on the same vertex set V, with each edge e replaced by $f_e^H \cdot d$ parallel copies (note that $f_e^H \cdot d$ is integral). By construction, the number of edges in this multigraph is $\sum_e f_e^H \cdot d$. By feasibility of f^H, this multigraph has maximum degree at most $\max_v \sum_{e\ni v} f_v^H \cdot d \leqslant d$. By Vizing's Theorem [261], the simple subgraph obtained by ignoring parallel edges corresponding to edges in $H\cap M^*$ can be $(d+1)$-edge colored. But for each edge $e=(u,v)\in H\cap M^*$, such a coloring uses at most $d_H(u)-1+d_H(v)-1$ distinct colors on edges other than (u,v) which are incident on u or v. To extend this $d+1$ edge coloring to a proper coloring of the multigraph, we color the $\max\{d-(d_H(u)-1+d_H(v)-1),0\}$ multiple edges (u,v) in this multigraph using some $\max\{d-(d_H(u)-1+d_H(v)-1),0\}$ colors of the palette of size $d+1$ which were not used on the other edges incident on u and v. We conclude that this multigraph, whose edges are contained in H and which has $\sum_e f_e\cdot d$ edges, is $(d+1)$-edge-colorable. Consequently, one of these $d+1$ colors (matchings) in this edge coloring is an integral matching in H of size at least

$$\mu(H) \geqslant \frac{1}{d+1} = \frac{1}{1+1/d}\cdot\sum_e f_e^H. \tag{10.6}$$

Taking expectation over $H\sim\mathcal{H}$ and combining (10.6) with (10.5), the lemma follows. □

As we show, the subgraph H output by Algorithm 15, when run on well-structured fractional matchings, contains such a kernel. Specifically, we show that H contains a kernel, provided the input fractional matching is *approximately maximal*, as defined by Arar et al. [14].

Definition 10.5.5. *A fractional matching $\vec{x}$ is (c,d)-*maximal *if every edge $e\in E$ either has fractional value $x_e > 1/d$ or it has one endpoint v with $\sum_{e\ni v} x_e \geqslant 1/c$ with all edges e' incident on this v having value $x_{e'}\leqslant 1/d$.*

As shown by Arar et al., sampling each edge e of a (c,d)-maximal fractional matching independently with probability $\min\{1, x_e\cdot d\}$ for sufficiently large $d=\Omega_\varepsilon(\log n)$ yields a deterministic $(c(1+O(\varepsilon), d(1+O(\varepsilon),0)$-kernel w.h.p. As we show, sampling each edge e with probability roughly as above, such that the indicator variables for edges to be sampled are NA, as in Algorithm 15, yields the same kind of kernel, w.h.p.

Lemma 10.5.6. *Let $c\geqslant 1$, $\varepsilon>0$ and $d\geqslant \frac{9c(1+\varepsilon)^2\cdot\log n}{\varepsilon^2}$. If $\vec{x}$ is a (c,d)-maximal fractional matching, then the subgraph H output by Algorithm 15 when run on $\vec{x}$ with ε and d is a deterministic $(c(1+O(\varepsilon), d(1+O(\varepsilon),0)$-kernel, w.h.p.*

Proof. Consider some vertex v. By Lemma 10.4.2, we have that each edge $e \ni v$ is sampled with probability at most $\Pr[X_e = 1] \leqslant \min\{1, x_e \cdot d\} \cdot (1+\varepsilon)$. Combined with the fractional matching constraint, $\sum_{e \ni v} x_e \leqslant 1$, this implies that the expected degree of v in H is at most

$$\mathbb{E}[d_H(v)] = \sum_{e \ni v} \mathbb{E}[X_e] \leqslant \sum_{e} x_e \cdot d \cdot (1+\varepsilon) \leqslant d(1+\varepsilon).$$

By Lemma 10.4.3, we have that the indicators $\{X_e \mid e \ni v\}$ are NA. Therefore, appealing to the upper tail bound of Lemma 2.4.10 for NA variables with $\delta = \varepsilon \in (0,1)$, we have that, since $d \geqslant \frac{9 \log n}{\varepsilon^2}$,

$$\Pr[d_H(v) \geqslant d(1+3\varepsilon)] \leqslant \Pr[d_H(v) \geqslant d(1+\varepsilon)^2] \leqslant \exp\left(\frac{-\varepsilon^2 d(1+\varepsilon)}{3}\right) \leqslant \frac{1}{n^3}.$$

Now, to prove the second property of kernels, consider some edge $e \in E$ such that $\Pr[e \notin H] > 0$. By Lemma 10.4.2, we have that $x_e \leqslant 1/d$. Therefore, as $\vec{x}$ is (c,d)-maximal, this implies that there exists some $v \in e$ with $\sum_{e' \ni v} x_{e'} \geqslant \frac{1}{c}$ and $x_{e'} \leqslant \frac{1}{d}$ for all $e' \ni v$. Therefore, by Lemma 10.4.2 each edge $e' \ni v$ is sampled with probability at least $\Pr[X_e] \geqslant x_e \cdot d/(1+\varepsilon)^2$. So, by linearity of expectation, the expected degree of v in H is at least

$$\mathbb{E}[d_H(v)] = \sum_{e \ni v} \mathbb{E}[X_e] \geqslant \sum_{e} x_e \cdot d/(1+\varepsilon)^2 \geqslant d/(c(1+\varepsilon)^2).$$

Recalling that the indicators $\{X_e \mid e \ni v\}$ are NA, we appeal to the lower tail bound of Lemma 2.4.10 with $\delta = \varepsilon > 0$, from which we obtain that, since $d \geqslant \frac{9c(1+\varepsilon)^2 \log n}{\varepsilon^2}$,

$$\Pr[d_H(v) \leqslant d(1-\varepsilon)/(c(1+\varepsilon)^2)] \leqslant \exp\left(\frac{-\varepsilon^2 d/(c(1+\varepsilon)^2)}{2}\right) \leqslant \frac{1}{n^3}.$$

Taking union bound over the $O(n^2)$ bad events which would make H not be kernel as desired, we find that H is a $(c(1+O(\varepsilon)), d(1+O(\varepsilon), 0)$-kernel w.h.p., as claimed. $\square$

Indeed, even taking d to be an appropriately-chosen constant yields (randomized) kernels, as we show in the following lemma, proved in Section 10.7.

Lemma 10.5.7. *Let $\varepsilon \in (0, 1/4)$, $c \geqslant \frac{1}{1-\varepsilon}$ and $d \geqslant \frac{4 \cdot \log(2/\varepsilon)}{\varepsilon^2}$. Let $\mathcal{H}$ be the distribution of subgraphs output by Algorithm 15 when run on a (c,d)-approximately maximal fractional matching $\vec{x}$ with ε and d as above. For any realization H of $\mathcal{H}$, we let H' be a graph obtained by removing all edges of vertices v of degree $d_H(v) > d(1+4\varepsilon)$. Then the distribution $\mathcal{H}'$ over H' is a $(c(1+O(\varepsilon)), d(1+4\varepsilon), \varepsilon)$-kernel.*

In light of lemmas 10.5.6 and 10.5.7, we now turn to discussing implications of Theorem 10.4.7.

Fast Worst-Case Algorithms: As shown in [14], the output fractional matching of [45] is $(1+\varepsilon, d)$-approximately-fractional, for some $d = \text{poly}(\log n, 1/\varepsilon)$ large enough to satisfy the conditions of Lemma 10.5.6. Therefore, plugging in this $\mathrm{poly}(\log n, 1/\varepsilon)$ worst-case update time deterministic algorithm into Theorem 10.4.7 in conjunction with the deterministic $O(\log n)$-time 2Δ-edge-coloring algorithm of [48], we obtain a Monte Carlo algorithm with guarantees similar to that of Theorem 10.1.1. Moreover, since we can verify in $O(|E(H)|)$ time the high-probability events implying that H is a kernel (broadly, we need only check whether any vertex has degree above d while sampling H, and verify that all vertices v of expected degree at least d/c have such a degree), we can re-sample H if ever it is not a kernel. Thus we obtain a Las Vegas randomized dynamic $(2+\varepsilon)$-approximate matching algorithm with $\mathrm{poly}\log n$ update time w.h.p, which works against adaptive adversaries, as stated in Theorem 10.1.1.

Constant-Time Algorithms: To obtain the constant-time algorithm of Theorem 10.1.2, we rely on the constant-time fractional matching algorithm of Bhattacharya and Kulkarni [41], which we show outputs a $(1+\varepsilon, d)$-maximal matching for any $d > 1+\varepsilon$ (see Section 10.8). Therefore, by Lemma 10.5.7, plugging this algorithm into the algorithm of Theorem 10.4.7 immediately yields a logarithmic-time $(2+\varepsilon)$-approximate algorithm against adaptive adversaries. Pleasingly, we can improve this bound further, and obtain a constant-time such algorithm. For this improvement, we show that the fractional matchings of [41] only define $O(\mu(G))$ subgraphs G_i, as they only assign one of $O(\mu(G))$ x-values to all edges. This implies in particular that Algorithm 15 can sample H from such $\vec{x}$ using only $O(\gamma \cdot d \cdot \mu(G))$ random choices (saving a factor of $\log n$), yielding a subgraph of expected size $O(d \cdot \sum_e x_e) = O(\gamma \cdot d \cdot \mu(G))$ (where the last inequality follows from the constant integrality gap of the fractional matching polytope). Using a simple constant-expected-time 3Δ-edge-coloring algorithm, this improves the update time to $\mathrm{poly}(1/\varepsilon)+O(\gamma \cdot d/\varepsilon)$. From the above we thus obtain the first constant-time $(2+\varepsilon)$-approximate algorithm against adaptive adversaries, as stated in Theorem 10.1.2.

10.6 Sampling Probabilities

Here we show that Algorithm 15 samples each edge into H with the probability given by (10.1) with $\frac{\log n}{\varepsilon^2}$ replaced by d, up to multiplicative $(1+\varepsilon)$ terms.

Lemma 10.4.2. *If $d \geqslant \frac{1}{\varepsilon}$ and $\gamma \geqslant 1$, then for every edge $e \in E$,*

$$\min\{1, x_e \cdot d\}/(1+\varepsilon)^2 \leqslant \Pr[e \in H] \leqslant \min\{1, x_e \cdot d\} \cdot (1+\varepsilon).$$

Moreover, if $x_e > \frac{1}{d}$, then $\Pr[e \in H] = 1$.

Proof. Let i be the integer for which $x_e \in ((1+\varepsilon)^{-i}, (1+\varepsilon)^{-i+1}]$. That is, the i for which $e \in E(G_i)$.

If $(1+\varepsilon)^{i-1} < d$, implying that $(1+\varepsilon)^i < d(1+\varepsilon)$, then Algorithm 15 samples all of the $\gamma\lceil(1+\varepsilon)^i\rceil = \min\{\gamma\lceil d(1+\varepsilon)\rceil, \gamma\lceil(1+\varepsilon)^i\rceil\}$ colors in the edge coloring of G_i. Consequently, the edge e is sampled with probability one. On the other hand, $(1+\varepsilon)^{i-1} < d$ also implies that $(1+\varepsilon)^{-i+1} > \frac{1}{d}$ and therefore that $x_e > (1+\varepsilon)^{-i} \geqslant \frac{1}{d(1+\varepsilon)}$. Thus, the edge e is sampled with probability at most

$$\Pr[e \in H] = 1 \leqslant \min\{1, x_e \cdot d\} \cdot (1+\varepsilon),$$

and trivially sampled with probability at least

$$\Pr[e \in H] = 1 \geqslant \min\{1, x_e \cdot d\}/(1+\varepsilon)^2.$$

Moreover, if $x_e > \frac{1}{d}$, then $(1+\varepsilon)^{-i+1} \geqslant x_e > \frac{1}{d}$, or put otherwise $(1+\varepsilon)^{i-1} < d$, and so we find that every edge e with $x_e > \frac{1}{d}$ is sampled with probability $\Pr[e \in H] = 1 (= \min\{1, x_e \cdot d\})$. It remains to consider edges e with $x_e \leqslant \frac{1}{d}$, for which $\min\{1, x_e \cdot d\} = x_e \cdot d$, and which in particular belong to subgraphs G_i with i satisfying $(1+\varepsilon)^{i-1} \geqslant d$.

Now, if i satisfies $(1+\varepsilon)^{i-1} \geqslant d$, then we sample some $\gamma\lceil d\rceil = \min\{\gamma\lceil d\rceil, \lceil\gamma \cdot (1+\varepsilon)^i\rceil\}$ colors in the edge coloring of G_i. As such, the probability of e appearing in H is precisely the probability that the color M containing e is one of the $\gamma\lceil d\rceil$ sampled colors in G_i, which by linearity of expectation happens with probability precisely

$$\Pr[e \in H] = \frac{\gamma\lceil d\rceil}{\gamma\lceil(1+\varepsilon)^i\rceil} = \frac{\lceil d\rceil}{\lceil(1+\varepsilon)^i\rceil}.$$

Now, since $d \geqslant \frac{1}{\varepsilon}$ implies that $d+1 \leqslant d(1+\varepsilon)$, the probability of e (which has $x_e \geqslant (1+\varepsilon)^{-i}$) appearing in H is at most

$$\Pr[e \in H] = \frac{\lceil d\rceil}{\lceil(1+\varepsilon)^i\rceil} \leqslant \frac{d+1}{(1+\varepsilon)^i} \leqslant \frac{d(1+\varepsilon)}{(1+\varepsilon)^i} \leqslant x_e \cdot d \cdot (1+\varepsilon).$$

On the other hand, since $(1+\varepsilon)^{i-1} \geqslant d$, and $d \geqslant \frac{1}{\varepsilon}$, we have that $(1+\varepsilon)^i \geqslant d \geqslant \frac{1}{\varepsilon}$, which implies that $(1+\varepsilon)^i + 1 \leqslant (1+\varepsilon)^{i+1}$. Consequently, the probability of e (which has $x_e \leqslant (1+\varepsilon)^{-i+1}$) appearing in H is at least

$$\Pr[e \in H] = \frac{\lceil d\rceil}{\lceil(1+\varepsilon)^i\rceil} \geqslant \frac{d}{(1+\varepsilon)^i + 1} \geqslant \frac{d}{(1+\varepsilon)^{i+1}} \geqslant \frac{x_e \cdot d}{(1+\varepsilon)^2}.$$

This completes the proof for edge e in $E(G_i)$ for i satisfying $(1+\varepsilon)^{i-1} \geqslant d$, as such edges e satisfy $(1+\varepsilon)^{-i+1} \leqslant \frac{1}{d}$ and consequently $\min\{1, x_e \cdot d\} = x_e \cdot d$. □

10.7 Randomized Kernels

In this section we show that running Algorithm 15 with $d = 1/\operatorname{poly}(\varepsilon)$ on a (c,d)-maximal fractional matching, and removing all edges of high-degree vertices in the output graph, yields a randomized kernel.

Lemma 10.5.7. *Let $\varepsilon \in (0, 1/4)$, $c \geqslant \frac{1}{1-\varepsilon}$ and $d \geqslant \frac{4\cdot\log(2/\varepsilon)}{\varepsilon^2}$. Let $\mathcal{H}$ be the distribution of subgraphs output by Algorithm 15 when run on a (c,d)-approximately maximal fractional matching $\vec{x}$ with ε and d as above. For any realization H of $\mathcal{H}$, we let H' be a graph obtained by removing all edges of vertices v of degree $d_H(v) > d(1+4\varepsilon)$. Then the distribution $\mathcal{H}'$ over H' is a $(c(1+O(\varepsilon)), d(1+4\varepsilon), \varepsilon)$-kernel.*

Proof. The fact that $\mathcal{H}'$ satisfies the first property of such a kernel is immediate, as we remove all edges of vertices of degree above $d(1+4\varepsilon)$ in H to obtain H'. The meat of the proof is dedicated to proving the second property.

Fix an edge e with $\Pr[e \notin \mathcal{H}'] > \varepsilon$. By Lemma 10.4.2 together with the fractional matching constraint ($\sum_{e' \ni v} x_{e'} \leqslant 1$), the expected $\mathcal{H}$-degree of any vertex $v \in e$ is at most

$$\mathbb{E}[d_{\mathcal{H}}(v)] = \sum_{e \ni v} \mathbb{E}[X_e] \leqslant \sum_{e} x_e \cdot d(1+\varepsilon) \leqslant d(1+\varepsilon).$$

Now, by Lemma 10.4.3, $d_{\mathcal{H}}(v) = \sum_{e' \ni v} X_{e'}$ is the sum of NA variables. So, by the upper tail bound of Lemma 2.4.10 with $\delta = \varepsilon$, combined with $d \geqslant \frac{4\log(2/\varepsilon)}{\varepsilon^2}$ and $\varepsilon \leqslant \frac{1}{2} \leqslant 1$, we find that

$$\Pr[d_{\mathcal{H}}(v) > d(1+4\varepsilon)] \leqslant \Pr\left[d_{\mathcal{H}}(v) \geqslant d(1+\varepsilon)^2\right] \leqslant \exp\left(\frac{-\varepsilon^2 \cdot d(1+\varepsilon)}{2}\right) \leqslant \varepsilon^2/2.$$

Therefore, by union bound, since $e = (u, v) \in H \setminus H'$ only if one (or both) of its endpoints have degree above $d(1+4\varepsilon)$ in H, we find that

$$\Pr[e \in H \setminus H'] \leqslant \sum_{v \in e} \Pr[d_{\mathcal{H}}(u) > d(1+4\varepsilon)] \leqslant \varepsilon^2. \tag{10.7}$$

By Equation (10.7), we have that

$$\Pr[e \notin H] = \Pr[e \notin H'] - \Pr[e \in H \setminus H'] \geqslant \Pr[e \notin H'] - \varepsilon^2.$$

Combining the above with $\Pr[e \notin H'] \geqslant \varepsilon$, we find that

$$\Pr[e \notin H] \geqslant \Pr[e \notin H'] \cdot (1-\varepsilon). \tag{10.8}$$

In what follows we use Equation (10.8) to prove the second property of kernels, namely, that for any edge e with $\Pr[e \notin \mathcal{H}'] > \varepsilon$, we have $\mathbb{E}[\max_{v \in e} d_{\mathcal{H}'}(v) \mid e \notin \mathcal{H}'] \geqslant \frac{d}{c}(1 - o(1))$.

By the law of total expectation, and since $d_{H'}(v) = 0$ if $e \in H \backslash H'$, we have that $\mathbb{E}[\max_v d_{H'}(v) \mid e \notin H']$ is equal to

$$\mathbb{E}[\max_v d_{H'}(v) \mid e \notin H] \cdot \Pr[e \notin H \mid e \notin H'],$$

which by Equation (10.8) implies

$$\mathbb{E}[\max_v d_{H'}(v) \mid e \notin H'] \geqslant \mathbb{E}[\max_v d_{H'}(v) \mid e \notin H] \cdot (1-\varepsilon). \tag{10.9}$$

We now turn to lower bounding $\mathbb{E}[\max_v d_{H'}(v) \mid e \notin H]$.

By Equation (10.8), we have that $\Pr[e \notin H] > \varepsilon \cdot (1-\varepsilon) > 0$. Therefore, by Lemma 10.4.2, $x_e \leqslant 1/d$. But then, by the (c,d)-approximate-maximality of $\vec{x}$, edge e contains a vertex v satisfying the following.

$$\sum_{e' \ni v} x_{e'} \geqslant 1/c. \tag{10.10}$$

$$x_{e'} \leqslant 1/d \qquad \forall e' \ni v. \tag{10.11}$$

We fix this v for the remainder of the proof, and turn to proving a lower bound on $\mathbb{E}[d_{H'}(v) \mid e \notin H]$, which by Equation (10.9) would imply the desired second property of kernels.

For notational simplicity, denote by Ω the probability space obtained by conditioning on the event $\overline{X_e} = [e \notin \mathcal{H}]$, or in other words, conditioning on the color of e in the edge coloring of G_i with $e \in E(G_i)$ not being sampled. (Recall that we use X_e as an indicator for $e \in H$.) First, this conditioning preserves the fact that colors in the different graphs are sampled without replacement—with colors in G_i not containing e sampled from a slightly smaller population. Consequently, Lemma 10.4.3 and Corollary 10.4.5, as well as Lemma 10.4.6, which only relied on colors being sampled without replacement and independently in the different graphs, hold for the probability space Ω. That is, we have the following.

$$\Pr_{\Omega}[X_{e'} \mid X_{e''}] \leqslant \Pr_{\Omega}[X_{e'}] \qquad \forall e', e'' : e' \cap e'' \neq \emptyset \tag{10.12}$$

$$\{[X_{e'} \mid X_{e''}, Xe] \mid e \ni v\} \text{ are NA}, \qquad \forall v \in V,\, e', e'' \ni v. \tag{10.13}$$

We now show that edges' sampling probabilities are hardly affected by conditioning on $e \notin H$. To this end, we note that this conditioning only affects the sampling probability of edges in the graph $G_i = G[\{e' \mid e' \in ((1+\varepsilon)^{-i}, (1+\varepsilon)^{-i+1}]\}]$ containing e. Now, since $(1+\varepsilon)^{-i} < x_e \leqslant 1/d$, we have that $(1+\varepsilon)^i \geqslant d$, and therefore the number of colors in the coloring of G_i is $\gamma \cdot \lceil (1+\varepsilon)^i \rceil \geqslant d$. Therefore, the sampling probability of edges $e' \in E(G_i)$ increases under conditioning on Ω by a multiplicative factor of at most

$$\frac{d}{d-1} \leqslant 1 + \varepsilon,$$

due to our choice of $d \geqslant \frac{4\log(2/\varepsilon)}{\varepsilon^2}$ and $\varepsilon \leqslant \frac{1}{2}$. From the above and Lemma 10.4.2 we conclude that for all edges $e' \in E(G_i) \setminus \{e\}$,

$$\Pr_{\Omega}[X_{e'}] \leqslant \Pr[X_{e'}] \cdot (1+\varepsilon) \leqslant d \cdot x_{e'} \cdot (1+\varepsilon)^2 \qquad \forall e' \neq e. \tag{10.14}$$

On the other hand, all colors other than that containing e have their probability of being sampled increase. In particular, we also have that

$$\Pr_{\Omega}[X_{e'}] \geqslant \Pr[X_{e'}] \qquad \forall e' : e' \cap e = \emptyset. \tag{10.15}$$

We now return to considering the vertex $v \in e$ satisfying (10.10) and (10.11), and we fix an edge (u, v). By Equation (10.12) and Equation (10.14), together with the fractional matching constraint $\sum_{e' \ni v} x'_{e'} \leqslant 1$, conditioned on the edge (u, v) appearing in H, the neighbor u has expected degree in H at most

$$\mathbb{E}_{\Omega}[d_H(u) \mid X_{(u,v)}] = \sum_{e' \ni u} \mathbb{E}_{\Omega}[X_{e'} \mid X_{(u,v)}] \leqslant 1 + \sum_{e' \ni u} x'_{e'} \cdot d \cdot (1+\varepsilon)^2 \leqslant 1 + d(1+\varepsilon)^2.$$

We recall that $[d_H(u) \mid X_{(u,v)}, X_e] = \sum_{e' \ni u}[X_{e'} \mid X_{(u,v)}, X_e]$ is the sum of NA variables, by (10.13). So, by the upper tail bound of Lemma 2.4.10 with $\delta = \varepsilon \in (0,1)$, we have that

$$\begin{aligned}\Pr_{\Omega}[d_H(u) > d(1+4\varepsilon) \mid X_{(u,v)}] &\leqslant \Pr_{\Omega}[d_H(u) \geqslant (1 + d(1+\varepsilon)^2)\cdot(1+\varepsilon) \mid X_{(u,v)}] \\ &\leqslant \exp\left(\frac{-\varepsilon^2(1+d(1+\varepsilon)^2)}{3}\right) \\ &\leqslant \varepsilon/2,\end{aligned}$$

where we relied on $d \geqslant \frac{4\log(2/\varepsilon)}{\varepsilon^2}$ and $\varepsilon \leqslant 1/4$. Denoting by B_u the bad event that u has more than $d(1+4\varepsilon)$ edges in H, we have that $\Pr_{\Omega}[B_u \mid X_{(u,v)}] \leqslant \varepsilon/2$. Analogously, we have that $\Pr_{\Omega}[B_v \mid X_{(u,v)}] \leqslant \varepsilon/2$.

Since each edge (u,v) in H is also in H' only if both B_u and B_v do not happen, the degree of v in H' is at least $d_{H'}(v) \geqslant \sum_{(u,v)} X_{(u,v)} \cdot (1 - \mathbb{1}[B_u] - \mathbb{1}[B_v])$. Now, by Equation (10.11), all edges $e' = (u,v)$ have $x'_{e'} \leqslant \frac{1}{d}$. Therefore, by Equation (10.15) and Lemma 10.4.2, these edges are sampled with probability $\Pr_{\Omega}[X_{e'}] \geqslant \Pr[X_{e'}] \geqslant x_{e'} \cdot d/(1+\varepsilon)^2$. So, since $\sum_{e' \ni v} x_{e'} \geqslant \frac{1}{c}$ by Equation (10.10), the expected degree of v in H' conditioned on $e \notin \mathcal{H}$, is at least

$$\begin{aligned}\mathbb{E}[d_{H'}(v) \mid e \notin \mathcal{H}] = \mathbb{E}_{\Omega}[d_{H'}(v)] &\sum_{(u,v)\neq e} \Pr_{\Omega}[X_{(u,v)}] \cdot (1 - \Pr_{\Omega}[B_u \mid X_{(u,v)}] - \Pr_{\Omega}[B_v \mid X_{(u,v)}]) \\ &\geqslant \sum_{\substack{e' \ni v \\ e' \neq e}} \left(x'_{e'} \cdot d/(1+\varepsilon)^2\right) \cdot (1-\varepsilon) \\ &\geqslant \left(\frac{1}{c} - \frac{1}{d}\right) \cdot (d/(1+\varepsilon)^2) \cdot (1-\varepsilon) \\ &\geqslant \frac{d(1+4\varepsilon)}{c(1+O(\varepsilon))},\end{aligned}$$

where the last inequality relied on $c \geqslant \frac{1}{1-\varepsilon}$, on $d \geqslant \frac{4\log(2/\varepsilon)}{\varepsilon^2}$, and $\varepsilon \leqslant \frac{1}{4}$.

To conclude, we have that $d_{H'}(v) \leqslant d(1+4\varepsilon)$ for every vertex v with probability one, while each edge e with $\Pr[e \notin \mathcal{H}'] > \varepsilon$, satisfies

$$\begin{aligned}\mathbb{E}[\max_{v \in e} d_{H'}(v) \mid e \notin \mathcal{H}'] &\geqslant \mathbb{E}[\max_{v\in e} d_{H'}(v) \mid e \notin \mathcal{H}] \cdot (1-\varepsilon) \\ &\geqslant d(1+4\varepsilon)/c(1+O(\varepsilon)).\end{aligned}$$

Thus, $\mathcal{H}'$ is a $(c(1+O(\varepsilon)), d(1+4\varepsilon), \varepsilon)$-kernel, as claimed, and the lemma follows. □

10.8 Constant-Time Algorithms

In order to obtain a constant-time algorithm using Lemma 10.5.7, we need in particular some approximately-maximal fractional matching algorithm with constant update time. As it so happens, the algorithm of Bhattacharya and Kulkarni [41] is precisely such an algorithm. As the structure of the fractional matching output by this algorithm will prove useful in several ways for our analysis, we take a moment to outline this fractional matching's structure.

We say a dynamic fractional matching algorithm maintains a (β, c)-*hierarchical partition* if it assigns each vertex v a *level* ℓ_v, and each edge e an $x-$value $x_e = \beta^{-\ell_e}$, where $\ell_e = \max_{v\in e}\{\ell_v\} \pm O(1)$, for some constant β. The second property this fractional matching must guarantee is that each vertex v with $\ell_v > 0$ has $\sum_{e\ni v} x_e \geqslant 1/c$. Most prior dynamic fractional matching algorithms [41, 44, 45, 47, 143], including that of [41], follow this approach, originally introduced by [47].

We now show that the fractional matching of [41] is approximately-maximal.

Lemma 10.8.1. *For all $\varepsilon > 0$, $d > 1+\varepsilon$, there exists a deterministic $(1+\varepsilon, d)$-maximal fractional matching algorithm with amortized update time $O(1/\varepsilon^2)$.*

Proof. The algorithm we consider is precisely that of [41]. As the update time of this algorithm was proven in [41], it remains only to prove that it outputs an approximately-maximal fractional matchings as stated.

The algorithm of Bhattacharya and Kulkarni [41] maintains a $((1+\varepsilon), (1+\varepsilon))$-hierarchical partition with $x_e = (1+\varepsilon)^{-\max_{v\in e}\{\ell_v\}-1}$. For such a partition, we have that for any value $d \geqslant 1+\varepsilon$, any edge e with $x_e \leqslant \frac{1}{d}$ must have an endpoint $v \in e$ of level $\ell_v \geqslant \log_{1+\varepsilon}(d) - 1$. But then all other incident edges $e' \ni v$ have x-value at most $x_{e'} \leqslant (1+\varepsilon)^{-\ell_v - 1} \leqslant \frac{1}{d}$. Moreover, since the level of v is at least $\ell_v \geqslant \log_{1+\varepsilon}(d) - 1 > 0$ (by our choice of $d > 1+\varepsilon$), we also have that $\sum_{e'\ni v} x_{e'} \geqslant \frac{1}{c}$. In other words, the fractional matching $\vec{x}$ output by the algorithm of [41] is $(1+\varepsilon, d)$-maximal. □

Lemmas 10.5.7 and 10.8.1 together with Theorem 10.4.7 imply a $(2+\varepsilon)$-approximate dynamic algorithm with logarithmic update time against adaptive adversaries. We now explain how to obtain such an approximation in *constant* time.

We note that any (β, c)-hierarchical partition must have at most $O(c \cdot \mu(G))$ vertices v of level $\ell_v > 0$. To see this, recall that all such vertices have $\sum_{e\ni v} x_e \geqslant 1/c$. Therefore,

$$\sum_{e\in E} x_e \geqslant \frac{1}{2} \sum_{v:\,\ell_v>0} \sum_{e\ni v} x_e \geqslant \frac{1}{2c} \cdot |\{v \mid \ell_v > 0\}.$$

But since the integrality gap of the fractional matching polytope is at most $\frac{3}{2}$, we also have that

$$\frac{3}{2} \cdot \mu(G) \geqslant \sum_{e\in E} x_e \geqslant \frac{1}{2c} \cdot |\{v \mid \ell_v > 0\}|.$$

That is, for constant c as we consider, the number of vertices of level $\ell_v > 0$ is at most $O(\mu(G))$. This implies in particular that there are only $O(\mu(G))$ distinct levels assigned to vertices. But an edge's value is determined by the level of its highest-level endpoint. Therefore, as there are only $O(\mu(G))$ many values $\max_{v\in e}\{\ell_v\}$ can take, we find that there are only $O(\mu(G))$ values any x_e can take. Hence, when running Algorithm 15 on $\vec{x}$ we only sample edges from $O(\mu(G))$ edge colorings of subgraphs G_i (which are induced by edges of similar x_e value). Thus, if we sample $d = \text{poly}(1/\varepsilon)$ colors per (non-empty) subgraph G_i, the choice of colors to sample can be done in

$O(\mu(G)/\operatorname{poly}(\varepsilon))$ time, yielding a graph of expected size $\mathbb{E}[|E(H)|] \leqslant \sum_e d \cdot x_e \leqslant d \cdot \frac{3}{2} \cdot \mu(G) = O(\mu(G)/\operatorname{poly}(\varepsilon))$. Extending the argument of Theorem 10.4.7 appropriately, using a 3Δ-edge-coloring algorithm with constant expected update time and the fractional matching algorithm of [41], together with Lemma 10.5.7, we obtain a $(2+\varepsilon)$-approximate dynamic algorithm with *constant* update time. Thus, we obtain Theorem 10.1.2.

10.9 Conclusion and Open Questions

This chapter provides the first randomized dynamic matching algorithms which work against adaptive adversaries and outperform deterministic algorithms for this problem. We obtain these results by leveraging a new framework we introduce for rounding fractional matchings dynamically against an adaptive adversary. Our work suggests several follow-up directions, of which we state a few below.

More Applications: A natural direction is to find more applications of our rounding framework. Recently, Bernstein et al. [40] applied our framework to a new decremental fractional matching algorithm to obtain a $(1+\varepsilon)$-approximate decremental matching algorithm for bipartite graphs in $\operatorname{poly}(\log n, 1/\varepsilon)$ amortized time (against adaptive adversaries). Are there more applications of our framework?

Maximum Weight Matching (MWM): The current best approximation for dynamic MWM with polylog worst-case update time against adaptive adversaries is $(4+\varepsilon)$, obtained by applying the reduction of [253] to our algorithm of Theorem 10.1.1. Indeed, even with amortization or the assumption of an oblivious adversary, no approximation below $(4+\varepsilon)$ is known to be achievable in sub-polynomial time. This is far from the ratios of 2 or $(2+\varepsilon)$ achievable efficiently for MWM in other models of computation, such as streaming [127, 232] and the CONGEST model of distributed computation [128, 199]. Attaining such bounds dynamically in polylog update time (even amortized and against an oblivious adversary) remains a tantalizing open problem.

Better Approximation: To date, no efficient (i.e., polylog update time) dynamic matching algorithm with approximation better than two is known. As pointed out by Assadi et al. [18], efficiently improving on this ratio of two for maximum matching has been a longstanding open problem in many models, and is known to be impossible to do in an online setting, as shown in Chapter 3. Is the dynamic setting "easier" than the online setting, or is an approximation ratio of 2 the best approximation achievable in polylog update time?

Chapter 11

Streaming Submodular Matching

How does this chapter differ from all other chapters (in this thesis)? In previous chapters, we mostly focused on online algorithms. In this chapter, we will consider *streaming* algorithms. In previous chapters, when studying matchings, our objective was to maximize the cardinality, or more generally the weight of a matching. In this chapter, based on joint work with Roie Levin [197], we study even more expressive, *submodular* objectives.

11.1 Background

Submodular functions are set functions which capture the notion of *diminishing returns*. The study of (approximately) maximizing such functions has a long history. For example, it has been known since the 70s that the greedy algorithm yields an $e/(e-1) \approx 1.582$ approximation for monotone submodular maximization subject to a cardinality constraint [226]. This is optimal among polytime algorithms with value oracle access [225], or assuming standard complexity-theoretic conjectures [81, 102, 204]. The same problem for *non-monotone* submodular functions is harder; it is hard to approximate to within a 2.037 factor [229]. Much work has been dedicated to improving the achievable approximation [51, 54, 93, 108, 142, 193, 229, 264]; the best currently stands at 2.597 [51].

Closer to the focus of this chapter is the study of submodular maximization subject to *matching* constraints, i.e., maximum submodular matching (MSM). For this problem, the greedy algorithm has long been known to be 3-approximate for monotone functions [114]. Improved approximations have since been obtained [109, 142, 192, 194], with the current best being $(2+\varepsilon)$ and $(4+\varepsilon)$ for monotone and non-monotone MSM respectively [109]. The papers above studied rich families of constraints (e.g. matroid intersection, matchoids, exchange systems), some of which were motivated explicitly by matching constraints (see [109]). Beyond theoretical interest, the MSM problem also has great practical appeal, since many natural objectives exhibit diminishing returns behavior. Applications across different fields include: machine translation [198], Internet advertising [80, 189], combinatorial auctions more broadly [57, 195, 263], and numerous other matching problem where the goal is a submodular notion of utility such as diversity [7, 8].

The proliferation of big-data applications such as those mentioned above has spurred a surge of interest in algorithms for the regime where the input is too large to even store in local memory.

To this end, it is common to formulate problems in the *streaming* model. Here the input is presented element-by-element to an algorithm that is restricted to use $\tilde{O}(S)$ memory, where S is the maximum size of any feasible solution. We study MSM in this model.

For our problem when the objective is *linear*, a line of work [73, 95, 104, 127, 205, 232] has shown that a $(2+\varepsilon)$-approximation is possible in the streaming model [127, 232]. Meanwhile, for submodular objectives under *cardinality* constraints (which are a special case of MSM in complete bipartite graphs), a separate line of work [9, 20, 65, 111, 180, 214] has culminated in the same $(2+\varepsilon)$ approximation ratio, for both monotone and non-monotone functions (the latter taking exponential time, as is to be expected from the lower bound of [229]); moreover, this $(2+\varepsilon)$ bound was recently proven to be tight [9, 112, 227]. On the other hand, for fully general MSM, the gap between known upper and lower bounds remain frustratingly large. Chakrabarti and Kale [58] gave a 7.75-approximate algorithm for MSM with monotone functions. For non-monotone functions, Chekuri et al. [65] gave a $(12+\varepsilon)$-approximate algorithm, later improved by Feldman et al. [111] to $5+2\sqrt{6} \approx 9.899$. The only known lower bound for monotone MSM is $\frac{e}{e-1} \approx 1.582$ for streaming or polytime algorithms, implied respectively by [174] and [102, 225]. For non-monotone functions, [229] implies a hardness of 2.037. Closing these gaps, especially from the algorithmic side, seems to require new ideas.

11.2 Our Contributions

We present a number of improved results for streaming maximum submodular matching (MSM) and related problems.

Our first result is an improvement on the 7.75 approximation of [58] for monotone MSM.

Theorem 11.2.1. *There exists a deterministic linear-time streaming* MSM *algorithm for monotone functions which is* $3+2\sqrt{2} \approx 5.828$ *approximate.*

Our algorithm extends in various ways: First, it yields the same approximation ratio for submodular b-*matchings*, where each node v can be matched b_v times, improving on the previous best 8-approximations [65, 111]. For the special case of linear functions (MWM), our algorithm—with appropriate parameters—recovers the $(2+\varepsilon)$-approximate algorithm of [232]. For weighted b-matching (MWbM), a slight modification of our algorithm yields a $(3+\varepsilon)$-approximate algorithm, improving on the previous best $(4+\varepsilon)$-approximation [73].

Next, we improve on the $5+2\sqrt{6} \approx 9.899$ approximation of [111] for non-monotone MSM.

Theorem 11.2.2. *There exists a randomized linear-time streaming* MSM *algorithm for non-monotone functions which is* $4+2\sqrt{3} \approx 7.464$ *approximate.*

Our non-monotone MSM algorithm's approximation ratio is better than the previous state-of-the-art 7.75-approximate *monotone* MSM algorithm [58]. Moreover, when applied to *monotone* functions, the algorithm of Theorem 11.2.2 yields the same approximation ratio as the deterministic algorithm of Theorem 11.2.1.

We turn to proving hardness for monotone MSM. As stated before, the previous best lower bounds for this problem were $\frac{e}{e-1} \approx 1.582$. These lower bounds applied to either space-bounded [174] or time-bounded algorithms [102, 225]. We show that the problem becomes harder for algorithms which are both space bounded and time bounded. This answers an open problem posed in the Bertinoro Workshop on Sublinear Algorithms 2014 [1], at least for time bounded algorithms.[1]

Theorem 11.2.3. *No polytime streaming* MSM *algorithm for monotone functions is better than* 1.914 *approximate.*

Finally, to demonstrate that our techniques have the potential for wider applicability, we also use them to provide an alternative and unified proof of the results of Chakrabarti and Kale [58] and Feldman et al. [111] for MSM.

11.2.1 Our Techniques and Overview

Our starting point is the breakthrough result of Paz and Schwartzman [232] for a special case of our problem—maximum weight matching (MWM). They gave a $(2+\varepsilon)$-approximate streaming algorithm by extending the local-ratio technique [26]. Subsequently, Ghaffari and Wajc [127] simplified and slightly improved the analysis of [232], by re-interpreting their algorithm in terms of the primal-dual method.[2] The primal-dual method is ubiquitous in the context of approximating linear objectives. In this chapter, we show that this method is also useful in the context of streaming submodular optimization, where to the best of our knowledge, it has not yet been used. For our primal-dual analysis, we rely on the *concave-closure extension* for submodular functions which has a "configuration LP"-like formulation. In particular, using this extension, we find that a natural generalization of the MWM algorithm of [232] (described in Section 11.4) yields improved bounds for monotone MSM and its generalization to b-matchings. Our primal-dual analysis is robust in the sense that it allows for extensions and generalizations, as we now outline.

Our approach in a nutshell (Sections 11.4+11.5). Our approach is to keep monotone dual solutions (initially zero), and whenever an edge arrives, discard it if its dual constraint is already satisfied. Edges whose dual constraint is not satisfied are added to a stack S, and relevant dual variables are increased, so as to satisfy their dual constraint. Finally, we unwind the stack S, constructing a matching M greedily. The intuition here is that the latter edge in the stack incident on a common edge have higher marginal gain than earlier such edges in the stack. More formally, we show that this matching M has value at least some constant times the dual objective cost. Weak LP duality and the choice of LP imply that $f(M) \geqslant \frac{1}{\alpha} \cdot f(S \cup OPT)$ for some $\alpha > 1$, which implies our algorithm is α-approximate for monotone MSM.

[1]We note briefly that such a bound does not follow from space lower for cardinality constrained submodular maximization [112, 227] in a stream (a special case of our setting, with a complete bipartite graph on n and k nodes), since a bound for that problem cannot be superlinear in n.

[2]A form of equivalence between local-ratio and primal-dual was established in [27], but not for the extension of the local ratio technique given in [232].

Extension 1 (Section 11.6). Extending our approach, which gives $f(M) \geqslant \frac{1}{\alpha} \cdot f(S \cup OPT)$, to non-monotone functions f seems challenging, since for such functions $f(S \cup OPT)$ can be arbitrarily smaller than $f(OPT)$. To overcome this challenge, we note that our dual updates over-satisfy dual constraints of edges in S. We can therefore afford to randomly discard edges whose dual is not satisfied on arrival (and not add them to S), resulting in these edges' dual constraints holding *in expectation*. This allows us to argue, via a generalization of the randomized primal-dual method of Devanur et al. [79] (on which we elaborate in Section 11.3), that $\mathbb{E}[f(M)] \geqslant \frac{1}{\alpha} \cdot \mathbb{E}[f(S \cup OPT)]$. As S contains each element with probability at most some q, a classic lemma of [54] allows us to show that $\mathbb{E}[f(S \cup OPT)] \geqslant (1-q) \cdot f(OPT)$, from which we get our results for non-monotone MSM.[3] Given the wide success of the randomized-primal dual method of [79] in recent years [97, 143, 162–166, 256], we believe that our extension of this method in the context of submodular optimization will likely find other applications.

Extension 2 (Section 11.7). For maximum weight *b-matching* (MWbM), the dual updates when adding an edge to the stack are not high enough to satisfy this edge's dual constraint. However, since we do cover each edge *outside* the stack S, weak duality implies that a maximum-weight b-matching M in the stack S has value at least as high as $f(M) \geqslant \frac{1}{2+\varepsilon} \cdot f(OPT \setminus S)$, and trivially at least as high as $f(M) \geqslant f(OPT \cap S)$. Combining these lower bounds on $f(M)$ imply our improved $(3+\varepsilon)$ approximation ratio for MWbM. This general approach seems fairly general, and could find uses for other sub-additive objectives subject to downward-closed constraints.

Unifying Prior Work (Section 11.8). To demonstrate the usefulness of our primal-dual analysis, we also show that this (randomized) primal-dual approach gives an alternative, unified way to analyze the MSM algorithms of [58, 111].

Lower bound (Section 11.9). Our lower bound instance makes use of two sources of hardness: computational hardness under ETH ([81, 102]) and information-theoretic hardness resulting form the algorithm not knowing the contents or order of the stream in advance ([132]). In particular, our proof embeds a submodular problem (specifically, set cover) in parts of the linear instance of [132], and hence exploits the submodularity in the MSM objective. Interestingly, our lower bound of 1.914 is higher than any convex combination of the previous hardness results we make use of, both of which imply a lower bound no higher than $e/(e-1)$.

11.3 Preliminaries

A set function $f : 2^N \to \mathbb{R}$ is *submodular* if the marginal gains of adding elements to sets, denoted by $f_S(e) := f(S \cup \{e\}) - f(S)$, satisfy $f_S(e) \geqslant f_T(e)$ for $e \notin T$ and $S \subseteq T \subseteq N$. We say f is *monotone* if $f(S) \leqslant f(T)$ for all $S \subseteq T \subseteq N$. Throughout this chapter we require only oracle access to the submodular function. The maximum submodular matching (MSM) problem is defined by a non-negative submodular function $f : 2^E \to \mathbb{R}_{\geqslant 0}$, where E is the edge-

[3] Incidentally, for monotone functions, for which $E[f(M)] \geqslant \frac{1}{\alpha} \cdot E[f(S \cup OPT)] \geqslant \frac{1}{\alpha} \cdot f(OPT)$, this algorithm is α approximate. This is somewhat surprising, as this algorithm runs an α-approximate monotone algorithm (and this analysis is tight, by Section 11.10) on a random q-fraction of the input, suggesting an α/q approximation. Nonetheless, we show that for q not too small in terms of α, we retain the same approximation ratio even after this sub-sampling.

set of some n-node graph $G = (V, E)$, and feasible sets are matchings in G. The more general maximum submodular b-matching (MSbM) problem, has as feasible sets subgraphs in which the degree of each vertex v does not exceed b_v, for some input vector $\vec{b}$. Our objective is to design algorithms with low approximation ratio $\alpha \geqslant 1$, that is algorithms producing solutions M such that $\mathbb{E}[f(M)] \geqslant \frac{1}{\alpha} \cdot f(OPT)$ for the smallest possible value of α.

For *streaming* MSM, edges of E are presented one at a time, and we are tasked with computing a matching in G at the end of the stream, using little memory. Since any matching's size is at most $n/2$, we restrict our algorithms to using the bare minimum space, $\tilde{O}(n)$ (while the entire graph can have size $\Omega(n^2)$). On a technical note, we will only allow the algorithm to query the value oracle for f on subsets currently stored in memory. As is standard (see e.g., [139, 269]), we assume the range of f is polynomially bounded. More precisely, we assume that $\frac{\max_{e,S} f_S(e)}{\min_{e,S} f_S(e)} = n^{O(1)}$, where the max and min are taken over e, S for which $f_S(e) \neq 0$. This implies in particular that we can store values of the form $f_S(e)$ using $O(\log n)$ bits.

Useful Notation: Throughout this chapter we will rely on the following notation. First, we denote by $e^{(1)}, e^{(2)}, \ldots,$ the edges in the stream, in order. For edges $e = e^{(i)}, e' = e^{(j)}$, we write $e < e'$ if and only if $i < j$, i.e., if e arrived before e'. Similarly to [65, 111], we will also use $f(e : S) := f_{S \cap \{e' < e\}}(e)$ as shorthand for the marginal gain from adding e to the set of elements which arrived before e in S. One simple yet useful property of this notation is that $\sum_{e \in S} f(e : S) = f(S)$ ([65, Lemma 1].) Other properties of this notation we will make use of, both easy consequences of submodularity, are $f(e : S) \leqslant f_S(e)$, as well as monotonicity of $f(e : S)$ in S, i.e., $f(e : A) \geqslant f(e : B)$ for $A \subseteq B$.

11.3.1 The Primal-Dual Method in Our Setting

As discussed in Section 11.2.1, the main workhorse of our algorithms is the primal-dual method. In this method, we consider some linear program (LP) relaxation, and its dual LP. We then design an algorithm which computes a (primal) solution of value P, and a feasible solution of value D, and show that $P \geqslant \frac{1}{\alpha} \cdot D$, which implies an approximation ratio of α, by weak duality, since

$$P \geqslant \frac{1}{\alpha} \cdot D \geqslant \frac{1}{\alpha} \cdot f(OPT).$$

For linear objectives, the first step of the primal-dual method—obtaining an LP relaxation—is often direct: write some integer linear program for the problem and drop the integrality constraints. For submodular objective functions, which are only naturally defined over vertices of the hypercube, $\vec{x} \in \{0, 1\}^E$, and are not defined over fractional points $\vec{x} \in [0, 1]^E \setminus \{0, 1\}^E$, the first step of defining a relaxation usually requires extending f to real vectors. For this, we use the *concave closure* (see e.g.[262] for a survey of its history and further properties).

Definition 11.3.1. *The* concave closure $f^+ : [0,1]^E \to \mathbb{R}$ *of a set function* $f : 2^E \to \mathbb{R}$ *is given by*

$$f^+(\vec{x}) := \max\left\{ \sum_{T\subseteq E} \alpha_T \cdot f(T) \;\middle|\; \sum_{T\subseteq E} \alpha_T = 1,\ \alpha_T \geqslant 0\ \forall T \subseteq E,\ \sum_{T\ni e} \alpha_T = x_e\ \forall e \in E \right\}.$$

In words, the concave closure is the maximum expected f-value of a random subset $T \subseteq E$, where the maximum is taken over all distributions matching the marginal probabilities given by $\vec{x}$. This is indeed an extension of set functions (and in particular submodular functions) to real-valued vectors, as this distribution must be deterministic for all $\vec{x} \in \{0,1\}^E$. Consequently, for any set $P \subseteq [0,1]^E$ containing the characteristic vector $\vec{x}_{OPT}$ of an optimal solution OPT, we have that $\max_{\vec{x}^* \in P} f^+(x) \geqslant f^+(x_{OPT}) = f(OPT)$.

Now, to define an LP relaxation for submodular maximization of some function g subject to some linear constraints $A\vec{x} \leqslant \vec{c}$, we simply consider $\max\left\{g^+(\vec{x}) \mid A\vec{x} \leqslant \vec{c}\right\}$. For MSbM, we obtain the primal and dual programs given in Figure 11.1.

Primal (P)		Dual (D)	
max	$\sum_{T\subseteq E} \alpha_T \cdot g(T)$	min	$\mu + \sum_{v\in V} b_v \cdot \phi_v$
subject to		subject to	
$\forall T \subseteq E:$	$\sum_{T\ni e} \alpha_T = x_e$	$\forall T \subseteq E:$	$\mu + \sum_{e\in T} \lambda_e \geqslant g(T)$
	$\sum_{T\subseteq E} \alpha_T = 1$	$\forall e \in E:$	$\sum_{v\in e} \phi_v \geqslant \lambda_e$
$\forall v \in V:$	$\sum_{e\ni v} x_e \leqslant b_v$		
$\forall e \in E, T \subseteq E:$	$x_e, \alpha_T \geqslant 0$	$\forall v \in V:$	$\phi_v \geqslant 0$

Figure 11.1: The LP relaxation of the MSbM problem and its dual

11.3.2 Non-Montone MSM: Extending the Randomized Primal-Dual Method

To go from monotone to non-monotone function maximization, we make use of our dual updates resulting in dual solutions which over-satisfy (some) dual constraints. This allows us to randomly sub-sample edges with probability q when deciding whether to insert them into S, and still have a dual solution which is feasible *in expectation* over the choice of S. This is akin to the randomized primal-dual method of Devanur et al. [79], who introduced this technique in the context of maximum *cardinality* and *weighted* matching. However, unlike in [79] (and subsequent work [97, 143, 162–166, 166, 256]), for our problem the LP is not fixed. Specifically, we consider a different submodular function in our LP based on S, denoted by $g^S(T) := f(T \cup S)$. This results in *random* primal and dual LPs, depending on the random set S. We show that our (randomized) dual solution is feasible for the obtained (randomized) dual LP in expectation over S. Consequently, our expected solution's value is at least as high as some multiple of an expected solution to the dual LP, implying

$$\mathbb{E}_S[f(M)] \geqslant \frac{1}{\alpha} \cdot \mathbb{E}_S[D] \geqslant \frac{1}{\alpha} \cdot \mathbb{E}_S[f(S \cup OPT)]. \tag{11.1}$$

Equation (11.1) retrieves our bound for monotone functions, as these satisfy $\mathbb{E}_S[f(S \cup OPT)] \geqslant f(OPT)$. To obtain bounds for non-monotone functions, we show that $\mathbb{E}_S[f(S \cup OPT)] \geqslant (1-q) \cdot f(OPT)$, by relying on the following lemma, due to Buchbinder et al. [54, Lemma 2.2].

Lemma 11.3.2. *Let $h : 2^N \to R_{\geqslant 0}$ be a non-negative submodular function, and let B be a random subset of N containing every element of N with probability at most q (not necessarily independently), then $\mathbb{E}[h(B)] \geqslant (1-q) \cdot h(\emptyset)$.*

11.4 Our Basic Algorithm

In this section we describe our monotone submodular b-matching algorithm, which we will use with slight modifications and different parameter choices in coming sections.

The algorithm maintains a stack of edges S, initially empty, as well as vertex potentials $\vec{\phi} \in \mathbb{R}^{|V|}$. When an edge e arrives, we compare the marginal value of this arriving edge with respect to the stack to the sum of vertex potentials of the edge's endpoints times a slack parameter C. If $C \cdot \sum_{v \in e} \phi_v$ is larger, we continue to the next edge. Otherwise, with probability q we add the edge to the stack and increment the endpoint vertex potentials. At the end of the stream, we construct a b-matching greedily by unwinding the stack *in reverse order*. The pseudocode is given in Algorithm 16.

Algorithm 16 The MSbM Algorithm

```
    Initialization
 1: $S \leftarrow$ emptystack
 2: $\forall v \in V : \phi_v^{(0)} \leftarrow 0$
    Loop
 3: for $t \in \{1, \dots, |E|\}$ do
 4:     $e \leftarrow e^{(t)}$
 5:     $\forall v \in V : \phi_v^{(t)} \leftarrow \phi_v^{(t-1)}$
 6:     if $C \cdot \sum_{v \in e} \phi_v^{(t-1)} \geqslant f(e : S)$ then
 7:         continue                    ▷ skip edge $e$
 8:     else
 9:         with probability $q$ do
10:             $S.push(e)$
11:             for $v \in e$ do
12:                 $w_{ev} \leftarrow \frac{f(e:S) - \sum_{v \in e} \phi_v^{(t-1)}}{b_v}$
13:             for $v \in e$ do
14:                 $\phi_v^{(t)} \leftarrow \phi_v^{(t-1)} + w_{ev}$

    Post-Processing
15: $M \leftarrow \emptyset$
16: while $S \neq$ emptystack do
17:     $e \leftarrow S.pop()$
18:     if $|M \cap N(e)| < b_v$ for all $v \in e$ then
19:         $M \leftarrow M \cup \{e\}$
20: return $M$
```

Algorithm 16 clearly outputs a feasible b-matching. In subsequent sections we analyze this algorithm for various instantiations of the parameters C and q. Before doing so, we show that this algorithm when run with $C = 1 + \Omega(1)$ is indeed a streaming algorithm, and in particular uses space $\tilde{O}(\sum_v b_v)$.

Lemma 11.4.1. *For any constant $\varepsilon > 0$, Algorithm 16 run with $C = 1 + \varepsilon$ uses $\tilde{O}(\sum_v b_v)$ space.*

The proof broadly relies on the observation that every edge incident on vertex v inserted to the stack increases ϕ_v by a multiplicative factor of $(C-1)/b_v$, coupled with the fact that the minimum and maximum non-zero values which ϕ_v can take are polynomially bounded in each other, due to f being polynomially bounded. See Section 11.11 for a proof.

We further note that as Algorithm 16 only evaluates f a constant number of times per edge arrival, followed by an algorithm with time $O(|S|) \leqslant O(|E|)$, this algorithm runs in time linear in $|E|$, times the time to evaluate f.

Lemma 11.4.2. *Algorithm 16 requires $O(1)$ operations and function evaluations per arrival, followed by $O(|E|)$ time post-processing.*

11.5 Monotone MSbM

In this section we will consider a *deterministic* instantiation of Algorithm 16 (specifically, we will set $q = 1$) in the context of monotone submodular b-matching.

To argue about the approximation ratio, we will fit a dual solution to this algorithm. Define the auxiliary submodular functions $g^S : 2^E \to \mathbb{R}^+$ to be $g^S(T) := f(S \cup T)$. We will work with the dual LP (D) for the function g^S, and consider the following dual solution.

$$\mu := f(S) = g^S(\emptyset),$$
$$\phi_v := C \cdot \phi_v^{(|E|)}$$
$$\lambda_e := \begin{cases} f(e : S) & e \notin S \\ 0 & e \in S. \end{cases}$$

We start by showing that the above is indeed dual feasible.

Lemma 11.5.1. *The dual solution $(\mu, \vec{\phi}, \vec{\lambda})$ is feasible for the LP (D) with function g^S.*

Proof. To see that the first set of constraints are satisfied, note that by submodularity of f

$$\sum_{e \in T} \lambda_e = \sum_{e \in T \setminus S} f(e : S) \geqslant \sum_{e \in T \setminus S} f_S(e) \geqslant f_S(T \setminus S) = f(S \cup T) - f(S) = g^S(T) - \mu.$$

For the second set of constraints, note that an edge $e = e^{(t)}$ is not added to the stack if and only if the check at Line 6 fails. Therefore, since $\phi_v^{(t)}$ values increase monotonically with t, we have

$$\sum_{v\in e}\phi_v = C\cdot\sum_{v\in e}\phi_v^{(|E|)} \geqslant C\cdot\sum_{v\in e}\phi_v^{(t-1)} \geqslant f(e:S) = \lambda_e. \qquad \square$$

It remains to relate the value of the solution M to the cost of this dual. We first prove an auxiliary relationship that will be useful:

Lemma 11.5.2. *The* b*-matching* M *output by Algorithm 16 satisfies*

$$f(M) \geqslant \frac{1}{2}\cdot\sum_{e\in S}\sum_{v\in e} b_v\cdot w_{ev}.$$

Proof. We first note that for any edge $e = e^{(t)}$ and $v \in e$, since $\phi_v^{(t-1)} = \sum_{e'\ni v, e'<e} w_e$, we have that

$$f(e:S) = b_v\cdot w_{ev} + \sum_{u\in e}\phi_u^{(t-1)} \geqslant b_v\cdot w_{ev} + \phi_v^{(t-1)} = b_v\cdot w_{ev} + \sum_{\substack{e'\ni v\\ e'<e}} w_{e'v}.$$

Combined with submodularity of f, the above yields the following lower bound on $f(M)$,

$$f(M) = \sum_{e\in M} f(e:M) \geqslant \sum_{e\in M} f(e:S) \geqslant \sum_{e\in M}\sum_{v\in e}\Big(b_v\cdot w_{ev} + \sum_{\substack{e'\ni v\\ e'<e}} w_{e'v}\Big).$$

On the other hand, the greedy manner in which we construct M implies that any edge $e' \in S\setminus M$ must have at least one endpoint v with b_v edges $e > e'$ in M. Consequently, the term $w_{e'v}$ for such e and v is summed b_v times in the above lower bound for $f(M)$. On the other hand, $b_v\cdot w_{ev} = b_u\cdot w_{eu}$ for $e = (u,v)$, by definition. From the above we obtain our desired inequality.

$$f(M) \geqslant \sum_{e\in M}\sum_{v\in e} b_v\cdot w_{ev} + \frac{1}{2}\cdot\sum_{e\in S\setminus M}\sum_{v\in e} b_v\cdot w_{ev} \geqslant \frac{1}{2}\cdot\sum_{e\in S}\sum_{v\in e} b_v\cdot w_{ev}. \qquad \square$$

We can now bound the two terms in the dual objective separately with respect to the primal, using the following two corollaries of Lemma 11.5.2.

Lemma 11.5.3. *The* b*-matching* M *output by Algorithm 16 satisfies* $f(M) \geqslant \frac{1}{2C}\sum_{v\in V} b_v\cdot\phi_v$.

Proof. Since $\phi_v = C\cdot\phi_v^{(|E|)}$, and $w_{ev} = \phi_v^{(t)} - \phi_v^{(t-1)}$ for all $v \in e = e^{(t)}$, Lemma 11.5.2 implies that

$$f(M) \geqslant \frac{1}{2}\cdot\sum_{e\in S}\sum_{v\in e} b_v\cdot w_{ev} = \frac{1}{2}\cdot\sum_{v\in V}\sum_{t=1}^{|E|} b_v\cdot\big(\phi_v^{(t)} - \phi_v^{(t-1)}\big) = \frac{1}{2}\cdot\sum_{v\in V} b_v\cdot\phi_v^{(|E|)} = \frac{1}{2C}\cdot\sum_{v\in V} b_v\cdot\phi_v.$$

$\square$

Lemma 11.5.4. *The b-matching M output by Algorithm 16 satisfies $f(M) \geqslant \left(1 - \frac{1}{C}\right)\mu$.*

Proof. We note that $w_e > 0$ for an edge $e = e^{(t)}$ if and only if $f(e : S) \geqslant C \cdot \sum_{v \in e} \phi_v^{(t-1)}$. Hence,

$$b_v \cdot w_{ev} = f(e : S) - \sum_{v \in e} \phi_v^{(t-1)} \geqslant \left(1 - \frac{1}{C}\right) \cdot f(e : S).$$

Combining the above with Lemma 11.5.2, and again recalling that for $e = (u, v)$, we have that $b_v \cdot w_{ev} = b_u \cdot w_{eu}$, by definition, we obtain the desired inequality.

$$f(M) \geqslant \frac{1}{2} \cdot \sum_{e \in S} \sum_{v \in e} b_v \cdot w_{ev} \geqslant \left(1 - \frac{1}{C}\right) \sum_{e \in S} f(e : S) = \left(1 - \frac{1}{C}\right) f(S). \qquad \square$$

Combining the above two corollaries and Lemma 11.5.1 with LP duality, we can now analyze the algorithm's approximation ratio.

Theorem 11.5.5. *Algorithm 16 run with $q = 1$ and C on a monotone* MSbM *instance outputs a b-matching M of value*

$$\left(2C + \frac{C}{C-1}\right) \cdot f(M) \geqslant f(\mathrm{OPT}).$$

This is optimized by taking $C = 1 + \frac{1}{\sqrt{2}}$, which yields a $3 + 2\sqrt{2} \approx 5.828$ approximation.

Proof. By weak LP duality and Lemma 11.5.1, together with monotonicity of f, we have that

$$C \cdot \sum_v b_v \cdot \phi_v + \mu \geqslant \max_T g^S(T) = \max_T f(S \cup T) \geqslant f(S \cup \mathrm{OPT}) \geqslant f(\mathrm{OPT}).$$

Combining Lemma 11.5.3 and Lemma 11.5.4 and rearranging, we get the desired inequality,

$$\left(2C + \frac{C}{C-1}\right) \cdot f(M) \geqslant C \cdot \sum_v b_v \cdot \phi_v + \mu \geqslant f(\mathrm{OPT}). \qquad \square$$

In Section 11.10 we show that our analysis of Algorithm 16 is tight.

We note that our analysis of this section required monotonicity, as we lower bounded $f(M)$ by (a multiple of) $f(S \cup OPT) \geqslant f(OPT)$, where the last step crucially relies on monotonicity. In the next section, we show how the use of randomness (namely, setting $q \neq 1$) allows us to obtain new results for *non-monotone* MSM.

11.6 Non-Monotone MSM

In this section we consider MSM (so, $b_v = 1$ for all v in this section), for *non-monotone* functions.

To extend our results to non-monotone MSM, we make use of the freedom to choose $q \notin \{0, 1\}$, resulting in a randomized algorithm. This will allow us to lower bound $\mathbb{E}_S[f(S \cup OPT)]$ in terms of $f(OPT)$. But first, we show that for appropriately chosen q, the output matching M has high value compared to $\mathbb{E}_S[f(S \cup OPT)]$. The analysis of this fact will follow the same outline of Section 11.5, relying on LP duality, but with a twist.

For our dual fitting, we use the same dual solution as in Section 11.5. However, this time this dual solution will only be feasible *in expectation*, in the following sense. Since we now have $q \notin \{0, 1\}$, Algorithm 16 is now a randomized algorithm, S is a random set, g^S is a random submodular function, and thus (D) is a random LP. Let $\mathbb{E}[(D)]$ denote this LP, which is (D) with the submodular function $g(T) := \mathbb{E}_S[g^S(T)]$. We now show that our dual solution's expectation is feasible for $\mathbb{E}[(D)]$.

Lemma 11.6.1. *For $q \in [1/(2C+1), 1/2]$, the expected dual solution $(\mathbb{E}[\mu], \mathbb{E}[\vec{\phi}], \mathbb{E}[\vec{\lambda}])$ is feasible for the expected LP $\mathbb{E}[(D)]$.*

Proof. The first set of constraints is satisfied for any realization of the randomness. Indeed, as in the proof of Lemma 11.5.1, for any realization of S, by submodularity of f, we have

$$\sum_{e \in T} \lambda_e = \sum_{e \in T \setminus S} f(e : S) \geqslant \sum_{e \in T \setminus S} f_S(e) \geqslant f_S(T \setminus S) = f(S \cup T) - f(S) = g^S(T) - \mu.$$

Consequently, taking expectation over S, we have that indeed, $\mathbb{E}_S[\mu] + \sum_{e \in T} \mathbb{E}_S[\lambda_e] \geqslant \mathbb{E}_S[g^S(T)]$. We now tun to proving the second set of constraints, which will only hold in expectation.

Fix an edge $e = e^{(t)}$, and define the event $A_e := [f(e : S) \leqslant C \cdot \sum_{v \in V} \phi_v^{(t-1)}]$. Then, by definition of A_e and monotonicity of $\phi_v^{(t)}$ in t, we have that

$$\mathbb{E}\left[\sum_{v \in e} \phi_v \,\middle|\, A_e\right] \geqslant \mathbb{E}\left[C \cdot \sum_{v \in e} \phi_v^{(t-1)} \,\middle|\, A_e\right] \geqslant \mathbb{E}[f(e : S) \mid A_e] = \mathbb{E}[\lambda_e \mid A_e]. \qquad (11.2)$$

We now prove the same inequality holds when conditioning on the complement, $\overline{A_e}$.

Fix a realization of the randomness R for which $\overline{A_e}$ holds. Then, $e = e^{(t)}$ fails the test in Line 6, and so with probability q, we have $\sum_{v \in e} \phi_v^{(t)} = \sum_{v \in e} (\phi_v^{(t-1)} + w_e) = 2 \cdot f(e : S) - \sum_{v \in e} \phi_v^{(t-1)}$, and with probability $(1-q)$, we have $\sum_{v \in e} \phi_v^{(t)} = \sum_{v \in e} \phi_v^{(t-1)}$. Hence, in this case, as $q \leqslant \frac{1}{2}$, we have

$$\mathbb{E}\left[\sum_{v \in e} \phi_v^{(t)} \,\middle|\, R\right] = 2q \cdot f(e : S) + (1 - 2q) \cdot \sum_{v \in e} \phi_v^{(t-1)} \geqslant 2q \cdot f(e : S).$$

Now, since $\phi_v \geqslant C \cdot \phi^{(t)}$, and $q \geqslant 1/(2C+1)$ and since λ_e is set to $f(e:S)$ if e is not added to S (with probability $1-q$) and set to zero otherwise, the above implies that

$$\mathbb{E}\left[\sum_{v \in e} \phi_v \,\middle|\, R\right] \geqslant 2qC \cdot f(e:S) \geqslant (1-q) \cdot f(e:S) = \mathbb{E}[\lambda_e \mid R].$$

By the law of total expectation, taken over all $R \subseteq \overline{A_e}$, we have

$$\mathbb{E}\left[\sum_{v \in e} \phi_v \,\middle|\, \overline{A_e}\right] \geqslant \mathbb{E}\big[\lambda_e \mid \overline{A_e}\big]. \tag{11.3}$$

Combining inequalities (11.2) and (11.3) with the law of total expectation gives the desired inequality,

$$\mathbb{E}\left[\sum_{v \in e} \phi_v\right] \geqslant \mathbb{E}[\lambda_e]. \qquad \square$$

To bound the performance of this section's randomized variant of Algorithm 16, we can reuse corollaries 11.5.3 and 11.5.4, since these follow from Lemma 11.5.1, which holds for every realization of the random choices of the algorithm. We now use these corollaries, LP duality and Lemma 11.6.1, together with Lemma 11.3.2, to analyze this algorithm.

Theorem 11.6.2. *Algorithm 16 run with $q = 1/(2C+1)$ and C on a non-monotone* MSM *instance outputs a matching M of value*

$$\left(\frac{4C^2-1}{2C-2}\right) \cdot f(M) \geqslant f(\mathrm{OPT}).$$

This is optimized by taking $C = 1 + \frac{\sqrt{3}}{2}$, resulting in an approximation ratio of $4 + 2\sqrt{3} \approx 7.464$. Moreover, the same algorithm is $2C + C/(C-1)$ approximate for monotone MSM.

Proof. First, by Lemma 11.5.3 and Lemma 11.5.4, for every realization of the algorithm, we have

$$\left(2C + \frac{C}{C-1}\right) \cdot f(M) \geqslant \sum_v \phi_v + \mu,$$

and thus this relationship holds in expectation as well.

$$\left(2C + \frac{C}{C-1}\right) \cdot \mathbb{E}[f(M)] \geqslant \mathbb{E}\left[\sum_v \phi_v + \mu\right]. \tag{11.4}$$

On the other hand, by Lemma 11.6.1, the expected dual LP solution is feasible for $\mathbb{E}[(D)]$. Therefore, by weak LP duality, we have

$$\mathbb{E}\left[\sum_v \phi_v + \mu\right] \geqslant \max_T \mathbb{E}[g(T)] = \max_T \mathbb{E}[f(S \cup T)] \geqslant \mathbb{E}[f(S \cup \mathrm{OPT})]. \tag{11.5}$$

The result for monotone MSM follows from equations (11.4) and (11.5), together with monotonicity implying $\mathbb{E}[f(S \cup \mathrm{OPT})] \geqslant f(\mathrm{OPT})$.

For non-monotone MSM, let us define the additional auxiliary function $h : 2^E \to \mathbb{R}^+$, with $h(T) := h(\mathrm{OPT} \cup T)$. Now note that by our sampling procedure, S is a random subset of E containing every edge with probability at most q. Hence, by Lemma 11.3.2, we have

$$\mathbb{E}[f(S \cup \mathrm{OPT})] = \mathbb{E}[h(S)] \geqslant (1-q) \cdot h(\emptyset) = (1-q) \cdot f(\mathrm{OPT}). \tag{11.6}$$

Combining equations (11.4), (11.5) and (11.6), together with our choice of $q = 1/(2C+1)$, the desired inequality follows by rearranging terms. □

Having explored the use of Algorithm 16 for submodular matchings, we now turn to consider this algorithm's use in the context of streaming *linear* objectives.

11.7 Linear Objectives

In this section we address the use of Algorithm 16 to matching and b-matching with linear objectives, i.e., MWM and MWbM, using a deterministic variant, with $q = 1$.

For MWM, this algorithm with $C = 1 + \varepsilon$ is essentially the algorithm of [232], and so it retrieves the state-of-the-art $(2+\varepsilon)$-approximation for this problem, previously analyzed in [127, 232]. We therefore focus on MWbM, for which a simple modification of Algorithm 16 yields a $3+\varepsilon$ approximation, improving upon the previous best $4+\varepsilon$ approximation due to [73].

The modification to Algorithm 16 which we consider is a natural one: instead of computing M greedily, we simply compute an optimal MWbM M in the subgraph induced by S, using a polytime linear-space offline algorithm (e.g., [12, 117]). Trivially, the b-matching M has weight at least

$$w(M) \geqslant w(OPT \cap S). \tag{11.7}$$

Moreover, this b-matching has weight no lower than the greedily-constructed b-matching of lines 15-20. We use LP duality to show that this modified algorithm with $C = 1 + \varepsilon$ outputs a b-matching M of weight at least $w(M) \geqslant \frac{1}{2+\varepsilon} \cdot w(OPT \setminus S)$.

Lemma 11.7.1. *Let M be a* MWbM *in the stack S obtained by running Algorithm 16 with $C = 1 + \varepsilon/2$ and $q = 1$ until Line 15. Then, we have $w(M) \geqslant \frac{1}{2+\varepsilon} \cdot w(OPT \setminus S)$.*

Proof. Consider the matching M' obtained by greedily unwinding the stack, as in Algorithm 16. Clearly, $w(M) \geqslant w(M')$. So, by Lemma 11.5.3, we have $w(M) \geqslant \frac{1}{2+\varepsilon} \cdot \sum_{v \in V} \phi_v$, for $\phi_v = C \cdot \phi_v^{(|E|)}$. To relate $\sum_{v \in V} \phi_v$ to $w(OPT)$, we show that the dual solution $(0, \vec{\phi}, \vec{w})$ is dual feasible for the LP (D) with function w.

The first set of constraints are trivially satisfied, due to linearity of w, as $0 + \sum_{e \in T} w_e = w(T)$.

For the second set of constraints, note that an edge $e = e^{(t)}$ is not added to the stack if and only if the check at Line 6 fails. Therefore, since $\phi_v^{(t)}$ values increase monotonically with t, we have

$$\sum_{v\in e}\phi_v = C\cdot\sum_{v\in e}\phi_v^{(|E|)} \geqslant C\cdot\sum_{v\in e}\phi_v^{(t-1)} \geqslant f(e:S) = w_e.$$

Therefore, by weak LP duality, we have $w(M) \geqslant 0 + \frac{1}{2+\varepsilon}\cdot\sum_{v\in V}\phi_v \geqslant \frac{1}{2+\varepsilon}\cdot w(OPT)$. □

We are now ready to analyze the approximation ratio of this MWbM algorithm.

Theorem 11.7.2. *For any $\varepsilon \geqslant 0$, Algorithm 16 run with $C = 1+\varepsilon/2$ and $q = 1$ until Line 15, followed by a linear-space offline* MWbM *algorithm run on S to compute a solution M is a $(3+\varepsilon)$-approximate streaming* MWbM *algorithm.*

Proof. To see that this is a streaming algorithm, we recall that $|S| = \tilde{O}(\sum_v b_v)$, by Lemma 11.4.1. Since we compute M by running an offline linear-space algorithm on the subgraph induced by S, therefore using $O(|S|)$ space for this last step, the desired space bound follows.

To analyze the algorithm's approximation ratio, let $\alpha \in [0,1]$ be the weighted fraction of OPT in S. That is, $w(OPT\cap S) = \alpha\cdot w(OPT)$, and by linearity, $w(OPT\setminus S) = (1-\alpha)\cdot w(OPT)$. Therefore, by Equation (11.7) and Lemma 11.7.1 we have the following.

$$w(M) \geqslant w(OPT\cap S) = \alpha\cdot w(OPT).$$
$$w(M) \geqslant \frac{1}{2+\varepsilon}\cdot w(OPT\setminus S) = \frac{1-\alpha}{2+\varepsilon}\cdot w(OPT).$$

We thus find that the approximation ratio of this algorithm is at most $1/\min\{\alpha, \frac{1-\alpha}{2+\varepsilon}\} \leqslant 3+\varepsilon$. □

Remark. We note that this approach—dual covering constraints for elements outside of the algorithm's memory S, and solving the problem optimally for S—is rather general. In particular, it applies to matching under any sub-additive (not just submodular) set function f, for which $f(OPT) \leqslant f(OPT\setminus S) + f(OPT\cap S)$. Moreover, this approach extends beyond matchings, to any downward-closed constraints, for which $OPT\setminus S$ and $OPT\cap S$ are both feasible solutions. So, it seems like this approach could find applications to streaming algorithms for other objectives and constraints, provided dual feasibility can be guaranteed using a dual solution of value bounded by that of the output solution.

11.8 Explaining Prior Work using LP Duality

In this section we further demonstrate the generality of our (randomized) primal-dual analysis, showing that it provides fairly simple alternative analyses of the algorithms of [58, 111], giving one unified analysis for these algorithms and ours. To keep things simple, we focus only on MSM, though [58, 111] show that their algorithms also work more broadly for k-matchoid and k-set system constraints.

In [58], Chakrabarti and Kale presented a reduction from MSM to MWM, by showing how to use a subclass of MWM algorithms to solve MSM. We now introduce the algorithm of [58] instantiated with the MWM algorithm of McGregor [205]. The algorithm is a natural and elegant one: when an edge e arrives, we consider its marginal gain with respect to the current matching. If this marginal gain is higher than some slack parameter C times the marginal gains of the currently blocking edges $e' \in N(e) \cap M$, we preempt those edges and add e to the matching.

In anticipation of our analysis of the algorithm of [111] in Section 11.8.2, we generalize the algorithm's description and allow the algorithm to preempt with some probability $q \in [0, 1]$. The full pseudo-code is given in Algorithm 17.

Algorithm 17 The MSM Algorithm of [58] and [111]

```
    Initialization
1:  $M \leftarrow \emptyset$
    Loop
2:  for $t \in \{1, \ldots, |E|\}$ do
3:      $e \leftarrow e^{(t)}$
4:      $B(e) \leftarrow \sum_{e' \in N(e) \cap M} f(e' : M)$
5:      if $f(e : M) \leqslant C \cdot B(e)$ then
6:          continue                                  ▷ skip edge $e$
7:      else
8:          with probability $q$ do
9:              $M \leftarrow (M \setminus N(e)) \cup \{e\}$
10: return $M$
```

This algorithm only ever adds an edge e to M upon its arrival. After adding an edge to M, this edge can be *preempted*, i.e., removed from M, after which it is never added back to M. Thus we note that this algorithm is not only a streaming algorithm, but also a so-called *preemptive* algorithm: it only stores a single matching in memory and therefore trivially requires $\tilde{O}(n)$ space.

For convenience, we let $M^{(t)}$ denote the matching M at time t, and let $S := \bigcup_t M^{(t)}$ denote the set of edges ever added to M. For an edge e let $B^{(t)}(e) = \sum_{e' \in N(e) \cap M^{(t)}} f(e' : M^{(t)})$. We will also denote by $P := S \setminus M$ the set of *preempted* edges.

11.8.1 The Framework of [58], Applied to the Algorithm of [205]

In this section we analyze the deterministic algorithm obtained by applying the framework of Chakrabarti and Kale [58] to the MWM algorithm of McGregor [205], corresponding to Algorithm 17 run with $q = 1$.

To argue about the approximation ratio, we will again fit a dual solution to this algorithm. Define the auxiliary submodular functions $g^S : 2^E \to \mathbb{R}^+$ to be $g^S(T) := f(S \cup T)$. Similarly to our analysis of Algorithm 16, we define the following dual.

$$
\begin{aligned}
\mu &:= f(S) = g^S(\emptyset), \\
\phi_v &:= C \cdot \max\{f(e : M^{(t)}) \mid t \in [|E|],\ v \in e \in M^{(t)}\}, \\
\lambda_e &:= \begin{cases} f(e : S) & e \notin S \\ 0 & e \in S. \end{cases}
\end{aligned}
$$

Note the difference here in the setting of ϕ_v from the algorithms of Sections 11.5 and 11.6. We start by showing that this is a dual feasible solution to the LP (D) for the function g^S.

Lemma 11.8.1. *The dual solution $(\vec{\lambda}, \vec{\phi}, \mu)$ is feasible for the LP (D) with function g^S.*

Proof. To see that the first set of constraints are satisfied, note that by submodularity of f,

$$\sum_{e \in T} \lambda_e = \sum_{e \in T \setminus S} f(e : S) \geqslant \sum_{e \in T \setminus S} f_S(e) \geqslant f_S(T \setminus S) = f(S \cup T) - f(S) = g^S(T) - \mu.$$

For the second set of constraints, we note that if $e = e^{(t)} \notin S$, then by the test in Line 5 and submodularity, we have that

$$\lambda_e = f(e : S) \leqslant f(e : M^{(t-1)}) \leqslant C \cdot B^{(t-1)}(e) \leqslant \sum_{v \in e} \phi_v. \qquad \square$$

It remains to relate the value of the solution M to the cost of this dual. For this, we introduce the following useful notation. For any edge $e \in S$, we define the weight of e to be

$$w_e := \begin{cases} f(e : M) & e \in M \\ f(e : M^{(t)}) & e \in M^{(t-1)} \setminus M^{(t)} \subseteq P. \end{cases}$$

In words, the weight of an edge in the matching is $f(e : M)$, and the weight of a preempted edge is frozen to its last value before the edge was preempted. One simple consequence of the definition of the weights w_e is the following relationship to $f(M)$.

Observation 11.8.2. $f(M) = \sum_{e \in M} f(e : M) = \sum_{e \in M} w_e = w(M)$.

We now show that the preempted edges' weight is bounded in terms of the weight of M.

Lemma 11.8.3. *The weights of P and M satisfy $w(P) \leqslant w(M) \cdot \frac{1}{C-1}$.*

Proof. For any edge e, we define the following set of preempted edges which are preempted in favor of$e$ or in favor of an edge (recursively) preempted due to e.[4] First, for an edge $e = e^{(t)}$, we let the set $P^1(e) := N(e) \cap M^{(t-1)}$ denote the edges preempted when e is added to M. For any $i > 1$, we let $P^i(e) := P^1(P^{i-1}(e))$ be the set of edges preempted by an edge with a trail of preemptions of length $i-1$ from e. By Line 6, we have that any edge $e \in S$ has weight at least $w_e \geqslant C \cdot P^1(e)$. By induction, this implies that $w_e \geqslant C \cdot w(P^{(i-1)}(e)) \geqslant C^i \cdot w(P^i(e))$. Now, since each preempted edge $e' \in P$ belongs to precisely one set $P^i(e)$ for some $i \geqslant 1$ and $e \in M$, we find that indeed,

$$w(P) = \sum_{e \in M} \sum_{i \geqslant 1} w(P^i(e)) \leqslant \sum_{e \in M} \sum_{i \geqslant 1} \frac{1}{C^i} \cdot w_e = w(M) \cdot \left(\frac{1}{C} + \frac{1}{C^2} + \frac{1}{C^3} + \dots \right) = w(M) \cdot \frac{1}{C-1}.$$

□

Using Lemma 11.8.3, we can now relate the value of the primal solution M to the cost of the our dual solution, $\mu + \sum_v \phi_v$. We start by bounding μ in terms of $f(M)$.

Lemma 11.8.4. *The matching M output by Algorithm 17 satisfies $f(M) \geqslant \left(1 - \frac{1}{C}\right) \cdot \mu$.*

Proof. By submodularity of f, Lemma 11.8.3, and Observation 11.8.2 we obtain the desired inequality,

$$\mu = f(S) = f(M \cup P) \leqslant f(M) + \sum_{e \in P} f(e : M) = w(M) + w(P) \leqslant \left(1 + \frac{1}{C-1}\right) \cdot f(M). \quad \square$$

We next bound $\sum_v \phi_v$ in terms of $f(M)$.

Lemma 11.8.5. *The matching M output by Algorithm 17 run with $C > 1$ satisfies*

$$f(M) \geqslant \frac{1}{2C + C/(C-1)} \cdot \sum_{v \in V} \phi_v.$$

Proof. Fix a vertex v and edge $e \in M^{(t-1)} \setminus M^{(t)} \subseteq P$ preempted at time t in favor of edge $e' = e^{(t)} \ni v$. For this edge e, by monotonicty in t' of $f(e' : M^{(t')})$, the test of Line 5, non-negativity of $f(e'' : M^{(t-1)})$ for any edge $e'' \in M^{(t-1)}$ and $C > 1$ we have that

$$w_{e'} \geqslant f(e' : M^{(t-1)}) \geqslant C \cdot B^{(t-1)}(e') \geqslant C \cdot f(e : M^{(t-1)}) = C \cdot w_e > w_e.$$

Consequently, again relying on monotonicity in t' of $f(e' : M^{(t')})$, we have that for any edge $e \in P$, there is at most one vertex $v \in e$ such that $w_e = f(e : M^{(t-1)})$ is equal to $\phi_v = \max\{f(e' : M^{(t')}) \mid v \in e' \in M^{(t')}\}$. Edges $e \in M$, on the other hand, clearly have $w_e = f(e : M)$ equal to

[4]In [104, 205], these sets are referred to by the somewhat morbid term "trail of the dead".

$\phi_v = \max\{f(e' : M^{(t-1)}) \mid v \in e' \in M^{(t-1)} \setminus M^{(t)}\}$ for at most two vertices $v \in e$. Combined with Lemma 11.8.3 and Observation 11.8.2, this yields the desired inequality,

$$\sum_{v\in V} \phi_v \leqslant C \cdot \left(2\sum_{e\in M} w_e + \sum_{e\in P} w_e\right) \leqslant \left(2C + \frac{C}{C-1}\right) \cdot w(M) = \left(2C + \frac{C}{C-1}\right) \cdot f(M). \quad \square$$

Equipped with the above lemmas, we can now analyze Algorithm 17's approximation ratio.

Theorem 11.8.6. *Algorithm 16 run with $C > 1$ and $q = 1$ on a monotone* MSM *instance outputs a matching M of value*

$$\left(2C + \frac{2C}{C-1}\right) \cdot f(M) \geqslant f(\mathrm{OPT}).$$

This is optimized by taking $C = 2$, resulting in an approximation ratio of 8.

Proof. By weak LP duality and Lemma 11.8.1, together with monotonicity of f, we have that

$$C \cdot \sum_v \phi_v + \mu \geqslant \max_T g^S(T) = \max_T f(S \cup T) \geqslant f(S \cup \mathrm{OPT}) \geqslant f(\mathrm{OPT}).$$

Combining Lemma 11.8.5 and $f(M) = \mu$ by definition and rearranging, we get the desired inequality,

$$\left(2C + \frac{2C}{C-1}\right) \cdot f(M) \geqslant C \cdot \sum_v \phi_v + \mu \geqslant f(\mathrm{OPT}). \qquad \square$$

As with our algorithm of Section 11.5, our analysis of Algorithm 17 relied on monotonicity, crucially using $f(S \cup OPT) \geqslant f(OPT)$. To extend this algorithm to non-monotone MSM, we again appeal to Lemma 11.3.2, setting $q = \frac{1}{2C+1}$. This is precisely the algorithm of Feldman et al. [111], which we analyze in the following section.

11.8.2 The Algorithm of Feldman et al. [111]

In [111], Feldman et al. showed how to generalize the algorithm of [58] to non-monotone function maximization. Here we show an analysis of their algorithm in our primal dual framework. Our proof is an extension of the one in Section 11.8.1 in a way that is analogous to how Section 11.6 extends Section 11.5.

We reuse the same dual from Section 11.8.1, only this time, both our dual object and the function g^S are random variables. The proof of expected dual feasibility for this variant of the algorithm of Section 11.8.1 is analogous to that of Lemma 11.6.1, so we only outline the differences here.

We start with expected feasibility.

Lemma 11.8.7. *The dual solution* $(\mathbb{E}[\vec{\lambda}], \mathbb{E}[\vec{\phi}], \mathbb{E}[\mu])$ *is feasible for the expected LP* $\mathbb{E}[(D)]$.

Proof (Sketch). The first set of constraints is satisfied for any random realization. Indeed, as in the proof of Lemma 11.8.1, for any realization of S, by submodularity of f, we have

$$\sum_{e\in T} \lambda_e = \sum_{e\in T\setminus S} f(e:S) \geqslant \sum_{e\in T\setminus S} f_S(e) \geqslant f_S(T\setminus S) = f(S\cup T) - f(S) = g^S(T) - \mu.$$

Consequently, taking expectation over S, we have that indeed, $\mathbb{E}_S[\mu]+\sum_{e\in T}\mathbb{E}_S[\lambda_e] \geqslant \mathbb{E}_S[g^S(T)]$.

For the second set of constraints, the proof is nearly identical to that of Lemma 11.6.1, where we show that

$$\mathbb{E}\left[\sum_{v\in e}\phi_v\right] \geqslant \mathbb{E}[\lambda_e].$$

This is proved by taking total probability over the event $A_e := [f(e:S) \leqslant C\cdot\sum_{v\in V}\phi_v^{(t-1)}]$ and its complement. The key inequality to prove here is that for any realization of randomness R for which $\overline{A_e}$ holds, we have that

$$\mathbb{E}\left[\sum_{v\in e}\phi_v^{(t)}\;\middle|\; R\right] = 2q\cdot f(e:S) + (1-2q)\cdot\sum_{v\in e}\phi_v^{(t-1)} \geqslant 2q\cdot f(e:S).$$

And indeed, conditioned on R, the edge $e = e^{(t)}$ fails the test in Line 5, and so with probability q, we have $\sum_{v\in e}\phi_v^{(t)} = 2\cdot f(e:S)$. To see this, note that if e is added to the matching, then for both $v\in e$, by definition $\phi_v^{(t)}$ must be at least $f(e:S)$. Hence, in this case

$$\mathbb{E}\left[\sum_{v\in e}\phi_v^{(t)}\;\middle|\; R\right] \geqslant 2q\cdot f(e:S).$$

The proof then proceeds as that of Lemma 11.6.1. ☐

To relate the value of the solution M to the cost of the dual, we can define weights as in Section 11.8.1 and reuse lemmas 11.8.3, 11.8.4, and 11.8.5, which hold for every realization of the random choices of the algorithm. From here, following our template, we can use these along with LP duality, Lemma 11.8.7 and Lemma 11.3.2, to analyze this algorithm.

Theorem 11.8.8. *Algorithm 17 run with* $q = 1/(2C+1)$ *and* C *on a non-monotone* MSM *instance outputs a matching* M *of value*

$$\left(\frac{2C^2+C}{C-1}\right)\cdot f(M) \geqslant f(\mathrm{OPT}).$$

This is optimized by taking $C = 1+\frac{\sqrt{3}}{2}$, *resulting in an approximation ratio of* $5+2\sqrt{6}\approx 9.899$. *Moreover, the same algorithm is* $2C + 2C/(C-1)$ *approximate for* monotone MSM, *yielding an approximation ratio of* 8 *for* $C = 2$.

11.9 Lower Bound for MSM

Previous work shows that beating a $\frac{e}{e-1} \approx 1.582$ approximation for MSM in the streaming model is impossible for quasilinear space bounded algorithms [174], or polytime bounded algorithms [81, 102, 204]. In this section, we show that assuming the exponential time hypothesis (ETH), whereby NP $\not\subseteq$ TIME$(2^{o(n)})$ [167], beating 1.914 is impossible for any algorithm that is both space and time bounded. In particular, we will rely on seminal hardness of approximation results SET COVER from [81]. Recall:

Definition 11.9.1. *A* SET COVER *instance consists of a set system $(\mathcal{U}, \mathcal{S})$, with $\mathcal{S} \subseteq 2^{\mathcal{U}}$. The goal is to pick the smallest number k of sets $S_1, \ldots S_k$ such that $\left|\bigcup_{i \in [k]} S_i\right| = |\mathcal{U}|$. We use K to denote the size of minimal cover for the instance $(\mathcal{U}, \mathcal{S})$, and $N = |\mathcal{U}| + |\mathcal{S}|$ to denote the description size.*

Lemma 11.9.2. *Assuming ETH, every algorithm achieving an approximation ratio $(1 - \alpha) \ln |\mathcal{U}|$ for* SET COVER *runs in time strictly greater than $2^{N^{\gamma \cdot \alpha}}$ for some $\gamma > 0$. Furthermore, this holds even under the assumptions that $|\mathcal{S}| \leqslant K^{1/(\gamma\alpha)}$ and $|\mathcal{U}| \leqslant |\mathcal{S}|^{1/(\gamma\alpha)}$.*

See Section 11.12 for a proof that the hardness holds even under the extra assumptions. To describe the instance, we will also use some extremal graph theory results from [132].

Definition 11.9.3. *An α-Ruzsa-Szemerédi graph (α-RS graph) is a bipartite graph $G = (P, Q, E)$ with $|P| = |Q| = n$ that is a union of induced matchings of size exactly αn.*

Theorem 11.9.4 (Lemma 53 of [132])**.** *For any constant $\varepsilon > 0$, there exists a family of balanced bipartite $(1/2 - \varepsilon)$-RS graphs with $n^{1+\Omega(1/\log\log n)}$ edges.*

In what follows we will show a randomized reduction from SET COVER to streaming MSM. Specifically, we will show that if there is a polytime streaming algorithm for MSM achieving ratio better than 1.914, then there is an algorithm for SET COVER violating Lemma 11.9.2. We proceed to describing our reduction.

The Reduction. The input is a SET COVER instance $(\mathcal{U}, \mathcal{S})$ for which the minimal cover contains K sets. Fix $n = 2^{k^{1/d}}$ for a degree d to be determined later.

We create an underlying bipartite graph $G = (L, R, E)$ with $n \operatorname{poly} \log n$ vertices as follows. The left/right vertex sets are partitioned into $L = P \sqcup P'$, $R = Q \sqcup Q'$. We let $|P| = |Q| = 2n$, and we let $|P'| = |Q'| = n \cdot |\mathcal{S}|/K$.

The edge set E arrives in two phases. In phase 1, all the edge of a set E_1 arrive, in phase 2 the edges of E_2 arrive. To define E_1, let G_0 be a fixed $(1/2-\varepsilon)$-RS graph with $m = \Omega\left(n^{1+1/\log\log n}\right)$

edges between P and Q, and let this graph be the union of the matchings $M_1, \ldots, M_t$. Let M'_i be a random subset of M_i of size $(1/2-\delta)n$ for a parameter $\varepsilon < \delta < 1/4$ and let $E_1 = M'_1 \cup \ldots \cup M'_t$. Choose one index $r \in [k]$ uniformly at random, and call M'_r the *distinguished matching*. Note that the index r is unknown to the algorithm.

Define E_2 as follows. Let $P_1 \sqcup P_2 \sqcup \ldots \sqcup P_{n/K}$ and $Q_1 \sqcup Q_2 \sqcup \ldots \sqcup Q_{n/K}$ be partitions of the the vertices of P and Q respectively not matched by M_r into subsets of size K. Similarly, let $P'_1 \sqcup P'_2 \sqcup \ldots \sqcup P'_{n/K}$ and $Q'_1 \sqcup Q'_2 \sqcup \ldots \sqcup Q'_{n/K}$ be partitions of P' and Q' into subsets of size $|\mathcal{S}|$. Let F_i be the edges of the complete bipartite graph between P_i and Q'_i, and let G_i be the edges of the complete bipartite graph between Q_i and P'_i. Finally, set $E_2 = \bigcup_i F_i \cup G_i$. See Figure 11.2.

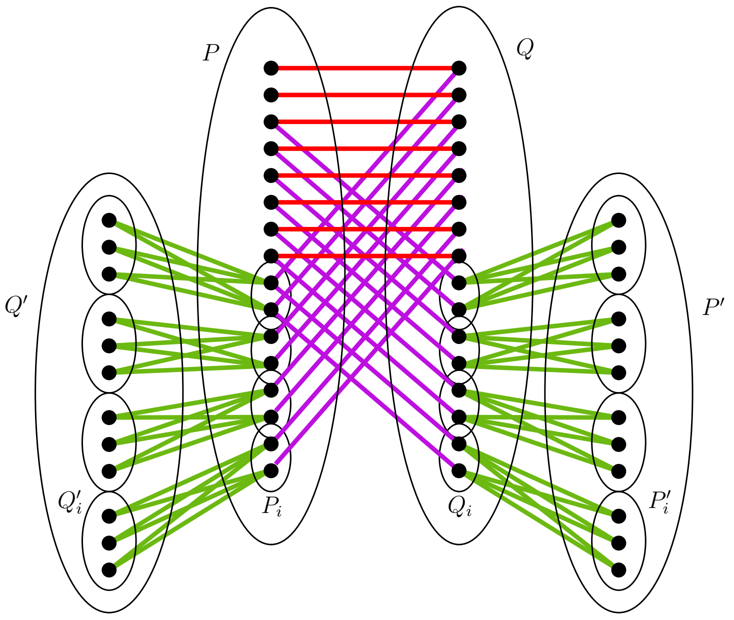


Figure 11.2: Illustration of lower bound instance.
Red edges represent the edges of the distinguished matching M_r in E_1, purple edges represent other edges in E_1, green edges represent edges of E_2. The red and purple edges together form the $(1/2-\varepsilon)$-RS graph G_0, subsampled.

It remains to describe the submodular function f. First, define the set function $f_1(E) = |E \cap E_1|$. Next, we define the function f_2 which is parametrized by the SET COVER instance. We identify each set of vertices P'_i and Q'_i with a disjoint copy of $\mathcal{S}$. For every edge $e \in E_2$, let $\phi(e)$ denote the set with which the endpoint of e in $P' \cup Q'$ is associated. Now, for some parameter $\eta > 0$ to be determined later, we define

$$f_2(E) := \frac{\eta K}{|\mathcal{U}|} \cdot \left| \bigcup_{e \in E} \phi(e) \right|.$$

Finally, set $f := f_1 + f_2$. Note that f is submodular since it the sum of a scaled coverage function and a linear function. On a technical note, since we assume that $|\mathcal{U}|$ is polynomially bounded in $|\mathcal{S}|$, we can represent the values of this function with $\operatorname{poly}\log n$ bits.

Some intuition. Intuitively, we can imagine that all edges of E_1 are worth 1. We imagine that each edge of E_2 is a set in one of the copies of the instance $(\mathcal{S}, \mathcal{U})$, and we let the value of all edges selected in the second phase be the coverage of all the associated sets (scaled by $\eta K/|\mathcal{U}|$). First we we will argue that the algorithm can output almost none of the edges of M_r', since it after phase 1 it has no information as to which matching is the distinguished one. Hence the majority of the edges it uses from phase 1 must be from $E_1 \backslash M_r'$. However, each edge the algorithm chooses from $E_1 \backslash M_r'$ precludes it from taking between 1 and 2 edges of E_2. Furthermore, maximizing the value of edges of E_2 amounts to solving a hard MAX K-COVERAGE instance. The coverage is scaled by the parameter η, and as a result, the algorithm is incentivized to take some $k := cK$ edges from each of the bipartite graphs (P_i, Q_i') (and (Q_i, P_i')) of E_2 and the remaining edges from $E_1 \backslash M_r'$. Meanwhile, OPT can take the distinguished matching edges M_r' as well as the edges of E_2 maximizing the coverage instance. Our bound will follow by setting η to maximize the ratio between these.

To start, we show that no streaming algorithm can "remember" more than a $o(1)$ fraction of the edges of the distinguished matching M_r. Since phase 1 of our construction is identical to the one in Appendix H of [132] which shows a $3/2$ semi-streaming lower bound for max *weight* matching., we can reuse their result here.

Lemma 11.9.5. *For any constants $\gamma, \delta \in (0, 1/4)$, let $\mathcal{A}$ be an algorithm that at the end of phase 1, with constant probability, outputs at least γn of the the edges of M_r'. Then $\mathcal{A}$ uses $\Omega(E_1) \geqslant n^{1+\Omega(1/\log\log n)}$ bits of space.*

We reproduce a version of the proof in Section 11.12 for completeness. With this, we are finally ready to prove the main theorem of the section.

Theorem 11.9.6. *Assuming ETH, there exists a distribution over* MSM *instances such that any deterministic algorithm achieving an* 1.914 *approximation must use either* $n^{1+\Omega(1/\log\log n)}$ *space or* $2^{(\log n)^1 0}$.

Proof. Our proof is a randomized polytime reduction from SET COVER to streaming MSM. We will show that if there is a randomized streaming algorithm achieving ratio better than 1.914 for MSM, then there is an algorithm for SET COVER achieving approximation ratio $(1-\alpha)\ln(|\mathcal{U}|)$ for constant $\alpha > 0$ that only requires polynomial extra overheard. We then argue that Lemma 11.9.2 implies that the streaming MSM algorithm must run in super polynomial time, assuming ETH.

Fix a deterministic algorithm A for streaming MSM. Now, given an instance of SET COVER $(\mathcal{U}, \mathcal{S})$ with minimum cover size K and description size $N = |\mathcal{U}| + |\mathcal{S}|$, create a random instance of streaming MSM according to the reduction described in this section. For each bipartite graph (P_i, Q_i') (or (Q_i, P_i')), if the algorithm A chooses cK edges from this graph, it can select at most $2(1-c)K$ edges from E_1 that are adjacent to P_i (or Q_i). Suppose WLOG that it can always achieve the $2(1-c)k$ bound. In this case we can also assume WLOG the algorithm chooses the same number $c \cdot K$ of edges from each such graph, and furthermore that it selects the same sets in the set system $(\mathcal{S},\mathcal{U})$. Otherwise it can locally improve its solution by copying the solution for the best index i.

Suppose this solution achieves coverage of $(1-e^{-c}+\gamma)\cdot|\mathcal{U}|$. Since the matchings $M_1,\ldots,M_t$ are induced, and by Lemma 11.9.5 w.h.p. the algorithm can only output $o(n)$ edges of M'_r after phase 1, the algorithm can only select $o(n)$ edges *not* incident to some P_i or Q_i. Thus the total value achieved by the algorithm is at most:

$$\begin{aligned}
&\big[2(1-e^{-c}+\gamma)\cdot\eta\cdot K+2(1-c)\cdot K\big]\cdot\frac{n}{K}+o(n)\\
&\leqslant 2(1-e^{-c}+\gamma)\cdot\eta n+2(1-c)\cdot n+o(n)\\
&\leqslant (2\eta-2\ln(\eta)+2\gamma)\cdot n+o(n),
\end{aligned}$$

where the last step follows since the expression is maximized at $c=\ln\eta$. On the other hand, the optimal solution can select the distinguished matching edge M'_r, as well as K edges adjacent to each set P_i corresponding to the minimum SET COVER solution. Thus the total value of OPT is at least:

$$(1-\delta+2\eta)\cdot n.$$

Thus the ratio between the maximum value achievable by the algorithm and the optimal value is bounded by:

$$\frac{1+2\eta-\delta}{2\eta-2\ln(\eta)+2\gamma+o(1)}.$$

Finally, we set $\eta=2.09$ and let $\delta\to 0$. If this ratio converges to a value strictly below 1.914, then we can conclude that $\gamma=\Omega(1)$ and $\gamma>0$.

We have shown that A can be used to pick cK sets with coverage $(1-e^{-c}+\gamma)\cdot|\mathcal{U}|$. To finish the proof, we now show that this can be used to recover an approximation algorithm B for SET COVER. For convenience, set constant γ' such that $(1-e^{-c-\gamma'}):=(1-e^{-c}+\gamma)$. Then, guess K, and repeat algorithm A recursively $\lceil\ln|\mathcal{U}|/(c+\gamma')\rceil\leqslant\ln|\mathcal{U}|/(c+\gamma')+1$ times, each time on the residual uncovered set system. Each call to A covers $(1-e^{-c-\gamma'})$ of the elements remaining, so after this number of iterations, the fraction of uncovered elements is less than $1/|\mathcal{U}|$, i.e. all elements are covered. Since each iteration costs $c\cdot K$, the total number of sets picked here is at most

$$c\left(\frac{\ln|\mathcal{U}|}{c+\gamma'}+1\right)\cdot K=\left(\frac{c}{c+\gamma'}+\frac{c}{\ln|\mathcal{U}|}\right)\cdot\ln|\mathcal{U}|\cdot K.$$

Defining the constant $\alpha=\gamma'/(c+\gamma')$, this is a $(1-\alpha-o(1))\ln|\mathcal{U}|$ approximation to the SET COVER instance. Furthermore, if A runs in time T then B runs in time $\mathrm{poly}(N)\cdot T$ (where $N=|\mathcal{U}|+|\mathcal{S}|$).

To conclude, if $T<2^{N^{\Delta}}$ for a constant $\Delta<\gamma\cdot\alpha$, then B runs faster than $2^{N^{\gamma\alpha}}$, contradicting Lemma 11.9.2. Thus the algorithm A must run in time at least $2^{N^{\gamma\cdot\alpha}}\geqslant 2^{|\mathcal{S}|^{\gamma\cdot\alpha}}\geqslant 2^{(\log n)^{d\cdot\gamma\cdot\alpha}}$. Setting $d=10/(\gamma\cdot\alpha)$, this running time is $2^{(\log n)^{10}}$, which is superpolynomial in n. $\square$

Theorem 11.2.3 therefore follows from Theorem 11.9.6 and Yao's minimax principle [270].

11.10 Tight instance for Algorithm 16

In this section we show that there exists a family of instances of MSM instances parametrized by C for which Algorithm 16 with parameter $C > 1$ yields an approximation factor of $2C + C/(C-1)$.

Lemma 11.10.1. *The approximation ratio of Algorithm 16 with $C > 1$ and $q = 1$ for monotone MSM is at least $2C + \frac{C}{C-1}$.*

Proof. Define the graph G as follows. The vertex set $V(G)$ consists of $\{x_i, y_i\}_{i\in[0,n]}$. For convenience, for every $i \in [1, n]$ we define the edges $d_i = (x_0, x_i)$ and $e_i = (x_i, y_i)$. Then the edge set $E(G)$ consists of the edges $\{d_i\}_{i=1}^n \cup \{e_i\}_{i=0}^n$.

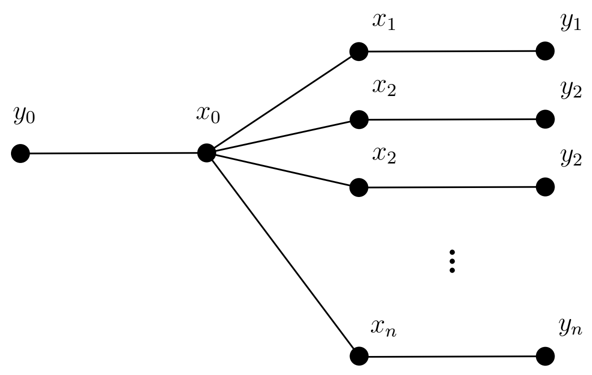


Figure 11.3: Tight Example for Algorithm 16

To define the (monotone) submodular function, we first define an auxiliary weight function $w : E(G) \to \mathbb{R}_{\geqslant 0}$. The weights are:

$$\begin{aligned} w(d_i) &= C^{i-1} && (n \geqslant i \geqslant 1)\\ w(e_1) &= 1 + C - \varepsilon \\ w(e_i) &= C^i - \varepsilon && (n \geqslant i \geqslant 2)\\ w(e_0) &= C^n - \varepsilon \end{aligned}$$

Now the submodular function is:

$$f(T) := w(T \cap \{e_0\}) + \sum_{i=0}^{n} \min(w(T \cap \{d_i, e_i\}), w(e_i))$$

Since weights are non-negative, this function is monotone. Submodularity follows from preseresvation of subdmodularity under linear combinations (and in particular sums), and $\min\{w(S), X\}$ being submodular for any linear function w.

The stream reveals the edges $d_1, \ldots, d_n$ in order, and subsequently reveals $e_0, e_1, \ldots, e_n$ in order. For a run of Algorithm 16 with this choice of C and $q = 1$, several claims hold inductively:

(a) On the arrival of edge d_i, we have $\phi_{x_0} = C^{i-2}$ (except for the arrival of d_1, at which point $\phi_0 = 0$) and $\phi_{x_i} = 0$.
(b) The algorithm takes every edge d_i into the stack.
(c) After d_i is taken into the stack, we have $\phi_{x_0} = C^{i-1}$ and $\phi_{x_i} = C^{i-1} + C^{i-2}$ (except for ϕ_{x_1} which is set to 1).
(d) The algorithm does not take e_i into the stack.

Let Λ_t be the statement that these claims holds for time t. By inspection Λ_1 holds, now consider some time $i > 1$. Claim (a) follows directly from claim (c) of Λ_{i-1}. Claim (b) follows from (a) since $f_S(d_i) = C^{i-1} = C \cdot \phi_0$ when d_i arrives. Claim (c) is a consequence of how the algorithm increases the potentials ϕ when taking edges into the stack. Claim (d) holds since $f_S(e_i) = w(e_i) - w(d_i) = C^i - C^{i-1} - \varepsilon < C \cdot \phi_{x_i}$.

From the above, we find that Algorithm 16 with parameter C as above and $q = 1$ will have all edges $d_1, \ldots, d_n$ in its stack by the end, resulting in it outputting the matching consisting of the single edge d_n. The value of this edge (and hence this matching) is C^{n-1}, while on the other hand OPT can take the edges $\{e_i\}_{i=0}^n$, which have value

$$\sum_{i=0}^{n} w(e_i) = C^n + \sum_{i=0}^{n} C^i - \varepsilon(n+1) \to C^{n-1}\left(2C + \frac{C}{C-1}\right). \qquad \text{(as } n \to \infty \text{ and } \varepsilon \to 0\text{)}$$

so long as $C > 1$. Hence $c(\mathrm{OPT})/c(\mathrm{ALG}) \to 2C + C/(C-1)$. The lemma follows. $\square$

11.11 Space Bound of Algorithm 16

In this section we bound the space usage of Algorithm 16, as restated in the following lemma.

Lemma 11.4.1. *For any constant $\varepsilon > 0$, Algorithm 16 run with $C = 1 + \varepsilon$ uses $\tilde{O}(\sum_v b_v)$ space.*

Proof. Fix a vertex $v \in V$. If an edge $e \ni v$ is added to S at time t, then by the test in Line 6, $f(e : S) \geqslant (1+\varepsilon) \cdot \sum_{u \in e} \phi_u^{(t-1)}$. Consequently, and since ϕ values are easily seen to always be positive, we have

$$\phi_v^{(t)} - \phi_v^{(t-1)} = w'_{ev} = \frac{f(e : S) - \sum_{u\in e} \phi_u^{(t-1)}}{b_v} \geqslant \frac{\varepsilon \cdot \sum_{u \in e} \phi_u^{(t-1)}}{b_v} \geqslant \frac{\varepsilon \cdot \phi_v^{(t-1)}}{b_v}.$$

Thus, adding this edge $e \ni v$ to S results in $\phi_v^{(t)} \geqslant \phi_v^{(t-1)} \cdot (1 + \varepsilon/b_v)$. Moreover, if e is the first edge of v added to S, then, letting $f_{min} := \min\{f(e : S) \neq 0 \mid e \in E, S \subseteq E\}$ be the minimum non-zero marginal gain, we have

$$\phi_v^{(t)} = \frac{f(e : S) - \sum_{u\in e} \phi_u^{(t-1)}}{b_v} \geqslant \frac{(\varepsilon/(1+\varepsilon)) \cdot f(e : S)}{b_v} \geqslant \frac{(\varepsilon/(1+\varepsilon)) \cdot f_{min}}{b_v}.$$

Therefore, if v had k edges added to the stack by time t, then

$$\phi_v^{(t)} \geqslant (\varepsilon/(1+\varepsilon)) \cdot (f_{min}/b_v) \cdot (1+\varepsilon/b_v)^{k-1}. \tag{11.8}$$

On the other hand, since f is polynomially bounded, we have that for some constant d

$$\phi_v^{(t)} \leqslant \sum_{e \ni v} f_{S_e}(e)/b_v \leqslant n^d \cdot (f_{min}/b_v). \tag{11.9}$$

Combining equations (11.8) and (11.9) and simplifying, we obtain $(1+\varepsilon/b_v)^{k-1} \leqslant n^d \cdot (1+\varepsilon)/\varepsilon$. Taking out logarithms and simplifying further, we find that

$$k \leqslant 1 + \frac{d \log n + \log(1+\varepsilon) + \log(1/\varepsilon)}{\log(1+\varepsilon/b_v)} = O((b_v/\varepsilon) \cdot (\log n + \log(1/\varepsilon))) = \tilde{O}(b_v).$$

That is, the number of edges of v in the stack is at most $\tilde{O}(b_v)$. Since each edge requires only $O(\log n)$ bits of space (and the ϕ_v variables can be specified using $O(\log n)$ bits each), the algorithm's space usage is indeed at most $\tilde{O}(\sum_v b_v)$. □

11.12 Deferred Proofs of Section 11.9

Lemma 11.12.1. *Assuming ETH, every algorithm achieving an approximation ratio $(1-\alpha)\ln|\mathcal{U}|$ for* SET COVER *runs in time strictly greater than $2^{N^{\gamma\cdot\alpha}}$ for some $\gamma > 0$. Furthermore, this holds even under the assumptions that $|\mathcal{S}| \leqslant K^{1/(\gamma\alpha)}$ and $|\mathcal{U}| \leqslant |\mathcal{S}|^{1/(\gamma\alpha)}$.*

Proof. The first statement is precisely Corollary 1.6 of [81].

For the extra assumptions, if $K < |\mathcal{S}|^{\gamma\alpha}$ then the brute force algorithm that checks all subsets of size K runs in time $|\mathcal{S}|^K < 2^{|\mathcal{S}|^{\gamma\alpha}\log|\mathcal{S}|} \leqslant 2^{N^{\gamma\alpha}}$. If $|\mathcal{S}| < |\mathcal{U}|^{\gamma\alpha}$, then one can brute force over all sub collections of S in time $2^{|\mathcal{S}|} \leqslant 2^{|\mathcal{U}|^{\gamma\alpha}} \leqslant 2^{N^{\gamma\alpha}}$. Both running times contradict Lemma 11.9.2. □

Lemma 11.12.2. *For any constants $\gamma, \delta \in (0, 1/4)$, let $\mathcal{A}$ be an algorithm that at the end of phase 1, with constant probability, outputs at least γn of the the edges of M_r'. Then $\mathcal{A}$ uses $\Omega(E_1) \geqslant n^{1+\Omega(1/\log\log n)}$ bits of space.*

Proof. Let $\mathcal{A}$ be an algorithm that outputs γn of the edges of M_r' at the end of phase 1 that uses fewer than $s = n \operatorname{poly}\log n$ bits. We will show that $\gamma = o(1)$.

Let $\mathcal{G}$ be the set of possible first phase graphs. Then

$$|\mathcal{G}| = \binom{n/2}{\delta n}^t = 2^{\gamma m}$$

for some $\gamma > 0$. Let $\phi : \mathcal{G} \to \{0,1\}^s$ be the function that takes an input graph G to the state of the algorithm $\mathcal{A}$ after running $\mathcal{A}$ on G. Let $\Gamma(G) = \{H \mid \phi(G) = \phi(H)\}$, that is the set of graphs inducing the same internal state for $\mathcal{A}$ at the end of phase 1.

Define $\Psi(G) = \bigcap_{H \in \Gamma(G)} E(H)$. Note that for any input graph G, the algorithm $\mathcal{A}$ can output an edge e if and only if $e \in \Psi(G)$. Also, for any G let t' be the number of matchings in the RS graph G_0 for which $\Psi(G)$ contains at least γn edges. Since algorithm $\mathcal{A}$ outputs γn edges of M_r', the number of graphs in $\Gamma(G)$ is bounded by

$$\binom{(1/2-\gamma)n}{\delta n}^{t'} \binom{n/2}{\delta n}^{t-t'} = \left(2^{-\Omega(\gamma n)} \binom{n/2}{\delta n}\right)^{t'} \binom{n/2}{\delta n}^{t-t'} = 2^{-\Omega(t'\gamma n)} 2^{\gamma m} \qquad (*)$$

On the other hand, since the first phase graph G is chosen uniformly at random, by a counting argument, with probability at least $1 - o(1)$ we have that $|\Gamma(G)| \geqslant 2^{(\gamma - o(1))m}$. Conditioning on this happening, we also know that $t' \geqslant \Omega(t)$ since the input graph is uniformly chosen within $\Gamma(G)$, and the algorithm succeeds with constant probability. These two facts together with $(*)$ imply that $\gamma = o(1)$. $\square$

11.13 Conclusion and Open Questions

In this chapter, we presented improved algorithms and lower bounds for streaming maximum submodular matching. The most natural question for this problem is determining the optimal approximation ratio of streaming algorithms for MSM. Can one show a lower bound strictly higher than 2 for monotone MSM? This would provide a separation between the streaming version of this problem and its offline counterpart, for which $(2+\varepsilon)$-approximate algorithms are known [109, 194]. More broadly, looking beyond submodular matching, the natural question is whether the techniques presented here can be leveraged to obtain results for other problems. In this chapter, we provide a principled way of analyzing streaming algorithms for submodular (matching) maximization; in particular, we show the usefulness of the (randomized) primal-dual method, and its extensions, for streaming MSM. We use this machinery to improve the upper bounds for this problem, and also show how to analyze known algorithms in this framework. The most appealing (and most open-ended) question here is to find more applications of this framework to other streaming (submodular) problems.

Chapter 12

Conclusion and Open Questions

In this thesis, we make progress on a number of long-standing open problems in the area of matching theory under uncertainty. For some of these problems we give a complete characterization of achievable approximation/competitive ratios, while for others, we improve on the prior bounds. There are naturally still many intriguing open questions which remain, some of which we have already addressed in earlier chapters. Beyond the intellectual interest in these problems for their own merit, progress on these problems will likely require new tools and techniques, and hopefully shed light on computation under uncertainty more broadly. We list some such problems here.

Edge Arrivals: In Chapter 3 we show that no online algorithm, even a fractional one, can achieve competitive ratio beyond $1/2 + O(1/n)$ under edge arrivals. Given that greedy gives a competitive ratio of $1/2$, this problem is in some sense completely resolved. There does, however, remain the question of pinning down the correct $o(1)$ term in the optimal $1/2 + o(1)$ competitive ratio. Buchbinder et al. [55] and [191] give $(1/2 + \Omega(2^{-n}))$-competitive randomized algorithms for this problem. What is the correct $o(1)$ term in the optimal $1/2 + o(1)$ competitive ratio? Is it $\Theta(1/n)$? Is it $\Theta(2^{-n})$? Somewhere in between? Determining the correct $o(1)$ term will give us a more detailed and nuanced understanding of the problem, and possibly shed light on online computation more broadly. Next, the nearly-complete picture we have for online matching under adversarial edge arrivals serves as a renewed motivation to (re)study this problem under various relaxations. For example, Guruganesh and Singla [146], building on a streaming algorithm of Konrad et al. [187], gave a $(1/2 + \Omega(1))$-competitive algorithm for this online matching under random-order edge arrivals. There has been much work on improving the results for streaming under random-order arrivals [17, 34, 99, 119, 186]. It would be interesting to see what improvements are achievable for *online* matching under random-order edge arrivals.

General vertex arrivals: In Chapter 4 we broke the barrier of $1/2$ for online matching under general arrivals, presenting a $(1/2+\Omega(1))$-competitive algorithm for this problem. This $\Omega(1)$ is rather small, and is left unspecified. It would be interesting to see what better (explicit) competitive ratios are achievable for this problem. Can one match the 0.526 bound of Wang and Wong [267] for *fractional* matching under general arrivals? Can one surpass this bound? Improving on this 0.526 bound would be interesting even for fractional algorithms, as it would likely require some non-trivial ideas to improve on the factional algorithm of [267]. On the hardness front, it seems plausible that the 0.591 lower bound of Buchbinder et al. [55] can be strengthened. How much harder is online matching under general vertex arrivals than the bipartite one-sided vertex arrival model of Karp et al. [179]? Finally, we note the recent work of Huang et al. [163, 165, 166] for online matching with deadlines. Online matching under general arrivals is more general the fully-online problem studied by Huang et al., and so any competitive ratio achievable for our problem yields a competitive ratio in theirs. Huang and Zhang [162] recently showed a separation between these problems for fractional algorithms, showing that the fully-online matching problem is strictly easier than online matching under general arrivals. Can one show a similar separation for randomized (integral) algorithms?

Online Dependent Rounding: In Chapter 5 we give an online dependent rounding scheme for bipartite fractional matching, in some sense mirroring the offline dependent rounding scheme of Gandhi et al. [122]. We give applications of this scheme to matching in regular graphs, as well as the online edge coloring in Chapter 6. Given the number of applications of the offline rounding scheme of [122] and its generalization over the years, the most natural question here is to find further applications of our online rounding schemes and its possible generalizations.

Online Edge Coloring: In chapters 6 and 9 we obtain optimal online edge coloring algorithms in the large $\Delta = \omega(\log n)$ regime. In particular, we show that for both adversarial one-sided vertex arrivals in bipartite graphs and random-order edge arrivals, the optimal competitive ratio is $1 + o(1)$ (provided Δ is known), thus resolving a conjecture of Bar-Noy et al. [25]. We note that these results still leave something to be desired, as the lower bound of 2 for both problems holds for $\Delta = O(\sqrt{\log n})$. While intuitively a super-logarithmic threshold seems natural for making the problem "easier", a proof of this fact remains illusive. Alternatively, an algorithm whose competitive ratio outperforms greedy for some sub-logarithmic $\Delta = \omega(\sqrt{\log n})$ would be surprising. Next, for adversarial bipartite edge coloring, we proved that not knowing Δ results in a strictly harder problem. Can the same be said for random-order edge arrivals? (In this case, one would need to also consider unknown number of edges, m, as estimating Δ given m in such a random-order arrival model is easy.) Finally, the question of achieving optimal competitive ratio for online edge coloring under adversarial edge arrivals remains a tantalizing open question. Making progress on more general adversarial vertex arrivals would likely prove a useful stepping stone in this direction.

Online Matching under Structural Assumptions: In Chapter 7 we give an explanation for the empirical success of online matching and AdWords algorithms on real-world data, by introducing and studying a class of graphs with properties arising in practice in the context of Internet advertising. We further present optimal deterministic algorithms for such inputs. What can be said about randomized algorithms for these inputs? Generally, studying online matching and other online problems under structural assumptions relevant to their motivating applications is a worthwhile objective. See the upcoming book of Roughgarden [243] for surveys on problem-specific approaches for algorithm analysis beyond the worst case.

Online Stochastic Metric Matching: In Chapter 8, we give a doubly-exponential improvement in the competitive ratio for stochastic metric matching over the previous best, which is implied by the optimal bound for metric matching under random-order arrivals [239]. Beyond the natural question of proving our algorithm (or indeed, any other) is $O(1)$ approximate, several questions remain. The first is whether similar improvements over the optimal random-order arrivals bounds [239] are achievable for stochastic arrivals drawn from an *unknown* distribution. Our approach does not work in this scenario. Can one do better than $O(\log n)$ approximation? Another point not addressed in our work is the question of *imperfect* matching, when the number of requests k is smaller than n. Our analysis does not seem to extend to this regime. What can be said of the optimal competitive ratio as k varies?

Dynamic Matching: In Chapter 10 we present a framework for deriving randomized dynamic matching algorithm against adaptive adversaries from fractional matching algorithms against adaptive adversaries. It is natural to ask for further instantiations of this framework, motivating further study of the dynamic fractional matching problem. Beyond the problems specifically related to dynamic matching, there are broader questions which still remain for dynamic algorithms more broadly, some of which our work touched upon: is there a separation between worst-case and amortized update time? Is there a separation between randomized algorithms against adaptive and oblivious adversaries? Is there a separation between deterministic and randomized algorithms? As a field, our current proof techniques do not seem to be refined enough to separate these computational models. Making progress on these meta-questions, even for specific problems, would therefore likely require new ideas.

Streaming Matching: In Chapter 11 we obtain improved bounds for streaming submodular matchings. The most natural research direction here is to improve on these bounds further, and possibly close the gap between upper and lower bounds for these problems. For the weighted matching problem, for which a $(2+\varepsilon)$-approximation is known due to the work of [232] (see also [127] and Chapter 11), we are at something of an impasse, as any improvement beyond a 2 approximation would require an improvement for the *unweighted* problem. Indeed, the most tantalizing open problem here is whether a $(2-\varepsilon)$-approximate maximum cardinality matching algorithm is possible. This is even open for any algorithm using $O(n^{1.999})$ space. As mentioned above, the same question has been studied intensely under random-order edge arrivals [17, 34, 99, 119, 186, 187]. In a recent result, Bernstein [34] gave a $^3/_2$-approximate algorithm for maximum matching under random-order edge arrivals. This bound is optimal for his approach, which relies on the EDCS matching sparsifiers introduced in the context of dynamic matching by Bernstein and Stein [37, 38]. Does this approach indeed yield the optimal approximation ratio for this problem, or are better bounds possible?

We hope the progress we have made in this thesis will initiate follow-up work in many of the above directions, and that the techniques developed here will prove useful for such follow ups and other problems.

Bibliography

[1] Bertinoro workshop 2014, problem 63. https://sublinear.info/index.php?title=Open_Problems:63. Accessed: 2020-06-20. 11.2

[2] Amir Abboud and Søren Dahlgaard. Popular conjectures as a barrier for dynamic planar graph algorithms. In *Proceedings of the 57th Symposium on Foundations of Computer Science (FOCS)*, pages 477–486, 2016. 10.1.3

[3] Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *Proceedings of the 55th Symposium on Foundations of Computer Science (FOCS)*, pages 434–443, 2014. 10.1.3

[4] Alexander A Ageev and Maxim I Sviridenko. Pipage rounding: A new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*, 8(3):307–328, 2004. 2.3.4, 4.1.1, 5.1.1

[5] Gagan Aggarwal, Rajeev Motwani, Devavrat Shah, and An Zhu. Switch scheduling via randomized edge coloring. In *Proceedings of the 44th Symposium on Foundations of Computer Science (FOCS)*, pages 502–512, 2003. 1.1, 1.2, 1.4, 5.1, 6.1, 6.1.4, 9, 9.1, 9.1, 9.3.1, 9.7

[6] Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1253–1264, 2011. 2.3.3, 7.1, 7.1, 8.1.2

[7] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. In *Proceedings of the 2nd International Conference on Web Search and Data Mining (WSDM)*, pages 5–14, 2009. 11.1

[8] Faez Ahmed, John P Dickerson, and Mark Fuge. Diverse weighted bipartite b-matching. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 35–41, 2017. 11.1

[9] Naor Alaluf, Alina Ene, Moran Feldman, Huy L Nguyen, and Andrew Suh. Optimal streaming algorithms for submodular maximization with cardinality constraints. In *Proceedings of the 47th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 6:1–6:19, 2020. 11.1

[10] Noga Alon. A simple algorithm for edge-coloring bipartite multigraphs. *Information Processing Letters (IPL)*, 85(6):301–302, 2003. 5.1, 6.1

[11] Noga Alon and Joel H Spencer. *The probabilistic method*. John Wiley & Sons, 2004. 9.1.1

[12] Richard P Anstee. A polynomial algorithm for b-matchings: an alternative approach. *Information Processing Letters (IPL)*, 24(3):153–157, 1987. 11.7

[13] Antonios Antoniadis, Neal Barcelo, Michael Nugent, Kirk Pruhs, and Michele Scquizzato. A $o(n)$-competitive deterministic algorithm for online matching on a line. In *Proceedings of the 12th Workshop on Approximation and Online Algorithms (WAOA)*, pages 11–22, 2014. 8.1

[14] Moab Arar, Shiri Chechik, Sarel Cohen, Cliff Stein, and David Wajc. Dynamic matching: Reducing integral algorithms to approximately-maximal fractional algorithms. In *Proceedings of the 45th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 7:1–7:16, 2018. 1.3, 1.5, 10, 10.1, 10.1.1, 10.1.2, 10.1.3, 10.5.2, 10.5.2, 10.5.2, 10.5.2

[15] Sepehr Assadi and Aaron Bernstein. Towards a unified theory of sparsification for matching problems. In *Proceedings of the 2nd Symposium on Simplicity in Algorithms (SOSA)*, 2019. 10.1.3

[16] Sepehr Assadi, Sanjeev Khanna, and Yang Li. The stochastic matching problem with (very) few queries. In *Proceedings of the 17th ACM Conference on Economics and Computation (EC)*, pages 43–60, 2016. 10.1.3

[17] Sepehr Assadi, MohammadHossein Bateni, Aaron Bernstein, Vahab Mirrokni, and Cliff Stein. Coresets meet edcs: algorithms for matching and vertex cover on massive graphs. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1616–1635, 2019. 12

[18] Sepehr Assadi, Mohammadhossein Bateni, and Vahab Mirrokni. Distributed weighted matching via randomized composable coresets. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 333–343, 2019. 10.9

[19] Yossi Azar, Ilan Reuven Cohen, and Alan Roytman. Online lower bounds via duality. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1038–1050, 2017. 5.1.2

[20] Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: Massive data summarization on the fly. In *Proceedings of the 20th International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 671–680, 2014. 11.1

[21] Bahman Bahmani and Michael Kapralov. Improved bounds for online stochastic matching. In *Proceedings of the 18th Annual European Symposium on Algorithms (ESA)*, pages 170–181. 2010. 1.1, 5.1, 5.1.2, 7.1.5, 8.1.2

[22] Bahman Bahmani, Aranyak Mehta, and Rajeev Motwani. Online graph edge-coloring in the random-order arrival model. *Theory of Computing*, 8(1):567–595, 2012. 1.2, **??**, 1.4, 6.1.4, 9, 9.1, 9.1, 9.3.1, 9.7

[23] Nikhil Bansal, Niv Buchbinder, Anupam Gupta, and Joseph (Seffi) Naor. An $O(log^2 k)$-

competitive algorithm for metric bipartite matching. In *Proceedings of the 15th Annual European Symposium on Algorithms (ESA)*, pages 522–533, 2007. 8.1, 8.9

[24] Nikhil Bansal, Anupam Gupta, Jian Li, Julián Mestre, Viswanath Nagarajan, and Atri Rudra. When lp is the cure for your matching woes: Improved bounds for stochastic matchings. *Algorithmica*, 63(4):733–762, 2012. 5.1.1

[25] Amotz Bar-Noy, Rajeev Motwani, and Joseph Naor. The greedy algorithm is optimal for on-line edge coloring. *Information Processing Letters (IPL)*, 44(5):251–253, 1992. 1.1, 1.2, 1.4, 6.1.1, 6.1.1, 6.1.2, 6.1.2, 6.1.4, 6.6, 9, 9.1, 9.4, 9.7, 12

[26] Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph Naor, and Baruch Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM (JACM)*, 48(5):1069–1090, 2001. 11.2.1

[27] Reuven Bar-Yehuda and Dror Rawitz. On the equivalence between the primal-dual schema and the local ratio technique. *SIAM Journal on Discrete Mathematics*, 19(3):762–797, 2005. 2

[28] Surender Baswana, Manoj Gupta, and Sandeep Sen. Fully dynamic maximal matching in $O(\log n)$ update time. In *Proceedings of the 52nd Symposium on Foundations of Computer Science (FOCS)*, pages 383–392, 2011. 10.1.3

[29] Surender Baswana, Sumeet Khurana, and Soumojit Sarkar. Fully dynamic randomized algorithms for graph spanners. *ACM Transactions on Algorithms (TALG)*, 8(4):35, 2012. 10.1

[30] Soheil Behnezhad, Jakub Łącki, and Vahab Mirrokni. Fully dynamic matching: Beating 2-approximation in Δ^{ε} update time. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2492–2508, 2020. 10.1.1, 10.1.3

[31] Daniel Berend and Aryeh Kontorovich. A sharp estimate of the binomial mean absolute deviation with applications. *Statistics & Probability Letters*, 83(4):1254–1259, 2013. 8.4, 8.4

[32] Bonnie Berger and John Rompel. Simulating $(\log^c n)$-wise independence in nc. *Journal of the ACM (JACM)*, 38(4):1026–1046, 1991. 6.1

[33] Aaron Bernstein. Deterministic partially dynamic single source shortest paths in weighted graphs. In *Proceedings of the 44th International Colloquium on Automata, Languages and Programming (ICALP)*, 2017. 10.1

[34] Aaron Bernstein. Improved bound for matching in random-order streams. In *Proceedings of the 47th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 12:1–12:13, 2020. 12

[35] Aaron Bernstein and Shiri Chechik. Deterministic decremental single source shortest paths: beyond the $O(mn)$ bound. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC)*, pages 389–397, 2016. 10.1

[36] Aaron Bernstein and Shiri Chechik. Deterministic partially dynamic single source shortest paths for sparse graphs. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 453–469, 2017. 10.1

[37] Aaron Bernstein and Cliff Stein. Fully dynamic matching in bipartite graphs. In *Proceedings of the 42nd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 167–179, 2015. 10.1, 10.1.3, 12

[38] Aaron Bernstein and Cliff Stein. Faster fully dynamic matchings with small approximation ratios. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 692–711, 2016. **??**, 10.1.3, 12

[39] Aaron Bernstein, Sebastian Forster, and Monika Henzinger. A deamortization approach for dynamic spanner and dynamic maximal matching. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1899–1918. Society for Industrial and Applied Mathematics, 2019. 10.1, 10.1.3, 2

[40] Aaron Bernstein, Maximillian Probst Gutenberg, and Thatchaphol Saranurak. Deterministic decremental reachability, SCC, and shortest paths via directed expanders and congestion balancing. In *Proceedings of the 61st Symposium on Foundations of Computer Science (FOCS)*, 2020. To Appear. 1.3, 10.9

[41] Sayan Bhattacharya and Janardhan Kulkarni. Deterministically maintaining a $(2+\varepsilon)$-approximate minimum vertex cover in $O(1/\varepsilon^2)$ amortized update time. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1872–1885, 2019. 10.1.1, 10.1.2, 10.1.3, 10.5.2, 10.8, 10.8

[42] Sayan Bhattacharya, Monika Henzinger, and Giuseppe F Italiano. Deterministic fully dynamic data structures for vertex cover and matching. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 785–804, 2015. **??**

[43] Sayan Bhattacharya, Monika Henzinger, and Danupon Nanongkai. New deterministic approximation algorithms for fully dynamic matching. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC)*, pages 398–411, 2016. **??**, 10.1.1, 10.1.3, 10.5.1, 10.5.2

[44] Sayan Bhattacharya, Deeparnab Chakrabarty, and Monika Henzinger. Deterministic fully dynamic approximate vertex cover and fractional matching in $O(1)$ amortized update time. In *Proceedings of the 19th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 86–98, 2017. 10.1.3, 10.8

[45] Sayan Bhattacharya, Monika Henzinger, and Danupon Nanongkai. Fully dynamic approximate maximum matching and minimum vertex cover in $O(\log^3 n)$ worst case update time. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 470–489, 2017. 10.1.1, 10.1.2, 10.1.2, 10.1.3, 10.3, 10.5.2, 10.8

[46] Sayan Bhattacharya, Deeparnab Chakrabarty, Monika Henzinger, and Danupon Nanongkai. Dynamic algorithms for graph coloring. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1–20, 2018. 6.1, 10.1.2, 10.2, 10.3, 10.3, 10.3

[47] Sayan Bhattacharya, Monika Henzinger, and Giuseppe Italiano. Dynamic algorithms via the primal-dual method. *Information and Computation*, 261:219–239, 2018. 10.1.3, 10.8

[48] Sayan Bhattacharya, Monika Henzinger, and Giuseppe F Italiano. Deterministic fully

dynamic data structures for vertex cover and matching. *SIAM Journal on Computing (SICOMP)*, 47(3):859–887, 2018. 10.1, 10.1.3, 10.5.2, 10.5.2

[49] Benjamin Birnbaum and Claire Mathieu. On-line bipartite matching made simple. *ACM SIGACT News*, 39(1):80–87, 2008. 7.1

[50] Brian Brubach, Karthik Abinav Sankararaman, Aravind Srinivasan, and Pan Xu. New algorithms, better bounds, and a novel model for online stochastic matching. In *Proceedings of the 24th Annual European Symposium on Algorithms (ESA)*, pages 24:1–24:16, 2016. 8.1.2

[51] Niv Buchbinder and Moran Feldman. Constrained submodular maximization via a non-symmetric technique. *Mathematics of Operations Research*, 44(3):988–1005, 2019. 11.1

[52] Niv Buchbinder and Joseph (Seffi) Naor. The design of competitive online algorithms via a primal: dual approach. *Foundations and Trends® in Theoretical Computer Science (TCS)*, 3(2–3):93–263, 2009. 1.4.1, 7.1.4

[53] Niv Buchbinder, Kamal Jain, and Joseph (Seffi) Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *Proceedings of the 15th Annual European Symposium on Algorithms (ESA)*, pages 253–264. 2007. 7.1, 7.1, 7.1.2, 7.1.2, 7.1.3, 7.1.5

[54] Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1433–1452, 2014. 11.1, 11.2.1, 11.3.2

[55] Niv Buchbinder, Danny Segev, and Yevgeny Tkach. Online algorithms for maximum cardinality matching with edge arrivals. *Algorithmica*, pages 1–19, Aug 2018. 1.1, **??**, **??**, **??**, 3.1, 3.2, 4.1, 5.1.2, 12

[56] Jarosław Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median, and positive correlation in budgeted optimization. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 737–756, 2014. 5.1.1

[57] Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing (SICOMP)*, 40(6):1740–1766, 2011. 5.1.1, 11.1

[58] Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: Matchings, matroids, and more. *Mathematical Programming*, 154(1-2):225–247, 2015. (document), 1.3, **??**, **??**, **??**, 11.1, 11.2, 11.2, 11.2, 11.2.1, 11.8, 17, 11.8.1, 11.8.2

[59] Minjun Chang, Dorit S Hochbaum, Quico Spaen, and Mark Velednitsky. An optimally-competitive algorithm for maximum online perfect bipartite matching with iid arrivals. *Theory of Computing Systems*, pages 1–17, 2019. 8.1, 8.8, 8.8, 8.8

[60] Yi-Jun Chang, Qizheng He, Wenzheng Li, Seth Pettie, and Jara Uitto. The complexity of distributed edge coloring with small palettes. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2633–2652, 2018. 9.1, 9.1.1

[61] Yi-Jun Chang, Wenzheng Li, and Seth Pettie. An optimal distributed $(\Delta + 1)$-coloring algorithm? In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing*

(STOC), pages 445–456, 2018. 6.1

[62] Moses Charikar and Shay Solomon. Fully dynamic almost-maximal matching: Breaking the polynomial barrier for worst-case time bounds. In *Proceedings of the 45th International Colloquium on Automata, Languages and Programming (ICALP)*, 2018. 1.3, 10.1, 10.1.3

[63] Chandra Chekuri, Jan Vondrak, and Rico Zenklusen. Dependent randomized rounding via exchange properties of combinatorial structures. In *Proceedings of the 51st Symposium on Foundations of Computer Science (FOCS)*, pages 575–584, 2010. 5.1.1

[64] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Multi-budgeted matchings and matroid intersection via dependent rounding. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1080–1097, 2011. 5.1.1

[65] Chandra Chekuri, Shalmoli Gupta, and Kent Quanrud. Streaming algorithms for submodular function maximization. In *Proceedings of the 42nd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 318–330, 2015. 1.3, 11.1, 11.2, 11.3

[66] Ashish Chiplunkar, Sumedh Tirodkar, and Sundar Vishwanathan. On randomized algorithms for matching in the online preemptive model. In *Proceedings of the 23rd Annual European Symposium on Algorithms (ESA)*, pages 325–336. 2015. 1.1, 4.1

[67] Julia Chuzhoy and Sanjeev Khanna. A new algorithm for decremental single-source shortest paths with applications to vertex-capacitated flow and cut problems. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*, pages 389–400, 2019. 10.1

[68] Julia Chuzhoy and Thatchaphol Saranurak. On dynamic shortest paths with adaptive adversary. Unpublished manuscript, 2019. 10.1

[69] Ilan Reuven Cohen and David Wajc. Randomized online matching in regular graphs. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 960–979, 2018. 1.5, 4.1.1, 5, 8.1.2

[70] Ilan Reuven Cohen, Binghui Peng, and David Wajc. Tight bounds for online edge coloring. In *Proceedings of the 60th Symposium on Foundations of Computer Science (FOCS)*, pages 1–25, 2019. 1.5, 6, 6.3.2, 9.1

[71] Richard Cole and John Hopcroft. On edge coloring bipartite graphs. *SIAM Journal on Computing (SICOMP)*, 11(3):540–546, 1982. 5.1

[72] Richard Cole, Kirstin Ost, and Stefan Schirra. Edge-coloring bipartite multigraphs in $O(E \log D)$ time. *Combinatorica*, 21(1):5–12, 2001. 5.1, 6.1, 9.1

[73] Michael Crouch and Daniel M Stubbs. Improved streaming algorithms for weighted matching, via unweighted matching. In *Proceedings of the 17th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, page 96, 2014. **??**, 11.1, 11.2, 11.7

[74] J Csima and László Lovász. A matching algorithm for regular bipartite graphs. *Discrete Applied Mathematics*, 35(3):197–203, 1992. 5.1

[75] Søren Dahlgaard. On the hardness of partially dynamic graph problems and connections to diameter. In *Proceedings of the 43rd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 48:1–48:14, 2016. 10.1.3

[76] Sina Dehghani, Soheil Ehsani, MohammadTaghi Hajiaghayi, Vahid Liaghat, and Saeed Seddighin. Stochastic k-server: How should uber work? In *Proceedings of the 44th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 126:1–126:14, 2017. 8.1.2

[77] Nikhil R Devanur and Thomas P Hayes. The adwords problem: online keyword matching with budgeted bidders under random permutations. In *Proceedings of the 10th ACM Conference on Electronic Commerce (EC)*, pages 71–78, 2009. 7.1.5

[78] Nikhil R Devanur, Balasubramanian Sivan, and Yossi Azar. Asymptotically optimal algorithm for stochastic adwords. In *Proceedings of the 13th ACM Conference on Electronic Commerce (EC)*, pages 388–404, 2012. 7.1.2, 7.1.5, 8.1.2

[79] Nikhil R Devanur, Kamal Jain, and Robert D Kleinberg. Randomized primal-dual analysis of ranking for online bipartite matching. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 101–107, 2013. 1.4.1, 2.3.3, 5.1.1, 5.1.2, 7.1, 7.1.2, 7.1.3, 11.2.1, 11.3.2

[80] John P Dickerson, Karthik Abinav Sankararaman, Aravind Srinivasan, and Pan Xu. Balancing relevance and diversity in online bipartite matching via submodularity. In *Proceedings of the 53rd AAAI Conference on Artificial Intelligence (AAAI)*, volume 33, pages 1877–1884, 2019. 11.1

[81] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 624–633, 2014. 11.1, 11.2.1, 11.9, 11.12

[82] Ran Duan, Haoqing He, and Tianyi Zhang. Dynamic edge coloring with improved approximation. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1937–1945, 2019. 6.1, 9.1, 9.1, 10.2

[83] Devdatt Dubhashi and Desh Ranjan. Balls and bins: A study in negative dependence. *BRICS Report Series*, 3(25), 1996. 2.4.2, 2.4.4, 2.4.1

[84] Devdatt Dubhashi, David A Grable, and Alessandro Panconesi. Near-optimal, distributed edge colouring via the nibble method. *Theoretical Computer Science (TCS)*, 203(2):225–252, 1998. 6.1, 9.1.1, 9.2

[85] Devdatt P Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. 2009. 9.6.19, 9.6.20, 9.6.7

[86] Alon Eden, Michal Feldman, Amos Fiat, and Kineret Segal. An economic-based analysis of ranking for online bipartite matching. *arXiv preprint arXiv:1804.06637*, 2018. 7.1

[87] Jack Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of research of the National Bureau of Standards B*, 69(125-130):55–56, 1965. 1, 2.3.5

[88] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17(3):449–467, 1965. 1

[89] Jeno Egerváry. Matrixok kombinatorius tulajdonságairól. *Matematikai és Fizikai Lapok*, 38(1931):16–28, 1931. 2.3.2

[90] Martin R Ehmsen, Lene M Favrholdt, Jens S Kohrt, and Rodica Mihai. Comparing first-fit and next-fit for online edge coloring. *Theoretical Computer Science (TCS)*, 411(16-18): 1734–1741, 2010. 6.1.4

[91] Michael Elkin, Seth Pettie, and Hsin-Hao Su. (2δ—l)-edge-coloring is much easier than maximal matching in the distributed setting. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 355–370, 2014. 9.1.1

[92] Michael Elkin, Seth Pettie, and Hsin-Hao Su. (2Δ-1)-edge-coloring is much easier than maximal matching in the distributed setting. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 355–370, 2015. 6.1

[93] Alina Ene and Huy L Nguyen. Constrained submodular maximization: Beyond 1/e. In *Proceedings of the 57th Symposium on Foundations of Computer Science (FOCS)*, pages 248–257, 2016. 11.1

[94] David Eppstein, Zvi Galil, Giuseppe F Italiano, and Amnon Nissenzweig. Sparsification – a technique for speeding up dynamic graph algorithms. *Journal of the ACM (JACM)*, 44 (5):669–696, 1997. 10.1.3

[95] Leah Epstein, Asaf Levin, Danny Segev, and Oren Weimann. Improved bounds for online preemptive matching. In *Proceedings of the 30th International Symposium on Theoretical Aspects of Computer Science (STACS)*, page 389, 2013. 1.1, 3.1, 5.1.2, 11.1

[96] Hossein Esfandiari, Nitish Korula, and Vahab S. Mirrokni. Online allocation with traffic spikes: Mixing adversarial and stochastic models. In *Proceedings of the 16th ACM Conference on Economics and Computation (EC)*, pages 169–186, 2015. 8.1.2

[97] Matthew Fahrbach, Zhiyi Huang, Runzhou Tao, and Morteza Zadimoghaddam. Edge-weighted online bipartite matching. In *Proceedings of the 61st Symposium on Foundations of Computer Science (FOCS)*, 2020. To Appear. 1.4.1, 11.2.1, 11.3.2

[98] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, 2004. 8.3, 8.7.1

[99] Alireza Farhadi, Mohammad Taghi Hajiaghayi, Tung Mah, Anup Rao, and Ryan A Rossi. Approximate maximum matching in random streams. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1773–1785, 2020. 12

[100] Lene M Favrholdt and Jesper W Mikkelsen. Online dual edge coloring of paths and trees. In *Proceedings of the 12th Workshop on Approximation and Online Algorithms (WAOA)*, pages 181–192, 2014. 6.1.4

[101] Monrad Favrholdt and Nyhave Nielsen. On-line edge-coloring with a fixed number of colors. *Algorithmica*, 35(2):176–191, 2003. 6.1.4

[102] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998. 11.1, 11.2, 11.2.1, 11.9

[103] Uriel Feige. Tighter bounds for online bipartite matching. In *Building Bridges II*, pages

235–255. 2019. 4.1, 7.1

[104] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2-3): 207–216, 2005. 11.1, 4

[105] Björn Feldkord, Matthias Feldotto, Anupam Gupta, Guru Guruganesh, Amit Kumar, Sören Riechers, and David Wajc. Fully-dynamic bin packing with little repacking. In *Proceedings of the 45th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 51:1–51:24, 2018. 1.5

[106] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and S Muthukrishnan. Online stochastic matching: Beating 1-1/e. In *Proceedings of the 50th Symposium on Foundations of Computer Science (FOCS)*, pages 117–126, 2009. 7.1.5, 8.1.2, 8.8

[107] Jon Feldman, Monika Henzinger, Nitish Korula, Vahab S Mirrokni, and Cliff Stein. Online stochastic packing applied to display ad allocation. In *Proceedings of the 18th Annual European Symposium on Algorithms (ESA)*, pages 182–194. 2010. 7.8

[108] Moran Feldman, Joseph Naor, and Roy Schwartz. A unified continuous greedy algorithm for submodular maximization. In *Proceedings of the 52nd Symposium on Foundations of Computer Science (FOCS)*, pages 570–579, 2011. 11.1

[109] Moran Feldman, Joseph Seffi Naor, Roy Schwartz, and Justin Ward. Improved approximations for k-exchange systems. In *Proceedings of the 19th Annual European Symposium on Algorithms (ESA)*, pages 784–798, 2011. 11.1, 11.13

[110] Moran Feldman, Ola Svensson, and Rico Zenklusen. Online contention resolution schemes. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1014–1033, 2016. 5.1.1

[111] Moran Feldman, Amin Karbasi, and Ehsan Kazemi. Do less, get more: streaming submodular maximization with subsampling. In *Proceedings of the 31st Annual Conference on Neural Information Processing Systems (NIPS)*, pages 732–742, 2018. (document), 1.3, **??**, 11.1, 11.2, 11.2, 11.2.1, 11.3, 11.8, 17, 11.8.1, 11.8.2

[112] Moran Feldman, Ashkan Norouzi-Fard, Ola Svensson, and Rico Zenklusen. The one-way communication complexity of submodular maximization with applications to streaming and robustness. In *Proceedings of the 52nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 1363–1374, 2020. 11.1, 1

[113] Manuela Fischer, Mohsen Ghaffari, and Fabian Kuhn. Deterministic distributed edge-coloring via hypergraph maximal matching. In *Proceedings of the 58th Symposium on Foundations of Computer Science (FOCS)*, pages 180–191, 2017. 6.1

[114] Marshall L Fisher, George L Nemhauser, and Laurence A Wolsey. An analysis of approximations for maximizing submodular set functions—ii. In *Polyhedral combinatorics*, pages 73–87. Springer, 1978. 11.1

[115] Lisa K Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM Journal on Discrete Mathematics*, 13(4):505–520, 2000. 10.1

[116] Sebastian Forster and Gramoz Goranci. Dynamic low-stretch trees via dynamic low-

diameter decompositions. pages 377–388, 2019. 10.1

[117] Harold N Gabow. Data structures for weighted matching and extensions to b-matching and f-factors. *ACM Transactions on Algorithms (TALG)*, 14(3):1–80, 2018. 11.7

[118] Harold N Gabow, Takao Nishizeki, Oded Kariv, Daniel Leven, and Terada Osmau. Algorithms for edge-coloring graphs. Technical report, Tohoku University, 1985. 6.1

[119] Buddhima Gamlath, Sagar Kale, Slobodan Mitrović, and Ola Svensson. Weighted matchings via unweighted augmentations. In *Proceedings of the 38th ACM Symposium on Principles of Distributed Computing (PODC)*, 2019. 12

[120] Buddhima Gamlath, Michael Kapralov, Andreas Maggiori, Ola Svensson, and David Wajc. Online matching with general arrivals. In *Proceedings of the 60th Symposium on Foundations of Computer Science (FOCS)*, pages 26–37, 2019. 1.5, 3, 4, 5.1.2, 8.1.2

[121] Shashidhar Gandham, Milind Dawande, and Ravi Prakash. Link scheduling in wireless sensor networks: Distributed edge-coloring revisited. *Journal of Parallel and Distributed Computing*, 68(8):1122–1134, 2008. 6.1

[122] Rajiv Gandhi, Samir Khuller, Srinivasan Parthasarathy, and Aravind Srinivasan. Dependent rounding and its applications to approximation algorithms. *Journal of the ACM (JACM)*, 53(3):324–360, 2006. 1.1, 2.3.4, 4.1.1, 5.1.1, 12

[123] Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979. 2.3.3

[124] Naveen Garg and Jochen Koenemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM Journal on Computing (SICOMP)*, 37 (2):630–652, 2007. 10.1

[125] Naveen Garg, Anupam Gupta, Stefano Leonardi, and Piotr Sankowski. Stochastic analyses for online combinatorial optimization problems. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 942–951, 2008. 8.1.2

[126] RC Geary. The ratio of the mean deviation to the standard deviation as a test of normality. *Biometrika*, 27(3/4):310–332, 1935. 5.2

[127] Mohsen Ghaffari and David Wajc. Simplified and space-optimal semi-streaming $(2+\varepsilon)$-approximate matching. In *Proceedings of the 2nd Symposium on Simplicity in Algorithms (SOSA)*, 2019. 1.3, 1.5, 10.9, 11.1, 11.2.1, 11.7, 12

[128] Mohsen Ghaffari, Themis Gouleakis, Christian Konrad, Slobodan Mitrović, and Ronitt Rubinfeld. Improved massively parallel computation algorithms for mis, matching, and vertex cover. In *Proceedings of the 37th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 129–138, 2018. 10.9

[129] Mohsen Ghaffari, Fabian Kuhn, Yannic Maus, and Jara Uitto. Deterministic distributed edge-coloring with fewer colors. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC)*, pages 418–430, 2018. 6.1

[130] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. Perfect matchings in $O(n^{1.5})$ time in regular bipartite graphs. *arXiv preprint arXiv:0902.1617*, 2009. 5.1

[131] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. Perfect matchings via uniform sampling in regular bipartite graphs. *ACM Transactions on Algorithms (TALG)*, 6(2):27, 2010. 5.1

[132] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 468–485, 2012. 10.1.3, 11.2.1, 11.9, 11.9.4, 11.9

[133] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. Perfect matchings in $O(n \log n)$ time in regular bipartite graphs. *SIAM Journal on Computing (SICOMP)*, 42(3):1392–1404, 2013. 5.1, 6.1

[134] Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 982–991, 2008. 2.3.3, 7.1, 7.1.5, 7.8, 8.1.2

[135] Michel X Goemans and David P Williamson. The primal-dual method for approximation algorithms and its application to network design problems. pages 144–191. 1997. 1.4.1

[136] Mark K Goldberg. On multigraphs of almost maximal chromatic class. *Diskret. Analiz*, 23(3):7, 1973. 6.3.1

[137] David A. Grable. A large deviation inequality for functions of independent, multi-way choices. *Combinatorics, Probability and Computing*, 7(1):57–63, 1998. 9.1.1

[138] Fabrizio Grandoni, Anupam Gupta, Stefano Leonardi, Pauli Miettinen, Piotr Sankowski, and Mohit Singh. Set covering with our eyes closed. *SIAM Journal on Computing (SICOMP)*, 42(3):808–830, 2013. 8.1.2

[139] Anupam Gupta and Roie Levin. The online submodular cover problem. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1525–1537, 2020. 11.3

[140] Anupam Gupta and Kevin Lewi. The online metric matching problem for doubling metrics. In *Proceedings of the 39th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 424–435, 2012. 8.1

[141] Anupam Gupta and Sahil Singla. Random-order models. *arXiv preprint arXiv:2002.12159*, 2020. 9.1

[142] Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained non-monotone submodular maximization: Offline and secretary algorithms. In *Proceedings of the 6th Conference on Web and Internet Economics (WINE)*, pages 246–257, 2010. 11.1

[143] Anupam Gupta, Ravishankar Krishnaswamy, Amit Kumar, and Debmalya Panigrahi. Online and dynamic algorithms for set cover. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing (STOC)*, pages 537–550, 2017. 1.4.1, 10.1.3, 10.8, 11.2.1, 11.3.2

[144] Anupam Gupta, Guru Guruganesh, Binghui Peng, and David Wajc. Stochastic online metric matching. In *Proceedings of the 46th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 67:1–67:14, 2019. 1.5, 8

[145] Manoj Gupta and Richard Peng. Fully dynamic $(1+\varepsilon)$-approximate matchings. In *Proceedings of the 54th Symposium on Foundations of Computer Science (FOCS)*, pages 548–557, 2013. 10.1, 10.1.2, 10.1.3

[146] Guru Prashanth Guruganesh and Sahil Singla. Online matroid intersection: Beating half for random arrival. In *Proceedings of the 19th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 241–253, 2017. 1.1, 3.1, 12

[147] Maximilian Probst Gutenberg and Christian Wulff-Nilsen. Decremental sssp in weighted digraphs: Faster and against an adaptive adversary. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2542–2561, 2020. 10.1

[148] Maximilian Probst Gutenberg and Christian Wulff-Nilsen. Deterministic algorithms for decremental approximate shortest paths: Faster and simpler. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2522–2541, 2020. 10.1

[149] Bernhard Haeupler and David Wajc. A faster distributed radio broadcast primitive. In *Proceedings of the 35th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 361–370, 2016. 1.5

[150] Bernhard Haeupler, Vahab S Mirrokni, and Morteza Zadimoghaddam. Online stochastic weighted matching: Improved approximation algorithms. In *Proceedings of the 7th Conference on Web and Internet Economics (WINE)*, pages 170–181. 2011. 7.1.5, 8.8

[151] Bernhard Haeupler, D Ellis Hershkowitz, and David Wajc. Round-and message-optimal distributed graph algorithms. In *Proceedings of the 37th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 119–128, 2018. 1.5

[152] Bernhard Haeupler, D Ellis Hershkowitz, and David Wajc. Near-optimal schedules for simultaneous multicasts. *arXiv preprint arXiv:2001.00072*, 2019. 1.5

[153] Bernhard Haeupler, David Wajc, and Goran Zuzic. Network coding gaps for completion times of multiple unicasts. In *Proceedings of the 61st Symposium on Foundations of Computer Science (FOCS)*, 2020. To Appear. 1.5

[154] Bernhard Haeupler, David Wajc, and Goran Zuzic. Shortcuts are universal lower bounds for distributed optimization. 2020. 1.5

[155] Philip Hall. On representatives of subsets. *Journal of the London Mathematical Society*, 1(1):26–30, 1935. 5.1

[156] Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC)*, pages 21–30, 2015. 10.1.3

[157] Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. Dynamic approximate all-pairs shortest paths: Breaking the $O(mn)$ barrier and derandomization. *SIAM Journal on Computing (SICOMP)*, 45(3):947–1006, 2016. 10.1

[158] Monika R Henzinger and Valerie King. Randomized fully dynamic graph algorithms with polylogarithmic time per operation. *Journal of the ACM (JACM)*, 46(4):502–516, 1999.

10.1

[159] Jacob Holm, Kristian De Lichtenberg, and Mikkel Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *Journal of the ACM (JACM)*, 48(4):723–760, 2001. 10.1

[160] Ian Holyer. The np-completeness of edge-coloring. *SIAM Journal on Computing (SICOMP)*, 10(4):718–720, 1981. 6.1, 9.1

[161] John E Hopcroft and Richard M Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973. 10.4.3

[162] Zhiyi Huang and Qiankun Zhang. Online primal dual meets online matching with stochastic rewards: configuration lp to the rescue. In *Proceedings of the 52nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 1153–1164, 2020. 1.4.1, 11.2.1, 11.3.2, 12

[163] Zhiyi Huang, Ning Kang, Zhihao Gavin Tang, Xiaowei Wu, Yuhao Zhang, and Xue Zhu. How to match when all vertices arrive online. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC)*, pages 17–29, 2018. 1.1, 4.1, 8.1.2, 12

[164] Zhiyi Huang, Zhihao Gavin Tang, Xiaowei Wu, and Yuhao Zhang. Online vertex-weighted bipartite matching: Beating 1-1/e with random arrivals. In *Proceedings of the 45th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 1070–1081, 2018.

[165] Zhiyi Huang, Binghui Peng, Zhihao Gavin Tang, Runzhou Tao, Xiaowei Wu, and Yuhao Zhang. Tight competitive ratios of classic matching algorithms in the fully online model. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2875–2886, 2019. 1.1, **??**, 3.1, 4.1, 5.1.2, 8.1.2, 12

[166] Zhiyi Huang, Zhihao Gavin Tang, Xiaowei Wu, and Yuhao Zhang. Fully online matching ii: Beating ranking and water-filling. In *Proceedings of the 61st Symposium on Foundations of Computer Science (FOCS)*, 2020. To Appear. 1.1, 1.4.1, 11.2.1, 11.3.2, 12

[167] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. 11.9

[168] Zoran Ivkovic and Errol L Lloyd. Fully dynamic maintenance of vertex cover. In *Proceedings of the 19th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, pages 99–111, 1993. 10.1.3

[169] CGJ Jacobi. About the research of the order of a system of arbitrary ordinary differential equations, 1890. 2

[170] Patrick Jaillet and Xin Lu. Online stochastic matching: New algorithms with better bounds. *Mathematics of Operations Research*, 2013. 7.1.5

[171] Kumar Joag-Dev and Frank Proschan. Negative association of random variables with applications. *The Annals of Statistics*, pages 286–295, 1983. 2.4.1, 2.4.3, 2.4.4, 10.4.2

[172] Bala Kalyanasundaram and Kirk Pruhs. Online weighted matching. *Journal of Algorithms*, 14(3):478–488, 1993. 1.2, 8.1

[173] Bala Kalyanasundaram and Kirk R Pruhs. An optimal deterministic algorithm for online b-matching. *Theoretical Computer Science (TCS)*, 233(1):319–325, 2000. 5.1.1, 5.1.2, 7.1

[174] Michael Kapralov. Better bounds for matchings in the streaming model. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1679–1697, 2013. **??**, 11.1, 11.2, 11.9

[175] Bruce M Kapron, Valerie King, and Ben Mountjoy. Dynamic graph connectivity in polylogarithmic worst case time. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1131–1142, 2013. 10.1

[176] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. Online bipartite matching with unknown distributions. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 587–596, 2011. 1.1, 5.1, 5.1.2, 7.1.5, 8.1.2, 9.1

[177] Howard J. Karloff and David B. Shmoys. Efficient parallel algorithms for edge coloring problems. *J. Algorithms*, 8(1):39–52, 1987. 6.1, 9.1

[178] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. 1972. 2.3.1

[179] Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. An optimal algorithm for online bipartite matching. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 352–358, 1990. 1.1, **??**, 1.4, 2.3.3, 3.1, 4, 4.1, 5.1, 5.1.2, 6, 7.1, 7.1, 8.1.2, 12

[180] Ehsan Kazemi, Marko Mitrovic, Morteza Zadimoghaddam, Silvio Lattanzi, and Amin Karbasi. Submodular streaming in all its glory: Tight approximation, minimum memory and low adaptive complexity. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 3311–3320, 2019. 11.1

[181] Thomas Kesselheim, Andreas Tönnis, Klaus Radke, and Berthold Vöcking. Primal beats dual on online packing lps in the random-order model. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 303–312, 2014. 9.1

[182] Samir Khuller, Stephen G Mitchell, and Vijay V Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. *Theoretical Computer Science (TCS)*, 127(2): 255–267, 1994. 1.2, 8.1

[183] Alam Khursheed and KM Lai Saxena. Positive dependence in multivariate distributions. *Communications in Statistics - Theory and Methods*, 10(12):1183–1196, 1981. 2.4.1, 2.4.4

[184] Dénes König. Über graphen und ihre anwendung auf determinantentheorie und mengenlehre. *Mathematische Annalen*, 77(4):453–465, 1916. 1.1, 2.1, 5.1, 6.1, 6.2.4, 9.1

[185] Dénes König. Gráfok és mátrixok. *Matematikai és Fizikai Lapok*, 38(1931):116–119, 1931. 2.1

[186] Christian Konrad. A simple augmentation method for matchings with applications to streaming algorithms. In *Proceedings of the 43rd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, 2018. 12

[187] Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum matching in semi-

streaming with few passes. In *Proceedings of the 15th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 231–242, 2012. 12

[188] Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3sum conjecture. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1272–1287, 2016. 10.1.3

[189] Nitish Korula, Vahab Mirrokni, and Morteza Zadimoghaddam. Online submodular welfare maximization: Greedy beats 1/2 in random order. *SIAM Journal on Computing (SICOMP)*, 47(3):1056–1086, 2018. 9.1, 11.1

[190] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 1, 1.4.1

[191] Euiwoong Lee and Sahil Singla. Maximum matching in the online batch-arrival model. In *Proceedings of the 19th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, pages 355–367, 2017. 1.1, **??**, **??**, 3.1, 3.2, 10.1.3, 12

[192] Jon Lee, Vahab S Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Non-monotone submodular maximization under matroid and knapsack constraints. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 323–332, 2009. 11.1

[193] Jon Lee, Vahab S Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Maximizing nonmonotone submodular functions under matroid or knapsack constraints. *SIAM Journal on Discrete Mathematics*, 23(4):2053–2078, 2010. 11.1

[194] Jon Lee, Maxim Sviridenko, and Jan Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. *Mathematics of Operations Research*, 35 (4):795–806, 2010. 11.1, 11.13

[195] Benny Lehmann, Daniel Lehmann, and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, 55(2):270–296, 2006. 11.1

[196] Gavriela Freund Lev, Nicholas Pippenger, and Leslie G Valiant. A fast parallel algorithm for routing in permutation networks. *IEEE transactions on Computers*, (2):93–100, 1981. 6.1

[197] Roie Levin and David Wajc. Streaming submodular matching meets the primal-dual method. *arXiv preprint arXiv:2008.10062*, 2020. 11

[198] Hui Lin and Jeff Bilmes. Word alignment via submodular maximization over matroids. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL)*, pages 170–175, 2011. 11.1

[199] Zvi Lotker, Boaz Patt-Shamir, and Seth Pettie. Improved distributed approximate matching. *Journal of the ACM (JACM)*, 62(5):38, 2015. 10.9

[200] László Lovász and Michael D Plummer. *Matching theory*, volume 367. American Mathematical Society, 2009. 1, 6.3.1

[201] Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps. In *Proceedings of the 43rd Annual ACM*

Symposium on Theory of Computing (STOC), pages 597–606, 2011. 7.1.5, 8.1.2, 9.1

[202] Mohammad Mahdian, Hamid Nazerzadeh, and Amin Saberi. Allocating online advertisement space with unreliable estimates. In *Proceedings of the 8th ACM Conference on Electronic Commerce (EC)*, pages 288–294, 2007. 7.1.5, 8.1.2

[203] Vahideh H Manshadi, Shayan Oveis Gharan, and Amin Saberi. Online stochastic matching: Online actions based on offline statistics. *Mathematics of Operations Research*, 37 (4):559–573, 2012. 7.1.5

[204] Pasin Manurangsi. Tight running time lower bounds for strong inapproximability of maximum k-coverage, unique set cover and related problems (via t-wise agreement testing theorem). In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 62–81, 2020. 11.1, 11.9

[205] Andrew McGregor. Finding graph matchings in data streams. In *Proceedings of the 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 170–181. 2005. (document), 11.1, 11.8, 11.8.1, 4

[206] Aranyak Mehta. Online matching and ad allocation. *Foundations and Trends® in Theoretical Computer Science*, 8(4):265–368, 2013. 1.1, 3.1, 7.1, 7.8, 8.1.2, 8.9

[207] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *Journal of the ACM (JACM)*, 54(5):22, 2007. 1.2, 2.3.3, 7.1, 7.1, 7.1.2, 7.1.3, 7.8, 8.1.2

[208] Adam Meyerson. Online facility location. In *Proceedings of the 42nd Symposium on Foundations of Computer Science (FOCS)*, pages 426–431, 2001. 9.1

[209] Adam Meyerson, Akash Nanavati, and Laura Poplawski. Randomized online algorithms for minimum metric bipartite matching. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 954–959, 2006. 8.1

[210] Silvio Micali and Vijay V Vazirani. An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In *Proceedings of the 21st Symposium on Foundations of Computer Science (FOCS)*, pages 17–27, 1980. 10.1.2, 10.4.3

[211] Jesper W Mikkelsen. Optimal online edge coloring of planar graphs with advice. In *International Conference on Algorithms and Complexity (CIAC)*, pages 352–364. Springer, 2015. 6.1.4

[212] Jesper W Mikkelsen. Randomization can be as helpful as a glimpse of the future in online computation. In *Proceedings of the 43rd International Colloquium on Automata, Languages and Programming (ICALP)*, page 39, 2016. 6.1.4

[213] Vahab S Mirrokni, Shayan Oveis Gharan, and Morteza Zadimoghaddam. Simultaneous approximations for adversarial and stochastic online budgeted allocation. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1690–1701, 2012. 8.1.2

[214] Baharan Mirzasoleiman, Stefanie Jegelka, and Andreas Krause. Streaming non-monotone submodular maximization: Personalized video summarization on the fly. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, pages 1379–1386. 11.1

[215] Jayadev Misra and David Gries. A constructive proof of vizing's theorem. In *Information Processing Letters (IPL)*, 1992. 6.1

[216] Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge university press, 2005. 2.4, 8.3.1, 8.3.1

[217] Aleksander Mądry. Faster approximation schemes for fractional multicommodity flow problems via dynamic graph algorithms. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 121–130, 2010. 1.3, 10.1

[218] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Cambridge University Press, 2010. 5.1

[219] Rajeev Motwani, Joseph (Seffi) Naor, and Moni Naor. The probabilistic method yields deterministic parallel algorithms. *Journal of Computer and System Sciences*, 49(3):478–516, 1994. 6.1, 9.1

[220] Danupon Nanongkai and Thatchaphol Saranurak. Dynamic spanning forest with worst-case update time: adaptive, las vegas, and $O(n^{1/2-\varepsilon})$-time. In *Proceedings of the 49th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1122–1129, 2017. 10.1

[221] Danupon Nanongkai, Thatchaphol Saranurak, and Christian Wulff-Nilsen. Dynamic minimum spanning forest with subpolynomial worst-case update time. In *Proceedings of the 58th Symposium on Foundations of Computer Science (FOCS)*, pages 950–961, 2017. 10.1

[222] Joseph Seffi Naor and David Wajc. Near-optimum online ad allocation for targeted advertising. *ACM Transactions on Economics and Computation (TEAC)*, 6(3-4):16, 2018. 1.5, 7, 8.1.2

[223] Krati Nayyar and Sharath Raghvendra. An input sensitive online algorithm for the metric bipartite matching problem. In *Proceedings of the 58th Symposium on Foundations of Computer Science (FOCS)*, pages 505–515, 2017. 8.1

[224] Ofer Neiman and Shay Solomon. Simple deterministic algorithms for fully dynamic maximal matching. *ACM Transactions on Algorithms (TALG)*, 12(1):7, 2016. 10.1, 10.1.2, 10.1.3

[225] George L Nemhauser and Laurence A Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of operations research*, 3(3):177–188, 1978. 11.1, 11.2

[226] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14 (1):265–294, 1978. 11.1

[227] Ashkan Norouzi-Fard, Jakub Tarnawski, Slobodan Mitrovic, Amir Zandieh, Aidasadat Mousavifar, and Ola Svensson. Beyond 1/2-approximation for submodular maximization on massive data streams. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 3829–3838, 2018. 11.1, 1

[228] Krzysztof Onak and Ronitt Rubinfeld. Maintaining a large matching and a small vertex cover. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing*

(STOC), pages 457–464, 2010. 10.1.3

[229] Shayan Oveis Gharan and Jan Vondrák. Submodular maximization by simulated annealing. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1098–1116, 2011. 11.1

[230] Alessandro Panconesi and Aravind Srinivasan. Improved distributed algorithms for coloring and network decomposition problems. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing (STOC)*, pages 581–592, 1992. 5.3.4, 5.3.4

[231] Alessandro Panconesi and Aravind Srinivasan. Randomized distributed edge coloring via an extension of the chernoff–hoeffding bounds. *SIAM Journal on Computing (SICOMP)*, 26(2):350–368, 1997. 1.2, 6.1, 6.1.4

[232] Ami Paz and Gregory Schwartzman. A $(2+\varepsilon)$-approximation for maximum weight matching in the semi-streaming model. *ACM Transactions on Algorithms (TALG)*, 15(2):18, 2018. 1.3, 10.9, 11.1, 11.2, 11.2.1, 2, 11.7, 12

[233] David Peleg and Shay Solomon. Dynamic $(1+\varepsilon)$-approximate matchings: a density-sensitive approach. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 712–729, 2016. 10.1, 10.1.3

[234] Julius Petersen. Sur le théoreme de tait. *L'intermédiaire des Mathématiciens*, 5:225–227, 1898. 6.1, 9.1

[235] Leonard Brian Pitt. *A simple probabilistic approximation algorithm for vertex cover*. Yale University, Department of Computer Science, 1985. 2.3.3

[236] PricewaterhouseCoopers. IAB internet advertising revenue report – full year 2019 results & q1 2020 revenues, 2020. URL `https://www.iab.com/wp-content/uploads/2020/05/FY19-IAB-Internet-Ad-Revenue-Report_Final.pdf`. [Online; accessed 20-July-2020]. 7.1

[237] Ariel D Procaccia, David Wajc, and Hanrui Zhang. Approximation-variance tradeoffs in facility location games. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, pages 1185—-1192, 2018. 1.5

[238] Prabhakar Raghavan and Clark D Tompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987. 1.4.1, 2.3.4, 5.1.1

[239] Sharath Raghvendra. A robust and optimal online algorithm for minimum metric bipartite matching. In *Proceedings of the 19th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, volume 60, 2016. 1.2, **??**, 8.1, 8.1.2, 8.3, 8.9, 12

[240] Sharath Raghvendra. Optimal analysis of an online algorithm for the bipartite matching problem on a line. In *Proceedings of the 34th Symposium on Computational geometry (SoCG)*, pages 67:1–67:14, 2018. 8.1

[241] April Rasala and Gordon Wilfong. Strictly nonblocking wdm cross-connects. *SIAM Journal on Computing (SICOMP)*, 35(2):449–485, 2005. 6.1

[242] Tim Roughgarden. Beyond worst-case analysis. *Communications of the ACM (CACM)*,

62(3):88–96, 2019. 1.2

[243] Tim Roughgarden. Beyond the worst-case analysis of algorithms, 2020. 1.2, 12

[244] Piotr Sankowski. Faster dynamic matchings and vertex connectivity. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 118–126, 2007. 10.1.3

[245] Alexander Schrijver. Bipartite edge coloring in O(Δm) time. *SIAM Journal on Computing (SICOMP)*, 28(3):841–846, 1998. 5.1

[246] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998. 2.3

[247] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003. 5.1.1

[248] Paul D Seymour. On multi-colourings of cubic graphs, and conjectures of fulkerson and tutte. *Proceedings of the London Mathematical Society*, 3(3):423–460, 1979. 6.3.1

[249] Claude E Shannon. A theorem on coloring the lines of a network. *Journal of Mathematics and Physics*, 28(1-4):148–152, 1949. 6.1

[250] Noam Solomon and Shay Solomon. Reoptimization via gradual transformations. *arXiv preprint arXiv:1803.05825*, 2018. 2, 10.3

[251] Shay Solomon. Fully dynamic maximal matching in constant update time. In *Proceedings of the 57th Symposium on Foundations of Computer Science (FOCS)*, pages 325–334, 2016. 10.1.3

[252] Shay Solomon. Local algorithms for bounded degree sparsifiers in sparse graphs. In *Proceedings of the 9th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 52:1–52:19, 2018. 10.1.3

[253] Daniel Stubbs and Virginia Vassilevska Williams. Metatheorems for dynamic weighted matching. In *Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS)*, 2017. 10.1.1, 10.9

[254] Hsin-Hao Su and Hoa T. Vu. Towards the locality of vizing's theorem. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*, pages 355–364, 2019. 9.1

[255] P.G. Tait. Remarks on the colourings of maps. *Proc. R. Soc. Edinburgh*, 10:729, 1880. 6.1, 9.1

[256] Zhihao Gavin Tang, Xiaowei Wu, and Yuhao Zhang. Towards a better understanding of randomized greedy matching. In *Proceedings of the 52nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 1097–1110, 2020. 1.4.1, 11.2.1, 11.3.2

[257] Leandros Tassiulas and Anthony Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE transactions on automatic control*, 37(12):1936–1948, 1992. 6.1

[258] Mikkel Thorup. Near-optimal fully-dynamic graph connectivity. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 343–350, 2000.

10.1

[259] Sumedh Tirodkar and Sundar Vishwanathan. Maximum matching on trees in the online preemptive and the incremental dynamic graph models. In *Proceedings of the 23rd International Computing and Combinatorics Conference (COCOON)*, pages 504–515, 2017. 1.1, 3.1, 4.1

[260] Jan van den Brand, Danupon Nanongkai, and Thatchaphol Saranurak. Dynamic matrix inverse: Improved algorithms and matching conditional lower bounds. In *Proceedings of the 60th Symposium on Foundations of Computer Science (FOCS)*, pages 456–480, 2019. 10.1.3

[261] Vadim G Vizing. On an estimate of the chromatic class of a p-graph. *Diskret analiz*, 3: 25–30, 1964. 1.4.1, 2.1, 6.1, 9.1, 10.2, 10.5.2

[262] Jan Vondrák. Submodularity in combinatorial optimization. 2007. 11.3.1

[263] Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 67–74, 2008. 11.1

[264] Jan Vondrák. Symmetry and approximability of submodular maximization problems. *SIAM Journal on Computing (SICOMP)*, 42(1):265–304, 2013. 11.1

[265] David Wajc. Negative association: definition, properties, and applications. 2.4.1

[266] David Wajc. Rounding dynamic matchings against an adaptive adversary. In *Proceedings of the 52nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 194–207, 2020. 1.5, 6.1, 10

[267] Yajun Wang and Sam Chiu-wai Wong. Two-sided online bipartite matching and vertex cover: Beating the greedy algorithm. In *Proceedings of the 42nd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 1070–1081, 2015. 1.1, 4.1, 4.1.1, 4.2, 4.2.1, 4.2.1, 4.2.1, 4.2.1, 4.2.1, 4.2.1, 4.4, 4.6, 12

[268] David P Williamson. The primal-dual method for approximation algorithms. *Mathematical Programming*, 91(3):447–478, 2002. 1.4.1

[269] Laurence A Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982. 11.3

[270] Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th Symposium on Foundations of Computer Science (FOCS)*, pages 222–227, 1977. 5.2, 11.9