Resource Allocation under Incentive, Information, and Complexity Constraints

Ankit Sharma

CMU-CS-14-124

July 2014

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Avrim Blum and Anupam Gupta, Chair
Ariel Procaccia
Jan Vondrák, IBM Almaden

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

This research was sponsored by the National Science Foundation under grant numbers CCF-0830540, CCF-1101215, CCF-1116892, CCF-1215883, CCF-0905390, CCF-1016799, IIS-1065251, and IIS-1350598; Microsoft Research and Carnegie Mellon University Center for Computational Thinking; Google Inter-university center for Electronic Markets and Auctions; Israel Science Foundation; United States-Israel Binational Science Foundation (BSF); Israeli Ministry of Science (MoS)

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: Resource allocation, Mechanism design, Approximation algorithms

Dedicated to my parents for their unconditional love, support and guidance. Thank you, Ma and Pa, for always being with there with me through the highs and lows of life. You have given me several invaluable pieces of advice, have always patiently listened to me, and have helped me shape my life. Ma and Pa, it is hard for me to express my gratitude in words to you. Dedicating this thesis to you is a small way to say "Thank you" for everything you have done for me.

Abstract

This thesis studies the problem of resource allocation from multiple perspectives under a variety of constraints and objectives. Resource allocation is a central theme in many economic settings (markets, auctions etc.) and, more broadly, is prevalent in almost everything we do (for instance, everyday, we make decisions about allocating our money and time). The problem of resource allocation is usually driven by the fact that the resource under consideration is limited, and less than the number of claimants demanding it. And hence, we need to decide who to allocate the resource to and in what proportion. What makes resource allocation an extremely active area of research is that there is no single good allocation mechanism for the myriad constraints and objectives under which we encounter this problem. In this thesis, we make advances by designing new allocation mechanisms, studying the properties of existing ones, and understanding the complexity of how we value resources.

Specifically, our main contributions are: (a) we introduce a more general and realistic model of limitation of resources and under this limitation model, design allocation mechanisms that allocate resources to an online stream of self-interested buyers with combinatorial valuations through item pricing while approximately optimizing the objectives of social welfare and profit, (b) we design adaptive and non-adaptive allocation mechanisms for resource allocation under uncertain valuations, where the values can be determined only through expensive queries, (c) we analyze the performance of some of the popular auction formats in the presence of 'spiteful' bidders whose utility is negatively affected by other bidders' positive outcome, and (d) we understand the complexity of submodular valuation functions, a class of valuations characterized by decreasing marginal utilities, vis-á-vis some of its well-studied sub-classes such as budget additive, coverage and cut functions.

Acknowledgements

Avrim and Anupam, thank you for always giving me the freedom to explore my interests, for patiently listening to my research ideas, and for all the useful pieces of advice on every aspect of research and career. Ariel and Jan, thank you for giving me the opportunity to work on many interesting problems with you and helping me understand different ways to think about and attack research problems.

I am grateful to Tuomas Sandholm, Yishay Mansour, Mohit Singh, Nikhil Devanur, Nitish Korula and Vahab Mirrokni for giving me the immense opportunity to work with them. And special thanks to Alan Frieze, R. Ravi, Ryan O'Donnell and Venkat Guruswami for the invaluable research discussions, and to Mor Harchol-Balter for being a wonderful person and being extremely approachable whenever I was in need of advice. And Deb, thank a lot! You have always been there to help! Thank you, Marilyn!

Thank you, Anuj, Anvesh, Ashique, Mehdi, Malav, Mayur, Pallav, Shayak, Shrikant for the countless discussions over dinner! And, again, thank you, Anuj!! Thank you, Sarah, for the long fascinating conversations and for introducing me to contra dancing. Thank you, Jamie and Nika, for being wonderful co-researchers and friends! Thank you, John and Gabe, for being awesome office-mates. John and Carol, I owe you for introducing me to the wonderful world of western classical music. Thank you, Carla LaRocca, for being an awesome piano teacher! And thanks, Athula, for the impromptu conversations!

I want to thank NSF and the other funding agencies for helping fund and making possible the research work contained in this thesis.

Thank you, Pittsburgh! You have been a wonderful city to live in! And thank you, India and U.S.; there is a lot I owe you!

P.S. Dear reader, thank you for taking out the time to read the section on acknowledgments. I hope you will find the contents of this thesis interesting and useful! :)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Resource Allocation

Allocating limited resources among competing claimants is a primary driving force of several economic activities. Markets allocate goods to buyers. Art auctions allocate scarce works of art to bidders. Ad auctions allocate impressions to advertisers. Mechanisms such as selling in the free market or auctions serve as means to make the allocation among the claimants who are usually individuals or firms or some times, even countries. In recent years, one of the key areas of research at the intersection of Computer Science and Economics is the study and the design of mechanisms that make allocations that are *efficient to implement*, *optimize certain objective*, and *operate under certain constraints*. An example of this research is the study and design of *ad auctions* that are used to allocate impressions to advertisers with the objective of maximizing either revenue or the viewer's utility.

This thesis aims to understand

> how to allocate *limited resources* among *competing claimants* to optimize a *desired objective* under certain *constraints*.

The limited resources could be natural resources such as land, coal etc., or machines resources such as CPU, memory, network bandwidth etc., or the merchandise of an online retailer, or the budget of an online advertiser. The corresponding claimants could be humans, computer programs, online buyers and impressions. In most scenarios, the demand for these resources usually exceeds their supply and this mismatch of demand and supply sets up the basis for this research question – *which of the claimants should be allocated what resources*. To facilitate setting up the problem mathematically, we assume that each

claimant has a *valuation function* that assigns some value (maybe zero) for each subset of resources, and hence every allocation brings some value to every claimant.

Different settings of resource allocation exhibit extremely varied constraints and have different objectives. When an online retail website sells (and, as a result, allocates) its inventory, it wants to maximize its *revenue* while facing *game theoretic constraints* from an *online* stream of buyers. In contrast, in allocating servers to an online stream of job requests, usually there are no game theoretic constraints, and the objective is to maximize the *sum of values* of the requests that get allocated their desired resources. Hence, the objective of the resource allocation could be to maximize the total value of allocation (a.k.a. *social welfare* in game theoretic settings), or maximize the minimum allocation (i.e., maximize *fairness*), or minimize the maximum allocation (i.e., minimize *makespan*), or maximize *revenue*. And the *constraints could be computational, game-theoretic, informational (offline vs online)* among others. Such extremely varied constraints and objectives make the design of a single 'good' allocation mechanism for all settings a difficult proposition. This thesis aims to understand the interplay of the different constraints and crystallize the key ideas that should drive the design of resource allocation mechanisms.

Here is the roadmap for remaining parts of this Chapter: In Section 1.2, we preview Chapter 2, where we introduce a new model of resource limitation, one that we show to be more general than the previous model and that more accurately captures limitations in many settings than the previous model. In this model, we show how to allocation resources via item pricing to an online stream of self-interested agents to approximately maximize social welfare and profit. In Section 1.3, we preview Chapter 3, where we analyze settings where the value for allocating a resource to a claimant is unknown and expensive queries need to be made to find it out. With the help of available stochastic information, we show how to make a few queries and make an allocation with value close to that of a mechanism that makes all queries. In Section 1.4, we preview Chapter 4, where we study some of the common existing auction formats – first- and second-price auctions, in the presence of 'spiteful' bidders. Finally, in Section 1.5, we preview Chapter 5, we study the properties of submodular functions, a special class of valuation functions marked by decreasing marginal utility.

We also want to take this opportunity to point the reader to Chapter 6 that discusses some potential directions for future research.

1.2 Resource allocation under procurement costs

An important aspect of studying resource limitation is the modeling of limitation itself. Previous work [Bartal et al., 2003] has modeled limitation by considering the case where we have a fixed number of copies of each resource. While this model of limitation captures several resources of interest such as seats on an airplane, rooms in a hotel etc., it does not suit well to many other resources. To illustrate this point, let us consider a few resources and understand why they are considered limited. Human resource in a firm is limited since while a firm can always hire more people, *hiring every additional employee incurs a cost* – there is cost to interview and hire a suitable candidate, followed by the cost to remunerate the individual for his work, supply her with office space and provide her with support staff. Similarly, resources such as network bandwidth are limited not because additional capacity cannot be added but there is a cost required for such augmentation. Therefore, for many resources of interest,

> *the resource is limited not because there is only so much of it but more often there is a cost to procure extra copies of the resource and this procurement cost is the limiting factor*. In contrast to this, the traditional model of limitation that computer science literature has used is *granting only a limited number of copies of the resource* to the allocation mechanism.

It is not just that this traditional *fixed-number-of-copies* limitation model does not accurately capture limitation for these resources. For such resources, applying resource allocation algorithms that assume a fixed number of copies for the resource is not straightforward since *it is not obvious what this fixed number should be* (especially if extra copies can be procured at moderate costs).

In Chapter 2, we consider a new model of limitation where each resource has a *procurement cost curve*; the procurement cost curve gives, for every $i \in \mathbb{N}$, the *marginal* cost of procuring the i^{th} copy of the resource. We study the special case where these procurement cost curves are non-decreasing[1]. We consider how resources are allocated to self-interested agents to maximize either social welfare or profit in an online setting. The allocation scheme *prices individual resources*, and allows the self-interested agents to buy their utility-maximizing set of resources. The loss in social welfare compared to the optimal is bounded by a multiplicative (and an additive) constant factor in case the limitation is 'soft', and this loss becomes logarithmic in case of a 'hard' limitation.

[1]While some procurement cost curves in the real world may not be non-decreasing for the entire domain, for our study, we choose the class of non-decreasing curve since it is the simplest class of curves that captures limitation.

Interestingly, the limitation model using procurement costs can also capture the fixed-number-of-copies model. In terms of procurement costs, the fixed-number-of-copies model of limitation has the first so many copies 'free' and any copy beyond this number has an infinite cost. Hence, this modeling of limitation through procurement costs not only analyzes a more general model of limitation but it also helps *bridge our understanding between the two extreme modes of limitation* that have been considered by previous literature – the mode where there are only a fixed number of copies of a resource [Bartal et al., 2003] and the mode where there is no limitation at all, also called the digital-goods model.

1.2.1 Results

As mentioned in the previous section, we allocate resources through *item pricing*. Each 'potential' copy of each resource carries a price tag and at any point, the price for a resource is the price of its cheapest available copy. The claimants are utility maximizing agents and buy the set of resources that maximize their value (for the set) minus the sum of prices of the resources (in the set). We note that this implies that we make our allocation with the help of 'demand' queries: for any set of prices, we can compute for any buyer their utility maximizing bundle.

We first study procurement cost curve that model 'soft' limitation. Herein we study polynomial procurement cost curves and logarithmic cost curves. We price the copies of the resources using the *'Twice the Index' pricing scheme*. Here the *price* of the k^{th} copy of a resource is the *procurement cost* of the $(2k)^{th}$ copy of that resource. For the objective of social welfare, we show that this pricing scheme is able to guarantee a *constant factor multiplicative approximation* to the optimal welfare in addition to some additive loss, where the constant factor depends only the parameters defining the cost curve. The additive loss is linear in the number of the distinct resources. This results shows that with significantly soft limitations, we can achieve a constant factor approximation to social welfare, something that in the case of fixed-number-of-copies model we do not know for any case where the number of copies is sub-linear in the number of claimants, no matter how large this number is [Bartal et al., 2003]

Next, we analyze the case of *arbitrary procurement cost curves*. As we mentioned earlier, the procurement cost model also captures, as a special case, the fixed-number-of-copies model. This implies that any inapproximability hardness results known for the latter model also translate to the case where we have arbitrary procurement cost curves, and therefore we can only fear worse approximation guarantees for arbitrary cost curves. Nevertheless, we design a pricing scheme (that builds on the work of Bartal et al. [2003]) that guarantees a *logarithmic factor multiplicative approximation* to the optimal social

welfare, in addition to some additive loss. In contrast to the case of 'soft' cost curves, the multiplicative depends not only the cost curve but also on the number of buyers.

Thus far, we presented two pricing schemes. One that gives constant factor multiplicative loss for 'soft' cost curves and another that gives logarithmic factor multiplicative loss for arbitrary cost curves. Ideally, we would like to have a *single pricing scheme that guarantees a small loss for soft limitation and a moderate loss for hard limitation*. None of the two pricing schemes we presented has this property. While we are not able to give such a pricing scheme for arbitrary cost curves, we are able to design a third pricing scheme, called the *'Smoothing' pricing scheme* that achieves this goal for *convex* cost curves.

Profit Maximization The discussion so far focused on the objective of maximizing social welfare. We now present our results for the objective of profit maximization. Here we are able to show a general result, stated formally as Theorem 2.9.1 in Section 2.9. Here is an informal presentation of the result.

Theorem 1.2.1. *Given an* **online multi-buyer social-welfare maximization** *algorithm A with ρ multiplicative and β additive losses, and an* **offline single-buyer profit maximization** *algorithm B with μ multiplicative and κ additive losses, we can construct a* **randomized online multi-buyer profit-maximizing** *algorithm C with $(\rho + \mu)$ multiplicative and $(\beta + \mu \cdot m)$ additive losses. (Here m is the number of buyers.)*

This result helps us combine each of our social welfare maximizing schemes with the single buyer profit maximizing scheme of Balcan et al. [2008] to generate a profit maximizing scheme for a sequence of online buyers. Using this result, we achieve a pricing scheme whose expected profit, for arbitrary procurement cost curves, is within a logarithmic multiplicative loss of the optimal profit with some additive loss.

1.2.2 Techniques

A significant property of our results is that it allows for *arbitrary combinatorial valuation*. With arbitrary combinatorial valuations, in the very special setting where there is only one copy of every resource, several strong inapproximability results are known [Hastad, 1996]. Furthermore, analyzing allocation schemes for arbitrary combinatorial valuation functions, in general, is hard.

The crucial result that makes it amenable for us to analyze our pricing schemes for arbitrary valuation function is the *"Structural Lemma"*. This result lends its power by extending guarantees of a pricing scheme *from the case of a single resource to the case of*

an arbitrary number of resources. Analyzing a pricing scheme is much simpler for the case of a single resource since the problem loses it combinatorial nature in this special case. And the Structural Lemma makes the analysis for this much simpler case sufficient for deriving guarantees for the combinatorial case. Our proof of the Structural Lemma builds on a similar result shown by Bartal et al. [2003] albeit for the case with no procurement costs.

The second general result in our work, stated informally in Theorem 1.2.1, and one that powers the results for profit maximization builds on a similar result shown by Awerbuch et al. [2003] in the setting without procurement costs.

1.2.3 Related Work

There is a huge body of literature on combinatorial auctions and pricing algorithms. The setting of combinatorial auctions has been considered both in Bayesian (stochastic) settings, where the buyers' valuations are assumed to come from a known prior distribution, and non-Bayesian (adversarial) settings. Our work focuses on the non-Bayesian or adversarial setting. We refer the reader to Blumrosen and Nisan [2007] and Hartline and Karlin [2007] and the references therein—in particular, note Bartal et al. [2003], Lehmann et al. [2006], Dobzinski et al. [2005], Dobzinski [2007], Briest et al. [2005] and Lavi and Swamy [2005] – for a more comprehensive survey of the results in this area.

The works of Briest et al. [2005] and Bartal et al. [2003] are closely related to our setting. The algorithms of Briest et al. [2005] give truthful mechanisms that achieve constant approximations to social welfare for $\Omega(\log n)$ copies of each item (see also [Archer et al., 2004]) in the *offline* setting. For the *online* setting, Bartal et al. [2003] give posted-price welfare-maximizing algorithms for combinatorial auctions in the limited supply setting—the approximation guarantees they give are logarithmic (when there are $\Omega(\log n)$ copies of each item) or worse (when there are fewer copies); their results are (nearly) tight for the online limited-supply setting. Our pricing scheme, the smoothing algorithm, presented in Section 2.8, generalizes the results of Bartal et al. [2003] from the the fixed-number-of-copies model to our more general non-decreasing procurement costs.

The work of Awerbuch et al. [2003] shows how to convert deterministic (or some special kind of randomized) online mechanisms for allocation problems into (randomized) posted-pricing schemes that achieve $(\rho + \log V_{max})$-fraction of the optimal profit possible, where the online algorithm is ρ-competitive for the allocation problem and V_{max} is the maximum valuation of any agent over the set of items. We extend their analysis to convert our social welfare maximizing algorithms to profit maximizing algorithms (Theo-

rem 1.2.1).

1.3 Resource allocation with expensive queries on stochastic input

Thus far we assumed that the value of allocating a resource to a claimant is *known*, either to the allocation mechanism as part of the input or to the claimant who may be a self-interested agent, in which case the mechanism has to incentivize the agent to report it truthfully. In certain scenarios, however, the value of awarding a certain resource to a claimant is *unknown*. The value can be known only through *queries* and the queries are expensive. Hence, the mechanism has to make resource allocation without necessarily knowing the entire input. Rather, the mechanism has to choose what parts of the input to query based on which it decides how to allocate the resources. Since queries are expensive, their number should be small. *The information that guides the queries of the mechanism is stochastic* – the mechanism has the knowledge of the distribution from which the input is drawn. In Chapter 3, we consider the following question:

> Without access to the entire input, which parts of the input does the resource allocation mechanism (guided by stochastic information) query, to make an allocation that has the maximum value?

Resource allocation with expensive queries can be modeled theoretically in a variety of ways. We consider the problem of set-packing (U, A) where each set from the collection A of sets contains elements from U (with $|U| = n$). Each element of U represents a resources, and each set represents a claimant. The stochastic information available is the following: For each set e, we know the probability p_e, that the value of the set is found to be 1 on query; with probability $1 - p_e$, the value of the set is zero. We want to query sets in A and based on the answers to the queries, pick a collection $S \subseteq A$ of pairwise-disjoint sets of maximum value.

To model expensive queries, one approach is to put a *cap on the number of queries* that can be made to the input, and the allocation mechanism has to decide which set of queries (under this cap) are made, and based on the results of the queries, how the resources are allocated. And these queries can be either non-adaptive or adaptive. A related question is the minimum number of queries required for the allocation mechanism to achieve a solution of a desired quality. We consider questions of both flavors – the first where we

cap the number of queries and ask *how best to use this budget of queries*, and the second where we ask *what the minimum number of queries is to achieve a desired solution quality*.

There are two parameters that are important for this problem. The first parameter is the *total number of queries* that are made by the allocation mechanism. The second parameter is the *number of rounds* in which the algorithms makes these queries. A non-adaptive algorithm issues all its queries in one round. A completely sequential algorithm will take as many rounds as the number of queries it makes. If the same number of queries are issued in a smaller number of rounds then it implies that more queries can be answered in parallel and parallelism is usually desirable in a real world system. Hence, we aim to minimize both the number of rounds and the number of queries.

Our work is motivated from the practical application of kidney exchange, and we describe this connection now. One medical solution to kidney failure is to transplant a kidney from a willing donor to a patient. For the transplant to be successful, the kidney has to be biologically compatible with the patient. Many times, however, the donor is a friend or a relative of the patient, who is willing to donate her kidney but is biologically incompatible with the patient. A practical solution to this is to try to find two such pairs that are mutually compatible, i.e., the donor of the first pair is compatible with the patient of the second and vice-versa. Kidney exchange does precisely the task of finding several mutually compatible pairs from a large pool of patient-donor nodes. To find whether a patient and a potential donor are compatible requires an expensive laboratory test, called the cross-match test. For kidney exchange, it is prohibitively expensive to do a cross-match test for every pair of patient and potential donor. Rather, it would like to conduct a few cross-match tests and be able to find sufficiently many mutually compatible pairs so as to satisfy most of the patients. Hence, kidney exchange provides a setting for resource allocation, where the value of allocating kidneys (resources) to patients (claimants) is unknown and can be found only through the cross-match tests (queries). In Chapter 3, we detail the implications of our theoretical framework and results to kidney exchange.

1.3.1 Results

We have two sets of results corresponding to the two flavors of questions we mentioned earlier, and we start with the results of the first flavor. For stochastic k-set packing (each set has at most k elements of U), we give a adaptive algorithm that proceeds for $O_\epsilon(1)$ rounds and outputs a solution whose expected value is at least $\frac{2}{k} - \epsilon$ fraction of the expected value of the *omniscient optimum* – one that queries all the sets. For any given resource, among the sets that contain the resource, the adaptive algorithm queries at most one set. Hence, the algorithm makes $O_\epsilon(1)$ queries for any resource. For the case of stochastic

matching ($k = 2$), we therefore get $(1 - \epsilon)$-approximate adaptive algorithm. Furthermore, for the sake of comparison, the best known *polynomial-time* algorithm for optimizing k-set packing in the *standard non-stochastic setting* has an approximation ratio of $\frac{3}{k+1} - \epsilon$ [Fürer and Yu, 2013].

While our adaptive algorithm takes only $O(1)$ rounds, we ideally would like a non-adaptive algorithm that gets a $(1 - \epsilon)$ approximation.While it is an open question whether an non-adaptive algorithm can achieve $(1-\epsilon)$-approximation with $O(1)$ queries per vertex, we present a non-adaptive algorithm that achieves $0.5 - \epsilon$ approximation for stochastic matching ($k = 2$) and $(\frac{2}{k} - \epsilon)^2$-approximation for stochastic k-set packing (for $k > 2$).

In the second set of results corresponding to ones of the second flavor, we explore the computational complexity of finding the best non-adaptive solution with per-vertex query budget of two. We show that this problem is NP-hard. Furthermore, for a particular class of 'realistic' kidney exchange distributions, we provided a polynomial time algorithm that for most graphs from the distribution gives *an almost optimal* solution to this per-vertex budget constrained problem.

1.3.2 Techniques

Our adaptive algorithm makes use of the local search algorithms [Hurkens and Schrijver, 1989] known for the problem of k-set packing. Specifically, starting from any set of answered queries, it prepares its queries for the next round by finding several disjoint constant-size structures that can augment the current solution formed out of the answered queries. The fact that the structures are disjoint means that the answers to the queries are stochastically independent, and the constant-size of each structure implies that each will exist with constant probability. This allows the adaptive algorithm to make significant progress towards the omniscient optimal in each round, and proceeding for a constant (depending on ϵ) number of rounds bring it ϵ close to it.

We show the NP-hardness of finding the best non-adaptive solution with per-vertex budget of two via a structural result showing that if a collection of disjoint 4-cycles exists in a graph, then this collection is the optimal solution and finding a 4-cycle is hard via a reduction from 3D-matching.

1.3.3 Related Work

Chen et al. [2009] consider a scenario of resource allocation with *adaptive queries*. The problem they consider is inspired by kidney exchange. They consider the following prob-

lem: Given a random graph G with known edge probabilities p_e, in what order should one query the edges in order to maximize the expected cardinality of the matching? They impose two constraints on the query pattern of the algorithm: any queried edge that is found to exist must be added to the matching, and for every node v, there is a specified budget on the total number of queries that can be made on edges incident to the node. They show that a greedy algorithm which queries the edges in decreasing order of the edge probabilities gives a 0.25-approximation. Adamczyk [2011] later improved the analysis to show that the greedy algorithm in fact yields a 0.5-approximation.

Bansal et al. [2012] extend the work of Chen et al. [2009] by considering the weighted version of the problem, where in addition to edge probabilities p_e, each edge has a weight w_e and the objective is maximize the expected *weight* of matching (as opposed to cardinality). They give an LP-based solution that achieves a 0.25-approximation for the case of a weighted general graph and a 0.33-approximation for the case of a weighted bipartite graph. These results have been generalized to other packing problems by Gupta and Nagarajan [2013].

1.4 Study of existing allocation mechanisms in presence of spiteful agents

The previous sections have focused on designing new allocation mechanisms. While new mechanisms are needed when the existing mechanisms do not suffice, in some cases, the existing mechanisms cannot be replaced due to practical constraints. In such cases, we analyze the properties of the existing mechanisms to quantify their behavior in practical scenarios of interest.

Auctions are one of the principal ways of allocating limited resources to self-interested agents. The first-price and second-price auctions with their sealed-bid and dynamic variants are very commonly used in practice. These auctions are well studied under the assumption that the bidder's utility is determined solely by their own outcome, and their utility is zero on losing. This, however, is not always the case, and this gap in our understanding forms the basis of our work in Chapter 4, where we study the behavior of certain popular auction formats in the presence of 'spiteful' agents.

Spiteful agents A spiteful agent's utility function is determined not just by what they received from the auction, but also what others received. Furthermore, the dependence is of a *negative* nature – *when others are better-off, the agent is worse-off.* In particular, the

agent's utility on losing may be a negative number (and not zero). This negative dependence stems from both *strategic reasons* and from purely *psychological* ones [Saijo and Nakamura, 1995, Levine, 1998, Loewenstein et al., 1989]. An instance where this dependence stems from strategic reasons is in an auction where several companies that operate in different markets of the world are bidding. The utility of a company on losing, in part, depends on whether the winner of the auction conducts their business in the same geographical location as itself. If the winner of the auction has its primary market the same as the losing firm, then the utility on losing the auction is probably lower (or the negative utility is higher) than if the winner focuses on a different market.

We consider the question of *how spiteful agents bid in the common auction formats, and how these contrast to the case with no spite*. Furthermore, how does this spite affect the winner of the auction and the revenue that the auctioneer derives from it? Previous work by Morgan et al. [2003] studied the special case where all the agents are equally spiteful and moreover, have their utility equally affected by different bidders. *We analyze the more realistic case, where different agents are 'spiteful' to different extents and furthermore, can have asymmetric spite to different bidders*. And so our results apply to involved settings where an agent A can be more spiteful to agent B than to agent C, and agent C may not be spiteful at all to agent A.

1.4.1 Results

We study the one-item auction and give analytical expressions for the equilibrium bids made by the agents. In addition, we analyze who wins the auction and how much they pay. We show that unlike the setting without spite, the *revenue equivalence* of the first-price and second-price auction is *no longer true*. In fact, unlike the symmetric spite setting, *no one type of auction dominates the other in terms of revenue*. Depending on how different the bidders are in their spite, one type can bring more or less revenue than the other. Moreover, from the perspective of resource allocation, the auctioned item may be won by a player *who does not value it the most*.

1.4.2 Techniques

We derive a Nash equilibrium for the spiteful bidders using first principles. We augment this by making careful guesses of the equilibrium bidding function to come up with closed-form solutions for them. For many of the settings, we consider the bidding functions given by Morgan et al. [2003] in the symmetric setting, and make educated guesses about their

potential form in the asymmetric setting.

1.4.3 Related work

Morgan et al. [2003] initiated the study of the auctions for the case when the competing bidders are *spiteful*. They compute the equilibrium bidding functions of the spiteful bidders, and compare the revenue attained in spiteful settings to that attained in settings without spite.

1.5 Approximating valuation functions

One aspect of resource allocation mechanisms that we have not touched on so far is the valuation function. The valuation function gives the value that a particular claimant has for a given set of resources. While in Chapter 2, we considered the case of arbitrary combinatorial valuations, usually, allowing from arbitrary combinatorial valuations makes the computational problem of allocation extremely hard. For this reason, research has considered special classes of valuation functions and given mechanisms that achieve either optimal or approximate solutions .

In Chapter 5, we study the class of submodular functions. This class of functions is important due to two reasons. First, in many economic settings, buyers display valuation functions obeying the property of 'decreasing marginal return' and the class of submodular functions is used to capture this property mathematically. The second reason is that many combinatorial optimization functions turn out to be submodular.

There are some special classes of submodular functions that are well studied. These include the class of matroid rank functions, cut functions and coverage functions. The objective behind the work in Chapter 5 is to understand *how the complexity of expressiveness of these various subclasses compare to one another* and to the parent class of submodular function. Specifically, we ask how closely a member of one subclass can be approximated by members of a different class.

1.5.1 Results

Our main results are:

- General submodular functions[2] can be approximated by cut functions of directed graphs to a factor of $n^2/4$, which is tight.
- General symmetric submodular functions[1] can be approximated by cut functions of undirected graphs to a factor of $n - 1$, which is tight up to a constant.
- Budgeted additive functions can be approximated by coverage functions to a factor of $e/(e-1)$, which is tight.

Here n is the size of the ground set on which the submodular function is defined. We also observe that prior works imply that monotone submodular functions can be approximated by coverage functions with a factor between $O(\sqrt{n}\log n)$ and $\Omega(n^{1/3}/\log^2 n)$.

1.5.2 Techniques

Unlike some of the previous work [Goemans et al., 2009, Balcan and Harvey, 2011, Badanidiyuru et al., 2012] that analyze *non-negative monotone* submodular functions, in this work, we look at *arbitrary* submodular functions. For instance, we consider the question of approximating arbitrary submodular functions by cut functions. An arbitrary submodular function can take zero value on non-trivial sets, and in particular, is not monotone. Since we aim for a purely multiplicative approximation, this implies that our cut functions should be zero exactly where the submodular function takes the zero value. Therefore, we face some unique challenges. In the case of *symmetric* submodular function, for instance, we use the cut function of the Gomory-Hu tree [Gomory and Hu, 1961, Queyranne, 1993] representation for the function to approximate it.

For approximating budgeted additive functions by coverage functions, we design a randomized construction. Furthermore, to show the optimality of the achieved approximation, we write a Linear Program whose objective is the best approximation factor, and then analyze the dual of this program to show that the primal value is at least the achieved approximation factor.

1.5.3 Related work

Submodular function have been studied from a number of perspectives. Goemans et al. [2009] consider the question of approximating a non-negative monotone submodular function using a simpler class of functions. They give $O(\sqrt{n}\log(n))$ approximation. Balcan

[2]We additionally assume that the submodular function takes value 0 on the null set and the universe.

and Harvey [2011] take the *learning perspective* and study the problem of probabilistically learning a monotone submodular function, given the values the function takes on a polynomial sized sample of its domain. They provide a lower bound of $\Omega(n^{1/3})$ on the best possible approximation a learning algorithm can give to the submodular function. Badanidiyuru et al. [2012] consider whether there can be *polynomially sized 'sketches'* to sub-additive functions (submodular function is a sub-class of sub-additive functions) and present a $O(\sqrt{n}\ polylog(n))$ approximation sketch. Seshadhri and Vondrák [2011] and Chakrabarty and Huang [2012] consider the question of *testing* whether a function belongs to the class of sub-modular function and coverage functions respectively.

Chapter 2

Online Resource Allocation with Procurement Costs

2.1 Introduction

In this chapter, we consider resource allocation in an online setting under game theoretic constraints. The main focus of this work is understanding how the extent of limitation of the resources plays out and affects the guarantees we can achieve for resource allocation under the given setting. We consider two objectives – social welfare maximization and profit maximization.

In order to understand the effect of limitation of resources, we need a model of limitation. Prior work has considered capturing limitation of resources by allowing some fixed number of copies of each resource. This accurately models resources as the rooms in a hotel or seats on an airplane. Under this model of limitation, prior work has explored the welfare (or profit) guarantees that can achieved as we vary the number of copies available for the resources. This model of limitation however fails to capture several resource settings of interest. For instance, consider the following resources – human resource and petroleum. For these resources, there is no a priori fixed number of copies. We can always hire extra personnel to join the labour force and can procure extra gallons of petroleum. What then limits us in allocating as many quantities of these resources as demanded. The answer is that there is a *cost* associated in procuring additional quantities of these resources. In case of human personnel, we may have to expend costs in the form of conducting interviews for recruitment, for training and in remuneration. For petroleum, we can get extra gallons but again we have to expend cost in drilling and refining it. For these resources, what limits

us is not the total fixed number of copies of these resources. What limits us is the cost that we have to expend in procuring additional copies.

In this work, we introduce a new model of limitation that captures these settings more accurately. In this model, there is a procurement cost curve associated with each resource. This cost curve captures the cost that needs to expended in procuring extra copies of the resource. In addition to capturing these new setting, what makes this model extremely useful is that it can also capture settings that were modeled by prior work, i.e., where there are only a fixed number of copies of a resource. With this new model of limitation, we explore how to allocate resources to optimize the two objectives under consideration – social welfare and profit.

The allocation of resources is through item pricing. We set prices for individual resources and the total price for a bundle of resources is the sum of the resources in that bundle. The claimants for these resources are buyers with arbitrary combinatorial valuations. The claimants arrive online, one at a time, and consume that set of resources which maximize their utility at the current prices. In this work, we show that extremely simple pricing functions – the price for every resource depends only on how many copies of that resources have been allocated so far – can achieve good approximation factors to social welfare. For a broad class of cost curves that capture limitation that is not extremely restrictive, we have a constant factor approximation. In case, the cost curves capture extremely restrictive limitation, we achieve a logarithmic factor approximation.

The roadmap for the rest of the chapter is as follows. We first define the model of limitation that we use in this work. Then we summarize in greater details the results of this work. Later we present the item pricing schemes that we design for resource allocation, followed by their analysis. We conclude with some possible extensions of this work.

2.2 Model of Limitation

We consider the following setting. A seller is selling a set $U = \{1, \ldots, n\}$ of n items to a sequence B of m buyers who arrive one at a time. The seller can obtain (or produce) additional copies of each item but at increasing (or at least non-decreasing) procurement cost; specifically, let $c_i(k)$ denote the procurement cost to the seller for the kth copy of item i. For each item i, let $C_i(k)$ be the cumulative cost for the first k copies—i.e., $C_i(k) = \sum_{k' \leq k} c_i(k')$. Let $c_i^{\mathsf{inv}}(p)$ be the number of copies of item i available before the procurement cost exceeds p; in case $c_i(\cdot)$ is invertible, it follows that $c_i^{\mathsf{inv}}(p) = c_i^{-1}(p)$.

Before each buyer arrives, the seller may mark up the costs to determine a sales price

π_i for each item i. Every buyer b has some (unknown to the seller) valuation function $v_b : 2^U \to \mathbb{R}$ over possible bundles of items (we only require that $v_b(\phi) = 0$ i.e. value on the empty bundle is zero), and purchases the utility-maximizing bundle for herself at the current prices. That is, buyer b purchases the set S maximizing $v_b(S) - \sum_{i \in S} \pi_i$. After a buyer finishes purchasing her desired set, the seller may then readjust prices, and then the next buyer arrives, and so on.

2.2.1 Our Results and Techniques

For a wide range of reasonable cost functions (linear, low-degree polynomial, logarithmic), we present a pricing scheme that achieves a social welfare within a *constant* factor of the optimal social welfare allocation minus a necessary additive loss. This holds for buyers with arbitrary combinatorial valuation functions. Furthermore, the algorithm is quite 'natural' and reasonable: we price the k^{th} copy of any good at the procurement cost of the $2k^{th}$ copy[1]. This pricing scheme, that we call *twice-the-index*, appears in Section 2.6.

Twice-the-index pricing scheme however fails to give good guarantees for all increasing cost functions. For instance, for the $0-\infty$ case, where the first few copies are available at zero cost and thereafter the copies have an extremely high procurement cost, buyer instances can be easily be created where twice-the-index fails to give any any 'reasonable' guarantee (Section 2.6.4). Bartal et al. [2003] propose a pricing scheme for the $0 - \infty$ setting which fetches a logarithmic approximation to social welfare in case the number of copies available at zero cost are logarithmically many. We borrow their idea and apply it to any general increasing cost curve by breaking up the curve into contiguous chunks, each containing logarithmically many copies, and apply their pricing scheme separately for each chunk. This pricing scheme, presented in Section 2.7, gives roughly a logarithmic approximation to (optimal social welfare minus the procurement cost of logarithmically many initial copies).

While Twice-the-index pricing scheme gives constant approximation guarantees for 'nice' curves, the pricing scheme in Section 2.7 gives a logarithmic approximation for general increasing curves. We would ideally want a single algorithm that can give us constant approximation guarantees for 'nice' curves and logarithmic guarantees for general curves. We achieve this for the case of *convex* increasing curves. In Section 2.8, we present a *smoothing* pricing scheme that attains constant approximation to optimal social welfare for polynomial curves and a logarithmic approximation for general convex curves (plus some additive loss in both cases).

[1]For illustrative examples showing why some closely related algorithms *fail*, see Section 2.5.

Interestingly, in order to prove the approximation guarantee for all of the presented social welfare maximizing schemes, we use a crucial result which we call the *Structural Lemma*. This result is stated and proved in Section 2.4. The result reduces the problem of proving the social welfare guarantee of a pricing scheme for buyers with arbitrary valuations to a case of proving that for *every item*, the profit generated through sales of copies of the item is comparable to the area between the procurement curve of the item and a line parallel to x-axis and at a height equal to the prices of the lowest priced unsold copy of the item. Structural Lemma therefore simplies the analysis considerably since it allows the problem to be seen *per item* even though the original problem is combinatorial. Furthermore, we note that we are implicitly making use of *demand* queries to make the allocation: for any set of prices, we can compute for any buyer their utility maximizing bundle.

In Section 2.9, we consider the objective of *profit maximization*, with profit being defined the sum of prices of goods sold minus the procurement cost of the goods. Here we give a randomized pricing scheme that takes as input any social welfare maximizing scheme (with approximation factor say ρ) and a single-buyer profit maximizing pricing (with approximation factor say μ), and converts it to a profit maximization pricing scheme that achieve a $(\rho + \mu)$ approximation to optimal profit for any sequence of buyers. In particular, we use the single-buyer profit maximization algorithm of Balcan et al. [2008] and combine it with the social-welfare pricing schemes mentioned above to get a logarithmic approximation to optimal profit for any general increasing curve. Our recipe for combining a social welfare maximization pricing scheme with a single-buyer profit maximization algorithm borrows ideas heavily from a similar result presented in Awerbuch et al. [2003]. In fact, it extends their results to a more general setting with procurement costs and arbitrary valuations.

2.3 Notation

For any particular sequence of buyers, let opt be the allocation that maximizes the social welfare. Clearly, the social welfare achieved under opt, denoted by $W(\mathrm{opt})$, is an upper bound on both the maximum social welfare and maximum profit achievable by any online algorithm.

For any algorithm alg, $W(\mathsf{alg})$ shall denote the social welfare attained through the algorithm. The algorithm shall determine a pricing scheme for the seller and $\pi_i(k)$ shall denote the sales price charged for the k^{th} copy of item $i \in U$. While this could in principle depend on other items sold, for all our algorithms it depends only on k and the cost-curve for the item. x_i shall denote the total number of copies of item i sold by the algorithm, and

P_i^f shall denote the price of the first *unsold* copy of item i—i.e., $P_i^f = \pi_i(x_i + 1)$.

We shall denote the *total procurement cost* suffered by the algorithm by $C(\mathsf{alg})$ and and the *revenue* made by $R(\mathsf{alg})$. profit_i shall denote the *profit* made by the algorithm from the sales of item i. The total profit made by the algorithm is $\sum_{i \in U} \mathsf{profit}_i = R(\mathsf{alg}) - C(\mathsf{alg})$.

Since x_i are the total number of copies sold by the algorithm alg for item i, therefore, $C(\mathsf{alg}) = \sum_{i \in U} \sum_{k=1}^{x_i} c_i(k)$, $R(\mathsf{alg}) = \sum_{i \in U} \sum_{k=1}^{x_i} \pi_i(k)$ and $\mathsf{profit}_i = \sum_{k=1}^{x_i} \pi_i(k) - \sum_{k=1}^{x_i} c_i(k)$.

The total valuation of buyers on their allocated bundles under alg is denoted by $V(\mathsf{alg}) = \sum_{b \in B} v_b(\mathsf{alg}(b))$ where $\mathsf{alg}(b)$ denotes the set of items bought by buyer b from the algorithm alg. The *social welfare* made by the algorithm $W(\mathsf{alg})$ is $V(\mathsf{alg}) - C(\mathsf{alg})$.

For opt, the welfare-maximizing allocation, λ_i denotes the number of copies of item i allocated in opt. $C(\mathrm{opt})$, $V(\mathrm{opt})$ and $W(\mathrm{opt})$ are defined analogously.

2.3.1 (α, β) approximation factor definition

Definition 2.3.1 ((α, β)-welfare approximation)**.** *An allocation scheme is said to be (α, β)-welfare approximate if for every possible set σ of buyers arriving in any sequence, the welfare achieved through the allocation scheme is at least $(\mathrm{opt}(\sigma) - \beta)/\alpha$, where $\mathrm{opt}(\sigma)$ is the optimal welfare on the buyer set σ.*

2.4 Single Resource Goodness and Structural Lemma

We now present the central lemma that drives the results mentioned in Section 3.1.2. The lemma makes it sufficient to consider the one resource case. For instance, it says that if a pricing scheme can achieve an (α, β)–approximation factor for the single resource case, then it can achieve $(\alpha, n\beta)$ approximation in case the number of resources is n in the special case where all the resources have the same approximation factor. This allows us to focus on the special case where we have just one resource and we want to design a pricing function for it. This case is significantly simpler to reason about since it does not have combinatorial structure of the original problem where we have multiple resources and the buyers can have arbitrary valuations for various subsets of resources.

Definition 2.4.1. *A price curve π is (α, β)-single-resource-good for a resource with cost curve c, if in the single resource setting involving just that resource, the allocation scheme that uses the price curve π is an (α, β)-approximate.*

Lemma 2.4.2. *(Structural Lemma) Consider the allocation schemes that uses (α_i, β_i)-single-resource-good price curve π_i for the i^{th} resource.Then the allocation scheme is $(\max_{i\in[n]} \alpha_i, \sum_{i\in[n]} \beta_i)$-welfare approximate.*

Proof. When buyer $b \in B$ arrives, let $x_i^{(b)}$ be the number of copies of item i sold before b comes in. Hence, the price b sees for item i would be $\pi_i(x_i^{(b)} + 1)$; for brevity we denote this $q_b(i)$, and for a set $S \subseteq U$, $q_b(S) := \sum_{i\in S} q_b(i)$. The utility of a set S for buyer b therefore is $v_b(S) - q_b(S)$. Since each buyer buys the set that maximizes her utility, hence in particular it implies that the set $S_b = \mathsf{alg}(b)$ which buyer b bought from alg must be giving her at least as much utility as the set S_b^* that opt allocated to her i.e.

$$v_b(S_b) - q_b(S_b) \geq v_b(S_b^*) - q_b(S_b^*) \ .$$

Summing over all buyers, we get

$$\textstyle\sum_{b\in B}(v_b(S_b) - q_b(S_b)) \geq \sum_{b\in B}(v_b(S_b^*) - q_b(S_b^*)) \ .$$

Adding and subtracting $C(\mathsf{alg})$ and $C(\mathrm{opt})$ on the left hand and right hand sides respectively, we get

$$\begin{aligned}&\big(\sum_{b\in B} v_b(S_b) - C(\mathsf{alg})\big) - \big(\sum_{b\in B} q_b(S_b) - C(\mathsf{alg})\big)\\ \geq&\big(\sum_{b\in B} v_b(S_b^*) - C(\mathrm{opt})\big) - \big(\sum_{b\in B} q_b(S_b^*) - C(\mathrm{opt})\big) \ .\end{aligned}$$

Identifying the term $\sum_{b\in B} v_b(S_b) - C(\mathsf{alg})$ with $W(\mathsf{alg})$, the term $\sum_{b\in B} q_b(S_b) - C(\mathsf{alg})$ with $\sum_{i\in U} \mathsf{profit}_i$ and the term $\sum_{b\in B} v_b(S_b^*) - C(\mathrm{opt})$ with $W(\mathrm{opt})$ we get

$$W(\mathsf{alg}) - \sum_{i\in U} \mathsf{profit}_i \geq W(\mathrm{opt}) - \big(\sum_{b\in B} q_b(S_b^*) - C(\mathrm{opt})\big) \ . \tag{2.1}$$

Since prices are non-decreasing, hence the price faced by any buyer cannot be more than the final price of the various items. Therefore for each buyer b, $q_b(S_b^*) = \sum_{i\in S_b^*} \pi_i(x_i^{(b)} + 1) \leq \sum_{i\in S_b^*} \pi_i(x_i+1) = \sum_{i\in S_b^*} P_i^f$. Hence, the term $\sum_{b\in B} q_b(S_b^*)$ is at most $\sum_{b\in B}\sum_{i\in\mathrm{opt}(b)} P_i^f = \sum_{i\in U}(P_i^f \cdot \lambda_i)$ where recall that λ_i denotes the number of copies of item i allocated under opt. Moreover, since $C(\mathrm{opt}) = \sum_{i\in U}\sum_{k=1}^{\lambda_i} c_i(k)$, we have

$$\begin{aligned}\sum_{b\in B} q_b(S_b^*) - C(\mathrm{opt}) \leq& \sum_{i\in U}(P_i^f \cdot \lambda_i) - \sum_{i\in U}\sum_{k=1}^{\lambda_i} c_i(k)\\ =& \textstyle\sum_{i\in U}\sum_{k=1}^{\lambda_i}(P_i^f - c_i(k)) \ .\end{aligned} \tag{2.2}$$

The quantity $(P_i^f - c_i(k))$ is non-negative until $c_i(k) \leq P_i^f$, that is it is non negative for $k \leq c_i^{\mathsf{inv}}(P_i^f)$. Hence, we have $\sum_{b\in B} q_b(S_b^*) - C(\text{opt}) \leq \sum_{i\in U} \sum_{k=1}^{\lambda_i} (P_i^f - c_i(k)) \leq \sum_{i\in U} \sum_{k=1}^{c_i^{\mathsf{inv}}(P_i^f)} (P_i^f - c_i(k))$. Therefore, using Equation (2.1), we get

$$W(\mathsf{alg}) - \sum_{i\in U} \mathsf{profit}_i \geq W(\text{opt}) - \big(\sum_{b\in B} q_b(S_b^*) - C(\text{opt})\big) \geq W(\text{opt}) - \sum_{i\in U} \sum_{k=1}^{c_i^{\mathsf{inv}}(P_i^f)} (P_i^f - c_i(k)) \,.$$

It is easy to see that since the allocation schemes uses (α_i, β_i)-single-resource-good price curve for the i^{th} resource, hence by Lemma 2.4.3, it is the case that $\forall i \in U$, $\sum_{k=1}^{c_i^{\mathsf{inv}}(P_i^f)} (P_i^f - c_i(k)) \leq \alpha_i \,\mathsf{profit}_i + \beta_i$. Hence, on combining these inequalities with the earlier inequality, we get

$$\begin{aligned} W(\mathsf{alg}) - \sum_{i\in U} \mathsf{profit}_i &\geq W(\text{opt}) - \sum_{i\in U} (\alpha_i \,\mathsf{profit}_i + \beta_i) \\ \Rightarrow W(\mathsf{alg}) + (\max_{i\in U} \alpha_i - 1) \sum_{i\in U} \mathsf{profit}_i &\geq W(\text{opt}) - \sum_{i\in U} \beta_i \,. \end{aligned}$$

Finally using the social welfare generated by the algorithm is at least the profit made, i.e. $W(\mathsf{alg}) \geq \sum_{i\in U} \mathsf{profit}_i$, we get the desired result $W(\mathsf{alg}) \geq (W(\text{opt}) - \sum_{i\in U} \beta_i)/\max_{i\in U} \alpha_i$ ■

2.4.1 Proving (α, β)-single-resource-goodness

In this section, we show how to analyze the single resource case and prove that a price curve π is (α, β)-single-resource-good for a some cost curve c.

Lemma 2.4.3 (Condition for being single-resource-good)**.** *A non-decreasing price curve π is (α, β)-single-resource-good for a given cost curve c if and only if for every $x \in \mathbb{N}^{\geq 0}$,*

$$\sum_{i=1}^{x} (\pi(i) - c(i)) \geq \left(\sum_{i=1}^{c^{inv}(\pi(x+1))} (\pi(x+1) - c(i)) - \beta \right) / \alpha.$$

Proof. First, we prove that the inequality

$$\sum_{i=1}^{x} (\pi(i) - c(i)) \geq \left(\sum_{i=1}^{c^{\mathsf{inv}}(\pi(x+1))} (\pi(x+1) - c(i)) - \beta \right) / \alpha$$

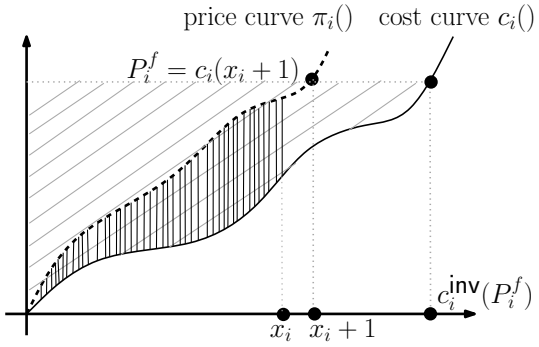


Figure 2.1: Structural Lemma: if the lightly shaded area is bounded by a small multiple of the doubly shaded area, then we get good social welfare. x_i is the last sold copy of the item and $x_i + 1$ is the first unsold copy. The lower continuous curve is the cost curve while the upper dashed curve is the price curve.

is a *necessary* condition for the price curve to be (α, β)-single-resource-good for cost curve c. Suppose not. Consider a (α, β)-single-resource-good price curve and assume that for some value of $x \in \mathbb{N}^{\geq 0}$, the inequality does not hold. Then consider the sequence of buyers with the first x buyers having values $\pi(1)$, $\pi(2)$, $\cdots$, $\pi(x)$ respectively, followed by $c^{\mathsf{inv}}(\pi(x+1))$ buyers each with value $\pi(x+1) - \epsilon$ (for arbitrarily small $\epsilon > 0$). It is not hard to see that the optimal allocation will allocate to the $c^{\mathsf{inv}}(\pi(x+1))$ buyers each with value $\pi(x+1) - \epsilon$, while the algorithm will allocate to the first x buyers. The welfare made by the algorithm is $\sum_{i=1}^{x}(\pi(i) - c(i))$ while the optimal welfare is $\sum_{i=1}^{c^{\mathsf{inv}}(\pi(x+1))}(\pi(x+1) - c(i)) - \epsilon \cdot c^{\mathsf{inv}}(\pi(x+1))$. (α, β)-single-resource-goodness will imply the desired inequality for arbitrarily small $\epsilon > 0$, leading to a contradiction.

The other direction that shows that the inequality is *sufficient* follows from the proof of Lemma 2.4.2 restricted to the case of a single resource. ■

2.4.2 Relaxed conditions for Single Resource Goodness

When we have more information about the buyers, we can satisfy (α, β)-single-resource-goodness for a price curve under more relaxed conditions than in Lemma 2.4.3. In particular, suppose we know the total (or an upper bound of the) number of buyers in the sequence, and let us call it m. Furthermore, we know an upper bound on the maximum

welfare a single buyer can contribute, and call this Z. Mathematically, Z is at least

$$U_{max} = \max_{b \in B} \max_{T \subseteq U} \left(v_b(T) - \sum_{i \in T} c_i(1) \right)$$

We now prove a variant of the structural theorem. For a given resource with cost curve $c()$, define $c^{\mathsf{invt}}(p) = \min\{c^{\mathsf{inv}}(p), m, c^{\mathsf{inv}}(U_{max})\}$ where m is the number of buyers and for a given set of buyers B and items U, is the maximum welfare any single buyer can achieve.

Lemma 2.4.4 (Relaxed condition for being single-resource-good with additional knowledge about buyer sequence)**.** *With knowledge of m and Z, and c^{invt} as defined above, a non-decreasing price curve π is (α, β)-single-resource-good for a given cost curve c if for every $x \in \mathbb{N}^{\geq 0}$,*

$$\sum_{i=1}^{x} (\pi(i) - c(i)) \geq \left(\sum_{i=1}^{c^{\mathsf{invt}}(\pi(x+1))} (\pi(x+1) - c(i)) - \beta \right) / \alpha. \tag{2.3}$$

Furthemore, an allocation schemes that uses (α_i, β_i)-single-resource-good price curve π_i for the i^{th} resource is $(\max_{i \in [n]} \alpha_i, \sum_{i \in [n]} \beta_i)$-welfare approximate.

Proof Sketch: Note that in the proof of Lemma 2.4.2 just after Equation (2.2), we argued that $\lambda_i \leq c_i^{\mathsf{inv}}(P_i^f)$. Instead of summing all the way to $c_i^{\mathsf{inv}}(P_i^f)$, we could stop the summation at $\min\{c_i^{\mathsf{inv}}(P_i^f), m, c_i^{\mathsf{inv}}(U_{max})\}$. Indeed, this is because

- $\lambda_i \leq m$: each buyer wants at most one copy of each item, so at most m copies of item i can be allocated in the optimal solution.
- $\lambda_i \leq c_i^{\mathsf{inv}}(U_{max})$: each copy beyond $c_i^{\mathsf{inv}}(U_{max})$ has cost strictly greater than U_{max}; allocation of any such copy can only decrease the social welfare.

■

2.5 Some 'natural' pricing schemes

We first give some natural pricing schemes and instances where they fail to achieve good social welfare.

2.5.1 Pricing at Cost

While the algorithm of pricing *at cost* (i.e., setting $\pi(k) = c(k)$) gives an optimal welfare for the unlimited supply setting (where procurement costs are zero), it is not a good algorithm even for "simple" cost curves. E.g., for a single item with linear costs $c(k) = k$, consider a sequence of m buyers with the i^{th} buyer having value i for $i \in \{1, \ldots, m\}$, followed by m buyers with value m each. Pricing at cost will sell to the first m buyers and give zero welfare for them, after which the procurement cost will be too high to sell any further copies. In contrast, the optimal solution is to sell to the second set of m buyers with welfare $m^2 - \frac{m(m+1)}{2} = \Omega(m^2)$.

2.5.2 Pricing at Twice the Cost

Another natural algorithm is to price at twice (or any fixed multiple) of the *cost* of each item. However, while this can be shown to perform well for linear and low-degree polynomial cost functions, it performs poorly for the case of logarithmic costs. Indeed, consider a single item with procurement cost $c(x) = \log x$, and suppose we price the i^{th} item at cost $\pi(i) = 2 \log i$. Suppose the first m buyers have valuations $2 \log 1, 2 \log 2, \ldots, 2 \log m$ respectively, and are followed by m^2 buyers with valuation $2 \log m = \log m^2$. The algorithm would sell to the first m buyers, getting a social welfare of $\sum_{i=1}^{m} (2 \log i - \log i) = O(m \log m)$, after which the cost would be too high for the remaining buyers. In contrast, optimum would sell to the last m^2 buyers, and get a social welfare of $\sum_{i=1}^{m^2} (\log m^2 - \log i) = \Omega(m^2)$.

2.6 Algorithm: Pricing at twice the index

The first two ideas for pricing items with procurement costs are perhaps to (a) sell at cost, or (b) sell at some constant times the cost; however, as we have seen in Section 2.5, these schemes fail even for simple cost functions like linear and logarithmic procurement costs, respectively. In this section, we consider the next natural pricing scheme: *The price $\pi_i(k)$ of the k^{th} copy of an item is the procurement cost of the $(2k)^{th}$ copy.* I.e.,

$$\pi_i(k) := c_i(2k).$$

There is nothing special about pricing at *twice* the index, other factors would work as well, just giving slightly different bounds. We shall analyze this algorithm for function classes including polynomial $c_i(x) = x^d$ and logarithmic $c_i(x) = \ln(1 + x)$. Since these functions

are strictly increasing and hence invertible, hence we shall have $c_i^{\mathsf{inv}}(c_i(x)) = x$ for all $x \geq 0$. To analyze this algorithm, we shall use the result of Lemma 2.4.3.

(**Single resource goodness condition for Twice the Index pricing scheme**)
To show (α, β)-single-resource-goodness, we want to show that for all $x \in \mathbb{N}^{\geq 0}$,

$$\sum_{k=1}^{x} (c(2\,k) - c(k)) \geq \left(\sum_{k=1}^{c^{\mathsf{inv}}(c(2\,(x+1)))} (c(2\,(x+1)) - c(k)) - \beta \right) / \alpha$$

This is since the price $\pi(k)$ of the k^{th} copy is $c(2\,k)$. For ease of notation, call $A(x) = \sum_{k=1}^{c^{\mathsf{inv}}(c(2\,(x+1)))} (c(2\,(x+1)) - c(k))$ and $B(x) = \sum_{k=1}^{x} (c(2\,k) - c(k))$.

2.6.1 Performance on some cost functions

We now show that for some "well-behaved" classes of functions, we get $A_i(x) \leq \alpha \cdot \mathsf{profit}_i(x) + \beta_i$; the β_i term will usually depend on the procurement cost of the first few copies of the items—hence we will guarantee that we get a multiplicative α-fraction of the welfare if we ignore the procurement cost of the first few copies.

- Linear procurement costs: $c(x) = a\,x + b$ for some constant $a, b \geq 0$. It is easy to verify that then we have $A(x) = a(x+1)(2x+1)$, and $B(x) = \frac{1}{2} a\,x(x+1)$, and hence we have $(6, a)$-single-resource-goodness. Lemma 2.4.2 implies that

$$\begin{aligned} W(\mathsf{alg}) &\geq \tfrac{1}{6} \left(W(\mathrm{opt}) - \textstyle\sum_{i \in U} a_i \right) \\ &= \tfrac{1}{6} \left(W(\mathrm{opt}) - \textstyle\sum_{i \in U} (c_i(2) - c_i(1)) \right) , \end{aligned}$$

  where $c_i(x) = a_i\,x + b_i$. This result, with suitably modified guarantees, can easily be extended to the case where the actual procurement cost lies between *two linear curves* whose slopes are within a constant factor of each other.

- Polynomial procurement costs: $c_i(x) = a_i x^d$ for $d > 1$. Then $A_i(x) \leq a_i \frac{d}{d+1} (2(x+1))^{d+1}$, whereas $B_i(x) \geq a_i \frac{1}{d+1} (2^d - 1)\, x^{d+1}$, so some algebra implies that $(12\,d, 2^{d+1} (d+2)^{d+1}\, a_i)$-single-resource-goodness. Hence

$$W(\mathsf{alg}) \geq \tfrac{1}{12\,d} \left(W(\mathrm{opt}) - 2\,(d+2)^{d+1} \textstyle\sum_{i \in U} c_i(2) \right) .$$

Such a bound also holds for $c_i(x)$ being a polynomial of degree at most d with positive coefficients. The additive loss of $2^{O(d \log(d))}$ should be compared to the lower bound of $\Omega(2^d/d)$ in Corollary 2.6.3

- Logarithmic procurement costs: $c_i(x) = \ln(1+x)$. By algebra, $A_i(x) \leq (2x+3)$, and $B_i(x) \geq \ln(\frac{3}{2})\, x$, so we have $(\frac{2}{\ln(3/2)}, 3)$-single-resource-goodness, and again, Lemma 2.4.2 implies $(\frac{\ln(3/2)}{2}, 3|U|)$-welfare approximation.

2.6.2 Trade-off between the multiplicative guarantee and additive loss

In the guarantees given above, gains in the multiplicative factor can be made while trading-off commensurate losses in the additive loss terms. Specifically, consider the polynomial procurement cost $c_i(x) = x^d$. For a given x_i, we have that $A_i(x_i) \leq \frac{d}{d+1}\left(2\left(x_i+1\right)\right)^{d+1}$ and $B_i(x_i) \geq \frac{(2^d-1)\, x_i^{d+1}}{d+1}$. Hence,

$$A_i(x_i) \leq \frac{d}{d+1}\left(2\left(x_i+1\right)\right)^{d+1} = \frac{d}{d+1}\left(1+1/x_i\right)^{d+1} 2^{d+1}\, x_i^{d+1} \leq 4\, d\left(1+1/x_i\right)^{d+1} B_i(x_i) \tag{2.4}$$

where we have used $\forall d \geq 1,\ 2^d - 1 \geq 2^{d-1}$. Therefore, using that

- for all $x_i \leq q$, $A_i(x_i) \leq \frac{d}{d+1}\left(2\left(x_i+1\right)\right)^{d+1} \leq \frac{d}{d+1}\left(2\left(q+1\right)\right)^{d+1}$, and
- for all $x_i > q$, $A_i(x_i) \leq 4\, d\left(1+1/x_i\right)^{d+1} B_i \leq 4\, d\left(1+1/q\right)^{d+1} B_i(x_i)$,

for any $q \geq 1$, we can write

$$\forall x_i \geq 0,\ A_i(x_i) \leq 4\, d\left(1+1/q\right)^{d+1} B_i(x_i) + \left(d/(d+1)\right)\left(2\left(q+1\right)\right)^{d+1}.$$

Denoting $\alpha_q = 4\, d\left(1+1/q\right)^{d+1}$ and $\beta_q = \left(d/(d+1)\right)\left(2\left(q+1\right)\right)^{d+1}$, we can write for any $q \geq 1$, we achieve (α_q, β_q)-single-resource-goodness. A large q means a higher additive loss but with the benefit of a lower multiplicative factor. Hence, depending on the specific situation, we can look for a sweet spot by varying the parameter q. In the previous section, we had chosen $q = d+1$ to give the result for polynomial case.

As we show in Section 2.6.3, a social-welfare maximizing algorithm which has no estimate of $W(\mathrm{opt})$ has to lose an additive factor. At a high level, q represents the number of initial copies which we are ready to lose.

While the "twice-the-index" algorithm works for the above cost functions, its behavior worsens if the function grows very fast; Section 2.6.4 shows a bad example for the algorithm. Hence, in the next section, we give a logarithmic-approximation algorithm for the case of general increasing procurement cost curve.

2.6.3 The Necessity of Additive Loss

If we do not have an estimates for $W(\mathrm{opt})$, we give a trade-off between the additive and multiplicative loss (even for a single item), for any algorithm where the prices are at least the procurement cost.

Lemma 2.6.1. *With no estimate of $W(\mathrm{opt})$ it is impossible for a deterministic algorithm to give a purely multiplicative guarantee i.e. a guarantee of the form*

$$W(\mathsf{alg}) \geq W(\mathrm{opt})/\alpha$$

for any finite α.

Proof. Suppose we have an algorithm $\mathcal{A}$ that gets such a α-approximation for all inputs. Consider a single item with procurement cost function $c(k) = k$. Suppose the price of the first copy is set to any $1 + \theta$, for $\theta > 0$. Then we can send in a single buyer with valuation $1 + \theta - \varepsilon$, getting a zero social welfare, whereas the optimal welfare is $\theta - \varepsilon > 0$. On the other hand, if the price of the first copy is 1, then first send a buyer with value 1, and then a buyer with value 1.9—the optimal welfare of 0.9 is achieved by selling to the second buyer, but we only sell to the first buyer, get zero welfare again. ■

Some Quantitative Trade-offs

Lemma 2.6.2. *For any deterministic pricing algorithm (in a single item setting) acting on procurement costs $c()$ and that price copies at at least their procurement cost, to give the guarantee $W(\mathsf{alg}) \geq (W(\mathrm{opt}) - \Delta)/\alpha$, it is necessary that $\alpha \geq \frac{c(2)-c(1)}{\Delta} - 1$.*

Proof. Let $\pi(1) = c(1) + \gamma$. Note that $\gamma \leq \Delta$ because otherwise a buyer sent in with valuation $c(1) + \gamma - \varepsilon$ would buy nothing and hence $W(\mathsf{alg}) = 0$ while $W(\mathrm{opt}) = \gamma - \epsilon$ and therefore $W(\mathsf{alg}) \geq (W(\mathrm{opt}) - \Delta)/\alpha$ would be false.

Now consider a sequence of two buyers, the first with valuation $c(1) + \gamma$ and the second with valuation $c(2) - \varepsilon$. The first buyer will buy the first copy. Since the price of second copy is at least $c(2)$, hence the second buyer won't buy. Hence, $W(\mathsf{alg}) = \gamma$ while

$W(\text{opt}) = c(2) - c(1) - \varepsilon$. In such a scenario, for the guarantee to hold we require that $\gamma \geq (c(2) - c(1) - \varepsilon - \Delta)/\alpha$ which implies that $\gamma\,\alpha + \Delta \geq c(2) - c(1) - \varepsilon$. Noting that $\gamma \leq \Delta$ and that the inequality needs to hold for any $\varepsilon \geq 0$, the claim follows. ■

The following corollary follows immediately.

Corollary 2.6.3. *For procurement curves $c(x) = x^d$, for $\alpha = 4\,d$, $\Delta = \Omega(2^d/d)$.*

2.6.4 Bad Example for Pricing at Twice the Index

Here is an example where twice-the-index algorithm fails to produce good social welfare—e.g., consider the limited supply-like setting where $c(k) = 0$ for $k \leq T$, and $c(k) = V$ for $k > T$. Consider sending in T buyers with valuation zero, followed by T buyers with valuation $V - \varepsilon$. Twice-the-index prices the first $T/2$ copies at zero, and the rest at V, whence we get zero welfare, whereas the optimal welfare of $T(V - \varepsilon)$ is achieved by selling to just the later T buyers.

2.7 General Increasing Cost Functions

In this section, we present an algorithm that applies to general increasing cost functions, giving a logarithmic approximation minus an additive term that depends on the cost function (Theorem 2.7.2). The guarantee is achieved through a simple discretization of the cost function that allows us to reduce to the case of step functions and apply the algorithm of Bartal et al. [2003]. In fact, we get a multiplicative logarithmic approximation to $W(\text{opt})$ as long as the procurement cost of the first few logarithmic copies of all the items is small compared to $W(\text{opt})$. For the $0 - \infty$ procurement cost setting (i.e. the first few copies at zero cost and subsequent at an extremely high cost), if we have $\Omega(\log nm)$ copies of each item available at zero cost, the additive loss is zero and the algorithm presented here gets a logarithmic fraction of the optimal social welfare just as in Bartal et al. [2003].

2.7.1 Algorithm

Before describing the algorithm, let us introduce some notation. Define U_{max} as the maximum welfare any single buyer can achieve. Mathematically,

$$U_{max}(U, B) = \max_{b \in B} \max_{T \subseteq U} \left(v_b(T) - \sum_{i \in T} c_i(1) \right), \tag{2.5}$$

Note that the optimal social welfare, $W(\text{opt})$, lies between U_{max} and $m \cdot U_{max}$. The algorithm requires a parameter Z which satisfies $Z \in (U_{max}, U_{max}/\epsilon]$ [2]. For item i, define $\ell_i = \min\{c_i^{\mathsf{inv}}(Z), m\}$ and $c_i^{\mathsf{invt}}(p) = \min\{c_i^{\mathsf{inv}}(p), c_i^{\mathsf{inv}}(Z), m\}$. We can think of ℓ_i as the 'effective' number of copies of item i that are available and of $c_i^{\mathsf{invt}}(p)$ as the function which gives the 'effective' number of copies of item i whose procurement cost is at most p; $c_i^{\mathsf{invt}}(p)$ is the maximum number of copies of item i that opt can allocate before the procurement cost exceeds p (Lemma 2.4.4). Note that using c_i^{invt} (as opposed to using c_i^{inv}) is a technicality; one can imagine $c_i^{\mathsf{invt}} \approx c_i^{\mathsf{inv}}$ for a first read.

We now describe the pricing algorithm. In order to price copies for an item i, the algorithm divides ℓ_i copies into contiguous steps and each step has τ_i number of copies where $\tau_i = \lceil \log(4\, n\, \ell_i/\epsilon) \rceil$; hence the first step contains copies 1 through τ_i, the second from $\tau_i + 1$ through $2\tau_i$ and so on. Let s_{rq} denote the q^{th} copy relative to the r^{th} step; note that q varies from 1 to τ. The procurement cost of copy s_{rq} is therefore $c_i((r-1) \cdot \tau_i + q)$; the first copy in step r has cost $c_i((r-1)\tau_i + 1)$ and the last copy has cost $c_i(r \cdot \tau_i)$.

The algorithm sets the price of copy s_{rq} as

$$\pi_i(s_{rq}) = \frac{\epsilon\, Z}{4\, n\, \ell_i} \cdot 2^q + c_i(r \cdot \tau_i)$$

so that the first copy in step r has price $\frac{\epsilon Z}{4\, n\, \ell_i} + c_i(r \cdot \tau_i)$ while the last copy has price at least $Z + c_i(r \cdot \tau_i)$. Note that since any copy in the r^{th} step has procurement cost at most $c_i(r \cdot \tau_i)$, therefore, the price of every copy in the r^{th} step is greater than its procurement cost.

For every item, the algorithm sells copies of the item in increasing order of prices, so it might so happen that after the sale of a few copies from the first step, copies from the second step start selling, even before all copies of the first step are exhausted, since the copies in the second step are cheaper than the copies remaining in the first step.

2.7.2 Analysis

The crucial lemma of this section that will help prove the social welfare guarantee is

Lemma 2.7.1. *For every item $i \in U$, the price curve is $(4 \cdot \tau_i, \frac{\epsilon Z}{2\, n} + (c_i(\tau_i) - c_i(1)) \cdot \tau_i)$-single-resource-good.*

[2] We can remove this assumption at a further loss of $O(\log W(\text{opt}) \, (\log \log W(\text{opt}))^2)$ in the approximation guarantee [Balcan et al., 2008].

We now use Lemma 2.7.1 to prove the main result of this section. Theorem 2.7.2 roughly states that the social welfare achieved by the algorithm is a logarithmic approximation to (optimal social welfare minus the sum of procurement cost of the first few copies of every item).

Theorem 2.7.2. *Given a parameter $Z \geq U_{max}$, alg is $(4 \cdot \max_{i \in U} \tau_i, \frac{\epsilon Z}{2} + \sum_{i \in U}(c_i(\tau_i) - c_i(1)) \cdot \tau_i)$-welfare approximate. Furthermore, if $Z \in (U_{max}, U_{max}/\epsilon]$, then*

$$W(\textsf{alg}) \geq \frac{W(\text{opt})/2 - \sum_{i \in U}(c_i(\tau_i) - c_i(1)) \cdot \tau_i}{4 \cdot \max_{i \in U} \tau_i}$$

where $\tau_i = \lceil \log(4\, n\, \ell_i/\epsilon) \rceil$ and $\ell_i = \min\{c_i^{\textsf{inv}}(Z), m\}$.

Proof. That alg is $(4 \cdot \max_{i \in U} \tau_i, \frac{\epsilon Z}{2} + \sum_{i \in U}(c_i(\tau_i) - c_i(1)) \cdot \tau_i)$-welfare approximate follows from Lemma 2.7.1 and Lemma 2.4.4. The "furthermore" part of the claim follows from the observation that if $Z \in (U_{max}, U_{max}/\epsilon]$, then $\epsilon Z/2 \leq W(\text{opt})/2$. ■

We now need to prove Lemma 2.7.1. The analysis below considers any particular item $i \in U$. Recall that P_i^f denotes the price of the lowest price unsold copy of item i. Let t be the step which contains the copy $c_i^{\textsf{invt}}(P_i^f)$. Define for $1 \leq r < t$, $s_r = \tau_i$, and $s_t = \min\{\tau_i, c_i^{\textsf{invt}}(P_i^f) - (t-1)\tau_i\}$ so that we have $\sum_{r=1}^{t} s_r = c_i^{\textsf{invt}}(P_i^f)$. Further, for item i, let $\textsf{profit}_i(r)$ denote the total profit made by the algorithm from the sales of copies of the item from its r^{th} step. Finally for convenience of analysis define $c_i(0) = c_i(1)$.

The following lemma bounds the left hand side of the inequality claimed in Lemma 2.7.1 in terms of a related quantity.

Lemma 2.7.3. $\sum_{k=1}^{c_i^{\textsf{invt}}(P_i^f)}(P_i^f - c_i(k)) \leq \sum_{r=1}^{t}(P_i^f - c_i((r-1) \cdot \tau_i)) \cdot s_r$

Proof. Note that $\sum_{k=1}^{c_i^{\textsf{invt}}(P_i^f)}(P_i^f - c_i(k)) = \sum_{r=1}^{t}\sum_{x=1}^{s_r}(P_i^f - c_i((r-1) \cdot \tau_i + x))$ where we have broken up the summation across the different steps. Finally, $c_i((r-1) \cdot \tau_i + x) \geq c_i((r-1) \cdot \tau_i)$ since we are dealing with a non-decreasing procurement curve $c_i()$ and therefore for each r, we have $\sum_{x=1}^{s_r}(P_i^f - c_i((r-1) \cdot \tau_i + x)) \leq (P_i^f - c_i((r-1) \cdot \tau_i)) \cdot s_r$. This gives us the desired result. ■

Lemma 2.7.4. *For each step r such that $2 \leq r \leq t$, $(P_i^f - c_i(r \cdot \tau_i)) \leq 2 \cdot \textsf{profit}_i(r) + \frac{\epsilon Z}{4\, n\, \ell_i}$.*

Proof. This is because

– either $(P_i^f - c_i(r \cdot \tau_i)) > \frac{\epsilon Z}{4\, n\, \ell_i}$, in which case $\mathsf{profit}_i(r) \geq (P_i^f - c_i(r \cdot \tau_i))/2$.

  This is because for every p such that $\frac{\epsilon Z}{4\, n\, \ell_i} \leq p \leq Z$, the r^{th} step has a copy whose price is in the range $[p/2 + c_i(r \cdot \tau_i), p + c_i(r \cdot \tau_i))$ and hence in particular, there is a copy in the r^{th} step whose price q is in the range $[(P_i^f - c_i(r \cdot \tau_i))/2 + c_i(r \cdot \tau_i), (P_i^f - c_i(r \cdot \tau_i)) + c_i(r \cdot \tau_i)) = [(P_i^f - c_i(r \cdot \tau_i))/2 + c_i(r \cdot \tau_i), P_i^f)$. Therefore the price q of such a copy is strictly less than P_i^f and since the P_i^f is the price of the lowest priced unsold copy of item i, therefore the copy at price q must have been sold. Any copy in r^{th} step has procurement cost at most $c_i(r \cdot \tau_i)$, hence the sale of a copy at price $q \geq (P_i^f - c_i(r \cdot \tau_i))/2 + c_i(r \cdot \tau_i)$ must result in a profit of at least $(P_i^f - c_i(r \cdot \tau_i))/2$.

– or $(P_i^f - c_i(r \cdot \tau_i)) \leq \frac{\epsilon Z}{4\, n\, \ell_i}$.

Since $\mathsf{profit}_i(r)$ is a non-negative quantity, hence we see that in both cases the desired inequality is satisfied. ■

Proof of Lemma 2.7.1 :

Note that

$$\sum_{r=1}^{t} (P_i^f - c_i((r-1) \cdot \tau_i)) \cdot s_r = (P_i^f - c_i(0)) \cdot s_1 + \sum_{r=2}^{t} (P_i^f - c_i((r-1) \cdot \tau_i)) \cdot s_r \quad (2.6)$$

First, using Lemma 2.7.4 we bound the second summation on the right hand side of equation (2.6).

$$\begin{aligned}
\sum_{r=2}^{t} (P_i^f - c_i((r-1) \cdot \tau_i)) \cdot s_r &\leq 2 \cdot \sum_{r=2}^{t} \mathsf{profit}_i(r-1) \cdot s_r + \sum_{r=2}^{t} \frac{\epsilon Z}{4\, n\, \ell_i} \cdot s_r \\
&\leq 2 \cdot (\max_r s_r) \cdot \sum_{r=2}^{t} \mathsf{profit}_i(r-1) + \frac{\epsilon Z}{4\, n\, \ell_i} \cdot \sum_{r=2}^{t} s_r \\
&\leq 2 \cdot \tau_i \cdot \mathsf{profit}_i + \frac{\epsilon Z}{4\, n} \qquad (2.7)
\end{aligned}$$

where in the last inequality we have used $\sum_{r=2}^{t} s_r = c_i^{\mathsf{invt}}(P_i^f) \leq \ell_i$ and that $\tau_i \geq s_r$ for any r.

Now we bound the first term on the right hand side of equation (2.6).

$$\begin{aligned}(P_i^f - c_i(0)) \cdot s_1 &= (P_i^f - c_i(\tau_i)) \cdot s_1 + (c_i(\tau_i) - c_i(0)) \cdot s_1 \\ &\leq 2 \cdot \tau_i \cdot \mathsf{profit}_i(1) + \frac{\epsilon Z}{4n} + (c_i(\tau_i) - c_i(0)) \cdot s_1 \\ &\leq 2 \cdot \tau_i \cdot \mathsf{profit}_i + \frac{\epsilon Z}{4n} + (c_i(\tau_i) - c_i(0)) \cdot s_1 \end{aligned} \tag{2.8}$$

where the first inequality follows from Lemma 2.7.4 and the second follows from noting that the total profit profit_i made through sales of copies of item i is at least as much as the profit $\mathsf{profit}_i(1)$ made through the sale of copies from the first step of the item.

Using Lemma 2.7.3 and Equations (2.6), (2.7) and (2.8) derived above we get

$$\sum_{k=1}^{c_i^{\mathsf{invt}}(P_i^f)} (P_i^f - c_i(k)) \leq 4 \cdot \tau_i \cdot \mathsf{profit}_i + 2 \cdot \frac{\epsilon Z}{4n} + (c_i(\tau_i) - c_i(0)) \cdot \tau_i$$

Using Claim 2.7.3, and noting that by definition $c_i(0) = c_i(1)$, and Lemma 2.4.4, we get the desired result. ■

2.8 Smoothing Algorithm

The pricing algorithm of Section 2.7 gives us a logarithmic multiplicative guarantee along with some additive loss for all increasing cost curves. Twice-the-index algorithm presented in Section 2.6 gives a constant approximation factor plus an additive loss for polynomial curves. This raises the question of whether there is a pricing algorithm which can achieve the best of both the worlds i.e. give logarithmic multiplicative guarantees for general curves but constant factor guarantee for nice curves such as polynomial and logarithmic. In this section we present a pricing algorithm that achieves that for the case of *convex* cost functions. It gives logarithmic guarantees for general convex curves (Corollary 2.8.15) but in addition, gives for polynomial cost curves, a constant factor approximation (Theorem 2.8.25).

2.8.1 Intuition

Ideally, we would like to set prices which are sufficiently far above the cost curve (so that we generate a large social welfare), yet not be too far above it (else the high prices

may result in no sales, causing a large additive loss). Hence, we run into problems when the cost curve increases sharply—and the intuitive goal is to create a price curve which smooths out these sharp changes in the cost curve while staying "close" to it.

The smoothing algorithm takes the cost curve, and creates a price function which is a monotone step function: copies of the item are grouped into intervals, with all copies in an interval having the same price. We call these intervals "price intervals". The algorithm creates the price curve from *right to left*. If we think of ℓ_i as the effective number of copies of item i and Z as the highest price, then the ℓ_i^{th} copy is priced first at price Z through creation of the price interval $[\lfloor \frac{2}{3}\,\ell_i \rfloor, \infty)$[3], with items in this interval priced at Z; subsequently, price intervals are created progressively moving leftwards until we have priced the first copy. At each point, we use the intuition from above: if the price is much higher than the cost, we set the price for the new interval such that the price-cost gap is slashed by a factor of 2, else we set the price to maintain a sufficient gap from the cost.

2.8.2 The smoothing algorithm

Before we give the algorithm (in Figure 2.2), let us give some definitions; we urge the impatient reader to jump to Section 2.8.3 to get a quick rough feel of the algorithm. We assume that the cost of the first copy of every item is 0 i.e. $\forall i, c_i(1) = 0$ [4]. Recall the notation U_{max} defined in Equation 2.5 in Section 2.7; it represented the maximum welfare which can be made through a single buyer. In the present scenario since $c_i(1) = 0$, hence U_{max} equals $\max_{b \in B} \max_{T \subseteq U} v_b(T)$

Define $\ell_i = \min\{c_i^{\mathsf{inv}}(Z), m\}$ and $B_i = \lceil 12 \log(4n\ell_i/\epsilon) \rceil$. Similar to Section 2.7, at a high level, think of ℓ_i as being the "effective number" of copies of item i available, and B_i as the "number of different price levels" we create in our price curve. $c_i^{\mathsf{invt}}(p)$, as in Section 2.7, is defined as the "truncated" value $\min\{c_i^{\mathsf{inv}}(p), c_i^{\mathsf{inv}}(Z), m\}$; please refer to Section 2.7 to get a sense of why c_i^{invt} is defined the way it is. Define $\mathsf{width}_i(p) := \left\lfloor \frac{c_i^{\mathsf{invt}}(p)}{B_i} \right\rfloor$; this function will determine the number of copies we group together in a price interval. We assume that

$$\ell_i \geq B_i \geq 12; \tag{2.9}$$

see Claim 2.8.6 for why this is without loss of generality.

Let $\pi_i : \mathbb{Z}_+ \to \mathbb{R}_+$ be the price function and let $\mathcal{J}_i$ denote the set of price intervals for item i, and with $z_i = |\mathcal{J}_i|$. We refer to the q^{th} interval of item i as J_{iq}, with J_{i1} being

[3] We abuse notation slightly by denoting the integer interval $\{r, r+1, \ldots, s-1\}$ as the half-open real interval $[r, s)$.

[4] In Lemma 2.8.9 we show that this is without loss of generality.

1: for all $x \geq \lfloor \frac{2}{3}\, \ell_i \rfloor$, **set** $\pi_i(x) := Z$
2: **set** $x \leftarrow \lfloor \frac{2}{3}\, \ell_i \rfloor$
3: **while** $x > 1$ **do**
4: **if** $\mathsf{width}_i(\pi_i(x)) \geq 1$ **then**
5: **set** $x' \leftarrow \max\{x - \mathsf{width}_i(\pi_i(x)), 1\}$
6:

$$\textbf{set } \Delta = \begin{cases} \frac{\pi_i(x) - c_i(x)}{2} & \text{if } \pi_i(x) \geq 3\, c_i(x) \\ \frac{c_i(x)}{2} & \text{otherwise} \end{cases}$$

7: for all $y \in [x', x)$, **set** $\pi_i(y) := c_i(x) + \Delta$
8: **set** $x \leftarrow x'$
9: **else**
10: for all $y \in [1, x)$, **set** $\pi_i(y) := \pi_i(x)$
11: **set** $x \leftarrow 1$

Figure 2.2: Smoothing algorithm

the price interval that contains the first copy of item i, and $J_{iz_i} = [\lfloor \frac{2}{3}\, \ell_i \rfloor, \infty)$. Let $\pi_i(J_{iq})$ be the price of the copies in the interval J_{iq}. Depending on the procurement curve, two consecutive price intervals may have the same price. Also, we will formally state later that the prices we generate are non-decreasing, and always stay above the procurement cost for all copies less than ℓ_i.

2.8.3 The main ideas

Smoothing: Step 6 ensures a smooth price curve: if the price is more than thrice the procurement cost, we slash the gap between the price and procurement cost by two else we allow the price to stay at a sufficient gap from the cost.

Price Interval Size : The idea of the analysis is to show that whenever the number of copies sold moves from a lower price interval to a higher one, the social welfare generated by selling copies at the lower price is enough to be competitive against opt, even if we sell no further copies at the higher price. Consequently, the size of a price interval J_{iq} must depend on the price of items in the next interval $J_{i\,q+1}$. It turns out that to get a

multiplicative approximation factor of $O(B_i)$, if the price of copies in $J_{i\,q+1}$ were P, it suffices to set the width of $J_{i\,q}$ to be $\lfloor \frac{c_i^{\mathsf{invt}}(P)}{B_i} \rfloor = \mathsf{width}_i(P)$. Here is a simple special case that illustrates why: suppose only item i was being sold and we sold all copies from J_{iq} but no copies from interval $J_{i\,q+1}$. We would like to apply Lemma 2.4.4. The final price P_i^f in that case is $P = \pi_i(J_{i\,q+1})$. Staring at the left hand side of Equation (2.3), we see that it is at most $P \cdot c_i^{\mathsf{invt}}(P)$. Since we sold all the copies in price interval J_{iq}, we sold at least $|J_{iq}| = \lfloor \frac{c_i^{\mathsf{invt}}(P)}{B_i} \rfloor$ many copies, each at profit at least $P/6$ (something we will prove later). Hence on the right hand side of Equation (2.3), the term profit_i is at least $P \cdot \lfloor \frac{c_i^{\mathsf{invt}}(P)}{B_i} \rfloor / 6$. Putting $\alpha = O(B_i)$ we satisfy Equation (2.3) and thereby get an $O(B_i)$ approximation. Since the width of J_{iq} depends on the price of $J_{i\,q+1}$, it is natural that our pricing algorithm creates price intervals from *right to left*.

Termination: The algorithm terminates in one of two ways: either while creating such appropriately sized price intervals, we hit the first copy (i.e., $x' \leftarrow 1$ in Step 5, and then the loop condition fails in Step 3) or the price p of some price interval is low enough that $p < c_i(B_i)$, which implies $c_i^{\mathsf{invt}}(p) < B_i$ (the proof of implication appears later) and therefore $\mathsf{width}_i(p) = \lfloor \frac{c_i^{\mathsf{invt}}(p)}{B_i} \rfloor < 1$: this causes $x \leftarrow 1$ in Step 11. In the latter case, the price has become low enough that we can simply group all remaining copies into the lowest priced interval J_{i1} at price p. The subsequent analysis will often have to separately consider these two cases: whether $x \leftarrow 1$ is achieved in Step 5 or in Step 11.

2.8.4 The Analysis

The main result of this section is the following:

Lemma 2.8.1. *Given an estimate $Z \in (U_{max}, U_{max}/\epsilon]$, the smoothing algorithm on a non-decreasing cost curve is $(12B_i, \pi_i(J_{i2}) \cdot c_i^{\mathsf{invt}}(\pi_i(J_{i2}))$-single-resource-good, where $B_i := \lceil 12 \log(4n\ell_i/\epsilon) \rceil$, and $\ell_i := \min\{c_i^{\mathsf{inv}}(Z), m\}$.*

And this result with the use of Lemma 2.4.4 immediately yields the welfare approximation guarantees of the smoothing algorithm.

Theorem 2.8.2. *The social welfare $W(\mathsf{alg})$ achieved by the smoothing algorithm on a non-decreasing cost curve given an estimate $Z \in (U_{max}, U_{max}/\epsilon]$ satisfies*

$$W(\mathsf{alg}) \geq \frac{W(\mathrm{opt}) - \sum_{i \in U} \pi_i(J_{i2}) \cdot c_i^{\mathsf{invt}}(\pi_i(J_{i2}))}{12 \max_{i \in U} B_i},$$

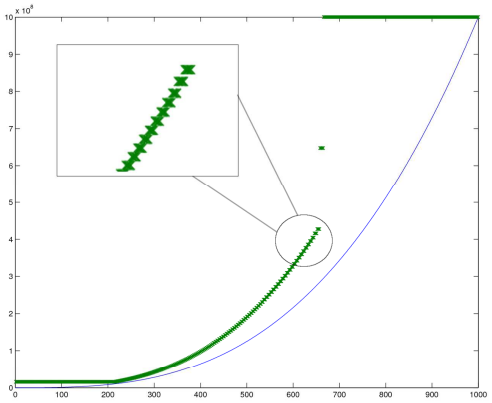


Figure 2.3: The figure shows the pricing curve drawn by the smoothing algorithm for the procurement curve $c_i(x) = x^3$. The lower line is the procurement curve. The upper thicker line is the pricing curve. We can observe that the price curve is flat towards the extreme right; this flat region contains the right-most price interval. Towards the extreme left the price curve *appears* to be a smooth curve. The inset shows the individual price intervals.

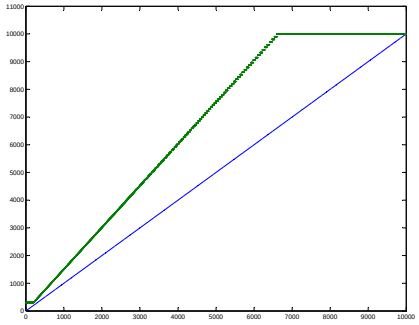


Figure 2.4: The figure shows the pricing curve drawn by the smoothing algorithm for the linear procurement curve

where $B_i := \lceil 12 \log(4n\ell_i/\epsilon) \rceil$, *and* $\ell_i := \min\{c_i^{\mathsf{inv}}(Z), m\}$.

Theorem 2.8.2 roughly states that the social welfare attained by the smoothing algorithm is a logarithmic approximation to (optimal social welfare minus the *price* of the first few copies of each item). Now to prove Lemma 2.8.1, we make use of two lemmas – Lemma 2.8.12 and Lemma 2.8.14. We now explain their role in the analysis.

Let us call an interval $J_{iq} = [r, s)$ to be *full-sized* if its width equals $\mathsf{width}_i(\pi_i(s))$. Note that the right-most interval J_{iz_i} is not full sized since it semi-infinite. Further, the left-most interval J_{i1} *may not* be full-sized either because the algorithm ran out of copies, or the price became too low so that all remaining unpriced copies were bunched together. We first show that *if we sell at least* $|J_{i1}| + |J_{i2}|$ *copies of item* i, i.e., we have sold at least one full-sized interval, we get a good approximation factor for the reasons we discussed in Section 2.8.3. This is proved in Lemma 2.8.12.

Then we consider the case when the *number of items sold is less than* $|J_{i1}| + |J_{i2}|$: in this case we cannot show a good multiplicative loss. Instead, we show that the price of items in the first two intervals is small in this case, which bounds the additive loss. This is

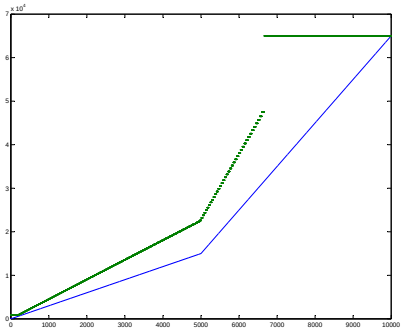


Figure 2.5: The figure shows the pricing curve drawn by the smoothing algorithm a piece-wise linear procurement curve. The lower line is the procurement curve. The upper thicker line is the pricing curve.

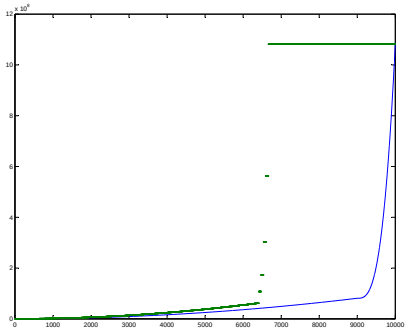


Figure 2.6: The figure shows the pricing curve drawn by the smoothing algorithm a procurement curve which grows as x^2 initially and as x^3 in the final phase. The lower line is the procurement curve. The upper thicker line is the pricing curve.

proved in Lemma 2.8.14.

All the intervals except the leftmost J_{i1} and rightmost J_{iz_i} ones are created in a similar fashion; intervals J_{i1} and J_{iz_i} have to treated as special cases at several points in the analysis. Also, the analysis which follows from this point onwards up till (and not including) Theorem 2.8.2 is *per item*. Hence the subscript i in the terms involved is irrelevant from the point of analysis and is present only to maintain uniformity in presentation.

To begin, we state some useful properties of the prices and widths of the intervals.

Lemma 2.8.3 (Prices and Widths)**.** *The following facts about interval prices hold for the intervals in $\mathcal{J}_i$ for any non-decreasing cost curve:*

a. For any $J_{iq} = [r, s)$ such that $q \neq z_i$, $\pi_i(J_{iq}) \geq \frac{3}{2}\, c_i(s)$. Hence, $\pi_i(x) \geq \frac{3}{2}\, c_i(x)$ for $x \in J_{iq}$.

a'. If the cost curve is convex, $\pi_i(\lfloor \frac{2}{3}\, \ell_i \rfloor) \geq \frac{3}{2}\, c_i(\lfloor \frac{2}{3}\, \ell_i \rfloor)$.

b. For consecutive J_{iq} and $J_{i\,q+1}$ and $q \neq z_i - 1$, we have $\pi_i(J_{iq}) \leq \pi_i(J_{i\,q+1}) \leq 2\, \pi_i(J_{iq})$. If the cost curve is convex the claim also holds for $q = z_i - 1$.

c. All price intervals $J_{iq} = [r, s)$ ($q \notin \{1, z_i\}$) have $|J_{iq}| = \mathsf{width}_i(\pi_i(s)) = \mathsf{width}_i(\pi_i(J_{i\,q+1}))$.

Lemma 2.8.3(a) states that the price of any copy is sufficiently far from the procurement cost of that copy. Lemma 2.8.3(a') states the same claim about the left end of the

right-most interval J_{iz_i} in case the cost curve is in addition convex. Lemma 2.8.3(b) states the price of copies in the interval $J_{i\,q+1}$ is higher than that of $J_{i\,q}$, but not too far from it. Lemma 2.8.3(d) states that all price intervals except possibly the left-most and the right-most are full-sized. Armed with these facts, we first show that if "many" copies of item i are sold, then we are in good shape. The other case where "few" copies are sold, is dealt with subsequently.

We now prove Lemma 2.8.3. We start with a couple of observations which are easy to prove.

Observation 2.8.4. *$\mathsf{width}_i(p)$ is non-decreasing in p.*

Observation 2.8.5. *Assuming the parameter $Z > 0$, for every copy x, the price set by the algorithm, $\pi_i(x) > 0$.*

Claim 2.8.6. *In the analysis of the smoothing algorithm, it is sufficient to consider only those items that have $\ell_i \geq B_i$.*

Proof Sketch: We would like to show that we can assume $\ell_i \geq B_i \geq 12$ without loss of generality. We first show that $B_i \geq 12$. Recall that $\ell_i := \min\{c_i^{\mathsf{inv}}(Z), m\}$ and $B_i = \lceil 12 \log(4n\ell_i/\epsilon) \rceil$. We can assume that $U_{max} > 0$, so $Z > 0$, and since $c_i(1) = 0$, hence $\ell_i \geq 1$; in turn this implies that $B_i \geq 12$.

Now, if the minimum $\ell_i < B_i$ because m is small, we can always add in dummy buyers, this does not change any of the arguments. Else, it must be the case that $\ell_i = c_i^{\mathsf{inv}}(Z) < B_i$, which means $c_i(B_i) > Z \geq U_{max}$. We claim that we can just drop all such items from the instance, and run our algorithm on the remaining items, with guarantees identical to those in Theorem 2.8.2.

Indeed, how many copies of item i could we possibly sell in the optimal solution? At most ℓ_i, since after that its cost is at least $c_i(\ell_i) \geq Z$, too high for opt to allocate to anyone without decreasing the social welfare as the cost exceeds the valuation. Therefore, since at most ℓ_i copies of such an item can be allocated, so ignoring this item entirely can drop $W(\mathrm{opt})$ by at most $\ell_i \cdot U_{max} < B_i \cdot c_i(B_i)$. Hence, dropping all such items implies that the remaining set of items U' (and the original set of buyers) have an modified optimal welfare of $W(\mathrm{opt}') \geq W(\mathrm{opt}) - \sum_{i \in U \setminus U'} B_i \cdot c_i(B_i)$. For this new instance, Theorem 2.8.2 gives a welfare of

$$W(\mathsf{alg}) \geq \frac{W(\mathrm{opt}')/2 - \sum_{i \in U'} B_i \cdot c_i(B_i)}{12 \max_{i \in U'} B_i} \geq \frac{W(\mathrm{opt})/2 - \sum_{i \in U} B_i \cdot c_i(B_i)}{12 \max_{i \in U} B_i} \tag{2.10}$$

Hence, we can assume $\ell_i \geq B_i \geq 12$ without loss of generality. ■

Lemma 2.8.7. *The number of price intervals,* $z_i \geq 3$

Proof. $z_i \neq 1$ since the first time the algorithm checks for condition $x > 1$ in Step 3, it evaluates to true because x is set to $\lfloor \frac{2}{3}\ell_i \rfloor$ by Step 2 and since by Equation (2.9), $\ell_i > 3$, therefore $x = \lfloor \frac{2}{3}\,\ell_i \rfloor \geq \ell_i/3 > 1$. Hence, the algorithm creates at least one price interval other than $[\lfloor \frac{2}{3}\,\ell_i \rfloor, \infty)$.

We now prove that z_i is at least 3. We prove by contradiction. If it were the case $z_i = 2$, then it implies that the algorithm terminates the second time it checks for the condition $x > 1$ in Step 3. As observed earlier, x can be set to 1 either by Step 5 or by Step 11. To disambiguate let the value of x be x_1 and x_2 the first and second time respectively, the `while` loop condition at Step 3 is checked. We know from Step 2, that $x_1 = \lfloor \frac{2}{3}\,\ell_i \rfloor$. For z_i to be 2, we require x_2 to be 1.

- If x_2 is set to 1 by Step 11, it implies that the condition $\mathsf{width}_i(\pi_i(x_1)) \geq 1$ in Step 4 must have evaluated to false. However, $\pi_i(x_1) = Z$ (by Step 1) and therefore, $\mathsf{width}_i(\pi_i(x_1)) = \lfloor c_i^{\mathsf{invt}}(Z)/B_i \rfloor$. Now $c_i^{\mathsf{invt}}(Z) = \min\{c_i^{\mathsf{inv}}(Z), \ell_i\}$ and $\ell_i = \min\{c_i^{\mathsf{inv}}(Z), m\}$ and therefore, $c_i^{\mathsf{invt}}(Z) = \ell_i$. Hence, $\mathsf{width}_i(\pi_i(x_1)) = \lfloor \ell_i/B_i \rfloor \geq 1$ since $\ell_i \geq B_i$ by Equation (2.9). Hence, x_2 could not have been set to 1 by Step 11.
- The other case is that x_2 is set to 1 by Step 5. This implies that $\max\{x_1 - \mathsf{width}_i(\pi_i(x_1)), 1\} = 1$. However, $x_1 - \mathsf{width}_i(\pi_i(x_1)) = \lfloor \frac{2}{3}\,\ell_i \rfloor - \lfloor \ell_i/B_i \rfloor \geq \ell_i/3 - \ell_i/B_i \geq 2$ which is satisfied due to Equation (2.9). Therefore, $x_2 > 2$ and hence could not have been set to 1 by Step 5.

This proves the contradiction. ■

Proposition 2.8.8 (The left-most interval)**.** *The following facts hold for the left-most interval* J_{i1}*:*

a. If the procedure terminated through Step 5 creating $J_{i1} = [1, s)$*, then* $|J_{i1}| \leq \mathsf{width}_i(\pi_i(s)) = \mathsf{width}_i(\pi_i(J_{i\,2}))$.

b. If the procedure terminated through Step 11, then $\pi_i(J_{i1}) = \pi_i(J_{i2})$.

Proof. If the algorithm terminated through Step 5, then by construction we have $|J_{i1}| \leq \mathsf{width}_i(\pi_i(s)) = \mathsf{width}_i(\pi_i(J_{i2}))$. If the algorithm terminated through Step 11, then we have no non-trivial bound on $|J_{i1}|$, however, by Step 10, we have $\pi_i(J_{i1}) = \pi_i(J_{i2})$. ■

Proof of Lemma 2.8.3 :

- Part (a): We first prove that for $J_{iq} = [r, s)$, $\pi_i(J_{iq}) > \frac{3}{2}c_i(s)$. First consider the case $q \neq 1$.
  - either $\pi_i(s) \geq 3\, c_i(s)$, in which case,

$$\pi_i(J_{iq}) = c_i(s) + \frac{\pi_i(s) - c_i(s)}{2} = \frac{\pi_i(s) + c_i(s)}{2} \geq \frac{3\, c_i(s) + c_i(s)}{2} = 2\, c_i(s).$$

  - or, $\pi_i(s) < 3\, c_i(s)$, in which case,

$$\pi_i(J_{iq}) = c_i(s) + \frac{c_i(s)}{2} = \frac{3}{2}\, c_i(s).$$

  In both cases, the inequality $\pi_i(J_{iq}) \geq \frac{3}{2}c_i(s)$ is true. Now for the case $q = 1$: the above argument also holds if the algorithm terminated in Step 5. If however the algorithm terminated in Step 11, then let $J_{i1} = [1, r)$ and $J_{i2} = [r, s)$ ($J_{i2} \neq J_{iz_i}$ by Lemma 2.8.7). The observations

  1. $\pi_i(J_{i1}) = \pi_i(J_{i2})$ implied by Proposition 2.8.8(b),
  2. $\pi_i(J_{i2}) \geq \frac{3}{2}\, c_i(s)$, which is at least $\frac{3}{2}\, c_i(r)$, the first implied by the above argument for $q \neq 1$ and the second implied by monotonicity of $c_i()$.

  together imply the result for J_{i1}.

  Having proved that $\pi_i(J_{iq}) > \frac{3}{2}c_i(s)$, note that since $c_i()$ is non-decreasing, therefore we have $c_i(x) \leq c_i(s)$ for all $x \in [r, s)$, so $\pi_i(x) = \pi_i(J_{iq}) \geq \frac{3}{2}c_i(s) \geq \frac{3}{2}c_i(x)$ which proves the second part of the claim.

- For part (a'), convexity implies that $c_i(\lfloor \frac{2}{3}\, \ell_i \rfloor) \leq \frac{2}{3}\, c_i(\ell_i)$. By the definition of ℓ_i, this is at most $\frac{2}{3}\, Z$. On the other hand, $\pi_i(\lfloor \frac{2}{3}\, \ell_i \rfloor) = \pi_i(J_{iz_i}) = Z$. Therefore, $\pi_i(\lfloor \frac{2}{3}\, \ell_i \rfloor) \geq \frac{3}{2}\, c_i(\lfloor \frac{2}{3}\, \ell_i \rfloor)$.

- For part (b), first consider the case where $q \notin \{1, z_i - 1\}$, where $J_{iq} = [r, s)$ and $J_{i\,q+1} = [s, t)$.
  - Either $\pi_i(s) \geq 3\, c_i(s)$: then

$$\begin{aligned} \pi_i(J_{iq}) = c_i(s) + \frac{\pi_i(s) - c_i(s)}{2} &= \frac{\pi_i(s) + c_i(s)}{2} \\ &\leq \frac{\pi_i(s) + \frac{1}{3}\, \pi_i(s)}{2} = \frac{2}{3}\, \pi_i(s) = \frac{2}{3}\, \pi_i(J_{i\,q+1})\ . \end{aligned}$$

  Also, $\pi_i(J_{iq}) = \frac{\pi_i(s) + c_i(s)}{2} \geq \frac{\pi_i(s)}{2} = \frac{1}{2}\pi_i(J_{i\,q+1})$. Hence, $\pi_i(J_{iq}) \leq \pi_i(J_{i\,q+1}) \leq 2\pi_i(J_{iq})$.

– Or $\pi_i(s) < 3\,c_i(s)$: then

$$\pi_i(J_{iq}) = c_i(s) + \frac{c_i(s)}{2} = \frac{3}{2}\,c_i(s) \leq \frac{3}{2}\,c_i(t) \leq \pi_i(t) = \pi_i(J_{i\,q+1})$$

where the first inequality follows from the monotonicity of c_i, and the second from Lemma 2.8.3(a). Further, $\pi_i(J_{iq}) = \frac{3}{2}\,c_i(s) > \frac{1}{2}\pi_i(s) = \frac{1}{2}\pi_i(J_{i\,q+1})$. Hence, we get $\pi_i(J_{iq}) \leq \pi_i(J_{i\,q+1}) \leq 2\,\pi_i(J_{iq})$.

Now for the case of J_{i1} ($q = 1$). Note that from Lemma 2.8.7, $z_i \geq 3$. Therefore in particular, $J_{i2} \neq J_{iz_i}$. The analysis above for J_{iq} also holds for J_{i1} if the algorithm terminated in Step 5. Otherwise, by Proposition 2.8.8(b), $\pi_i(J_{i1}) = \pi_i(J_{i2})$ in which case the both the inequalities trivially follow.

Finally for the case of $q = z_i - 1$. Let $J_{i\,z_i-1} = [r, s)$ and $J_{i\,z_i} = [s, \infty)$. Note that since s is left end point of J_{iz_i}, hence $s = \lfloor \frac{2}{3}\ell_i \rfloor$. Either $\pi_i(s) \geq 3\,c_i(s)$ in which case the analysis above for $q \notin \{1, z_i - 1\}$ holds for $q = z_i - 1$ as well and shows that $\pi_i(J_{i\,z_i-1}) \leq \pi_i(J_{iz_i}) \leq 2\pi_i(J_{i\,z_i-1})$; or $\pi_i(s) < 3\,c_i(s)$, in this case

$$\pi_i(J_{i\,z_i-1}) = c_i(s) + \frac{c_i(s)}{2} = \frac{3}{2}\,c_i(s) \leq \pi_i(s) = \pi_i(J_{iz_i})$$

where the second inequality follows from Lemma 2.8.3(a').

- Part (c): By construction (Step 5-7), for all price intervals $J_{iq} = [r, s)$ (except maybe J_{i1} and J_{iz_i}) we have $|J_{iq}| = \mathsf{width}_i(\pi_i(s))$. Since $s \in J_{i\,q+1}$, therefore, $\pi_i(s) = \pi_i(J_{i\,q+1})$ and hence we have $|J_{iq}| = \mathsf{width}_i(\pi_i(s)) = \mathsf{width}_i(\pi_i(J_{i\,q+1}))$.

■

Translating the Cost Curve We show that it is fine to translate the cost functions $c_i()$ to satisfy $c_i(1) = 0$.

Lemma 2.8.9. *Given a pricing algorithm $\mathcal{A}'$ for production cost curves $\{c'_i()\}$, which for any set of buyers achieves a guarantee of $(W(\mathrm{opt}) - \beta)/\alpha$, we can create a pricing algorithm $\mathcal{A}$ for the cost curves $c_i(x) = c'_i(x) + \delta_i \mathbf{1}_{x>0}$ for constants $\delta_i \geq 0$, that achieves the same guarantees.*

Moreover, in case $\mathcal{A}'$ needs an estimate of $\max_{b\in B} \max_{S\subseteq U} v_b(S)$, $\mathcal{A}$ takes as input an estimate of

$$\max_{b\in B} \max_{S\subseteq U} (v_b(S) - \sum_{i\in S} \delta_i).$$

Proof. The algorithm $\mathcal{A}$ just uses $\mathcal{A}'$ to generate the price functions $\pi'_i()$, and sets $\pi_i(x) = \pi'_i(x) + \delta_i \mathbf{1}_{x>0}$. To show the social welfare guarantee for $\mathcal{A}$, we consider any sequence of buyers $\sigma = b_1, b_2, \ldots, b_m$ for which the optimal welfare is $W(\text{opt}(\sigma, \{c_i\}))$.

Below, we show how to construct another sequence of fake buyers $\sigma' = b'_1, b'_2, \ldots, b'_m$, and prove that

$$W(\text{opt}(\sigma, \{c_i\})) = W(\text{opt}(\sigma', \{c'_i\})) \tag{2.11}$$
$$W(\mathcal{A}(\sigma, \{c_i\})) = W(\mathcal{A}'(\sigma', \{c'_i\})) \tag{2.12}$$

Now the algorithm $\mathcal{A}'$ gives the guarantee that for all σ',

$$W(\mathcal{A}'(\sigma', \{c'_i\})) \geq \frac{1}{\alpha}(W(\text{opt}(\sigma', \{c'_i\})) - \beta)$$

we would get the same guarantee for $\mathcal{A}$, and hence get the proof. The definition of the fake buyers b'_i is natural: their valuation function is $v'_i(S) = v_i(S) - \sum_{i \in S} \delta_i$—note that fake buyers may have non-monotone valuation functions, and they may also have negative values for some sets, but this is not a concern. Now to prove (2.11) and (2.12).

Claim 2.8.10. *For any $j \in [m]$, buyer $b_j \in \sigma$ buys the same set from $\mathcal{A}$ as $b'_i \in \sigma'$ buys from $\mathcal{A}'$.*

Proof. We prove this by induction. The base case is $j = 1$. Utility function $u_1(\cdot)$ for buyer b_1 is $\forall S \subseteq U,\ u_1(S) = v_1(S) - \sum_{i \in S} \pi_i(1)$ while the utility function $u'_1(\cdot)$ for buyer b'_1 is $\forall S \subseteq U,\ u'_1(S) = v'_1(S) - \sum_{i=1}^{S} \pi'_i(1)$. Using definition of v'_1 and π_i we get

$$\forall S \subseteq U,\ u'_1(S) = v'_1(S) - \sum_{i \in S} \pi'_i(1) = v_1(S) - \sum_{i \in S} c_i(1) - \sum_{i \in S} \pi'_i(1) = v_1(S) - \sum_{i \in S} \pi_i(1) = u_1(S).$$

Hence, b_1 and b'_1 have the same utility maximizing set and therefore they buy the same set of items.

Assume the induction hypothesis is true for $j < k$. We prove that buyer b_k and b'_k buy the same set of items. Since for all $j < k$, buyers b_j and b'_j bought the same set of items, therefore, the number of copies x_i and x'_i of item i sold by A and A' when b_k and b'_k arrive are equal. Therefore,

$$\begin{aligned}\forall S \subseteq U,\ u'_k(S) = v'_k(S) - \sum_{i \in S} \pi'_i(x'_i + 1) &= v_k(S) - \sum_{i \in S} c_i(1) - \sum_{i \in S} \pi'_i(x_i + 1) \\ &= v_i(S) - \sum_{i \in S} \pi_i(x_i + 1) = u_i(S).\end{aligned}$$

Hence, b_k and b'_k buy the same set of items. This completes the step of induction. Hence proved. ∎

Define an allocation vector $X_{ij} \in \{0,1\}^{n\times m}$ so that $X_{ij} = 1 \iff$ buyer b_j is assigned a copy of item i.

Claim 2.8.11. *Any allocation vector X achieves equal social welfare on buyer sequence σ with cost functions $\{c_i\}$, and on buyer sequence σ' with cost functions $\{c'_i\}$.*

Proof. Denote by y_i the number of copies of item i allocated under the scheme X_{ij}; hence $y_i = \sum_j X_{ij}$. Also, let $S_j \subseteq U$ denote the set of items allocated to j^{th} buyer.

Note that $W(X(\sigma, \{c_i\})) = \sum_j v_j(S_j) - \sum_{i\in U}\sum_{k=1}^{y_i} c_i(k)$ and $W(X(\sigma', \{c'_i\})) = \sum_j v'_j(S_j) - \sum_{i\in U}\sum_{k=1}^{y_i} c'_i(k)$. Using definition of $v'_i()$ and $\pi_i()$ we get

$$\begin{aligned}
W(X(\sigma, \{c_i\})) &= \sum_j v'_j(S_j) - \sum_{i\in U}\sum_{k=1}^{y_i} c'_i(k) \\
&= \sum_j \big(v_j(S_j) - \sum_{i\in S_j} c_i(1)\big) - \sum_{i\in U}\sum_{k=1}^{y_i} c'_i(k) \\
&= \sum_j v_j(S_j) - \sum_j \sum_{i\in U} X_{ij}\cdot c_i(1) - \sum_{i\in U}\sum_{k=1}^{y_i} c'_i(k) \\
&= \sum_j v_j(S_j) - \sum_{i\in U} y_i \cdot c_i(1) - \sum_{i\in U}\sum_{k=1}^{y_i} c'_i(k) \\
&= \sum_j v_j(S_j) - \sum_{i\in U}\sum_{k=1}^{y_i} (c_i(1) + c'_i(k)) \\
&= \sum_j v_j(S_j) - \sum_{i\in U}\sum_{k=1}^{y_i} c_i(k) = W(X(\sigma', \{c'_i\}))
\end{aligned}$$

which proves the claim. ■

Claim 2.8.11 says having the same allocations in the two settings achieves the same social welfare; this proves (2.11). Moreover, by Claim 2.8.10, the allocation made by $\mathcal{A}$ to buyer sequence σ is the same as that made by $\mathcal{A}'$ to buyers σ'; this proves (2.12).

Finally note that in case $\mathcal{A}'$ needs an estimate of $\max_{b'\in B}\max_{S\subseteq U} v'_b(S)$ for its guarantee to hold, then $\mathcal{A}$ passes the estimate of $\max_{b\in B}\max_{S\subseteq U}(v_b(S) - \sum_{i\in S}\delta_i$ since by definition of v'_b both quantities are equal. ■

The Case of Many Copies.

Suppose we sell *all* copies in some interval J_{iq} for $q > 1$: then we get that the profit made from that interval alone gives us a good approximation.

Lemma 2.8.12. *If the number of sold copies x_i of item i is at least $|J_{i1}| + |J_{i2}|$, then $P_i^f \cdot c_i^{\mathsf{invt}}(P_i^f) \leq 12B_i \cdot \mathsf{profit}_i$, where $\mathsf{profit}_i := \sum_{k=1}^{x_i}(\pi_i(k) - c_i(k))$.*

Proof. Let q be the largest integer such that $J_{iq} = [r, s)$ is completely sold out; hence $q \in [2, z_i)$. The final price is $P_i^f = \pi_i(J_{i\,q+1}) = \pi_i(s)$. We want to show we make a reasonable profit from the sales of copies in J_{iq}. From Lemma 2.8.3(c), there are $\mathsf{width}_i(\pi_i(s))$ many copies in J_{iq}. For each of these copies $k \in [r, s)$, the profit is $\pi_i(k) - c_i(k) \geq \pi_i(k) - c_i(s)$, because costs are non-decreasing.

However, by Step 7 of the pricing algorithm, for all $k \in J_{iq}$, $\pi_i(k) = c_i(s) + \Delta$, where Δ is determined by Step 6.

1. Either $\pi_i(s) \geq 3c_i(s)$, $\Delta = \frac{1}{2}(\pi_i(s) - c_i(s)) \geq \frac{1}{3}\pi_i(s)$,
2. Or $\pi_i(s) < 3c_i(s)$, $\Delta = c_i(s)/2 > \frac{1}{6}\pi_i(s)$.

So, we make a profit of at least $\pi_i(s)/6$ from each of the $\mathsf{width}_i(\pi_i(s)) = \lfloor \frac{c_i^{\mathsf{invt}}(\pi_i(s))}{B_i} \rfloor$ many copies in J_{iq}:

$$\mathsf{profit}_i \geq \tfrac{\pi_i(s)}{6} \cdot \lfloor \tfrac{c_i^{\mathsf{invt}}(\pi_i(s))}{B_i} \rfloor \geq \tfrac{\pi_i(s) \cdot c_i^{\mathsf{invt}}(\pi_i(s))}{12\,B_i},$$

where the last inequality is because $\lfloor t \rfloor \geq t/2$ for $t \geq 1$. Plugging in $P_i^f = \pi_i(s)$ completes the proof. ■

The Case of Few Copies

Now suppose item i is such that the number of copies we sell either lies within the left-most interval J_{i1}, or only covers a small fraction of the second interval J_{i2}: the argument given above does not hold in that case. However we can show the following result.

Lemma 2.8.13. *If the number of sold copies x_i of item i is less than $|J_{i1}| + |J_{i2}|$ then $\pi_i(P_i^f) \cdot c_i^{\mathsf{invt}}(P_i^f) \leq \pi_i(J_{i2}) \cdot c_i^{\mathsf{invt}}(\pi_i(J_{i2}))$.*

Proof. Since we end up selling less than $|J_{i1}| + |J_{i2}|$ copies, hence the final price P_i^f is at most $\max\{\pi_i(J_{i1}), \pi_i(J_{i2})\}$ which is $\pi_i(J_{i2})$ since Lemma 2.8.3(b) tell us that $\pi_i(J_{i1}) \leq \pi_i(J_{i2})$. Hence, $P_i^f \cdot c_i^{\mathsf{invt}}(P_i^f) \leq \pi_i(J_{i2}) \cdot c_i^{\mathsf{invt}}(\pi_i(J_{i2}))$ ($c_i^{\mathsf{invt}}(p)$ is non-decreasing function of p). ■

Finishing the Analysis

Lemma 2.8.12 and Lemma 2.8.13 together give us the main result of this section.

Proof of Lemma 2.8.1 : For each item i, depending on the number of copies sold, either Lemma 2.8.12 or Lemma 2.8.13 applies, which implies that for each $i \in U$,

$$P_i^f \cdot c_i^{\mathsf{invt}}(P_i^f) \leq 12\, B_i \cdot \mathsf{profit}_i + \pi_i(J_{i2}) \cdot c_i^{\mathsf{invt}}(\pi_i(J_{i2})).$$

Using Lemma 2.4.4, our result follows. ■

2.8.5 Convex cost curves

Theorem 2.8.2 leaves us unsatisfied since the additive loss, which is the *price* of the first few copies of each item, is not stated in terms of quantities that are part of the problem statement such as procurement cost. For convex curves, we are able to overcome that deficiency. The additive loss would be roughly the sum of *procurement cost* of the first few copies of every item. The crucial lemma which we will prove in this section is:

Lemma 2.8.14. *For a convex cost curve,* $\pi_i(J_{i2}) \cdot c^{\mathsf{invt}}(\pi_i(J_{i2})) \leq \max\{B_i\, c_i(B_i), \frac{\epsilon Z}{2\,n}\}$.

which will suffice to prove the following result.

Corollary 2.8.15. *The social welfare* $W(\mathsf{alg})$ *achieved by the smoothing algorithm on a non-decreasing convex cost curve given an estimate* $Z \in (U_{max}, U_{max}/\epsilon]$ *satisfies*

$$W(\mathsf{alg}) \geq \frac{W(\mathrm{opt})/2 - \sum_{i \in U} B_i \cdot c_i(B_i)}{12\, \max_{i \in U} B_i},$$

where $B_i := \lceil 12 \log(4n\ell_i/\epsilon) \rceil$, *and* $\ell_i := \min\{c_i^{\mathsf{inv}}(Z), m\}$.

Corollary 2.8.15 gives us the same approximation factor to optimal social welfare as guaranteed by Theorem 2.8.2, except that it states the additive loss to be the sum of *procurement cost* of first few copies of each item.

Proof of Corollary 2.8.15 : Putting Theorem 2.8.2 and Lemma 2.8.14 together,

$$W(\mathsf{alg}) \geq \frac{W(\mathrm{opt}) - \epsilon\, Z/2 - \sum_{i \in U} B_i \cdot c_i(B_i)}{12\, \max_{i \in U} B_i}\,.$$

Using $\epsilon Z \leq U_{max} \leq W(\mathrm{opt})$, we get the desired result. ■

We now need to prove Lemma 2.8.14. The pricing algorithm terminates when it has priced all the copies, i.e. x is set to 1 and the `if` condition in Step 3 becomes false. x can be set to 1 either in Step 8 (preceded by x' being set to 1 in Step 5) or in Step 11. We consider these two cases separately.

- Algorithm terminates through Step 11: Lemma 2.8.17 proves that $c_i^{\mathsf{invt}}(\pi_i(J_{i2})) \cdot \pi_i(J_{i2}) < B_i \cdot c_i(B_i)$.
- Algorithm terminates through Step 5: Lemma 2.8.22 proves that $c_i^{\mathsf{invt}}(\pi_i(J_{i2})) \cdot \pi_i(J_{i2}) < \frac{\epsilon Z}{2\,n}$.

Proof of Lemma 2.8.14 : The algorithm terminates either through Step 5 or Step 11 and Lemma 2.8.22 and Lemma 2.8.17 together indicate that $\pi_i(J_{i2}) \cdot c^{\mathsf{invt}}(\pi_i(J_{i2})) \leq \max\{B_i\, c_i(B_i), \frac{\epsilon Z}{2\,n}\}$. ■

Before proving Lemma 2.8.17 and Lemma 2.8.22, we state and prove the following lemma that characterizes the circumstances under which the algorithm terminates in either condition.

Lemma 2.8.16. *The pricing algorithm terminates through Step 11 if and only if* $\pi_i(J_{i2}) < c_i(B_i)$.

Proof. Let $J_{i2} = [s, r)$. We first prove that if $\pi_i(J_{i2}) < c_i(B_i)$, then the algorithm terminates in Step 11. If $\pi_i(s) = \pi_i(J_{i2}) < c_i(B_i)$, then it implies that $c_i^{\mathsf{inv}}(\pi_i(s)) < B_i$, and by definition of $c^{\mathsf{invt}}()$, $c_i^{\mathsf{invt}}(\pi_i(s)) < B_i$ which implies that $\mathsf{width}_i(\pi_i(s)) = \lfloor \frac{c_i^{\mathsf{inv}}(\pi_i(s))}{B_i} \rfloor = 0$. Hence, right after creation of J_{i2}, when the algorithm checks for the `if` condition on point s in Step 4, it shall evaluate to false and therefore, the algorithm shall terminate through Step 11.

To prove the other direction, if the algorithm terminates through Step 11, then it must be the case that the `if` condition in Step 4 evaluated to false for some x. Further, x must be the left-end point of J_{i2}. This is because once the `if` condition evaluates to false, the algorithm jumps to Step 11 and creates a single price interval containing all copies that have not been priced yet and it includes the first copy and hence, this price interval must be J_{i1}. So x must be the left-end point of the price interval just after J_{i1}, i.e. J_{i2}.

Now, $\mathsf{width}_i(\pi_i(x)) = \mathsf{width}_i(\pi_i(J_{i2})) = \lfloor \frac{c_i^{\mathsf{invt}}(\pi_i(J_{i2}))}{B_i} \rfloor < 1$ implies that $\frac{c_i^{\mathsf{invt}}\pi_i(J_{i2}))}{B_i} < 1$ and so $c_i^{\mathsf{invt}}(\pi_i(J_{i2})) < B_i$. By definition of $c_i^{\mathsf{invt}}()$, this implies that $\min\{c_i^{\mathsf{inv}}(\pi_i(J_{i2}), \ell_i\} < B_i$. Since by Equation (2.9), $\ell_i \geq B_i$, it must be the case $c_i^{\mathsf{inv}}(\pi_i(J_{i2})) < B_i$, which by definition of $c_i^{\mathsf{inv}}()$ can occur only if $\pi_i(J_{i2}) < c_i(B_i)$. ■

We now prove Lemma 2.8.17 and Lemma 2.8.22 that treat the two conditions under which the algorithm can terminate.

Algorithm terminates through Step 11: The proof that price of J_{i2} is small follows almost immediately in this case.

Lemma 2.8.17. *If the algorithm terminated through Step 11 then* $\pi_i(J_{i2}) \cdot c_i^{\mathsf{invt}}(\pi_i(J_{i2})) < c_i(B_i)\, B_i$.

Proof. If the algorithm terminated through Step 11, then Lemma 2.8.16 implies that $\pi_i(J_{i2}) < c_i(B_i)$. By definition of $c_i^{\mathsf{invt}}()$, this implies that $c_i^{\mathsf{invt}}(\pi_i(J_{i2})) < B_i$ and hence we get the result. ■

Algorithm terminates through Step 5: We will prove that price of the interval J_{i2} is 'small' by showing that relative to the price of right-most interval J_{iz_i}, the prices for the subsequently created intervals on its left, have been slashed sufficiently often. For item i, label a copy x *close* if $\pi_i(x) < 3\, c_i(x)$, else label it as *far*. Depending on which of r and s are close or far, mark a price interval $J_{iq} = [r, s)$ as one of $\{(C, C), (F, C), (C, F), (F, F)\}$. Note that the right-most interval J_{iz_i} is not marked since it is semi-infinite. The following lemma indicates that in case prices are 'far' from the procurement cost, the algorithm slashes the prices exponentially.

Lemma 2.8.18. *If a contiguous sequence of price intervals* $J_{iq}, J_{i\,q+1}, \cdots, J_{i\,q+t-1}$ *are all marked* (F, F) *and* $J_{i\,q+t}$ *is marked* (F, C)*, then* $\pi_i(J_{iq}) \leq (\frac{2}{3})^t\, \pi_i(J_{i\,q+t})$.

Proof. If interval $J_{ip} = [r, s)$ is marked (F, F) which implies that $3\, c_i(s) < \pi_i(s) = \pi_i(J_{i\,p+1})$, then the pricing algorithm, by Step 6, sets

$$\pi_i(J_{ip}) = c_i(s) + \tfrac{\pi_i(s) - c_i(s)}{2} = \tfrac{\pi_i(s) + c_i(s)}{2} \leq \tfrac{\pi_i(s) + \frac{1}{3}\,\pi_i(s)}{2} = \tfrac{2}{3}\,\pi_i(s) = \tfrac{2}{3}\,\pi_i(J_{i\,p+1}).$$

Hence, $\pi_i(J_{iq}) \leq (\frac{2}{3})\, \pi_i(J_{i\,q+1}) \leq \cdots \leq (\frac{2}{3})^t\, \pi_i(J_{i\,q+t})$. ■

Lemma 2.8.19. *If the pricing algorithm terminated through Step 5, then for any* J_{iq}*, it is true that for all* $q' < q$, $|J_{iq'}| < \mathsf{width}_i(\pi_i(J_{iq}))$.

Proof. Consider any J_{iq}. By Lemma 2.8.3(b), we know that for $q' < q$, $\pi_i(J_{iq'}) \leq \pi_i(J_{i\,q'+1}) \leq \pi_i(J_{iq})$. Hence, for all price intervals $q' < q$ and $q' \neq 1$, by Lemma 2.8.3(c), $|J_{iq'}| = \mathsf{width}_i(\pi_i(J_{i\,q'+1})) \leq \mathsf{width}_i(\pi_i(J_{i\,q}))$ where the last inequality follows by Observation 2.8.4. Further, by Proposition 2.8.8(a), $|J_{i1}| \leq \mathsf{width}_i(\pi_i(J_{i2})) \leq \mathsf{width}_i(\pi_i(J_{i\,q}))$. This finishes the proof. ■

Lemma 2.8.20 states that if we ever have a price interval that is marked (F, C), there are 'many' price intervals to the left of that interval. Lemma 2.8.21 states that there are 'many' intervals to the left of the right-most interval J_{iz_i}.

Lemma 2.8.20. *Consider an interval $J_{iq} = [r, s)$ with $q \neq z_i$ that is marked (F, C). If the algorithm terminated through Step 5, then there are at least $B_i/4$ intervals $J_{iq'}$ with $q' < q$. In particular, J_{iq} cannot be the first price interval i.e. $q \neq 1$.*

Proof. Since s is close i.e. $\pi_i(s) < 3\, c_i(s)$, the algorithm, by Step 6, sets $\pi_i(J_{iq}) = c_i(s) + \frac{c_i(s)}{2} = \frac{3}{2}\, c_i(s)$. From the definition of r being marked far, $c_i(r) \leq \frac{1}{3}\, \pi_i(r) = \frac{1}{3}\, \frac{3}{2}\, c_i(s) = \frac{1}{2}\, c_i(s)$. Hence, across the interval J_{iq}, the cost function increases by at least $\frac{1}{2}\, c_i(s)$. Since $c_i(\cdot)$ is convex, the procurement cost should rise by at least $\frac{1}{2} c_i(s)$ starting from copy s onwards for every $|J_{iq}|$ copies. So, $c_i(r + 5 \cdot |J_{iq}|) = c_i(s + 4 \cdot |J_{iq}|) \leq c_i(s) + 4 \cdot \frac{1}{2}\, (c_i(s)) = 3 \cdot c_i(s)$ and hence

$$c_i^{\mathsf{invt}}(3\, c_i(s)) \leq c_i^{\mathsf{inv}}(3\, c_i(s)) < s + 4 \cdot |J_{iq}| = r + 5 \cdot |J_{iq}|. \tag{2.13}$$

By Lemma 2.8.3 (for $q \neq 1$) and Proposition 2.8.8(a) (for $q = 1$), we know that $|J_{iq}| \leq \mathsf{width}_i(\pi_i(s))$. Since $\pi_i(s) < 3\, c_i(s)$, $|J_{iq}| \leq \mathsf{width}_i(\pi_i(s)) \leq \mathsf{width}_i(3\, c_i(s)) \leq \frac{c_i^{\mathsf{invt}}(3\, c_i(s))}{B_i}$ where for the second inequality we have used Observation 2.8.4 which says $\mathsf{width}_i(p)$ is a non-decreasing function of p. Using (2.13), we get

$$|J_{iq}| \leq \tfrac{c_i^{\mathsf{invt}}(3\, c_i(s))}{B_i} \leq \tfrac{r + 5 \cdot |J_{iq}|}{B_i} \implies |J_{iq}| \leq \tfrac{r}{B_i - 5}$$

By Equation (2.9), $B_i \geq 12$ and therefore, $B_i - 5 \geq B_i/2$, hence the above equation implies that $|J_{iq}| \cdot \frac{B_i}{2} \leq r$. Since $|J_{iq}| \geq 1$ (any price interval contains at least one copy) and $B \geq 12$, hence $r \geq |J_{iq}| \cdot \frac{B_i}{2} \geq 6$. Therefore q cannot be 1, since for $q = 1$, we have $r = 1$ i.e. J_{i1}, by definition, is of the form $[1, s)$.

Now, since, $q \neq 1$, J_{iq} cannot contain the first copy i.e. $r \geq 2$ and hence $r - 1 \geq r/2$, and since we already have $|J_{iq}| \cdot \frac{B_i}{2} \leq r$, therefore we get,

$$|J_{iq}| \cdot \tfrac{B_i}{4} \leq r - 1 \tag{2.14}$$

Since the algorithm terminated in Step 5, by Lemma 2.8.19, for all $q' < q$, $|J_{iq'}| \leq \mathsf{width}_i(\pi_i(J_{iq}))$. Moreover, $|J_{iq}| = \mathsf{width}_i(\pi_i(J_{i\,q+1})) \geq \mathsf{width}_i(\pi_i(J_{iq}))$ where the equality follows from Lemma 2.8.3 and the inequality follows from Observation 2.8.4 and Lemma 2.8.3(b). Hence, we have that for all $q' < q$, $|J_{iq'}| \leq |J_{iq}|$. Since there are $r - 1$ copies to the left of J_{iq} and for all $q' < q$, $|J_{iq'}| \leq |J_{iq}|$, therefore, by (2.14), we get the desired result that there are at least $B_i/4$ price intervals $J_{iq'}$ with $q' < q$. ∎

Lemma 2.8.21. *If the algorithm terminated through Step 5, then there are at least $B_i/3$ intervals J_{iq} with $q < z_i$.*

Proof. Note that $\pi_i(J_{iz_i}) = Z$ and so $c_i^{\mathsf{invt}}(\pi_i(J_{iz_i})) = \ell_i$. Consequently, $\mathsf{width}_i(\pi_i(J_{iz_i})) \leq \frac{\ell_i}{B_i}$. Since the algorithm terminated through Step 5, hence by Lemma 2.8.19, for all $q' < z_i$, $|J_{iq'}| \leq \mathsf{width}_i(\pi_i(J_{iz_i})) \leq \frac{\ell_i}{B_i}$.

Since we have $\lfloor 2 \cdot \ell_i/3 \rfloor - 1$ copies to the left of J_{iz_i} (which due to Equation (2.9) is at least $\ell_i/3$), therefore, the number of intervals J_{iq} with $q < z_i$ is least $\frac{\ell_i/3}{\ell_i/B_i} = B_i/3$. ■

Lemma 2.8.22. *If the algorithm terminated through Step 5 then $\pi_i(J_{i2}) \cdot c_i^{\mathsf{invt}}(\pi_i(J_{i2})) < \frac{\epsilon Z}{2\,n}$.*

Proof. The interval J_{i1} can be marked either (F, C) or (F, F), since $c_i(1) = 0$ while $\pi_i(1) > 0$ (Observation 2.8.5). By Lemma 2.8.20, J_{i1} cannot be marked (F, C). Hence, the only case left is when J_{i1} is marked (F, F). Let q be the smallest value, if one exists, such that J_{iq} is marked (F, C); note that $q > B_i/4$ by Lemma 2.8.20, and in particular $q > 2$. If no such (F, C) interval exists, set $q \leftarrow z_i$.

By definition of J_{iq}, all intervals between J_{i1} and J_{iq} are marked (F, F). Depending on whether $q \neq z_i$ or $q = z_i$, Lemma 2.8.20 or Lemma 2.8.21 respectively imply there are at least $B_i/4$ of these intervals. By Lemma 2.8.18, $\pi_i(J_{i1}) \leq (\frac{2}{3})^{B_i/4}\, \pi_i(J_{iq}) \leq \frac{\pi_i(J_{iq})}{4n\ell_i/\epsilon}$, since $B_i = \lceil 12 \log(4n\ell_i/\epsilon) \rceil$.

Moreover, by Lemma 2.8.3(b), $\pi_i(J_{i2}) \leq 2 \cdot \pi_i(J_{i1}) \leq \frac{2\pi_i(J_{iq})}{4n\ell_i/\epsilon}$. By definition of $c^{\mathsf{invt}}()$, $c^{\mathsf{invt}}(\pi_i(J_{i2})) \leq \ell_i$; this gives $\pi_i(J_{i2}) \cdot c^{\mathsf{invt}}(\pi_i(J_{i2})) \leq \frac{2\pi_i(J_{iq})}{4n\ell_i/\epsilon} \cdot \ell_i \leq \frac{\epsilon Z}{2n}$. ■

The smoothing algorithm can give purely multiplicative guarantees as long as the cost of the first $O(\log n)$ copies of the items is small compared to $W(\mathrm{opt})$. As an example, suppose the cost functions are $c_i(k) = 0$ for $k \leq d \log n$, and $c_i(k) = \infty$ for $k > d \log n$ for some constant d. Then $\ell_i \leq d \log n$, and $B_i = O(\log n/\epsilon)$. So for d large enough constant, $B_i \cdot c(B_i) = 0$, and we get an $O(\log n)$ approximation to the social welfare, as in Bartal et al. [2003]. (This is best possible for online algorithms [Awerbuch et al., 1993].)

Polynomial procurement curves

For the case of polynomial procurement curves of the form[5] $c_i(x) = (x-1)^d$, we show that the smoothing algorithm gives approximation guarantees close to that of pricing at twice the index; refer Theorem 2.8.25 and the approximation guarantees given by twice-the-index algorithm on polynomial curves in Section 2.6.

In the analysis below we make a few assumptions. First, we assume that $m \geq c_i^{\mathsf{inv}}(Z)$ for all items i. This case interests us since it is here that the number of copies of an item that are available are less than the number of buyers. In this scenario, for all $p \leq Z$, $c_i^{\mathsf{invt}}(p) = c_i^{\mathsf{inv}}(p)$ where recall that Z is the parameter supplied to the smoothing algorithm that satisfies $Z \in (U_{max}, U_{max}/\epsilon]$. Second, we assume that

$$B_i \geq 18\,(2d+1) \text{ and } \ell_i \geq 2(B_i+1) \tag{2.15}$$

These requirements on ℓ_i and B_i subsume the ones mentioned in Equation 2.9.

The crucial result which will help us prove the improved bound is Lemma 2.8.23.

Lemma 2.8.23. *For all copies x in the range $[B_i, \lfloor(2/3)\ell_i\rfloor - (2d+1)\cdot\lfloor\ell_i/B_i\rfloor]$, $\frac{3}{2}\cdot c_i(x) < \pi_i(x) < 3\cdot c_i(x)$.*

The result says that apart possibly from a few copies on the left and right ends, the price is close to the procurement cost for all copies. We now show how such a result helps in proving in the improved bound. In preparation for applying the structural lemma in Theorem 2.8.25, the following result gives us the per-item profit equation.

Lemma 2.8.24. *For every item i, we have $c_i^{\mathsf{inv}}(P_i^f)\cdot P_i^f \leq 18\,(d+1)\,(27/16)^{d+1}\,\mathsf{profit}_i + 18\,c_i(B_i)\cdot B_i$.*

Proof. We consider three cases based on the number of copies x_i of item i that were sold by the smoothing algorithm alg.

- The algorithm sold at most B_i copies of item i.

  By Lemma 2.8.23, the price of B_i^{th} copy is at most $3\cdot c_i(B_i) = 3\,(B_i-1)^d$. Since algorithm sold less than B_i copies of item i, and by Lemma 2.8.3(b), prices are non-decreasing from left to right, hence, $P_i^f \leq 3\,(B_i-1)^d$ and therefore, $c_i^{\mathsf{inv}}(P_i^f) \leq 3^{1/d}(B_i-1)+1$. We have $P_i^f \cdot c_i^{\mathsf{inv}}(P_i^f) \leq 3\,(B_i-1)^d\,(3^{1/d}(B_i-1)+1)$.

[5] In case the reader is curious on why we choose the polynomial cost curve to be $(x-1)^d$ instead of the more natural choice of x^d, we recall that the smoothing algorithm analysis assumed that $c_i(1) = 0$ and hence we made the choice of $(x-1)^d$.

– The algorithm sold at least B_i copies and less than $\lfloor (2/3)\ell_i \rfloor - (2d+1) \cdot \lfloor \ell_i / B_i \rfloor$ copies.

  Let x_i be the last copy sold. We have $P_i^f = \pi_i(x_i+1) < 3 \cdot c_i(x_i+1) = 3 \cdot x_i^d$ where the inequality follows from Lemma 2.8.23. Hence, $c_i^{\mathsf{inv}}(P_i^f) \leq 3^{1/d} \cdot x_i + 1$. We have $P_i^f \cdot c_i^{\mathsf{inv}}(P_i^f) < (3^{1/d} \cdot x_i + 1) \cdot 3 \cdot (x_i)^d < 6 \cdot 3^{1/d} \cdot x_i^{d+1}$.

  For every copy x uptil x_i, alg earned profit at least $c_i(x)/2$. From Lemma 2.8.23, $\pi_i(x) - c_i(x) > c_i(x)/2$. Therefore, the profit profit_i earned by alg from the sales of item i is at least $\sum_{k=1}^{x_i} \frac{c_i(k)}{2} = \sum_{k=1}^{x_i} \frac{1}{2}(k-1)^d \geq \int_1^{x_i} \frac{1}{2}(k-1)^d \mathrm{d}k \geq \frac{1}{2(d+1)}(x_i - 1)^{d+1}$. Since $x_i \geq B_i$ and by Equation 2.15, $B_i \geq 18(2d+1) \geq 54$, therefore, $(x_i - 1) \geq (53/54) \cdot x_i$. Hence, we have $c_i^{\mathsf{inv}}(P_i^f) \cdot P_i^f < 2(d+1)(\frac{54}{53})^{d+1} \cdot 6 \cdot 3^{1/d} \cdot \mathsf{profit}_i$.

– The algorithm sold at least $\lfloor (2/3)\ell_i \rfloor - (2d+1) \cdot \lfloor \ell_i / B_i \rfloor$ copies.

  First we note that $P_i^f \cdot c_i^{\mathsf{inv}}(P_i^f) \leq \ell_i^d \cdot \ell_i = \ell_i^{d+1}$. Now following the same argument as in previous case, we know that the profit profit_i is at least $\sum_{k=1}^{x_i} \frac{1}{2} c_i(x) \geq \frac{1}{2(d+1)}(x_i - 1)^{d+1}$ where $x_i \geq \lfloor (2/3)\ell_i \rfloor - (2d+1) \cdot \lfloor \ell_i / B_i \rfloor$. Since by Equation 2.15, $B_i \geq 18(2d+1)$, hence, $\lfloor (2/3)\ell_i \rfloor - (2d+1) \cdot \lfloor \ell_i / B_i \rfloor \geq (2/3)\ell_i - 1 - (2d+1) \cdot \ell_i / B_i \geq (2/3)\ell_i - \ell_i/18 - 1 = (11/18)\ell_i - 1$.

  Hence the profit profit_i is at least $\frac{1}{2(d+1)}((11/18)\ell_i - 1 - 1)^{d+1} \geq \frac{1}{2(d+1)}(16/27)^{d+1}\ell_i^{d+1}$ where we have used that $((11/18)\ell_i - 2 \geq (16/27)\ell_i$ since by Equation 2.15, $\ell_i \geq 2(B_i+1) \geq 2 \cdot (18(2d+1)+1) \geq 110$. Hence $c_i^{\mathsf{inv}}(P_i^f) \cdot P_i^f \leq 2(d+1)(27/16)^{d+1} \cdot \mathsf{profit}_i$.

In all three cases, $P_i^f \cdot c_i^{\mathsf{inv}}(P_i^f) \leq 18\,(d+1)\,(27/16)^{d+1}\mathsf{profit}_i + 3\,(B_i - 1)^d\,(3^{1/d}(B_i - 1) + 1)$. Now note that $c_i(B_i) = (B_i - 1)^d$ and $3\,(3^{1/d}(B_i - 1) + 1) \leq 18\,B_i$. Hence we have the desired result. ■

We now present the main result of this section.

Theorem 2.8.25. *The social welfare $W(\mathsf{alg})$ achieved by the smoothing algorithm on the polynomial curve $c_i(x) = (x-1)^d$ satisfies*

$$W(\mathsf{alg}) \geq \frac{W(\mathrm{opt}) - 18 \sum_{i \in U} B_i \cdot c_i(B_i)}{18\,(d+1)\,(27/16)^{d+1}}$$

where we assume $B_i \geq 18(2d+1)$, $\ell_i \geq 2(B_i+1)$ and that for all items i, the number of buyers m, exceeds $c_i^{\mathsf{inv}}(Z)$.

Theorem 2.8.25 roughly states that social welfare attained by the smoothing algorithm on polynomial procurement curve $(x-1)^d$ is a constant approximation to the (optimal social welfare minus the procurement cost of the first d many copies of each item).

Proof of Theorem 2.8.25 : Since for all items i, $m \geq c_i^{\mathsf{inv}}(Z)$, hence for all $p \leq Z$, $c_i^{\mathsf{invt}}(p) = c_i^{\mathsf{inv}}(p)$. Hence, the result of Lemma 2.8.24 along with Lemma 2.4.3 proves that the price curve for item i is $(18\,(d+1)\,(27/16)^{d+1}, 18\,c_i(B_i) \cdot B_i)$-single-resource-good. Using Lemma 2.4.2, we have the desired result. ■

We would now like to prove Lemma 2.8.23. In preparation for that, we shall next prove a few lemmas. The following lemma states for a price interval $J_{iq} = [s,t)$, if at copy t, price is close to the procurement cost, then so it is at copy s. This implies in particular that the price is close to procurement curve for all copies in the price interval J_{iq}.

Lemma 2.8.26. *For the polynomial procurement curve $(x-1)^d$ ($d \geq 1$), consider a price interval $J_{iq} = [s,t)$ created by the smoothing algorithm such that $t \geq B_i$. If $\pi_i(t) < 3 \cdot c_i(t)$, then $\pi_i(s) < 3 \cdot c_i(s)$. In particular, for all $x \in [s,t)$, $\pi_i(x) < 3 \cdot c_i(x)$.*

Proof. First note that since $t \geq B_i$, therefore, $c_i(t) \geq c_i(B_i)$. Also, from Lemma 2.8.3, we know that $\pi_i(t) \geq \frac{3}{2}\, c_i(B_i)$. Hence, $c_i^{\mathsf{inv}}(\pi_i(t)) \geq B_i$, and since $c_i^{\mathsf{inv}}() = c_i^{\mathsf{invt}}()$, therefore, it implies that $\mathsf{width}_i(\pi_i(t)) \geq 1$. The width and price of J_{iq} shall therefore be decided by Steps 5- 7.

Since $\pi_i(t) < 3 \cdot c_i(t)$, therefore, $\pi_i(J_{iq}) = \frac{3}{2} c_i(t)$. Further, since $\pi_i(t) < 3 \cdot c_i(t)$, hence $\mathsf{width}_i(\pi_i(t)) = \lfloor c_i^{\mathsf{invt}}(\pi_i(t))/B_i \rfloor \leq \lfloor (3^{1/d} \cdot (t-1)+1)/B_i \rfloor$ which implies that $s \geq t - 3^{1/d} \cdot (t-1)/B_i - 1$.

Moreover, since $\pi_i(J_{iq}) = \frac{3}{2}\, c_i(t)$, hence the condition $\pi_i(s) < 3 \cdot c_i(s)$ is equivalent to $\frac{3}{2} c_i(t) < 3 \cdot c_i(s)$ or $c_i(s) > c_i(t)/2$. Since $c_i(x) = (x-1)^d$, therefore, we require $(s-1) > (t-1)/2^{1/d}$. Since $s \geq t - 3^{1/d} \cdot (t-1)/B_i - 1$, it suffices to have $(t-1) - 3^{1/d} \cdot (t-1)/B_i - 1 > (t-1)/2^{1/d}$ which for $t \geq B_i$, is equivalent to demanding $B_i \geq 3^{1/d}/(1 - 1/(t-1) - 1/2^{1/d})$. By Equation 2.15, we have $t \geq B_i \geq 18(2d+1)$ and hence the inequality is satisfied.

Since for all $x \in [s,t)$, $\pi_i(x) = \pi_i(s)$ and $c_i(x) \geq c_i(s)$, hence, $\pi_i(s) < 3 \cdot c_i(s)$ implies that $\forall x \in [s,t)$, $\pi_i(x) < 3 \cdot c_i(x)$. ■

Corollary 2.8.27 states that in case there is a copy in the range $[B_i, \lfloor (2/3)\, \ell_i \rfloor]$ in the range that is the left end point of a price interval and has its price close to its cost, then for all copies from B_i uptil that copy, the price curve is close to the procurement curve.

Corollary 2.8.27. *If at point x such that $x \in [B_i, \lfloor (2/3)\,\ell_i \rfloor]$, $\pi_i(x) < 3 \cdot c_i(x)$ and x is the left end point of a price interval, then for all copies x' in the range $[B_i, x]$ (i.e. for all copies to the left of x and to the right of B_i), $\pi_i(x') < 3 \cdot c_i(x')$.*

Proof. Consider a point x such that $\lfloor (2/3)\,\ell_i \rfloor > x > B_i$ and $\pi_i(x) < 3 \cdot c_i(x)$. Say x is the left end point of the price interval J_{iq}. Let $J_{i\,q-1} = [r, x)$. By Lemma 2.8.26, for all copies $y \in J_{i\,q-1}$, $\pi_i(y) < 3\,c_i(y)$. If the left end point r of $J_{i\,q-1}$ is such that $r \leq B_i$, then we have completed the proof of our claim.

Else if the left end point r of $J_{i\,q-1}$ is such that $r > B_i$, we can repeat the argument above since we have $\pi_i(r) < 3\,c_i(r)$ and r is the left end point of $J_{i\,q-1}$ and therefore inductively, we have proved the claim. ■

Corollary 2.8.27 is sufficient to prove Lemma 2.8.23 in case we can show that a copy which is the left end point of a price interval and has its price close to cost exists in the appropriate range. The following lemma proves the existence of such a copy.

Lemma 2.8.28. *For at least one price interval J_{ip} to the right of the point $\lfloor (2/3)\ell_i \rfloor - (2d+1) \cdot \lfloor \ell_i / B_i \rfloor$, it is the case that the left end point of J_{ip} is marked close.*

Proof. Denote the point $\lfloor (2/3)\ell_i \rfloor - (2d+1) \cdot \lfloor \ell_i/B_i \rfloor$ by w_1 and the point $\lfloor (2/3)\ell_i \rfloor - 2d \cdot \lfloor \ell_i/B_i \rfloor$ by w_2. Let I be the interval $[w_1, w_2]$.

We prove the claim by contradiction; assume that for all price intervals J_{ip} to the right of w_1, the left end point of J_{ip} is marked far. Since the width of any price interval is at most $\lfloor \ell_i/B_i \rfloor$ and $|I| = \lfloor \ell_i/B_i \rfloor$, hence there must be a price interval whose left end point, say τ, lies in the interval I.

In order to the contradict the assumption, we need to prove that $3\,c_i(\tau) > \pi_i(\tau)$. For this it suffices to show that $3 \cdot c_i(w_1) > \pi_i(w_2)$. This is because for any $x \in [w_1, w_2]$, since procurement curve $c_i()$ is non-decreasing, therefore, $c_i(x) \geq c_i(w_1)$; also by Lemma 2.8.3(b), $\pi_i(x) \leq \pi_i(w_2)$. Therefore, $3 \cdot c_i(w_1) > \pi_i(w_2)$ implies that for any x in $[w_1, w_2]$, $3 \cdot c_i(x) \geq 3 \cdot c_i(w_1) > \pi_i(w_2) > \pi_i(x)$ and hence in particular $3\,c_i(\tau) > \pi_i(\tau)$.

We now prove that $3 \cdot c_i(w_1) > \pi_i(w_2)$. We have

- $c_i(w_1) \geq (\ell_i/2 - (2d+1) \cdot \ell_i/B_i)^d$

  This is because $c_i(w_1) = (\lfloor (2/3)\ell_i \rfloor - (2d+1) \cdot \lfloor \ell_i/B_i \rfloor - 1)^d \geq ((2/3)\ell_i - 1 - (2d+1) \cdot \ell_i/B_i - 1)^d \geq (\ell_i/2 - (2d+1) \cdot \ell_i/B_i)^d$ where we use $(2/3)\ell_i - 2 \geq \ell_i/2$ since by Equation 2.15, $\ell_i \geq 2(B_i + 1) \geq 2\,(18(2d+1)+1) \geq 110$.

– $\pi_i(w_2) = \pi_i(J_{iq}) \leq (\frac{2}{3})^{2d} \cdot \ell_i^d$.

To see this, let w_2 lie in price interval of J_{iq}. Note that J_{iq} cannot be the rightmost price interval J_{iz_i} since the left end-point of J_{iz_i} is $\lfloor (2/3)\,\ell_i \rfloor$ and $w_2 < \lfloor (2/3)\ell_i \rfloor$. As all price intervals $J_{iq'}$ to the right of J_{iq} have their left end point marked far, hence, in other words, all price intervals between J_{iq} and J_{iz_i} are marked (F, F). Since a price interval has size at most $\lfloor \ell_i/B_i \rfloor$, therefore, there are at least $2d$ price intervals between J_{iq} and J_{iz_i}. By Lemma 2.8.18, $\pi_i(w_2) = \pi_i(J_{iq}) \leq (\frac{2}{3})^{2d} \cdot \ell_i^d$.

To prove that $3 \cdot c_i(w_1) > \pi_i(w_2)$, it suffices to have $(\frac{2}{3})^{2d} \cdot \ell_i^d < 3 \cdot (\ell_i/2 - (2d + 1) \cdot \ell_i/B_i)^d$ which is equivalent to demanding $(\frac{2}{3})^2 \cdot \frac{1}{3^{1/d}} < \frac{1}{2} - (2d + 1) \cdot \frac{1}{B_i}$ or $B_i > (2d+1)/(\frac{1}{2} - (\frac{2}{3})^2 \cdot \frac{1}{3^{1/d}})$; by Equation 2.15, $B_i > 18(2d+1)$, and hence we have satisfied the desired inequality. ■

We now prove Lemma 2.8.23 which recall roughly states that apart from the 'few' left-most copies and right-most copies, the price curve is close to the procurement curve for all copies.

Proof of Lemma 2.8.23 : From Lemma 2.8.3(a), we can infer one side of the inequality i.e. for all x in the desired range, $\pi_i(x) \geq 3\,c_i(x)/2$. Now for the other side of the inequality i.e. $\pi_i(x) < 3c_i(x)$.

Denote the point $\lfloor (2/3)\ell_i \rfloor - (2d+1) \cdot \lfloor \ell_i/B_i \rfloor$ by w. First we note that w lies to the right of B_i i.e. $w \geq B_i$. In order to see this, it suffices to show $(2/3)\,\ell_i - 1 - (2d+1) \cdot \ell_i/B_i \geq B_i$. By Equation 2.15, we have $B_i \geq 18(2d + 1)$, and hence $(2/3)\ell_i - 1 - (2d + 1) \cdot \ell_i/B_i \geq (2/3)\ell_i - 1 - \ell_i/18 = (11/18)\ell_i - 1$. And hence it suffices to have $\ell_i \geq (18/11) \cdot (B_i + 1)$ which by Equation 2.15 is true.

From Lemma 2.8.28, we know that there is at least one price interval J_{ip} to the right of w whose left end point say τ is marked close. Note that $\tau \geq w \geq B_i$. Further since τ is the *left end point* of a price interval, therefore, $\tau \leq \lfloor (2/3)\ell_i \rfloor$. Hence, we have $B_i \leq \tau \leq \lfloor (2/3)\,\ell_i \rfloor$. Thus, by Corollary 2.8.27, we get the desired result. ■

2.9 Profit Maximization

In this section, we present a result that shows how any *social welfare maximizing item pricing scheme for a stream of buyers* can be combined with a *single-buyer profit maximizing item pricing scheme* to yield a *profit maximizing scheme for a stream of buyers*. This result helps us in converting the item-pricing social welfare maximizing schemes from previous

sections into item-pricing profit maximizing schemes. Specifically, suppose we are given access to two algorithms:

1. a deterministic item pricing (α, β)-welfare approximate algorithm A, that is, given cost curves $\{c_i\}_{i \in U}$, A outputs pricing schemes $\{\pi_i(\cdot)\}_{i \in U}$ such that on any sequence σ of buyers,
$$\rho \cdot W(A(\sigma)) + \beta \geq \text{opt}(\sigma) \tag{2.16}$$

2. and, a randomized single-buyer profit maximization algorithm B, which outputs a non-negative price vector $\vec{\pi}$ for items $i \in U$ and gives the guarantee that for any buyer b, with valuation $v_b()$,
$$\mu \cdot \mathbb{E}_\tau[\sum_{i \in S_{\vec{\pi}}} \vec{\pi}_i] + \kappa \geq \max_{s \subseteq U} v_b(s) \tag{2.17}$$
where $S_{\vec{\pi}}$ is the set of items bought by the buyer b when the price vector $\vec{\pi}$ is presented and so $\sum_{i \in S_{\vec{\pi}}} \vec{\pi}_i$ is the resultant profit generated from buyer b.

Note that $\max_{s \subseteq U} v_b(s)$ is an upper bound on maximum profit that can be generated from buyer b. Further algorithm B operates in a world with zero procurement costs, and may take as input a parameter T such that $\max_{S \subseteq U} v_b(S) < T$; the parameters μ, κ may be functions of T.[6] We further assume that for every item $i \in U$ and $k \in \mathbb{N}$, the price $\pi_i(k)$ set by algorithm A is at least as much as the procurement cost of that copy of the item. The main result of this section is

Theorem 2.9.1. *Given a (ρ, β)-social welfare maximization algorithm A (satisfying Equation (2.16)) and a (μ, κ)-single buyer profit maximization algorithm B (satisfying Equation (2.17)), we can construct a randomized profit-maximizing algorithm C whose expected profit over any sequence σ of buyers is at least* $\left(\text{opt}(\sigma) - O(\beta + \kappa \cdot |\sigma|)\right) / O(\rho + \mu)$.

We now construct an algorithm C which uses A and B, and gives expected profit of approximately $\frac{W(\text{opt}(\sigma))}{(2\,\rho + 8\,\mu)}$ (with some additive loss) over any sequence σ of buyers. The construction of algorithm C and subsequent analysis heavily borrows ideas from a similar result proved in Awerbuch et al. [2003]. Our result can be seen as extension of their result to situations with production costs and arbitrary valuations.

[6] Balcan et al. [2008] give such a single-buyer profit maximization algorithm, a slight variant of which has $\kappa = T/(2\,m\,n)$ and $\mu = O(\log(m\,n))$. The algorithm picks a uniform price on a geometric scale for all items. It can be combined with either of the social welfare maximizing algorithms in this chapter to give a $O(\log(m\,n))$-profit maximizing algorithm+additive loss.

The Algorithm C: In case algorithm B requires an estimate T, such that $\max_{S \subseteq U} v_b(S) < T$, algorithm C takes as input a parameter T such that $U_{max} < T$, where U_{max} is defined as in (2.5). The parameter T is used each time algorithm B is invoked by C.

On a sequence σ of buyers, for buyer j, let x_i^j denote the number of copies of item i already sold when buyer j walks in.

1. With probability 1/2, set $t_j = \mathbf{0}$ and with probability 1/2, generate a random price vector $\vec{\pi}$ using algorithm B and set $t_j = \tau$.
2. Let the price of each item i be $\omega_i(x_i^j + 1) + t_j(i)$.

Simply put, algorithm C maintain a copy of algorithm A running in the background and keeps updating A's state with the sets buyers are buying. When buyer j walks in, with probability 1/2, algorithm C presents the price vector as specified by the current state of A (determined by the number of various items sold up till then), and with probability 1/2, adds a random price vector, generated using B, to the price vector specified by A.

We now present an analysis of profit generated by algorithm C with the final result mentioned in Theorem 2.9.1.

Analysis: For any allocation $\eta : B \to 2^U$ (where $\eta(j)$ denotes the set of items allocated to buyer j), $\sum_{k<j:i\in\eta_k} 1$ denotes the number of copies of item i allocated to buyer $k < j$ under η. Hence, $\chi_\eta^j(i) = c_i(1+\sum_{k<j:i\in\eta(k)} 1)$ denotes the cost of allocating a copy of item i to buyer j, given that the buyers previous to her have received their allocations under η. The cost of allocating $\eta(j)$ to buyer j is therefore $\sum_{i\in\eta(j)} \chi_\eta^j(i)$ and the social welfare achieved by allocating $\eta(j)$ to buyer j is $v_j(\eta(j)) - \sum_{i\in\eta(j)} \chi_\eta^j(i)$.

Let $\hat{s}_j = \mathrm{opt}(j)$ be the set allocated to buyer j under the optimal allocation, opt, when she is part of sequence σ. Denote by $\hat{\gamma}_j$ the social welfare achieved by allocating $\hat{s}_j$ to j, which is equal to $v_j(\hat{s}_j) - \sum_{i\in\hat{s}_j} \chi_{\mathrm{opt}}^j(i)$.

Consider a particular run of algorithm C and for $r < s$, let $t_{r:s}$ denote the set of random choices t_j made for buyers $j \in \{r, r+1, \cdots, s\}$. Let s_j, a random variable determined completely by $t_{1:j-1}$, be the set which buyer j would have bought if t_j were chosen to be $\mathbf{0}$, and let γ_j be the social welfare achieved by allocating s_j to j. Note that s_j *may not* be the set actually bought by buyer j, depending on whether or not t_j is zero.

Let p_j, a random variable determined completely by $t_{1:j}$, be the profit made by C from buyer j. Therefore, the total profit made by algorithm C is $\sum_{j\in Q} p_j$ where Q is the set of buyers.

We partition the set of buyers Q into two sets:

1. Let Q_1 be the set of buyers for whom $\gamma_j \geq \frac{1}{2}\,\hat{\gamma}_j$.

2. Let Q_2 be the set of buyer for whom $\gamma_j < \frac{1}{2}\,\hat{\gamma}_j$.

The partition of Q into Q_1 and Q_2 is determined by the set of random choices $t_{1:m}$. The following observation follows from the definition of $\hat{\gamma}_j$.

Observation 2.9.2. *The optimal profit is at most the optimal social welfare and that is* $\sum_{j \in Q} \hat{\gamma}_j$.

We now state and prove Lemma 2.9.3 and Lemma 2.9.4 which bound the welfare made by the optimal allocation for the buyer sets Q_1 and Q_2 respectively.

Note that for any j, the offsets $t_{1:j-1}$ completely determine the bundles bought by buyers 1 through $j-1$. Given $t_{1:j-1}$, for convenience let us define $\pi_j(s) = \sum_{i \in s} \omega_i(x_i^j + 1)$. $\pi_j(s)$ is equal to the price that would be offered to buyer j for set s, in case the offset t_j were chosen to be zero.

Lemma 2.9.3 (Low welfare buyers)**.** *For any set of values* $t_{1:m}$, $\sum_{j \in Q_2} \hat{\gamma}_j \leq 2\,\rho \sum_{j \in Q_1 \cup Q_2} p_j + 2\,\beta$.

Proof. Consider any set of values $t_{1:m}$. Consider a buyer j in Q_2. s_j is defined to be the utility-maximizing set if t_j were to be 0, therefore, $v_j(s_j) - \pi_j(s_j) \geq v_j(\hat{s}_j) - \pi_j(\hat{s}_j)$. Now, $\gamma_j \geq v_j(s_j) - \pi_j(s_j)$ and $\hat{\gamma}_j \leq v_j(\hat{s}_j)$. Since buyer j is in Q_2, hence $\gamma_j < \frac{1}{2}\,\hat{\gamma}_j$ and therefore we get

$$v_j(\hat{s}_j) - \pi_j(\hat{s}_j) \leq v_j(s_j) - \pi_j(s_j) \leq \gamma_j < \frac{1}{2}\,\hat{\gamma}_j \tag{2.18}$$

which implies that

$$\pi_j(\hat{s}_j) > v_j(\hat{s}_j) - \frac{1}{2}\,\hat{\gamma}_j \tag{2.19}$$

Let s'_j be the actual set bought by buyer j from algorithm C.

Now consider the sequence σ' composed of buyers j' defined as follows

- for every buyer j in Q_2, we introduce a buyer j' who has non-zero valuation for exactly two sets – she values set s'_j at $\pi_j(s'_j)$ and set $\hat{s}_j$ at $v_j(\hat{s}_j) - \frac{1}{2}\,\hat{\gamma}_j$, and

- for every buyer j in Q_1, we introduce buyer j', such that she is single-minded and has valuation $\pi_j(s'_j)$ for set s'_j.

The sequence of buyers in σ' is the natural ordering i.e. $m' < n'$ if and only if $m < n$. It is not difficult to verify that when algorithm A is run on sequence σ',

1. for all $j \in Q_1$, buyer j' shall buy the set s'_j from A,
2. for all $j \in Q_2$, buyer j'_1 shall buy the set s'_j from A (and not the set $\hat{s}_j$ by (2.19))

Consider the allocation η for sequence σ', wherein for every $j \in Q_2$, j' is allocated set $\hat{s}_j$ and rest of the buyers are allocated nothing. The social welfare achieved by η is $\sum_{j \in Q_2} \left(v_j(\hat{s}_j) - \frac{1}{2}\hat{\gamma}_j - \sum_{i \in \hat{s}_j} \chi^j_\eta(i) \right)$ where for each $j \in Q_2$, $v_j(\hat{s}_j) - \frac{1}{2}\hat{\gamma}_j$ is the value of buyer j' for set $\hat{s}_j$ and $\sum_{i \in \hat{s}_j} \chi^j_\eta(i)$ is the cost of allocating that set. Now observe that for each j, $\sum_{i \in \hat{s}_j} \chi^j_\eta(i) \leq \sum_{i \in \hat{s}_j} \chi^j_{\mathrm{opt}}(i)$ i.e. the cost of allocating the set $\hat{s}_j$ to buyer j' under η on sequence σ' is at most the cost of allocating that set to the buyer j under opt on sequence σ. This is because for any prefix of buyers, η allocates only at most as many copies of any item as opt for that prefix. Therefore, for each $j \in Q_2$, $v_j(\hat{s}_j) - \sum_{i \in \hat{s}_j} \chi^j_\eta(i) \geq v_j(\hat{s}_j) - \sum_{i \in \hat{s}_j} \chi^j_{\mathrm{opt}}(i) = \hat{\gamma}_j$, and therefore, $v_j(\hat{s}_j) - \frac{1}{2}\hat{\gamma}_j - \sum_{i \in \hat{s}_j} \chi^j_\eta(i) \geq \frac{1}{2}\hat{\gamma}_j$. Hence the allocation η achieves a social welfare of at least $\sum_{j \in Q_2} \frac{1}{2}\hat{\gamma}_j$

The (ρ, β)-approximation guarantee of A should hold on σ' as well and therefore using (2.16), and the fact optimal welfare on σ' is at least as much the welfare made through allocation η we have, $\sum_{j \in Q_2} \frac{1}{2}\hat{\gamma}_j \leq \rho \cdot \sum_{j \in Q_1 \cup Q_2} (\pi_j(s'_j) - c_j(s'_j)) + \beta$.

However, the profit p_j made by C on sequence σ is $\pi_j(s'_j) - c_j(s'_j) + \sum_{i \in s'_j} t_j(i)$ and therefore in particular, $p_j \geq \pi_j(s'_j) - c_j(s'_j)$. Hence, we get the desired claim i.e. $\sum_{j \in Q_2} \hat{\gamma}_j \leq 2\,\rho \sum_{j \in Q_1 \cup Q_2} p_j + 2\,\beta$. ■

Lemma 2.9.4 (High welfare buyers)**.** $\mathop{\mathbb{E}}_{t_{1:m}} \left[\sum_{j \in Q_1} \hat{\gamma}_j\right] \leq 8\,\mu \mathop{\mathbb{E}}_{t_{1:m}} \left[\sum_{j \in Q_1} p_j\right] + 4\,\kappa \mathop{\mathbb{E}}_{t_{1:m}} [|Q_1|]$.

Proof. For a buyer j in Q_1, we know that $\gamma_j = (v_j(s_j) - c_j(s_j)) \geq \frac{1}{2}\hat{\gamma}_j$.

- Either, $(\pi_j(s_j) - c_j(s_j)) \geq \frac{1}{2}\gamma_j \geq \frac{1}{4}\hat{\gamma}_j$: With probability $1/2$, we choose $t_j = \mathbf{0}$, and by definition of s_j, we know that buyer j would buy set s_j and therefore the profit from buyer j, $p_j = (\pi_j(s_j) - c_j(s_j)) \geq \frac{1}{2}\gamma_j \geq \frac{1}{4}\hat{\gamma}_j$.
- Or, $(\pi_j(s_j) - c_j(s_j)) < \frac{1}{2}\gamma_j$. In this case, $v_j(s_j) - \pi_j(s_j) \geq \frac{1}{2}\gamma_j$ because $(v_j(s_j) - \pi_j(s_j)) + (\pi_j(s_j) - c_j(s_j)) = \gamma_j$.

  With probability $1/2$, we set t_j to be a random vector $\vec{\pi}$ generated using algorithm B. Consider a setting with zero production cost and a buyer b whose valuation $v_b()$

is given as $\forall s \subseteq U, v_b(s) = v_j(s) - \pi_j(s)$. For any $\vec{\pi}$ and for any set s, buyer j and buyer b have the same utility as we can see in the following equation:

$$v_b(s) - \sum_{i \in s} \vec{\pi}_i = v_j(s) - \pi_j(s) - \sum_{i \in s} \vec{\pi}_i$$

Hence on being presented with price vector $\vec{\pi}$, buyer b shall the buy the same set as buyer j, call the set $S_{\vec{\pi}}$. Therefore, the expected value of $\sum_{i \in S_{\vec{\pi}}} \vec{\pi}_i$ is equal to the expected profit made from buyer b which by (2.17) is

$$\frac{\max_{s \subseteq U} v_b(s) - \kappa}{\mu}.$$

Since we are in the case where $(\pi_j(s_j) - c_j(s_j)) < \frac{1}{2}\gamma_j$, therefore $\max_{s \subseteq U} v_b(s) = v_j(s_j) - \pi_j(s_j) \geq \frac{1}{2}\gamma_j$. Since for buyer j, we choose a random offset $\vec{\pi}$ with probability $1/2$, therefore, the expected profit from buyer j is at least

$$\frac{\max_{s \subseteq U} v_b(s) - \kappa}{2\,\mu} \geq \frac{\frac{1}{2}\gamma_j - \kappa}{2\,\mu}.$$

Therefore, taking both of the above cases into account, for any buyer $j \in Q_1$,

$$\mathop{\mathbb{E}}_{t_j | t_{1:j-1}}[p_j] \geq \frac{\frac{1}{2}\gamma_j - \kappa}{2\,\mu} \geq \frac{\frac{1}{4}\hat{\gamma}_j - \kappa}{2\,\mu}.$$

Taking expectation over $t_{1:j}$, we get

$$\mathop{\mathbb{E}}_{t_{1:j}}[p_j \cdot \mathbb{I}[j \in Q_1]] \geq \frac{\mathop{\mathbb{E}}_{t_{1:j}}[(\frac{1}{4}\hat{\gamma}_j - \kappa) \cdot \mathbb{I}[j \in Q_1]]}{2\,\mu}$$

where $\mathbb{I}[\cdot]$ is the indicator function. The expectation can be extended to be over $t_{1:m}$ since $t_{1:j}$ completely determine p_j and $\mathbb{I}[j \in Q_1]$. Therefore,

$$\mathop{\mathbb{E}}_{t_{1:m}}[p_j \cdot \mathbb{I}[j \in Q_1]] \geq \frac{\mathop{\mathbb{E}}_{t_{1:m}}[(\frac{1}{4}\hat{\gamma}_j - \kappa) \cdot \mathbb{I}[j \in Q_1]]}{2\,\mu}$$

Using Linearity of expectation, we get the desired result. ■

We now state the main theorem which as we later show is equivalent to Theorem 2.9.1 and thereby prove the claimed profit guarantee.

Theorem 2.9.5 (Profit Guarantee)**.**

$$\sum_{j \in Q} \hat{\gamma}_j \leq (2\,\rho + 8\,\mu)\,\mathbb{E}[\sum_{j \in Q} p_j] + 4\,\kappa\,|Q| + 2\,\beta.$$

Proof. From Lemma 2.9.3, we get that for any set of values $t_{1:m}$, $\sum_{j \in Q_2} \hat{\gamma}_j \leq 2\,\rho\,\sum_{j \in Q_1 \cup Q_2} p_j + 2\,\beta$ and therefore, $\mathop{\mathbb{E}}_{t_{1:m}}[\sum_{j \in Q_2} \hat{\gamma}_j] \leq 2\,\rho\,\mathop{\mathbb{E}}_{t_{1:m}}[\sum_{j \in Q} p_j] + 2\,\beta$.

From Lemma 2.9.4, we get

$$\mathop{\mathbb{E}}_{t_{1:m}}[\sum_{j \in Q_1} \hat{\gamma}_j] \leq 8\,\mu\,\mathop{\mathbb{E}}_{t_{1:m}}[\sum_{j \in Q_1} p_j] + 4\,\kappa\,\mathbb{E}[|Q_1|].$$

Hence using Linearity of Expectation, we get

$$\sum_{j \in Q} \hat{\gamma}_j \leq (2\,\rho + 8\,\mu)\,\mathop{\mathbb{E}}_{t_{1:m}}[\sum_{j \in Q} p_j] + 4\,\kappa\,|Q| + 2\,\beta.$$

■

Proof of Theorem 2.9.1 : Recall that the optimal social welfare, and hence the optimal profit, on sequence σ is upper-bounded by $\sum_{j \in Q} \hat{\gamma}_j$ (Observation 2.9.2), while the expected profit generated by algorithm is C is given by $\mathbb{E}[\sum_{j \in Q} p_j]$. Hence Theorem 2.9.5 is equivalent to result quoted in Theorem 2.9.1. ■

Remark 2.9.6. *1. In case the social-welfare maximizing algorithm A takes estimate of U_{max}: Suppose the estimate given to algorithm C (which passes it on to the copy of A running in the background) is that $U_{max} \in [\delta\,Z, Z)$. Note that the only place where we use the guarantee is in Lemma 2.9.3. In the proof, in the stream σ', add a fake $\delta\,Z$- valuation buyer at the end of the stream to make the guarantee hold. The profit guarantee changes to*

$$\sum_{j \in Q} \hat{\gamma}_j \leq (2\,\rho + 8\,\mu)\,\mathbb{E}[\sum_{j \in Q} p_j] + 4\,\kappa\,|Q| + 2\,\beta + 2\,\rho\,\delta\,Z$$

2. Balcan et al. [2008] give a single-buyer profit maximization algorithm under zero production cost, which with slight modification, given a parameter $T > U_{max}$, has values of parameters $\kappa = \frac{T}{2\,m\,n}$ and $\mu = O(\log(m\,n))$. This profit maximization algorithm picks a uniform price on a geometric scale for all items and can be combined with either of the social welfare maximizing algorithms in this chapter to give a $O(\log(m\,n))$-profit maximizing algorithm with some additive loss.

2.10 Acknowledgment

The results in this chapter are part of a joint work with Avrim Blum, Anupam Gupta and Yishay Mansour [Blum et al., 2011].

Chapter 3

Resource allocation with expensive queries on stochastic input

3.1 Introduction

In many scenarios of interest, the value of allocation a resource to a claimant is a priori unknown. Examples include allocating an advertisement to a particular impression (a priori we do not know the value that the user has for a certain impression), allocating resources to a project (a priori we do not know how successful the project is), and matching donor-recipient pairs in a kidney exchange (a priori we do not know whether the recipient of one pair is compatible with the donor of other pair, and vice versa).

In such situations, there is usually a way to 'query' the value of allocating the resource to a claimant. For instance, by displaying an advertisement to a user, we can guess its value for the user depending on whether the user clicks on it or not. In allocation of resources to projects, by making a temporary allocation of a resources to a project and letting the project run for some time, we can ascertain the value of the project by looking at its short term success and market response. In case of kidney exchange, we can have lab tests done to ascertain the compatibility of the donor and the recipient.

In many cases, the queries that we form are expensive. For instance, lab tests in case of kidney exchange are monetarily expensive (and there is a hard-to-determine cost for the patients waiting to receive their kidneys). In project allocation, by making poor temporary allocation decisions early on, we pay the opportunity cost of making bad resource allocation decisions. For online advertisements, when we show "poor" ad to a user, we have lose the opportunity to show the user a good ad, which usually means loss in revenue.

Hence, ideally what we would like to do is to be able to perform few queries and within those few queries make good allocation decisions. There are two questions that are natural at this point. The first one is what benchmark we compare our allocation, made based on a few queries, against and the second is what information we have at our disposal that will guide us to make 'good' queries. We assume stochastic information is present that tells the probability distribution for the value of every potential allocation. We compare the value of the allocation that the algorithm makes to the highest expected value that any allocation could have achieved. We will detail the nature of stochastic information when we define the problem formally.

An important point of interest both from theoretical and application perspective is whether the queries we make are adaptive or non-adaptive, and on a related note, whether we can issue the queries in parallel or whether we need to issue them sequentially, i.e., wait for the result of the last query and then issue the new query. The ideal thing would be to have an algorithm that issues all necessary queries in parallel, and then makes allocations decisions once it receives the values of the different queries. We will explore the level of parallelism and non-adaptivity we can incorporate in the algorithm.

3.1.1 Model

We formalize our model and start with the simple case where each claimant wants exactly two resources and has value either 0 or 1, that is a priori unknown. We model this using a graph where each node of the graph represents a resources, and each edge represents the claimant that wants the two resources which form the end points of the edge. For each edge (claimant) e of the graph, we are given a number p_e which denote the prior probability that the value is 1. That is, on querying the value of the claimant e, with probability p_e, the value returned will be 1, and with probability $1 - p_e$, the value returned is zero. We want to query a 'few' edges of the graph, and give an allocation whose value in expectation is close to the expected value of *omniscient optimum* – one that queries all the edges and computes the maximum matching (which is the optimal allocation) among the existing edges.

In the general case where each claimant wants at most k resources, we are given a hypergraph, where each resource is mapped to a node and claimant is mapped to a hyperedge containing the nodes corresponding to the resources. Again, for each hyperedge e, we are given a number p_e that corresponds to the probability that the value returned is 1.

We measure the number of queries that the allocation mechanism makes on a per-resource level. That is, we ask the maximum number of queries that have been made

by the allocation mechanism to any single resource. Clearly, this maximum number of queries to a node (resource) in a graph is the number of edge (or hyperedges) incident to it. In case of a graph, this number can be at most n, the number of nodes (resources) in the graph.

Our main question is: In order to perform as well as the omniscient optimum, do we need to query (almost) all the edges, that is, do our per-vertex budgets need to be $\Theta(n)$, where n is the number of vertices? Or, can we, for any arbitrarily small $\epsilon > 0$, achieve performance that is very close to the omniscient optimal solution by using an $o(n)$ per-vertex budget?

3.1.2 Our Results and Techniques

Our main result gives a positive answer to our question, by showing that, surprisingly, a *constant* per-vertex budget is sufficient to get ϵ-close to the omniscient optimum. Indeed, we design a polynomial-time algorithm with the following properties: for any constant $\epsilon > 0$, the algorithm queries at most $O(1)$ edges incident to any particular vertex, requires $O(1)$ rounds of parallel queries, and achieves a $(1-\epsilon)$ fraction of the omniscient optimum.[1]

The foregoing algorithm is *adaptive*, in the sense that its queries are conditioned on the answers to previous queries. Even though it requires only a constant number of rounds, it is natural to ask whether a non-adaptive algorithm — one that issues all its queries in one round — can also achieve a similar guarantee. We do not give a complete answer to this question, but we do present a non-adaptive algorithm that achieves a $0.5(1-\epsilon)$-approximation (for arbitrarily small $\epsilon > 0$) to the omniscient optimum[2].

While as mentioned it is still an open question whether a non-adaptive solution can achieve a $(1-\epsilon)$-approximation, we prove in Section 3.7 that finding the best non-adaptive solution with a per-vertex guarantee is NP-hard. Specifically, if we are to place a per-vertex budget constraint of 2, then finding the subset of edges that obeys this constraint and has the maximum expected size of matching is NP-hard. This hardness result is shown using a characterization of the optimal non-adaptive solution followed by a reduction from the 3D-MATCHING problem. Interestingly, for the application of kidney exchange that in part motivates our work, we show a polynomial time algorithm to find an almost optimal non-adaptive solution with per-vertex budget constraint of two (Section 3.8).

We extend our algorithmic results to the stochastic k-set packing problem, where we

[1]This guarantee holds as long as all the non-zero p_e's are bounded away from zero by some constant. The constant can be arbitrarily small but should not depend on n.

[2]We extend our matching results to a more general stochastic model in Section 3.6.

are given a collection of sets, each with cardinality at most k. Stochastic Matching is a special case of Stochastic k-set packing: each set (which corresponds to an edge) has cardinality 2, that is, $k = 2$. In stochastic k-set packing, each set s exists with some known probability p_s, and we need to query the sets to find whether they exist. Our objective is to output a collection of *disjoint* sets of maximum cardinality. We present adaptive and non-adaptive polynomial-time algorithms that achieve, for any constant $\epsilon > 0$, at least $(\frac{2}{k} - \epsilon)$ and $(1 - \epsilon)\frac{(2/k)^2}{2/k+1}$ fraction, respectively, of the omniscient optimum, again using $O(1)$ queries per element and hence $O(n)$ overall. For the sake of comparison, the best known *polynomial-time* algorithm for optimizing k-set packing in the *standard non-stochastic setting* has an approximation ratio of $\frac{3}{k+1} - \epsilon$ [Fürer and Yu, 2013].

To better appreciate the result, we show how naïve algorithms fail to achieve our goal, even if they are allowed many queries. For example, querying a sublinear number of edges incident to each vertex, chosen uniformly at random, gives a vanishing fraction of the omniscient optimum — as we show in Section 3.2.

The primary technical ingredient in the design of our *adaptive algorithm* is that if, in any round r of the algorithm, the solution computed by round r (based on previous queries) is small compared to the omniscient optimum, then the current structure must admit a *large collection of disjoint constant-sized 'augmenting' structures*. These augmenting structures are composed of sets that have not been queried so far. Of course, we do not know whether these structures we are counting on to help augment our current matching actually exist; but we do know that these augmenting structures have constant size (and so each structure exists with some constant probability) and are *disjoint* (and therefore the outcomes of the queries to the different augmenting structures are independent). Hence, by querying all these structures in parallel in round r, in expectation, we can close a constant fraction of the gap between our current solution and the omniscient optimum. By repeating this argument over a constant number of rounds, we achieve a $(1-\epsilon)$ fraction of the omniscient optimum. In the case of stochastic matching, these augmenting structures are simply augmenting paths; in the more general case of k-set packing, we borrow the notion of augmenting structures from Hurkens and Schrijver [1989].

3.1.3 Related Work

Our model of the resource allocation problem where each claimant wants two resources goes by the name of stochastic matching in literature. Prior work has considered multiple variants of stochastic matching. The primary variant that has been considered is that of *query-commit*, where the algorithm is *forced* to add any queried edge to the matching if

the edge is found to exist (that is, any claimant found to have value 1 has to be allocated the resources). Furthermore, the query-commit setting has been studied with *per-vertex budget* constraints (see, e.g., [Bansal et al., 2012]). Here there is a given budget b_v for every vertex v, which denotes the maximum number of edges incident to this vertex that can be queried by the algorithm. Several papers, including those by Chen et al. [2009], Adamczyk [2011], and Bansal et al. [2012], design algorithms that achieve *constant-factor* approximations to the optimal maximum matching, subject to the foregoing constraints. In this work, we eschew the query-commit requirement since it has been show that the best approximation factor achievable with this constraint is bounded away from 1 [Costello et al., 2012], and our goal is to achieve $(1-\epsilon)$-approximation for arbitrarily small $\epsilon > 0$.

3.2 Preliminaries

For any graph $G = (V, E)$, let $M(E)$ denote its maximum (cardinality) matching.[3] In addition, for two matchings M and M', we denote their *symmetric difference* by $M \Delta M' = (M \cup M') \setminus (M \cap M')$; it includes only paths and cycles consisting of alternating edges of M and M'.

In the stochastic setting, given a set of edges X, define X_p to be the random subset formed by including each edge of X independently with probability p. We will assume for ease of exposition that $p_e = p$ for all edges $e \in E$. Our results hold when p is a lower bound, i.e., $p_e \geq p$ for all $e \in E$. Furthermore, in Section 3.6, we show that we can extend our results to a more general setting where the existence probabilities of edges incident to any particular vertex are correlated.

Given a graph $G = (V, E)$, define $\overline{M}(E)$ to be $\mathbb{E}[|M(E_p)|]$, where the expectation is taken over the random draw E_p. In addition, given the results of queries on some set of edges T, define $\overline{M}(E|T)$ to be $\mathbb{E}[|M(X_p \cup T')|]$, where $T' \subseteq T$ is the subset of edges of T that are known to exist based on the queries, and $X = E \setminus T$.

In the *non-adaptive* version of our problem, the goal is to design an algorithm that, given a graph $G = (V, E)$ with $|V| = n$, queries a subset X of edges in parallel such that $|X| = O(n)$, and maximizes the ratio $\overline{M}(X)/\overline{M}(E)$.

In contrast, an *adaptive* algorithm proceeds in rounds, and in each round queries a subset of edges in parallel. Based on the results of the queries up to the current round, it can choose the subset of edges to test in the next round. Formally, an R-*round adaptive*

[3] In the notation $M(E)$, we intentionally suppress the dependence on the vertex set V, since we care about the maximum matchings of different subsets of edges for a fixed vertex set.

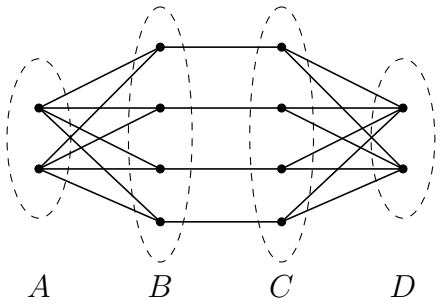


Figure 3.1: Illustration of the construction in Example 3.2.1, for $t = 4$ and $\beta = 1/2$.

stochastic matching algorithm selects, in each round r, a subset of edges $X_r \subseteq E$, where X_r can be a function of the results of the queries on $\bigcup_{1 \leq i \leq r} X_i$. The objective is to maximize the ratio $\mathbb{E}[|M(\bigcup_{1 \leq i \leq R} X_i)|]/\overline{M}(E)$, where the expectation in the numerator is taken over the outcome of the query results and the sets X_i chosen by the algorithm.

To gain some intuition about our goal of arbitrarily good approximations to the omniscient optimum, and why it is challenging, let us consider the naïve (non-adaptive) algorithm which queries t^α, for any $\alpha < 1$, random neighbors of each vertex. The following example shows that this algorithm performs poorly.

Example 3.2.1. *Consider the graph $G = (V, E)$ whose vertices are partitioned into sets A, B, C, and D, such that $|A| = |D| = t^\beta$ and $|B| = |C| = t$, for some $1 > \beta > \alpha$. Let E consist of one perfect matching between vertices of B and C, and two complete bipartite graphs, one between A and B, and another between C and D. See Figure 3.1 for an illustration. Let $p = 0.5$ be the existence probability of any edge.*

The omniscient optimal solution can use any edge, and, in particular, it can use the edges between B and C. Since, these edges form a matching of size t and $p = 0.5$, they alone provide a matching of expected size $t/2$. Hence, $\overline{M}(E) \geq t/2$.

Now, consider the algorithm described above. For every vertex in B, the probability that its edge to C is chosen is at most $\frac{t^\alpha}{t^\beta+1}$ (similarly for the edges from C to B). Therefore, the expected number of edges chosen between B and C is at most $\frac{2t^{1+\alpha}}{t^\beta+1}$, and the expected number of existing edges between B and C, after the coin tosses, is at most $\frac{t^{1+\alpha}}{t^\beta+1}$. A and D each have t^β vertices, so they contribute at most $2t^\beta$ edges to any matching. Therefore, the expected size of the overall matching is no more than $t^{1+\alpha-\beta} + 2t^\beta$. Using $n = \Theta(t)$, we conclude that the approximation ratio of the naïve algorithm approaches 0, as $n \to \infty$. For $\alpha = 0.5$ and $\beta = 0.75$, the approximation of the naïve algorithm ratio is $O(1/n^{0.25})$, at best.

3.3 Adaptive Algorithm: $(1-\epsilon)$-approximation

In this section, we present our main result: an adaptive algorithm — formally given as Algorithm 1 — that achieves a $(1-\epsilon)$ approximation to the omniscient optimum for arbitrarily small $\epsilon > 0$, using $O(1)$ queries per vertex and $O(1)$ rounds.

The algorithm is initialized with the empty matching M_0. At the end of each round r, our goal is to maintain a maximum matching M_r on the set of edges that are known to exist (based on queries made so far). To this end, at round r, we compute the maximum matching O_r on the set of edges that are known to exist *and* the ones that have not been queried yet (Step 2a). We consider augmenting paths in $O_r \Delta M_{r-1}$, and query all the edges in them (Steps 2b and 2c). Based on the results of these queries (Q_r), we update the maximum matching (M_r). Finally, we return the maximum matching M_R computed after $R = \frac{\log(2/\epsilon)}{p^{2/\epsilon}}$ rounds. (Let us assume that R is an integer for ease of exposition.)

Input: A graph $G = (V, E)$.

Parameter: $R = \frac{\log(2/\epsilon)}{p^{2/\epsilon}}$

Algorithm:

1. Initialize M_0 to the empty matching and $W_1 \leftarrow \emptyset$.
2. For $r = 1, \ldots, R$, do
   (a) Compute a maximum matching, O_r, in $(V, E \setminus W_r)$.
   (b) Set Q_r to the collection of all augmenting paths in $O_r \Delta M_{r-1}$.
   (c) Query the edges in Q_r. Let Q'_r and Q''_r represent the set of existing and non-existing edges.
   (d) $W_{r+1} \leftarrow W_r \cup Q''_r$.
   (e) Set M_r to the maximum matching in $\left(V, \bigcup_{j=1}^{r} Q'_j\right)$.
3. Output M_R.

Algorithm 1: ADAPTIVE ALGORITHM FOR STOCHASTIC MATCHING: $(1-\epsilon)$ APPROXIMATION

It is easy to see that *this algorithm queries at most* $\frac{\log(2/\epsilon)}{p^{2/\epsilon}}$ *edges per vertex*: In a given round r, the algorithm queries edges that are in augmenting paths of $O_r \Delta M_{r-1}$. Since there is at most one augmenting path using any particular vertex, the algorithm queries

at most one edge per vertex in each round. Furthermore, the algorithm executes $\frac{\log(2/\epsilon)}{p^{2/\epsilon}}$ rounds. Therefore, the number of queries issued by the algorithm per vertex is as claimed.

The rest of the section is devoted to proving that the matching returned by this algorithm after R rounds has cardinality that is, in expectation, at least a $(1-\epsilon)$ fraction of $\overline{M}(E)$.

Theorem 3.3.1. *For any graph $G = (V, E)$ and any $\epsilon > 0$, Algorithm 1 returns a matching whose expected cardinality is at least $(1-\epsilon)\,\overline{M}(E)$ in $R = \frac{\log(2/\epsilon)}{p^{(2/\epsilon)}}$ rounds.*

As mentioned in Section 3.1.2, one of the insights behind this result is the existence of many *disjoint* augmenting paths of *bounded length* that can be used to augment a matching that is far from the omniscient optimum, that is, a lower bound on the number of elements in Q_r of a given length L. This observation is formalized in the following lemma. (We emphasize that the lemma pertains to the non-stochastic setting.)

Lemma 3.3.2. *Consider a graph $G = (V, E)$ with two matchings M_1 and M_2. Suppose $|M_2| > |M_1|$. Then in $M_1 \Delta M_2$, for any odd length $L \geq 1$, there exist at least $|M_2| - (1 + \frac{2}{L+1})|M_1|$ augmenting paths of length at most L, which augment the cardinality of M_1.*

Proof. Let x_l be the number of augmenting paths of length l (for any odd $l \geq 1$) found in $M_1 \Delta M_2$ that augment the cardinality of M_1. Each augmenting path increases the size of M_1 by 1, so the total number of augmenting paths $\sum_{l\geq 1} x_l$ is at least $|M_2| - |M_1|$. Moreover, each augmenting path of length l has $\frac{l-1}{2}$ edges in M_1. Hence, $\sum_{l\geq 1} \frac{l-1}{2} x_l \leq |M_1|$. In particular, this implies that $\frac{L+1}{2} \sum_{l\geq L+2} x_l \leq |M_1|$. We conclude that

$$\sum_{l=1}^{L} x_l = \sum_{l\geq 1} x_l - \sum_{l\geq L+2} x_l \geq (|M_2| - |M_1|) - \frac{2}{L+1}|M_1| = |M_2| - \left(1 + \frac{2}{L+1}\right)|M_1|.$$

■

The rest of the theorem's proof requires some additional notation. At the beginning of any given round r, the algorithm already knows about the existence (or non-existence) of the edges in $\bigcup_{i=1}^{r-1} Q_i$. We use Z_r to denote the expected size of the maximum matching in graph $G = (V, E)$ given the results of the queries $\bigcup_{i=1}^{r-1} Q_i$. More formally, $Z_r = \overline{M}(E | \bigcup_{i=1}^{r-1} Q_i)$. Note that $Z_1 = \overline{M}(E)$.

For a given r, we use the notation $\mathbb{E}_{Q_r}[X]$ to denote the expected value of X where the expectation is taken *only* over the outcome of query Q_r, and fixing the outcomes on the results of queries $\bigcup_{i=1}^{r-1} Q_i$. Moreover, for a given r, we use $\mathbb{E}_{Q_r,\ldots,Q_R}[X]$ to denote the

expected value of X with the expectation taken over the outcomes of queries $\bigcup_{i=r}^{R} Q_i$, and fixing an outcome on the results of queries $\bigcup_{i=1}^{r-1} Q_i$.

In Lemma 3.3.3, for any round r and for *any* outcome of the queries $\bigcup_{i=1}^{r-1} Q_i$, we lower-bound the *expected increase in the size of* M_r over the size of M_{r-1}, with the expectation being taken only over the outcome of edges in Q_r. This lower bound is a function of Z_r.

Lemma 3.3.3. *For any $r \in [R]$, odd L, and $Q_1, \cdots, Q_{r-1}$, it holds that $\mathbb{E}_{Q_r}[|M_r|] \geq (1-\gamma)|M_{r-1}| + \alpha Z_r$, where $\gamma = p^{(L+1)/2}\left(1+\frac{2}{L+1}\right)$ and $\alpha = p^{(L+1)/2}$.*

Proof. By Lemma 3.3.2, there exist at least $|O_r| - (1+\frac{2}{L+1})|M_{r-1}|$ augmenting paths in $O_r \Delta M_{r-1}$ that augment M_{r-1} and are of length at most L. The O_r part of every augmenting path of length at most L exists independently with probability at least $p^{(L+1)/2}$. Therefore, the expected increase in the size of the matching is:

$$\begin{aligned}\mathbb{E}_{Q_r}[|M_r|] - |M_{r-1}| \;\geq\; p^{\frac{L+1}{2}}\left(|O_r| - \left(1+\frac{2}{L+1}\right)|M_{r-1}|\right) \;&=\; \alpha|O_r| - \gamma|M_{r-1}| \\ &\geq\; \alpha Z_r - \gamma|M_{r-1}|,\end{aligned}$$

where the last inequality holds by the fact that Z_r, which is the expected size of the optimal matching with expectation taken over non-queried edges, cannot be larger than O_r, which is the maximum matching assuming that every non-queried edge exists. ■

We are now ready to prove the theorem.

Proof of Theorem 3.3.1. Let $L = \frac{4}{\epsilon} - 1$; it is assumed to be an odd integer for ease of exposition.[4] By Lemma 3.3.3, we know that for every $r \in [R]$, $\mathbb{E}_{Q_r}[|M_r|| \geq (1-\gamma)|M_{r-1}| + \alpha Z_r$, where $\gamma = p^{(L+1)/2}(1+\frac{2}{L+1})$, and $\alpha = p^{(L+1)/2}$. We will use this inequality repeatedly to derive our result. We will also require the equality

$$\mathbb{E}_{Q_{r-1}}[Z_r] = \mathbb{E}_{Q_{r-1}}\left[\overline{M}(E|\bigcup_{i=1}^{r-1} Q_i)\right] = \overline{M}(E|\bigcup_{i=1}^{r-2} Q_i) = Z_{r-1}. \tag{3.1}$$

First, applying Lemma 3.3.3 at round R, we have that $\mathbb{E}_{Q_R}[|M_R|] \geq (1-\gamma)|M_{R-1}| + \alpha Z_R$. This inequality is true for any fixed outcomes of $Q_1, \ldots, Q_{R-1}$. In particular, we can take the expectation over Q_{R-1}, and obtain

$$\mathbb{E}_{Q_{R-1},Q_R}[|M_R|] \geq (1-\gamma)\,\mathbb{E}_{Q_{R-1}}[|M_{R-1}|] + \alpha\,\mathbb{E}_{Q_{R-1}}[Z_R].$$

[4]Otherwise there exists $\epsilon/2 \leq \epsilon' \leq \epsilon$ such that $\frac{4}{\epsilon'} - 1$ is an odd integer. We use a similar simplification in the proofs of other results.

By Equation (3.1), we know that $\mathbb{E}_{Q_{R-1}}[Z_R] = Z_{R-1}$. Furthermore, we can apply Lemma 3.3.3 to $\mathbb{E}_{Q_{R-1}}[|M_{R-1}|]$ to get the following inequality:

$$\begin{aligned}\mathbb{E}_{Q_{R-1},Q_R}[|M_R|] &\geq (1-\gamma)\,\mathbb{E}_{Q_{R-1}}[|M_{R-1}|] + \alpha\,\mathbb{E}_{Q_{R-1}}[Z_R] \\ &\geq (1-\gamma)\,\left((1-\gamma)\,|M_{R-2}| + \alpha\,Z_{R-1}\right) + \alpha\,Z_{R-1} \\ &= (1-\gamma)^2\,|M_{R-2}| + \alpha\,(1+(1-\gamma))\,Z_{R-1}.\end{aligned}$$

We repeat the above steps by sequentially taking expectations over Q_{R-2} through Q_1, and at each step applying Equation (3.1) and Lemma 3.3.3. This gives us

$$\begin{aligned}\mathbb{E}_{Q_1,\ldots,Q_R}[|M_R|] \;&\geq\; (1-\gamma)^R|M_0| + \alpha\,(1+(1-\gamma)+\cdots+(1-\gamma)^{R-1})Z_1 \\ &=\; \alpha\,\frac{1-(1-\gamma)^R}{\gamma}Z_1,\end{aligned}$$

where the second transition follows from the initialization of M_0 as an empty matching. Since $L = \frac{4}{\epsilon} - 1$ and $R = \frac{\log(2/\epsilon)}{p^{2/\epsilon}}$, we have

$$\begin{aligned}\frac{\alpha}{\gamma}\big(1-(1-\gamma)^R\big) = \left(1-\frac{2}{L+1}\right)\big(1-(1-\gamma)^R\big) \geq 1-\frac{2}{L+1}-e^{-\gamma R} &\geq 1-\frac{\epsilon}{2}-\frac{\epsilon}{2} \\ &= 1-\epsilon,\end{aligned}$$

where the second transition is true because $e^{-x} \geq 1-x$ for all $x \in \mathbb{R}$. We conclude that $\mathbb{E}_{Q_1,\ldots,Q_R}[|M_R|] \geq (1-\epsilon)\,Z_1$. Because $Z_1 = \overline{M}(E)$, it follows that expected size of the algorithm's output is at least $(1-\epsilon)\,\overline{M}(E)$. ■

In Section 3.6, we extend our results to the setting where edges have correlated existence probabilities — an edge's probability is determined by parameters associated with its two vertices. This generalization gives a better model for kidney exchange, as some patients are *highly sensitized* and therefore harder to match in general; this means that all edges incident to such vertices are less likely to exist. We consider two settings, first, where an adversary chooses the vertex parameters, and second, where these parameters are drawn from a distribution. Our approach involves excluding from our analysis edges that have too low an existence probability. We do so by showing that (under specific conditions) excluding any augmenting path that includes such edges still leaves us with a large number of constant-size augmenting paths.

3.4 Non-adaptive algorithm: 0.5-approximation

The adaptive algorithm, Algorithm 1, augments the current matching by computing a maximum matching on queried edges that are known to exist, and edges that have not been queried. One way to extend this idea to the non-adaptive setting is the following: we can simply choose several edge-disjoint matchings, and hope that they help in augmenting each other. In this section, we ask: How close can this non-adaptive interpretation of our adaptive approach take us to the omniscient optimum?

In more detail, our non-adaptive algorithm — formally given as Algorithm 2 — iterates $R = \frac{\log(2/\epsilon)}{p^{2/\epsilon}}$ times. In each iteration, it picks a maximum matching and removes it. The set of edges queried by the algorithm is the union of the edges chosen in some iteration. We will show that, for any arbitrarily small $\epsilon > 0$, the algorithm finds a $0.5(1-\epsilon)$-approximate solution. Since we allow an arbitrarily small (though constant) probability p for stochastic matching, achieving a 0.5-approximation independently of the value of p, while querying only a linear number of edges, is nontrivial. For example, a naïve algorithm that only queries one maximum matching clearly does not guarantee a 0.5-approximation — it would guarantee only a p-approximation. In addition, the example given in Section 3.2 shows that choosing edges at random performs poorly.

Input: A graph $G(V, E)$.

Parameter: $R = \frac{\log(2/\epsilon)}{p^{2/\epsilon}}$

Algorithm:

1. Initialize $W_1 \leftarrow \emptyset$.
2. For $r = 1, \ldots, R$, do
   (a) Compute a maximum matching, O_r, in $\left(V, E \setminus \bigcup_{1 \le i \le r-1} W_i\right)$.
   (b) $W_r \leftarrow W_{r-1} \cup O_r$.
3. Query all the edges in W_R, and output the maximum matching among the edges that are found to exist in W_R.

Algorithm 2: NON-ADAPTIVE ALGORITHM FOR STOCHASTIC MATCHING: 0.5-APPROXIMATION

The number of edges incident to any particular vertex that are queried by the algorithm is at most $\frac{\log(2/\epsilon)}{p^{2/\epsilon}}$, because the vertex can be matched with at most one neighbor in each round. The next theorem establishes the approximation guarantee of Algorithm 2.

Theorem 3.4.1. *Given a graph $G = (V, E)$ and any $\epsilon > 0$, the expected size $\overline{M}(W_R)$ of the matching produced by Algorithm 2 is at least a $0.5(1 - \epsilon)$ fraction of $\overline{M}(E)$.*

Lemma 3.4.2. *Let E_1 be an arbitrary subset of edges of E, and let $E_2 = E \setminus E_1$. Then $\overline{M}(E) \leq \overline{M}(E_1) + \overline{M}(E_2)$.*

Proof. Let E' be an arbitrary subset of edges of E, and let $E'_1 = E_1 \cap E'$ and $E'_2 = E_2 \cap E'$. We claim that $|M(E')| \leq |M(E'_1)| + |M(E'_2)|$. This is because if T is the set of edges in a maximum matching in graph (V, E'), then clearly $T \cap E'_1$ and $T \cap E'_2$ are valid matchings in E'_1 and E'_2 respectively, and thereby it follows that $|M(E'_1)| \geq |T \cap E'_1|$ and $|M(E'_2)| \geq |T \cap E'_2|$, and hence $|M(E')| \leq |M(E'_1)| + |M(E'_2)|$. Expectation is a convex combination of the values of the outcomes. For every subset E' of edges in E, multiplying the above inequality by the probability that the outcome of the coin tosses on the edges of E is E', and then summing the various inequalities, we get $\overline{M}(E) \leq \overline{M}(E_1) + \overline{M}(E_2)$. ■

In order to lower bound $\overline{M}(W_R)$, we first show that for any round r, either our current collection of edges has an expected matching size $\overline{M}(W_{r-1})$ that compares well with $\overline{M}(E)$, or in round r, we have a significant increase in $\overline{M}(W_r)$ over $\overline{M}(W_{r-1})$.

Lemma 3.4.3. *At any iteration $r \in [R]$ of Algorithm 2 and odd L, if $\overline{M}(W_{r-1}) \leq \overline{M}(E)/2$, then*

$$\overline{M}(W_r) \geq \frac{\alpha}{2}\,\overline{M}(E) + (1 - \gamma)\overline{M}(W_{r-1}),$$

where $\gamma = p^{(L+1)/2}(1 + \frac{L+1}{2})$ and $\alpha = p^{(L+1)/2}$.

Proof. Define $U = E \setminus W_{r-1}$. Assume that $\overline{M}(W_{r-1}) \leq \overline{M}(E)/2$. By Lemma 3.4.2, we know that $\overline{M}(U) \geq \overline{M}(E) - \overline{M}(W_{r-1})$. Hence, $|O_r| = |M(U)| \geq \overline{M}(U) \geq \overline{M}(E) - \overline{M}(W_{r-1}) \geq \overline{M}(E)/2$.

In a thought experiment, say at the beginning of round r, we query the set W_{r-1} and let W'_{r-1} be the set of edges that are found to exist. By Lemma 3.3.2, there are at least $|O_r| - (1 + \frac{2}{L+1})|M(W'_{r-1})|$ augmenting paths of length at most L in $O_r \Delta M(W'_{r-1})$ that augment $M(W'_{r-1})$. Each of these paths succeeds with probability at least $p^{(L+1)/2}$. We have,

$$\begin{aligned}\overline{M}(O_r \cup W'_{r-1}|W'_{r-1}) - |M(W'_{r-1})| &\geq p^{(L+1)/2}\left(|O_r| - (1 + \frac{2}{L+1})|M(W'_{r-1})|\right)\\ &\geq p^{(L+1)/2}\left(\frac{1}{2}\,\overline{M}(E) - (1 + \frac{2}{L+1})|M(W'_{r-1})|\right),\end{aligned}$$

where the expectation on the left hand side is taken only over the outcome of the edges in O_r. Therefore, we have $\overline{M}(O_r \cup W'_{r-1}|W'_{r-1}) \geq \frac{\alpha}{2}\,\overline{M}(E) + (1-\gamma)|M(W'_{r-1})|$, where $\alpha = p^{(L+1)/2}$ and $\gamma = p^{(L+1)/2}\,(1+\frac{2}{L+1})$. Taking expectation over the coin tosses on W_{r-1} that create outcome W'_{r-1}, we have our result, i.e.,

$$\overline{M}(W_r) \geq \mathbb{E}_{W_{r-1}}[\overline{M}(O_r \cup W'_{r-1}|W'_{r-1})] \geq \overline{M}(O_r \cup W_{r-1}) \geq \frac{\alpha}{2}\,\overline{M}(E) + (1-\gamma)\overline{M}(W_{r-1}).$$

■

Proof of Theorem 3.4.1. For ease of exposition, assume $L = \frac{4}{\epsilon} - 1$ is an odd integer. Then, either $\overline{M}(W_R) \geq \overline{M}(E)/2$ in which case we are done. Or otherwise, by repeatedly applying Lemma 3.4.3 for R steps, we have

$$\overline{M}(W_R) \;\geq\; \frac{\alpha}{2}(1 + (1-\gamma) + (1-\gamma)^2 + \cdots + (1-\gamma)^{R-1})\overline{M}(E) \;\geq\; \frac{\alpha}{2}\frac{(1-(1-\gamma)^R)}{\gamma}\overline{M}(E).$$

Now, $\frac{\alpha}{\gamma}(1-(1-\gamma)^R) \geq 1 - \frac{2}{L+1} - e^{-\gamma R} \geq 1 - \epsilon$ for $R = \frac{\log(2/\epsilon)}{p^{2/\epsilon}}$. Hence, we have our $0.5(1-\epsilon)$ approximation. ■

As explained in Section 3.9, we do not know whether in general non-adaptive algorithms can achieve a $(1-\epsilon)$-approximation with $O(1)$ queries per vertex. However, if there is such an algorithm, it is not Algorithm 2! Indeed, the next theorem shows that the algorithm cannot give an approximation ratio better than $5/6$ to the omniscient optimum. In fact, it holds even when $R = \Theta(\log n)$.

Theorem 3.4.4. *Let $p = 0.5$. For any $\epsilon > 0$ there exists n and a graph (V, E) with $|V| \geq n$ such that Algorithm 2, with $R = O(\log n)$, returns a matching with expected size of at most $\frac{5}{6}\overline{M}(E) + \epsilon$.*

Claim 3.4.5. *Let $G = (U \cup V, U \times V)$ be a complete bipartite graph between U and V with $|U| = |V| = n$. For any constant probability p, $\overline{M}(E) \geq n - o(n)$.*

Proof. Denote by E_p the random set of edges formed by including each edge in $U \times V$ independently with probability p. We show that with probability at least $1 - \frac{1}{n^8}$, over the draw E_p, the maximum matching in the graph $(U \cup V, E_p)$ is at least $n - c\log(n)$, where $c = 10/\log(\frac{1}{(1-p)})$, and this will complete our claim.

In order to show this, we prove that with probability at least $1 - \frac{1}{n^8}$, over the draw E_p, all subsets $S \subseteq U$ of size at most $n - c\log(n)$, have a neighborhood of size at least $|S|$. By Hall's theorem, our claim will follow.

Consider any set $S \subseteq U$ of size at most $n - c\log(n)$. We will call set S 'bad' if there exists some set $T \subseteq V$ of size $(|S| - 1)$ such that S does not have edges to $V \setminus T$. Fix any set $T \subseteq V$ of size $|S| - 1$. Over draws of E_p, the probability that S has no outgoing edges to $V \setminus T$ is at most $(1-p)^{|S||V\setminus T|} = (1-p)^{|S|(n-|S|+1)}$. Hence, by union bound, the probability that S is bad is at most $\binom{n}{|S|-1}(1-p)^{|S|(n-|S|+1)}$.

Again, by union bound, the probability that some set $S \subseteq U$ of size at most $n - c\log(n)$ is bad is at most $\sum_{1\leq|S|\leq n-c\log(n)} \binom{n}{|S|}\binom{n}{|S|-1}(1-p)^{|S|(n-|S|+1)}$ and this in turn is at most

$$\sum_{1\leq|S|\leq n-c\log(n)} n^{|S|}n^{|S|}(1-p)^{|S|(n-|S|+1)} \leq \sum_{1\leq|S|\leq n-c\log(n)} e^{|S|\cdot(2\log(n)+(n+1)\log(1-p)-|S|\log(1-p))}$$

Note that the exponent in the summation achieves its maximum for $|S| = 1$. For $c = 10/\log(\frac{1}{1-p})$, we have that the given sum is at most $\exp(-\frac{n}{2}\log(\frac{1}{1-p}))$, and hence with high probability, no set $S \subseteq U$ of size at most $n - c\log(n)$ is bad. ■

Proof of Theorem 3.4.4. Let (V, E) be a graph, illustrated in Figure 3, whose vertices are partitioned into sets A, B, C, and D, such that $|A| = |D| = \frac{t}{2}$, $|B| = |C| = t$. The edge set E consists of one perfect matching between vertices of B and C, and two complete bipartite graphs, one between A and B, and another between C and D. Let $p = 0.5$ be the existence probability of any edge.

We first examine the value of the omniscient optimal, $\overline{M}(E)$. Since $p = 0.5$, in expectation, half of the edges in the perfect matching between B and C exist, and therefore half of the vertices of B and C will get matched. By Claim 3.4.5, with high probability, the complete bipartite graph between the remaining half of B and A has a matching of size at least $t/2 - o(t)$. And similarly, with high probability, the complete bipartite graph between remaining half of C and D has a matching of size at least $t/2 - o(t)$. Therefore, $\overline{M}(E)$ is at least $\frac{3}{2}t - o(t)$.

Next, we look at Algorithm 2. For ease of exposition, let B_1 and B_2 denote the top and bottom half of the vertices in B. Similarly, define C_1 and C_2. Since Algorithm 2 picks maximum matchings arbitrarily, we show that there exists a way of picking maximum matchings such that the expected matching size of the union of the edges picked in the matching is at most $\frac{5}{4}t$ $(= \frac{5}{6}\,\frac{3}{2}t)$.

Consider the following choice of maximum matching picked by the algorithm: In the first round, the algorithm picks the perfect matching between B_1 and C_1, and a perfect matching between A and B_2, and a perfect matching between C_2 and D. In the second round, the algorithm picks the perfect matching between B_2 and C_2, and a perfect matching each between A and B_1, and between C_1 and D. After these two rounds, we can see

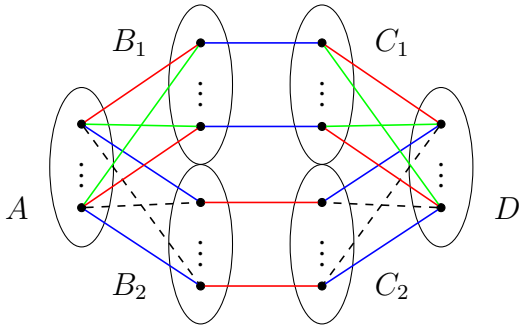


Figure 3.2: The blue and red edges represent the matching picked at rounds 1 and 2, respectively. The green edges represent the edges picked at round 3 and above. The dashed edges are never picked by the algorithm.

that there are no more edges left between B and C. For the subsequent $R-2$ rounds, in each round, the algorithms picks a perfect matching between A and B_1, and a perfect matching between C_1 and D. It is easy to verify that in every round, the algorithm has picked a maximum matching from the remnant graph.

We analyze the expected size of matching output by the algorithm. For each of the vertices in B_2 and C_2, the algorithm has picked only two incident edges. For any vertex in B_2 and C_2, with probability at least $(1-p)^2 = \frac{1}{4}$, none of these two incident edges exist. Hence, the expected number of vertices that are *unmatched* in B_2 and C_2 is at least $\frac{1}{4}(\frac{t}{2}+\frac{t}{2}) = \frac{t}{4}$. Since the vertices in A can only be matched with vertices in B, and the vertices in D can only be matched with vertices in C, it follows that at least $\frac{t}{4}$ of the vertices in A and C are unmatched in expectation. Hence, the total number of edges included in the matching is at most $\frac{5}{4}t$. This completes our claim. ■

3.5 Generalization to k-Set Packing

So far we have focused on stochastic matching, for ease of exposition. But our approach directly generalizes to the k-set packing problem.

Formally, a k-set packing instance (U, A) consists of a set of elements U, $|U| = n$, and a collection of subsets A, such that each subset S in A contains at most k elements of U, that is, $S \subseteq U$ and $|S| \leq k$. Given such an instance, a feasible solution is a collection of sets $B \subseteq A$ such that any two sets in B are disjoint. We use $K(A)$ to denote the largest feasible solution B.

Finding an optimal solution to the non-stochastic k-set packing problem is **NP**-hard (see, e.g., [Abraham et al., 2007] for the special case of k-cycle packing). Hurkens and Schrijver [1989] designed a polynomial-time local search algorithm with an approximation ratio of $(\frac{2}{k} - \eta)$, using local improvements of *constant size* that depends only on η and k. We denote this constant by $s_{\eta,k}$. More formally, consider an instance (U, A) of k-set packing and let $B \subseteq A$ be a collection of disjoint k-sets. (C, D) is said to be an *augmenting structure* for B if removing D and adding C to B increases the cardinality and maintains the disjointness of the resulting collection, i.e., if $(B \cup C) \setminus D$ is a disjoint collection of k-sets and $|(B \cup C) \setminus D| > |B|$, where $C \subseteq A$ and $D \subseteq B$.

Turning to the stochastic version of the problem, given (U, A), let A_p be a random subset of A where each set from A is included in A_p independently with probability p. We then define $\overline{K}(A)$ to be $\mathbb{E}[|K(A_p)|]$, where the expectation is taken over the random draw A_p. Similarly to the matching setting, this is the omniscient optimum — our benchmark.

A notation that we will use in some parts of the analysis is $\overline{K}(A|B)$ that we define as follows: Given a collection $B \subseteq A$ that has been queried and $B' \subseteq B$ that exists, we use $\overline{K}(A|B)$ to denote $\mathbb{E}[|K(X_p \cup B')|]$ where X_p is the random set formed by including every element of $A \setminus B$ independently with probability p.

In Section 3.5.1, we give a polynomial time algorithm that finds a linear number of disjoint constant-sized augmenting structures for a given solution of k-set packing. Next, in Sections 3.5.2 and 3.5.3, we introduce and analyze the adaptive and non-adaptive algorithms for k-set packing, respectively.

3.5.1 Disjoint Constant-Size Augmenting Structures for k-Set Packing

Hurkens and Schrijver [1989] prove the following on augmenting structures for k-set packing:

Lemma 3.5.1 ([Hurkens and Schrijver, 1989])**.** *Given a collection B of disjoint sets such that $|B| < (2/k - \eta)|K(A)|$, there exists an augmenting structure (C, D) for B such that both C and D have at most $s_{\eta,k}$ sets, for a constant $s_{\eta,k}$ which depends only on η and k.*

However, we need to find *many* augmenting structures. We use Lemma 3.5.1 to prove:

Lemma 3.5.2. *If $|B| < |K(A)|$, then there exist $\frac{1}{k\, s_{\eta,k}}(|K(A)| - \frac{|B|}{\frac{2}{k}-\eta})$ disjoint augmenting structures that augment the cardinality of B, each with size at most $s_{\eta,k}$. Moreover, this collection of augmenting structures can be found in polynomial time.*

We introduce Algorithm 3 that runs on a k-set instance and a current solution, and returns a linear number of disjoint augmenting structures.

Input: k-set instance (U, A) and a collection $B \subseteq A$ of disjoint sets.
Output: Collection Q of disjoint augmenting structures (C, D) for B.
Parameter: $s_{\eta,k}$ (the desired maximum size of the augmenting structure)
Algorithm:

1. Initialize $A_1 \leftarrow A$ and $Q \leftarrow \phi$ (empty set).
2. For $t = 1, \cdots, |A|$
   (a) Find a $s_{\eta,k}$-sized augmenting structure (C, D) for B on the k-set instance (U, A_t).
   (b) Add (C, D) to Q. (If C is an empty set, break out of the loop.)
   (c) Set A_{t+1} to be A_t minus the collection C and any set in $A_t \setminus B$ that intersects with C.
3. Output Q.

Algorithm 3: FINDING CONSTANT-SIZE DISJOINT AUGMENTING STRUCTURES FOR k-SETS

Proof of Lemma 3.5.2. We prove this lemma using Algorithm 3. We claim that if we run this algorithm on the k-set instance (U, A) and the collection B, then it will return a collection Q of at least $T = \frac{1}{k\, s_{\eta,k}}(|K(A)| - \frac{|B|}{\frac{2}{k}-\eta})$ disjoint augmenting structures (C, D) for B. By Step 2c, we are guaranteed that Q consists of *disjoint* augmenting structures. Hence, all that is left to show is that in each of the first T iterations, at Step 2a, we are able to find a *non-empty* augmenting structure (C, D) for B.

By Lemma 3.5.1, we know that if at iteration t, it is the case that $|B| < (\frac{2}{k}-\eta)|K(A_t)|$, then we will find a $s_{\eta,k}$-sized augmenting structure (C, D) for B. To prove that the inequality holds at each iteration $t \leq T$, we first claim that for all t, $|K(A_t)| \geq |K(A)| - (t - 1) \cdot k \cdot s_{\eta,k}$. We prove this by induction. The claim is clearly true for the base case of $t = 1$. For the inductive step, suppose it is true for t, then we know that $|K(A_t)| \geq |K(A)| - (t-1) \cdot k \cdot s_{\eta,k}$. At iteration t, the augmenting structure (C, D) can intersect with at most $s_{\eta,k} \cdot k$ many sets of $K(A_t)$. This is true since $K(A_t)$ consists of *disjoint* sets, and the augmenting structure (C, D) is of size at most $s_{\eta,k}$. Hence, Step 2c reduces $|K(A_t)|$ by at most $k \cdot s_{\eta,k}$. So, $|K(A_{t+1})| \geq |K(A_t)| - k \cdot s_{\eta,k}$. Combining the two

inequalities, $|K(A_t)| \geq |K(A)| - (t-1) \cdot k \cdot s_{\eta,k}$ and $|K(A_{t+1})| \geq |K(A_t)| - k \cdot s_{\eta,k}$, we have $|K(A_{t+1})| \geq |K(A)| - t \cdot k \cdot s_{\eta,k}$.

Hence, if the for-loop adds *non-empty* augmenting structures only for the first t rounds, it must be the case that $|B| \geq (\frac{2}{k} - \eta)|K(A_{t+1})|$, and therefore that $|B| \geq (\frac{2}{k} - \eta)(|K(A)| - t \cdot k \cdot s_{\eta,k})$ which implies that $t \geq \frac{1}{k\, s_{\eta,k}}(|K(A)| - \frac{|B|}{\frac{2}{k} - \eta})$. ■

3.5.2 Adaptive Algorithm for k-Set Packing

In this section, we extend the ideas introduced earlier for obtaining the adaptive solution for matching, together with Lemma 3.5.2 and additional ingredients, to obtain the following result.

Theorem 3.5.3. *There exists an adaptive polynomial-time algorithm that, given a k-set instance (U, A) and $\epsilon > 0$, uses $O(1)$ rounds and $O(n)$ queries overall, and returns a set B_R whose expected cardinality is at least a $(1-\epsilon)\frac{2}{k}$ fraction of $\overline{K}(A)$.*

We introduce Algorithm 4 that is used to find a packing that approximates the omniscient optimum. In each round r, the algorithm maintains a feasible k-set packing B_r based on the k-sets that have been queried so far. It then computes a collection Q_r of several disjoint bounded sized augmenting structures to the current solution B_r, where the augmenting structures are composed of the sets that are not queried so far. It issues queries to these augmenting structures, and uses those that are found to exist for augmenting the current solution. The augmented solution is fed into the next round.

For any element $v \in U$, the number of sets that it belongs to and are also queried is at most R. This is because in each of the R rounds, the algorithm issues queries to *disjoint* augmenting structures, and each augmenting structure includes *at most one set per element*.

We introduce some notation that is used in the remainder of the proofs in this section. At the beginning of the r^{th} iteration of Algorithm 4, we know the results of the queries $\bigcup_{i=1}^{r-1} Q_i$. We define Z_r to be the expected cardinality of the instance (U, A) given the result of these queries. More formally, $Z_r = \overline{K}(A | \bigcup_{i=1}^{r-1} Q_i)$. We note that $Z_1 = \overline{K}(A)$.

For a given r, we use the notation $\mathbb{E}_{Q_r}[X]$ to denote the expected value of X where the expectation is taken over *only* the outcome of query Q_r, and fixing the outcomes on the results of queries $\bigcup_{i=1}^{r-1} Q_i$. Moreover, for a given r, we use $\mathbb{E}_{Q_r,\ldots,Q_R}[X]$ to denote the expected value of X with the expectation taken over the outcomes of queries $\bigcup_{i=r}^{R} Q_i$, and fixing an outcome on the results of queries $\bigcup_{i=1}^{r-1} Q_i$.

Input: A k-set instance (U, A), and $\epsilon > 0$.

Parameters: $\eta = \frac{\epsilon}{k}$ and $R = \frac{(\frac{2}{k} - \eta)\ k\ s_{\eta,k}}{p^{s_{\eta,k}}} \log(\frac{2}{\epsilon})$ (For a $(1-\epsilon)(\frac{2}{k})$-approximation)

Algorithm:

1. Initialize $r \leftarrow 1$, $B_0 \leftarrow \emptyset$ and $A_1 \leftarrow A$.
2. For $r = 1, \ldots, R$, do
   (a) Initialize B_r to B_{r-1}.
   (b) Let Q_r be the set of augmenting structures output by Algorithm 3 on the input of k-set instance (U, A_r), the collection B_r, and parameter $s_{\eta,k}$.
   (c) For each augmenting structure $(C, D) \in Q_r$.
      i. Query all elements in C.
      ii. If all elements of C exist, augment the current solution: $B_r \leftarrow (B_r \setminus D) \cup C$.
   (d) Set A_{r+1} to be A_r minus any queried sets that were found to not exist.
3. Return B_R.

Algorithm 4: ADAPTIVE ALGORITHM FOR STOCHASTIC k-SET PACKING

The next result, Lemma 3.5.4, proves a lower bound on the expected increase in the cardinality of B_r (the solution at round r) with respect to B_{r-1} (the solution in the previous round).

Lemma 3.5.4. *For every $r \in [R]$, it is the case that $\mathbb{E}_{Q_r}[|B_r|] \geq (1-\gamma)|B_{r-1}|+\gamma(\frac{2}{k}-\eta)Z_r$, where $\gamma = \frac{p^{s_{\eta,k}}}{(\frac{2}{k}-\eta)k\, s_{\eta,k}}$.*

Proof. By Lemma 3.5.2, Q_r is a collection of at least $\frac{1}{k\, s_{\eta,k}}(|K(A_r)| - \frac{|B_{r-1}|}{\frac{2}{k}-\eta})$ *disjoint $s_{\eta,k}$-size augmenting structures* (C, D) for B_{r-1}. Since in each augmenting structure (C, D), C has at most $s_{\eta,k}$ sets, on querying, the set C exists with probability at least $p^{s_{\eta,k}}$. Therefore, the expected increase in the size of the solution at Step 2c is:

$$\mathbb{E}_{Q_r}[|B_r|] - |B_{r-1}| \geq p^{k s_{\eta,k}}\, |Q_r| \geq \frac{p^{s_{\eta,k}}}{k\, s_{\eta,k}} \left(|K(A_r)| - \frac{|B_{r-1}|}{\frac{2}{k}-\eta} \right) \geq \gamma\big((\frac{2}{k}-\eta)\, |K(A_r)| - |B_{r-1}|\big).$$

Noting that $|K(A_r)| \geq Z_r$, we have our result. ■

Proof of Theorem 3.5.3. First, we make a technical observation about Z_r: For every $r \leq R$, $\mathbb{E}_{Q_{r-1}}[Z_r] = Z_{r-1}$. This is since

$$\mathbb{E}_{Q_{r-1}}[Z_r] = \mathbb{E}_{Q_{r-1}}[\overline{K}(A|\bigcup_{i=1}^{r-1} Q_i)] = \overline{K}(A|\bigcup_{i=1}^{r-2} Q_i) = Z_{r-1}. \tag{3.2}$$

Now, similar to the proof of Theorem 3.3.1, we first apply Lemma 3.5.4 to the R^{th} step and get $\mathbb{E}_{Q_R}[|B_R|] \geq (1-\gamma)|B_{R-1}| + \gamma(\frac{2}{k}-\eta)Z_R$. Next taking expectation on both sides with respect to Q_{R-1}, we get $\mathbb{E}_{Q_{R-1},Q_R}[|B_R|] \geq (1-\gamma)\mathbb{E}_{Q_{R-1}}[|B_{R-1}|] + \gamma(\frac{2}{k}-\eta)\mathbb{E}_{Q_{R-1}}[Z_R]$. Applying Lemma 3.5.4 to $\mathbb{E}_{Q_{R-1}}[|B_{R-1}|]$ and Equation (3.2) to $\mathbb{E}_{Q_{R-1}}[Z_R]$, we get

$$\begin{aligned}\mathbb{E}_{Q_{R-1},Q_R}[|B_R|] &\geq (1-\gamma)((1-\gamma)|B_{R-2}| + \gamma(\frac{2}{k}-\eta)Z_{R-1}) + \gamma(\frac{2}{k}-\eta)\, Z_{R-1} \\ &= (1-\gamma)^2|B_{R-2}| + \gamma(\frac{2}{k}-\eta)(1+(1-\gamma))\, Z_{R-1}.\end{aligned}$$

We can repeat the above steps, by sequentially taking expectation over Q_{R-2} through Q_1, and applying Lemma 3.5.4 and Equation (3.2) at each step, to achieve

$$\begin{aligned}\mathbb{E}_{Q_1,\ldots,Q_R}[|B_R|] &\geq (1-\gamma)^R|B_0| + \gamma(\frac{2}{k}-\eta)(1+(1-\gamma)+\cdots+(1-\gamma)^{R-1})\, Z_1 \\ &\geq (\frac{2}{k}-\eta)(1-(1-\gamma)^R)\, \overline{K}(A) \geq \frac{2}{k}(1-\frac{\eta k}{2})(1-e^{-\gamma R}).\, \overline{K}(A)\end{aligned}$$

We complete the claim by noting that

$$\frac{2}{k}(1-\frac{\eta k}{2})(1-e^{-\gamma R}) \geq \frac{2}{k}(1-\frac{\epsilon}{2})(1-\frac{\epsilon}{2}) \geq (1-\epsilon)\frac{2}{k},$$

where the penultimate inequality comes from the fact that $\eta = \epsilon/k$ and

$$R = \frac{(\frac{2}{k}-\eta)\ k\ s_{\eta,k}}{p^{s_{\eta,k}}} \log(\frac{2}{\epsilon}) = \frac{1}{\gamma}\log(\frac{2}{\epsilon}).$$

Therefore, the cardinality of B_R in expectation is at least a $(1-\epsilon)\frac{2}{k}\overline{K}(A)$. ■

3.5.3 Non-Adaptive Algorithm for k-Set Packing

Theorem 3.5.5. *There exists a non-adaptive polynomial-time algorithm that, given a k-set instance $(U, \mathcal{A})$ and $\epsilon > 0$, uses $O(n)$ queries overall and returns a k-set packing with expected cardinality $(1-\epsilon)\frac{(2/k)^2}{2/k+1}\overline{K}(\mathcal{A})$.*

To prove Theorem 3.5.5, we introduce a simple non-adaptive algorithm (Algorithm 5) that proceeds as follows. For R rounds, at every round, using the local improvement algorithm of Hurkens and Schrijver [1989], we find a $(\frac{2}{k}-\eta)$-approximate solution to the k-set instance and remove it from the instance. Then, we query every set that is included in these R solutions. We show that the expected cardinality of the maximum packing on the chosen sets is a $(1-\epsilon)\frac{(2/k)^2}{2/k+1}$ of the expected optimal packing.

It is simple to see that for any $v \in U$, there are at most R sets that include v and are queried by the algorithm. This is because the sets chosen at each round are *disjoint* since they form a feasible k-set packing solution.

Before proving Theorem 3.5.5, we present a technical claim.

Claim 3.5.6. *Let $\mathcal{A}_1 \subseteq \mathcal{A}$ and $\mathcal{A}_2 = \mathcal{A} \setminus \mathcal{A}_1$. Then $\overline{K}(\mathcal{A}) \leq \overline{K}(\mathcal{A}_1) + \overline{K}(\mathcal{A}_2)$.*

Proof. Let $\mathcal{A}'$ be any subset of $\mathcal{A}$, $\mathcal{A}'_1 = \mathcal{A}_1 \cap \mathcal{A}'$, and $\mathcal{A}'_2 = \mathcal{A}_2 \cap \mathcal{A}'$. Since the k-set packing of $\mathcal{A}'$ restricted to $\mathcal{A}'_1$ and $\mathcal{A}'_2$ are valid k-set packings for these subsets, hence $|K(\mathcal{A}')| \leq |K(\mathcal{A}'_1))| + |K(\mathcal{A}'_2)|$. For every $\mathcal{A}' \subseteq \mathcal{A}$, the above inequality holds. Expectation is a linear combination of the values of the outcomes, and so this inequality also holds in expectation. That is, $\overline{K}(\mathcal{A}) \leq \overline{K}(\mathcal{A}_1) + \overline{K}(\mathcal{A}_2)$. ■

Input: A k-set instance (U, A), and $\epsilon > 0$.

Parameters: $\eta = \frac{\epsilon}{2k}$ and $R = \frac{(\frac{2}{k}-\eta)\ k\ s_{\eta,k}}{p^{s_{\eta,k}}} \log(\frac{2}{\epsilon})$. (For $(1-\epsilon)\frac{(2/k)^2}{2/k+1}$-approximation)

Algorithm:

1. Let $B_0 \leftarrow \emptyset$.
2. For $r = 1, \ldots, R$, do
   (a) Assign O_r to a $(\frac{2}{k} - \eta)$-approximate solution to the k-set instance $(U, \mathcal{A} \setminus \bigcup_{i=1}^{r-1} B_i)$. ($O_r$ is found using the local improvement algorithm of Hurkens and Schrijver [1989].)
   (b) Set $B_r \leftarrow B_{r-1} \cup O_r$.
3. Query the sets in O_1, and assign Q_1 to be the sets that are found to exist.
4. For $r = 2, \cdots, R$, do
   (a) Find augmenting structures in O_r that augment Q_{r-1}. This is achieved by giving k-set instance $(U, Q_{r-1} \cup O_r)$ and solution Q_{r-1} as input (with parameter $s_{\eta,k}$) to Algorithm 3.
   (b) Query all the augmenting structures in O_r, and augment Q_{r-1} with the ones that are found to exist. Call the augmented solution Q_r.
5. Output Q_R.

Algorithm 5: NON-ADAPTIVE ALGORITHM FOR STOCHASTIC k-SET PACKING

Proof of Theorem 3.5.5. We claim that the expected cardinality of the k-set solution output by Algorithm 5 is at least $(1-\frac{\epsilon}{2})\frac{(\frac{2}{k}-\eta)^2}{1+\frac{2}{k}-\eta}\overline{K}(\mathcal{A})$. The claimed approximation will follow since $\eta = \frac{\epsilon}{2k}$.

For ease of exposition, let $\alpha = \frac{\frac{2}{k}-\eta}{1+\frac{2}{k}-\eta}$, and now note that $\frac{(\frac{2}{k}-\eta)^2}{1+\frac{2}{k}-\eta} = \alpha(\frac{2}{k}-\eta) = (1-\alpha)(\frac{2}{k}-\eta)^2$.

Assume that $\overline{K}(B_R) \leq \alpha \cdot \overline{K}(\mathcal{A})$ (else it will be immediately follow that the expected cardinality of the k-set solution output by the algorithm is at least $(\frac{2}{k}-\eta)\alpha\overline{K}(A)$ and this will complete the claim).

First, we make an observation. For each round $r \in [R]$, we have $\overline{K}(B_r) \leq \overline{K}(B_R) \leq \alpha\overline{K}(\mathcal{A})$. If we denote $A_r = A \setminus B_{r-1}$, then it follows that

$$|O_r| \geq (\frac{2}{k}-\eta)|K(\mathcal{A}_r)| \geq (\frac{2}{k}-\eta)\overline{K}(\mathcal{A}_r) \geq (\frac{2}{k}-\eta)(\overline{K}(\mathcal{A})-\overline{K}(B_{r-1})) \geq (\frac{2}{k}-\eta)(1-\alpha)\overline{K}(\mathcal{A})\,,$$

where the first inequality follows from the fact that O_r is $(\frac{2}{k}-\eta)$-approximation to A_r, and the second inequality follows from Claim 3.5.6.

We analyze the expected cardinality of the output solution Q_R by analyzing the R stages that the algorithm adopts at Steps 3 and 4 to create solution Q_R. For this analysis, we use the following notation: For a given r, we use the notation $\mathbb{E}_{O_r}[X]$ to denote the expected value of X where the expectation is taken over *only* the outcome of query O_r, and fixing the outcomes on the results of queries $\bigcup_{i=1}^{r-1} O_i$. Moreover, for a given r, we use $\mathbb{E}_{O_r,\ldots,O_R}[X]$ to denote the expected value of X with the expectation taken over the outcomes of queries $\bigcup_{i=r}^{R} O_i$, and fixing an outcome on the results of queries $\bigcup_{i=1}^{r-1} O_i$.

In the first stage, Q_1 is assigned to the collection of k-sets that are found to exist in O_1. In the second stage, we try to augment Q_1 by finding augmenting structures from O_2 and querying them. By Lemma 3.5.2, it finds at least $\frac{1}{ks_{\eta,k}}\left(|O_2| - \frac{|Q_1|}{\frac{2}{k}-\eta}\right)$ disjoint augmenting structures from O_2 that have size at most $s_{\eta,k}$ and augment Q_1. Since each augmenting structure exists independently with probability at least $p^{s_{\eta,k}}$, *in expectation over the outcomes of queries to* O_2, the size of Q_2, $\mathbb{E}_{O_2}[Q_2]$, is at least

$$\begin{aligned}|Q_1| + p^{s_{\eta,k}}\left(\frac{1}{ks_{\eta,k}}\big(|O_2| - \frac{|Q_1|}{\frac{2}{k}-\eta}\big)\right) =& \frac{p^{s_{\eta,k}}}{ks_{\eta,k}}\,|O_2| + (1 - \frac{p^{s_{\eta,k}}}{ks_{\eta,k}(\frac{2}{k}-\eta)})|Q_1| \\ \geq& \frac{p^{s_{\eta,k}}}{ks_{\eta,k}}\,(\frac{2}{k}-\eta)(1-\alpha)\overline{K}(\mathcal{A}) + (1 - \frac{p^{s_{\eta,k}}}{ks_{\eta,k}(\frac{2}{k}-\eta)})\,|Q_1|,\end{aligned}$$

and hence the expected size of Q_2 is at least $\beta\ \overline{K}(A) + (1-\gamma)|Q_1|$, where $\beta = \frac{p^{s_{\eta,k}}}{ks_{\eta,k}}(\frac{2}{k}-\eta)(1-\alpha)$ and $\gamma = \frac{p^{s_{\eta,k}}}{k\ s_{\eta,k}\ (\frac{2}{k}-\eta)}$.

For the third stage, a similar analysis shows that the expected size of Q_3, $\mathbb{E}_{O_3}[Q_3]$, *with expectation taken only over the outcomes of the queries to* O_3, is at least $\beta\ \overline{K}(A) + (1-\gamma)|Q_2|$. If we now, in addition, take expectation over the outcomes of queries to O_2, we get the expected size of Q_3, $\mathbb{E}_{O_2,O_3}[Q_3]$, is at least $\beta\ \overline{K}(A)+(1-\gamma)\ (\beta\ \overline{K}(A)+(1-\gamma)|Q_1|) = \beta(1+(1-\gamma))\ \overline{K}(A) + (1-\gamma)^2\ |Q_1|$.

Repeating the above steps, the procedure creates the k-set solution Q_R (from $O_1, \cdots, O_R$) whose expected size, with expectation taken over the outcomes of queries to O_2 through O_R, is at least

$$\beta(1+(1-\gamma)+\cdots+(1-\gamma)^{R-2})\overline{K}(A) + (1-\gamma)^{R-1}|Q_1|\ .$$

Finally, taking expectation over outcomes of queries to O_1, since the expected size of $|Q_1|$ is at least $p|O_1| \geq p\ (\frac{2}{k}-\eta)\overline{K}(A) \geq \beta\ \overline{K}(A)$, we have that the expected size of Q_R is at least

$$\begin{aligned}&\beta\ (1+(1-\gamma)+\cdots+(1-\gamma)^{R-1})\overline{K}(A)\\ &\quad= \frac{\beta}{\gamma}(1-(1-\gamma)^R)\overline{K}(A) \geq \frac{\beta}{\gamma}(1-e^{-\gamma R})\overline{K}(A) \geq (1-\frac{\epsilon}{2})\frac{(\frac{2}{k}-\eta)^2}{\frac{2}{k}-\eta+1}\overline{K}(\mathcal{A})\end{aligned}$$

■

3.6 Matching Under Correlated Edge Probabilities

In this section, we extend our framework to a more general setting. Here, the existence probability of an edge depends on parameters that are associated with the endpoints of the edge. Specifically, every vertex $v_i \in V$ is associated with parameter p_i, and an edge $e_{ij} = (v_i, v_j)$ exists with probability $p_i p_j$.

Importantly, this model is a generalization of the model studied above: we can still think of each edge $e \in E$ as existing with a given probability, and these events are *independent*. However, using vertex parameters gives us a formal framework for correlating the probabilities of edges incident to any particular vertex. The motivation for this comes from kidney exchange: Some *highly sensitized* patients are less likely than other patients to be compatible with potential donors. Such patients correspond to a small p_i parameter.

We consider two settings: adversarial and stochastic. In the adversarial setting, the vertex parameters p_i are selected by an adversary, whereas in the stochastic model, the parameters are drawn from a distribution. In the former setting, for $\delta > 0$, define f_δ to

be the *number of vertices* that have $p_i < \delta$. In the latter setting, for a distribution D and $\delta > 0$, let g_δ indicate the *probability* that a vertex has its parameter less than δ, i.e., $g_\delta = \Pr_{p_i \sim D}[p_i < \delta]$. We formulate our results in terms of δ, f_δ, and g_δ, and the desired value of δ can depend on the application. For example, in kidney exchange, δ would be the probability that a highly-sensitized patient is compatible with a random donor (a patient is typically considered to be highly sensitized when this probability is 0.2), and f_δ would be the number of highly-sensitized patients in the kidney exchange pool.

3.6.1 Adaptive Algorithm in Adversarial Setting

In this section, we consider the case where an adversary chooses the values of vertex parameters. We give guarantees on the performance of Algorithm 1 in this setting.

Theorem 3.6.1. *For any graph (V, E), any $\epsilon > 0$, and $\delta > 0$, Algorithm 1 returns a matching with expected size of $(1-\epsilon)(\overline{M}(E) - f_\delta)$ in $R = \frac{\log(2/\epsilon)}{\delta^{4/\epsilon}}$ iterations.*

The proof of this theorem and the subsequent lemmas are similar to the proofs of Section 3.3, and are included here for completeness. In the next lemma, $\mathbb{E}_{Q_r}[|M_r|]$ indicates the expected size of M_r, where the expectation is over the query outcome of Q_r. More formally, $\mathbb{E}_{Q_r}[|M_r|] = \overline{M}(\bigcup_{j=1}^{r} Q_j | \bigcup_{j=1}^{r-1} Q_j)$. We use Z_r to denote the expected size of the maximum matching in graph (V, E) given the results of the queries $\bigcup_{j=1}^{r-1} Q_j$. More formally, $Z_r = \overline{M}(E | \bigcup_{j=1}^{r-1} Q_i)$. Note that $Z_1 = \overline{M}(E)$.

Lemma 3.6.2. *For all $r \in [R]$ and odd L, $\mathbb{E}_{Q_r}[|M_r|] \geq (1-\gamma)|M_{r-1}| + \alpha(Z_r - f_\delta)$, where $\gamma = \delta^{L+1}(1 + \frac{2}{L+1})$ and $\alpha = \delta^{L+1}$.*

Proof. By Lemma 3.3.2, there exists $|O_r| - (1 + \frac{2}{L+1})|M_{r-1}|$ many augmenting paths in $O_r \Delta M_{r-1}$ that augment M_{r-1} and have length at most L. These augmenting paths are disjoint, so at most f_δ of them include a vertex v_i, with $p_i \leq \delta$. We will ignore these paths. Among the remaining augmenting paths, each path of length L, has at most $\frac{L+1}{2}$ edges that have not been queried yet. These edges do not share a vertex, so each one exists, independently of others, with probability at least δ^2. Therefore, the expected increase in the size of the matching from these augmenting paths is:

$$\mathbb{E}_{Q_r}[|M_r|] - |M_{r-1}| \geq \delta^{L+1}\left(|O_r| - (1 + \frac{2}{L+1})|M_{r-1}| - f_\delta\right) \geq \alpha(Z_r - f_\delta) - \gamma|M_{r-1}|.$$

where the last inequality holds by the fact that Z_r, which is the expected size of the optimal matching with expectation taken over the non-queried edges, cannot be larger than O_r, which is the maximum matching assuming that every non-queried edge exists. ■

Proof sketch of Theorem 3.6.1. Let $L = \frac{4}{\epsilon} - 1$. First note that for all r, it is true that

$$\mathbb{E}_{Q_{r-1}}[Z_r - f_\delta] = \mathbb{E}_{Q_{r-1}}[Z_r] - f_\delta = \mathbb{E}_{Q_{r-1}}\left[\overline{M}(E|\bigcup_{i=1}^{r-1} Q_i)\right] - f_\delta = \overline{M}(E|\bigcup_{i=1}^{r-2} Q_i) - f_\delta$$
$$= Z_{r-1} - f_\delta.$$

The remainder of the proof is similar to that of Theorem 3.3.1 with $Z_r - f_\delta$ replacing Z_r. Following similar analysis, we have

$$\mathbb{E}_{Q_1,\ldots,Q_R}[|M_R|] \geq \alpha \frac{1-(1-\gamma)^R}{\gamma}(\overline{M}(E) - f_\delta).$$

Since $R = \frac{\log(2/\epsilon)}{\delta^{4/\epsilon}}$, we have

$$\frac{\alpha}{\gamma}\big(1-(1-\gamma)^R\big) \geq (1 - \frac{2}{L+1})\big(1-(1-\gamma)^R\big) \geq (1-\frac{\epsilon}{2})(1-e^{-\gamma R}) \geq (1-\epsilon). \tag{3.3}$$

Therefore, Algorithm 1 returns a matching with expected size of $(1-\epsilon)(\overline{M}(E) - f_\delta)$. ■

3.6.2 Adaptive Algorithm in Stochastic Setting

In this section, we consider the case where the vertex parameters are drawn independently from a distribution.

Corollary 3.6.3. *Given any graph (V, E) with vertex parameters that are drawn from distribution D and any $\epsilon, \delta > 0$, Algorithm 1 returns a matching with expected size of $(1-\epsilon)(\overline{M}(E) - ng_\delta)$ in $R = \frac{\log(2/\epsilon)}{\delta^{4/\epsilon}}$ iterations.*

Proof. The result of Theorem 3.6.1 holds for any value of f_δ. Hence, on taking expectation over the value of f_δ, we have our result. ■

The next corollary shows the implication of Corollary 3.6.3 for the uniform distribution.

Corollary 3.6.4. *For a given graph (V, E) with vertex parameters that are drawn from the uniform distribution, and any $\epsilon > 0$, Algorithm 1 returns a matching with expected size of $(1-\epsilon)(\overline{M}(E) - \epsilon n)$ in $R = \frac{\log(2/\epsilon)}{\epsilon^{4/\epsilon}}$ iterations.*

Proof. This follows from Corollary 3.6.3 by setting $\delta = \epsilon$ and noting that $g_\epsilon = \epsilon$ for the uniform distribution. ■

3.6.3 Non-adaptive algorithm in Adversarial Setting

In this section, we consider the case where an adversary chooses the values of vertex parameters. We prove performance guarantees for Algorithm 2 in this adversarial setting.

Theorem 3.6.5. *Given a graph (V, E) with vertex parameters that are selected by an adversary, and any $\epsilon, \delta > 0$, Algorithm 2 returns a matching with expected size of $\frac{1}{2}(1-\epsilon)(\overline{M}(E) - f_\delta)$ in $R = \frac{\log(2/\epsilon)}{\delta^{4/\epsilon}}$ iterations.*

The proof of Theorem 3.6.5 and the subsequent lemma are similar to Section 3.4, and are included here for completeness.

Lemma 3.6.6. *For any iteration $r \in [R]$ of Algorithm 2 and odd L, if $\overline{M}(W_{r-1}) \leq \overline{M}(E)/2$, then $\overline{M}(W_r) \geq \frac{\alpha}{2}\ (\overline{M}(E) - f_\delta) + (1-\gamma)\overline{M}(W_{r-1})$, where $\alpha = \delta^{L+1}$ and $\gamma = \delta^{L+1}(1 + \frac{2}{L+1})$.*

Proof. Define $U = E \setminus W_{r-1}$. Assume that $\overline{M}(W_{r-1}) \leq \overline{M}(E)/2$. By Claim 3.4.2, we know that $\overline{M}(U) \geq \overline{M}(E) - \overline{M}(W_{r-1})$. Hence, $|O_r| = |M(U)| \geq \overline{M}(U) \geq \overline{M}(E) - \overline{M}(W_{r-1}) \geq \overline{M}(E)/2$.

Let W'_{r-1} represent one possible outcome of existing edges when edges are drawn from W_{r-1}. By Lemma 3.3.2, there are at least $|O_r| - (1 + \frac{2}{L+1})|M(W'_{r-1})|$ augmenting paths of length at most L in $O_r \Delta M(W'_{r-1})$ that augment $M(W'_{r-1})$. Among these paths, at most f_δ have a vertex v_i, with $p_i < \delta$. We ignore these paths. Each remaining path succeeds with probability $(\delta^2)^{(L+1)/2}$. Hence, the expected increase in the size of $|M(W'_{r-1})|$ using the remaining paths of length L is,

$$\begin{aligned}\overline{M}(O_r \cup W'_{r-1}|W'_{r-1}) - |M(W'_{r-1})| &\geq \delta^{L+1}\left(|O_r| - (1+\frac{2}{L+1})|M(W'_{r-1})| - f_\delta\right) \\ &\geq \delta^{L+1}\left(\frac{1}{2}\,\overline{M}(E) - (1+\frac{2}{L+1})|M(W'_{r-1})| - f_\delta\right).\end{aligned}$$

Re-arranging the inequality, we get $\overline{M}(O_r \cup W'_{r-1}|W'_{r-1}) \geq \frac{\alpha}{2}\ (\overline{M}(E) - f_\delta) + (1-\gamma)|M(W'_{r-1})|$. Taking expectation over the coin tosses on W_{r-1} that create outcome W'_{r-1}, we have

$$\overline{M}(W_r) \geq \mathbb{E}_{W_{r-1}}[\overline{M}(O_r \cup W'_{r-1}|W'_{r-1})] \geq \frac{\alpha}{2}\ (\overline{M}(E) - f_\delta) + (1-\gamma)\overline{M}(W_{r-1}).$$

■

Proof sketch of Theorem 3.6.5. Let $L = \frac{4}{\epsilon} - 1$. The proof is similar to that of Theorem 3.4.1 with the value of $\overline{M}(E)$ being replaced by $\overline{M}(E) - f_\delta$. Following a similar analysis, we get

$$\overline{M}(W_R) \geq \frac{\alpha}{2} \frac{(1-(1-\gamma)^R)}{\gamma} (\overline{M}(E) - f_\delta).$$

Now, $\frac{\alpha}{\gamma}(1-(1-\gamma)^R) \geq (1-\frac{2}{L+1})(1-e^{-\gamma R}) \geq (1-\epsilon)$ for $R = \frac{\log(2/\epsilon)}{\delta^{4/\epsilon}}$. Hence, Algorithm 2 returns a matching with expected size of $0.5(1-\epsilon)(\overline{M}(E) - f_\delta)$. ■

3.6.4 Non-adaptive algorithm in Stochastic Setting

We examine the performance of Algorithm 2 in the setting where the vertex parameters are chosen independently from a distribution.

Corollary 3.6.7. *Given a graph (V, E) with vertex parameters that are selected from distribution D, and $\epsilon, \delta > 0$, Algorithm 2 returns a matching with expected size of $\frac{1}{2}(1-\epsilon)(\overline{M}(E) - ng_\delta)$ with $R = \frac{\log(2/\epsilon)}{\delta^{4/\epsilon}}$ non-adaptive queries.*

Proof. The result of Theorem 3.6.5 holds for any value of f_δ. Hence, on taking expectation over the values of f_δ, we have our result. ■

Corollary 3.6.8. *For any $G = (V, E)$ with vertex parameters that are drawn from the uniform distribution, and any $\epsilon > 0$, Algorithm 2 returns a matching with expected size of $0.5(1-\epsilon)(\overline{M}(E) - n\epsilon)$ with $R = \frac{\log(2/\epsilon)}{\epsilon^{4/\epsilon}}$ non-adaptive queries.*

Proof. This follows from Corollary 3.6.7 by setting $\delta = \epsilon$ and noting that $g_\epsilon = \epsilon$ for the uniform distribution. ■

3.7 Computational complexity of budget-constrained non-adaptive solution

In previous sections, we aimed to find (either adaptively or non-adaptively) the subset of edges whose expected size is close to that of the omniscient optimal. While it is an open question whether we can find a $(1-\epsilon)$-approximate solution *non-adaptively*, in this section, we explore the aspect of non-adaptivity from a different direction. We constrain the subset

of edges to be such that each vertex has at most two edges incident to it (i.e., each resource is queried at most twice). We ask whether in polynomial time, we can find the subset of edges that observes this constraint and has the maximum expected size of matching. Note that here we no longer compare ourselves against the omniscient optimal. Rather, we fix a per-vertex budget of queries and ask for the optimal non-adaptive solution within that budget. Specifically, we fix our per-vertex budget to 2 and we call this NONADAPTIVE$_2$ problem. Our main result is that finding the optimal solution is NP-hard. This result, stated formally as Theorem 3.7.6, will be shown via a reduction from the 3D-MATCHING problem.

Unlike results in the previous sections, the results in this and the following sections are restricted to the case where the graph G has each non-zero $p_e = p$ for some constant p. To emphasize, we introduce subscript p in the notation $\overline{M}_p(H)$ that was used to denote the expected size of matching of the collection of edges H. Furthermore, we use NAOPT$_2(G)$ to denote the optimal solution to NONADAPTIVE$_2$, i.e., NAOPT$_2(G) =\triangleq \max_H \overline{M}_p(H)$, where the maximum is over all subsets of edges so that each vertex has maximum incident degree 2.

3.7.1 4-cycle cover is optimal

For the reduction from 3D-MATCHING problem, we will need the following crucial result, which states that if the graph has a 4-cycle cover[5], then the 4-cycle cover is the *unique* optimal subgraph.

Theorem 3.7.1. *For any $0 < p < 1$, if the graph G admits a 4-cycle cover, then every optimal H is a 4-cycle cover of G.*

Before we prove Theorem 3.7.1, we make a couple of easy observations.

Observation 3.7.2. *Due to the degree constraints $\delta_{E'}(v) \leq 2$, the subgraph H is a collection of disjoint cycles and paths, and maybe isolated vertices.*

Observation 3.7.3. *A cycle of length $l + 1$ has higher expected size of matching than a path of length l (the length of a path or cycle is the number of* edges *in it).*

Corollary 3.7.4. *If in $H(V, E')$, there exists a path P, whose end points share an edge in $G(V, E)$, then adding that edge to E' does not reduce the size of the expected matching in H.*

[5] A 4-cycle cover is a collection of cycles each of length 4, such that every vertex lies in exactly one cycle.

In order to prove the theorem, we rely on the following crucial lemma. We note that Lemma 3.7.5 holds for *any* non-trivial value of p (i.e., $p \notin \{0, 1\}$).

Lemma 3.7.5. *For any $0 < p < 1$, a 4-cycle has strictly higher expected probability of a vertex being matched than a cycle or a path of any other length.*

Proof. By Observation 3.7.3, it suffices to show that in a 4-cycle, the average probability of a vertex being matched is strictly higher than that in any other cycle C. Each edge on this cycle exists independently with probability p. Let C_p be the space of outcomes of the edges. Since all edges on a cycle have the same probability of existence p, each vertex in the cycle has the same probability of being matched. We note that — in our analysis — to ensure that each vertex has the same probability of being matched, whenever there is more than one possible maximum matching in an instantiation in C_p, we choose each of the possible maximum matchings with equal probability.

Consider a vertex $v \in C$. Let us calculate the probability that v is matched by breaking up the outcome space into four cases.

1. Both edges incident to v exist. In this case v is definitely matched if $|C|$ is even (as is the case with a 4-cycle). For odd length cycles, v is matched with probability strictly less than one. This event occurs with probability p^2.

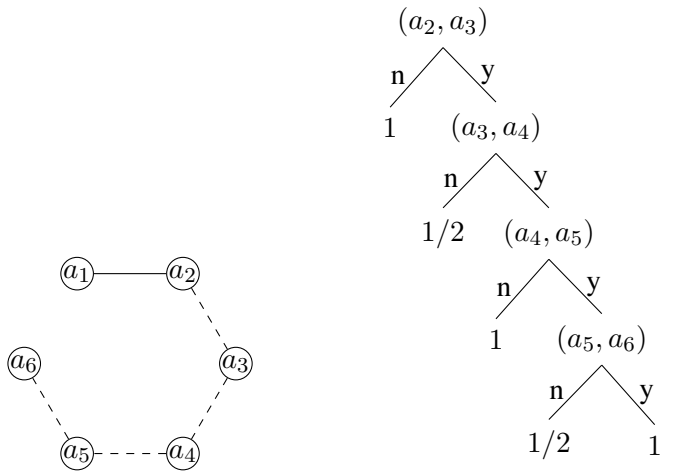
2. Both edges do not exist. In this case v is definitely not matched, and this occurs with probability $(1-p)^2$.

3. One of the edges incident to v exists and other does not. Each of these two cases occurs with probability $p(1-p)$.

To calculate the probability that v is matched in the third case, let us look at Figure 3.3(a) where $v = a_1$ and $n = 6$. The edge (a_1, a_6) is is absent, while the edge (a_1, a_2) is present. Clearly it holds that

$$\begin{aligned}\Pr[a_1 \text{ matched} | \nexists(a_n, a_1), \exists(a_1 a_2)] = (1-p)\cdot 1 + p(1-p)\cdot\frac{1}{2} + p^2(1-p)\cdot 1 \\ + p^3(1-p)\cdot\frac{1}{2} + \cdots + p^{n-3}(1-p)\cdot f(n) + p^{n-2}\cdot g(n),\end{aligned} \tag{3.4}$$

where $f(n)$ is 1 if n is odd and $\frac{1}{2}$ if n is even, and $g(n)$ is the opposite. In Equation (3.4) we have used the observation that if the path starting at a_1 is of even length then a_1 is



(a) Dashed edges may not be available.

(b) Probability that vertex v is matched.

Figure 3.3: The proof of Lemma 3.7.5 illustrated for the case of $n = 6$.

matched with probability $\frac{1}{2}$, and if it is of odd length then it is matched with probability 1. 1's and $\frac{1}{2}$'s alternate in the above expression; see Figure 3.3(b) for an illustration. For the case of a 4-cycle, the expression in Equation (3.4) is equal to $(1-p)\cdot 1+p(1-p)\cdot\frac{1}{2}+p^2\cdot 1$. For any cycle of length greater than 4, the expression is strictly smaller, because

$$\begin{aligned}
&(1-p)\cdot 1 + p(1-p)\frac{1}{2} + p^2(1-p)\cdot 1 + p^3(1-p)\frac{1}{2} + \cdots + p^{n-3}(1-p)\cdot f(n) + p^{n-2}\cdot g(n)\\
&< (1-p)\cdot 1 + p(1-p)\cdot\frac{1}{2} + p^2(1-p)\cdot 1 + p^3(1-p)\cdot 1 + \cdots + p^{n-3}(1-p)\cdot 1 + p^{n-2}\cdot 1\\
&= (1-p)\cdot 1 + p(1-p)\cdot\frac{1}{2} + p^2\cdot 1,
\end{aligned}$$

where the inequality is obtained by replacing all the $\frac{1}{2}$'s starting from the fourth term by 1's.

It follows that the expected probability that a vertex is matched is strictly higher in a 4-cycle than in a cycle of length greater than 4. The only other cycle length left to consider is 3. A similar analysis shows that the expected probability that a vertex is matched in a cycle of length 3 is

$$(1-p)^2\cdot 0 + p^2(p\cdot\frac{1}{2} + (1-p)\cdot 1) + 2\cdot p(1-p)\cdot((1-p)\cdot 1 + p\cdot\frac{1}{2}),$$

while for a 4-cycle the expression is

$$(1-p)^2 \cdot 0 + p^2 \cdot 1 + 2 \cdot p(1-p) \cdot ((1-p) \cdot 1 + p(1-p) \cdot \frac{1}{2} + p^2 \cdot 1) \, .$$

It is easy to verify that the 4-cycle expression is strictly greater than the 3-cycle expression for all $0 < p < 1$. ■

Lemma 3.7.5 is one of the main building blocks for our subsequent algorithmic results. We will use it here to establish Theorem 3.7.1 and that in turn can be applied to establish the computational hardness of our NONADAPTIVE$_2$ problem.

Proof of Theorem 3.7.1. Consider a graph G that admits a 4-cycle cover H, and consider any other subgraph H' with maximum degree 2. Recall that $\overline{M}_p(H)$, the expected size of the matching within subgraph H, is half the sum of the probabilities of the vertices being matched. By Observation 3.7.2, H' is also a collection of cycles and paths. And by Lemma 3.7.5, the average probability of a vertex being matched is highest in a 4-cycle. Hence if H' has any cycle of length other than 4 or a path, we have $\overline{M}_p(H') < \overline{M}_p(H)$. This completes the proof. ■

3.7.2 Hardness result

Theorem 3.7.6. NONADAPTIVE$_2$ *is NP-complete.*

Theorem 3.7.1 states that if a graph G admits a 4-cycle cover, then the optimal solution to NONADAPTIVE$_2$ for G is always a 4-cycle cover, i.e., any other collection of edges E' would yield a strictly smaller expected number of swaps. In other words, there is a collection of edges E' that has the same value as a 4-cycle cover of n vertices (a value that is easy to compute) if and only if a 4-cycle cover exists. Theorem 3.7.6 therefore follows directly from the following lemma that states that finding whether or not a 4-cycle cover exists is NP-hard. The proof of the lemma is similar to the proof that a cover by cycles of length *at most* l for $l \geq 3$ is NP-hard [Abraham et al., 2007, Theorem 1].

Lemma 3.7.7. *Deciding whether a graph G admits a cover by 4-cycles is an NP-complete problem.*

Proof. We reduce the 3D-MATCHING problem to the problem of finding whether a graph admits a 4-cycle cover. In 3D-MATCHING there are three vertex sets X, Y and Z, such that $|X| = |Y| = |Z|$. In addition, we are given a set S of 3-tuples of the form (x, y, z)

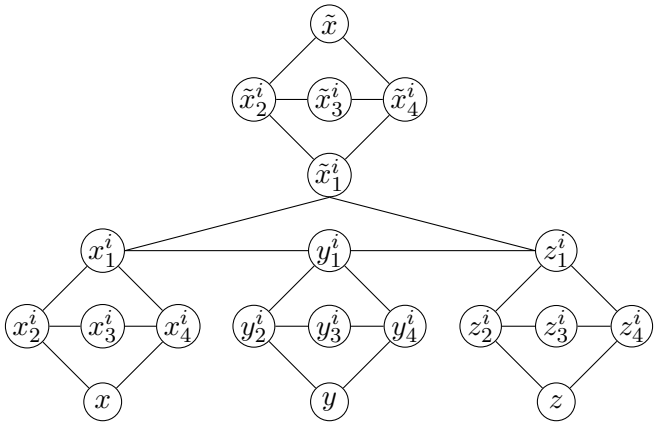


Figure 3.4: The gadget used in the proof of Lemma 3.7.7.

where $x \in X$, $y \in Y$ and $z \in Z$. The problem is to decide whether there exists a subset $S' \subseteq S$, such that $|S'| = |X| = |Y| = |Z|$ and no two tuples in S' share a vertex in either X or Y or Z. The set S' encodes a perfect matching — every $x \in X$ is matched to a unique $y \in Y$ and $z \in Z$.

For the reduction, we add a new set of vertices $\tilde{X} = \{\tilde{x} \mid x \in X\}$. And for each $v \in X \cup Y \cup Z$, for every tuple t_i in the 3D-MATCHING instance that v occurs in, we add the set of vertices $\{v_1^i, v_2^i, v_3^i, v_4^i\}$. Furthermore, if $v \in X$, we also add the set $\{\tilde{v}_1^i, \tilde{v}_2^i, \tilde{v}_3^i, \tilde{v}_4^i\}$. On all these vertices, we construct a graph G, where for every tuple $t_i = (x, y, z)$ in S, we introduce the gadget shown in Figure 3.4. Note that the vertices with superscript i only appear in a single gadget. The vertices $x, \tilde{x}, y, z$ can appear in multiple gadgets, and moreover $\tilde{x}$ appears in each gadget that contains x. The intuition is that x is covered if and only if $\tilde{x}$ is covered.

We claim that graph G has a 4-cycle cover if and only if the corresponding 3D-MATCHING problem has a perfect matching. First, if the 3D-MATCHING problem allows a perfect matching, then graph G has a cover through 4-cycles. Indeed, for every tuple $t_i = (x, y, z) \in S'$, we completely cover the corresponding gadget with 4-cycles using only the gadget's vertices (there is only one such cover). For all tuples $t_i = (x, y, z) \in S \setminus S'$, we cover all the vertices except $x, \tilde{x}, y, z$ with 4-cycles using the gadget's vertices. It is easy to verify that this is a complete cover by 4-cycles.

In the other direction, if the graph G has a cover via 4-cycles then the 3D-MATCHING problem admits a perfect matching. The first observation we make is that in a 4-cycle cover for G, for every $x \in X$, the 4-cycle which covers x has to be of the form (x, x_2^i, x_3^i, x_4^i), because the only other possible 4-cycle is (x, x_2^i, x_1^i, x_4^i) but now x_3^i cannot be covered. In

addition, once for a particular i the 4-cycle (x, x_2^i, x_3^i, x_4^i) is included the corresponding x_1^i can only be covered through the 4-cycle $(x_1^i, y_1^i, z_1^i, \tilde{x}_1^i)$. This in turn implies that we must completely cover the gadget using only the gadget's vertices. For every i such that $(x_1^i, y_1^i, z_1^i, \tilde{x}_1^i)$ is included in the cover for graph G, the tuple (x, y, z) is included in the set S'. It is easy to verify that S' encodes a solution to the 3D-MATCHING problem. ■

3.8 Almost optimal budget-constrained non-adaptive solution for Kidney Exchange Graphs

Our main result in this section is Theorem 3.8.20, stated informally as follows.

Theorem 3.8.1 (informal)**.** *For most realistic kidney exchange graphs G, we can algorithmically find in polynomial time an almost optimal subgraph H for* NONADAPTIVE$_2$ *problem.*

Note that we defined the NONADAPTIVE$_2$ problem (and the notation NAOPT$_2(G)$) in Section 3.7. To prove this result, we have to go through a number of steps. First, in Sections 3.8.2 and 3.8.3, we prove structural results for the optimal solution in bipartite and general graphs. Then in two steps – Sections 3.8.4 and 3.8.5, we define formally a realistic kidney exchange graph, and prove the result stated informally above.

3.8.1 Background

People who suffer from chronic kidney disease are best treated by transplanting a healthy kidney from a live donor. However, even patients who are fortunate enough to have a willing donor (typically a family member or a close friend) may be incompatible with him. This is where the recent innovation of *kidney exchange* comes in. The basic insight that drives kidney exchange is that two incompatible donor-patient pairs may be able to exchange kidneys so that both patients receive a healthy kidney. To pinpoint as many of these life-saving opportunities as possible, matching algorithms are run (on a weekly or monthly basis) on databases that contain the information of registered of donors and patients.

There are three hurdles that must be cleared before a donation can take place. First, the donor and patient must pass a *blood typing* test. There are four blood types (O, A, B, AB) – depending on the presence of A and B antigens — and only some are compatible with others. For example, a donor with blood type A can donate to a patient with blood

type A or AB, but not to a patient with blood type B or O. Second, the donor and patient must pass a *tissue typing* test. There are six tissue antigens; the more of them are shared by the patient and donor, the more likely it is that the transplant will be successful. Third, a *crossmatch* test is performed by (roughly speaking) mixing the donor and patient's blood in a tube and spinning it; depending on whether the blood is suspended or stuck together, doctors can predict whether the patient's body would attack the new kidney (confusingly called *positive crossmatch*) or would accept it (*negative crossmatch*).

The blood and tissue typing tests are fundamentally different from the crossmatch test, in that the relevant information can be collected from each donor and each patient even before matches are made. In contrast, for a crossmatch test (samples of) the blood of the patient and his intended donor must be physically in the same place. Therefore, existing kidney exchanges such as the one run by the United Network for Organ Sharing (UNOS) first compute a matching based only on blood typing and tissue typing tests. Then, crossmatches are performed only for patients and donors that were matched. Exchanges where all the relevant crossmatches are negative proceed to the operating room, while exchanges that involved a positive crossmatch fail.

In graph-theoretic terms, each incompatible donor-patient pair is represented by a vertex. We consider the undirected case where there is an edge between two vertices if each donor is compatible with the other patient *in terms of blood type and tissue type only*, that is, a pairwise exchange is potentially possible if the crossmatch test is negative[6]. Given a matching on this graph, a crossmatch is performed for each edge in the matching[7]; we model this as flipping an independent coin with some bias p for each edge to determine whether the edge succeeds or fails.

3.8.2 Complete Graphs and Bipartite Graphs

Our next goal is to characterize optimal solutions for complete graphs and complete bipartite graphs. We start with the complete graph. The following is an immediate corollary of Theorem 3.7.1.

Corollary 3.8.2. *Consider a complete graph $G(V, E)$, i.e., $E = \{(u, v) : u \neq v, u, v \in V\}$, such that $|V|$ is divisible by 4. Then the optimal subgraph H of G is composed of $|V|/4$ vertex disjoint 4-cycles.*

[6]In practice, kidney exchanges also use directed 3-cycles.

[7]By 'performing a crossmatch test on an edge', we really mean 'performing a *pair* of crossmatch tests' on that edge, one for each direction. However, we do not make this explicit, since we focus only on pairwise exchanges.

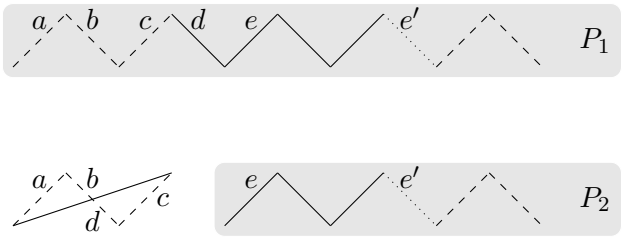


Figure 3.5: The proof of Claim 3.8.4 illustrated for the case of $l = 10$. Solid edges exist, dotted edges do not exist, and dashed edges may or may not exist.

We now move to complete bipartite graphs. Note that since a bipartite graph may not admit a 4-cycle cover, the results above do not imply what an optimal subgraph for a bipartite graph looks like. The main result for this section is the following.

Lemma 3.8.3. *For a complete bipartite graph $G(L \cup R, L \times R)$ with $|L| \leq |R| \leq 2|L|$, there exists an optimal subgraph H consisting only of 4-cycles, paths of length 2, and at most one path of length 4 or a single edge.*

As opposed to complete graphs where a cover by 4-cycles is uniquely optimal if one exists, here we do not claim uniqueness for this optimal subgraph. For our purposes, the aspect of the lemma which will prove crucial later is that only "small" structures are required. To prove this lemma, we first show that we do not lose anything by restricting our attention to "short" paths.

Claim 3.8.4. *For any $l \geq 6$, the expected size of the matching under a 4-cycle plus a path of length $l - 4$ is at least the expected size of the matching under a path of length l.*

Proof. Let P_1 be a path of length $l \geq 6$, and call its first four edges from the left a, b, c, d, e. Now we use the first four vertices to close a cycle C_4, and we also call its edges a, b, c, d; the remaining path of length $l - 4 \geq 2$, which starts with e, is denoted by P_2. We would like to claim that $\overline{M}_p(P_1) \leq \overline{M}_p(C_4 + P_2)$; recall that $\overline{M}_p(H)$ was the expected size of the maximum matching in the graph H. We first make the observation that in any instantiation of the edges, if the edge d is absent then the two structures have maximum matchings of the same size, so we need only consider outcomes where edge d is present.

Consider such an instantiation of the edges (with edge d present) and let edge e' be the first edge in P_2 to the right of edge s that fails; if there is no such edge e', let $e' = \perp$, i.e., null. See Figure 3.5 for an illustration. To the right of e', both paths P_1 and P_2 have the same maximum matchings, so we need only look to the left of e'. Denote the path

segments of P_1 and P_2 to the left of e' as P_1' and P_2' respectively. We want to show that $\overline{M}_p(P_1') \leq \overline{M}_p(C_4 + P_2')$.

We now look at all possible outcomes of edges a, b and c, and the length of the path P_2', and tabulate our observations in Table I; they are easy to verify. Here we use $M(H)$ to denote a maximum matching within the subgraph G. Since $\overline{M}_p(C_4 + P_2') = \mathbb{E}[|M(C_4 + P_2')|]$ and $\overline{M}_p(P_1') = \mathbb{E}[|M(P_1')|]$, we can use the table to get

$$\begin{aligned}
&\overline{M}_p(C_4 + P_2') - \overline{M}_p(P_1') = \mathbb{E}[|M(C_4 + P_2')|] - \mathbb{E}[|M(P_1')|] \\
&= p\big[((1-p)^3 + (1-p)^2 p + (1-p)p^2 + p^2(1-p))\Pr(|P_2'| \text{ odd}) - p(1-p)^2 \Pr(|P_2'| \text{ even})\big] \\
&= p(1-p) \cdot \big[(1 - p + 2p^2) \cdot \Pr(|P_2'| \text{ odd}) - p(1-p) \cdot \Pr(|P_2'| \text{ even})\big],
\end{aligned}$$

where in the first equality the leftmost factor of p stands for the probability of edge d being present, the term $((1-p)^3 + (1-p)^2 p + (1-p)p^2 + p^2(1-p))$ sums up the probabilities of the outcomes of edges a, b, c where $C_4 + P_2'$ has one more matched edge than P_1', and the term $p(1-p)^2$ is for the single case where P_1' has one more matched edge than $C_4 + P_2'$. Now, with $l' = l - 4$,

1. l is odd: $\Pr(|P_2'| \text{ odd}) = (1-p) \cdot \big(p + p^3 + p^5 + \cdots + p^{l'-2}\big) + p^{l'}$ and $\Pr(|P_2'| \text{ even}) = (1-p) \cdot (1 + p^2 + p^4 + \cdots + p^{l'-1})$. So, we have $\Pr(|P_2'| \text{ odd}) \geq p \cdot \Pr(|P_2'| \text{ even})$. And hence,

$$(1 - p + 2p^2) \cdot \Pr(|P_2'| \text{ odd}) \geq p(1-p) \cdot \Pr(|P_2'| \text{ even}) \,.$$

2. l is even: $\Pr(|P_2'| \text{ odd}) = (1-p) \cdot \big(p + p^3 + p^5 + \cdots + p^{l'-3} + p^{l'-1}\big)$ and $\Pr(|P_2'| \text{ even}) = (1-p) \cdot (1 + p^2 + p^4 + \cdots + p^{l'-2}) + p^{l'}$. So, we have $\Pr(|P_2'| \text{ odd}) = p \cdot \Pr(|P_2'| \text{ even}) -$

a	b	c	$\lvert M(4C + P_2')\rvert - \lvert M(P_1')\rvert$		a	b	c	$\lvert M(4C + P_2')\rvert - \lvert M(P_1')\rvert$	
			Even	Odd				Even	Odd
0	0	0	0	+1	1	0	0	-1	0
0	0	1	0	0	1	0	1	0	0
0	1	0	0	+1	1	1	0	0	+1
0	1	1	0	+1	1	1	1	0	0

Table 3.1: The table shows the difference in the size of matching between 4-cycle plus path P_2', and path P_1' for various possibilities of edge outcomes of a, b, c and whether $|P_2'|$ is even or odd. Edge d exists in all cases. An edge exists (resp., does not exist) if its column shows 1 (resp., 0).

$p^{l'+1}$. And hence,

$$(1-p+2p^2)\cdot\Pr(|P_2'| \text{ odd})-p(1-p)\cdot\Pr(|P_2'| \text{ even}) = 2p^3\cdot\Pr(|P_2'| \text{ even})-(1-p+2p^2)\cdot p^{l'+1}\ .$$

But, $\Pr(|P_2'| \text{ even})$ is at least $(1-p)+p^{l'}$ from the first expression that we wrote for that quantity. It follows that $2p^3 \cdot \Pr(|P_2'| \text{ even}) \geq (1-p+2p^2) \cdot p^{l'+1}$ since $l' \geq 2$ as $l \geq 6$.

Hence, in both cases, $\overline{M}_p(C_4 + P_2') - \overline{M}_p(P_1') \geq 0$, and the lemma follows. ■

Claim 3.8.4 implies that in a complete bipartite graph, paths of length at least six are useless. Next we compare 4-cycles and short paths.

Claim 3.8.5. *1. For any even $l \geq 4$ and any $p \in (0,1)$, the probability of a vertex being matched in a cycle of length l is strictly more than that in a cycle of length $l+2$.*

2. For any $p \in (0,1)$, the expected number of matched edges in a 4-cycle plus an edge is strictly more than the expected number of matched edges in a cycle of length 6.

3. The expected number of matched edges in a 4-cycle plus two paths of length 2 is equal to the expected number of matched edges in two paths of length 4.

Proof. 1. The proof of Lemma 3.7.5 shows not only 4-cycle has the highest expected number of matched edges, but that among even length cycles, a node has strictly higher expected probability of being matched in cycles of length l than in $l+2$ for all even $l \geq 4$.

2. The expected number of matched edges in a 4-cycle is $2p^2 + 2p(1-p)^2 + 2p(1-p)(1+p^2)$, for an edge is $1-(1-p)^2$ and for a cycle of length 6 is $6p(1-p)^2(1+p/2+p^2+p^3/2+p^4)+3p^2$. We can verify that the sum of the first two expressions strictly dominates the third for all $p \in (0,1)$.

3. For a 4-cycle, the expected number of matched edges is $2p^2 + 2p(1-p)^2 + 2p(1-p)(1+p^2)$; for a path of length 2, the expected number of matched edges is $1-(1-p)^2$ and for the case of a path of length 4, the expected number of matched edges is $2p^2 + 2p(1-p)^2 + p(1-p)^2 + p(1-p)(1+p^2)$.

   Summing the first expression with twice the second, and simplifying shows that it is equal to twice the third.

■

We now present the proof of Lemma 3.8.3.

Proof of Lemma 3.8.3. Consider an optimal choice of edges O for the complete bipartite graph G. If O contains a path of odd length ≥ 3, we can increase the quality of the solution by adding an edge between the end points to get an cycle. Also, cycles of odd length are impossible. Hence an optimal solution can contain only cycles of even length, edges, and paths of even length. Using the first two parts of Claim 3.8.5 we can assume that all cycles are of length 4. Next, by repeated application of Claim 3.8.4 to the even-length paths, we can convert O to a solution O' that is at least as good, but where we only have 4-cycles and paths of length 1, 2 and 4. In case, there are multiple paths of length 1 at this stage, they can be paired and missing edges added to give 4-cycles, thus giving multiple 4-cycles plus at most one path of length 1, and clearly keeping the expected matching size at least as much as before.

If at this stage there is more than one path of length 4 in O', we can use part 3 of Claim 3.8.5 to further prune these paths and replace them with 4-cycles and paths of length 2. At this point, we claim we cannot have both a path of length 4 and a path of length 1. If we did, this pair of structures would be worse than a path of length 6, which by Claim 3.8.4 would be worse than a 4-cycle plus a path of length 2. Hence we have only 4-cycles, paths of length 2, and either a single edge, or a path of length 4. ■

3.8.3 General Graphs

Having discussed the case of complete graphs and bipartite graphs (Section 3.8.2), we now move our attention to general graphs. The following lemma states that if there exists a vertex u which does not have any edge incident to it in the subgraph H, but which has an edge incident to it in the original graph G, then that edge can be included in the subgraph H (perhaps requiring some other edge in H to be deleted), without decreasing the expected size of matching of H.

Lemma 3.8.6. *(No vertex left behind.) Consider an undirected graph $G(V, E)$, and a subgraph $H(V, E')$ ($E' \subseteq E$) with $\delta_{E'}(v) \leq 2$. Suppose there exists a vertex $u \in V$ with $\delta_{E'}(u) = 0$ but $\delta_E(u) > 0$. Let v be a vertex which has an edge with u in E. Then we can add the edge (u, v) to E', and if needed, remove some other edge incident to v under E' in order to ensure $\delta_{E'}(v) \leq 2$, without reducing the expected size of matching of E'.*

From Lemma 3.8.6, we can infer the following result.

Corollary 3.8.7. *There exists an optimal solution $H(V, E')$ for the subgraph of $G(V, E)$ with the following property. For every vertex u that has $\delta_{E'}(u) = 0$,*

1. *either* $\delta_E(u) = 0$*, or*

2. *for every edge* (u, v) *present in* E*,* $\delta_{E'}(v) = 2$*, and if* b *and* d *are the two vertices adjacent to* v *under* E'*, then* $\delta_{E'}(b) = 1 = \delta_{E'}(d)$*.*

To prove the corollary, assume to the contrary that $\delta_{E'}(u) = 0$ but $\delta_E(u) > 0$ with $(u, v) \in E$ (and E' has the least number of such "violating" vertices among all those edge sets with the same expected matching size). Then if $\delta_{E'}(v) < 2$, we can add (u, v) to E'. Else if $\delta_{E'}(v) = 2$, and $\delta_{E'}(b) = 2$, say, then by Lemma 3.8.6 we can add edge (u, v) and drop (v, b) to get the set E'' with $\overline{M}_p(E'') \geq \overline{M}_p(E')$. Now E'' has fewer vertices that violate the property, which gives the desired contradiction. We now give the proof of Lemma 3.8.6.

Proof of Lemma 3.8.6. If $\delta_{E'}(v) < 2$, then we can add the edge (u, v) without removing any edge. Adding an edge cannot decrease the expected size of matching.

Hence, let us consider the case where $\delta_{E'}(v) = 2$, and say, the vertices to which v has an edge are b and d. If either d or b, has exactly one edge incident to it in E', say it is d, then we can drop the edge (v, d) in E' and add the edge (v, u). This does not change the expected size of matching since up to renaming, nothing has changed in the graph (u and d have exactly the same set of characteristics).

Therefore, we are left with the case where $\delta_{E'}(d) = 2 = \delta_{E'}(b)$. Consider E'', which the same as E' except that we drop the edge (v, d) and add the edge (v, u). We would like to show that the expected size of matching in E'' is at least as much as in E'.

In order to show that the expected size of matching in E'' is at least as much as in E', we shall partition the sample space of outcomes as follows:

1. Both (v, d) and (v, u) are present: Consider any outcome ω of edges in E where both (v, d) and (v, u) exist. Consider a maximum matching M for E' in ω. If M matches v to b, then M is also *a matching* in E'' since $E' \Delta E'' = \{(v, u), (v, d)\}$ and both don't exist in M. Hence the cardinality of the maximum matching in E'' is at least $|M|$.

   If M matches v to d, then consider matching M' for E'' which is exactly the same as M, but that it matches v to u (and not v to d). Again, $|M'| = |M|$, and hence the cardinality of the maximum matching in E'' is at least $|M'| = |M|$.

2. Both (v, d) and (v, u) are absent: Consider any outcome ω of edges in E where both (v, d) and (v, u) exist. Consider a maximum matching M for E' in ω. M is also *a*

matching in E'' since $E' \Delta E'' = \{(v,u),(v,d)\}$ and both don't exist in M. Hence the cardinality of the maximum matching in E'' is at least $|M|$.

3. Exactly one of (v,d) and (v,u) is present: Clearly for any outcomes of edges, the size of maximum matching in E' and E'' can differ by at most one. For a particular outcome of edges ω, denote the size of maximum matching for E' and E'' by $\phi(E',\omega)$ and $\phi(E'',\omega)$ respectively.

   We shall partition the sub-sample space (where exactly one of the two edges is present) into two halves. In one half, edge (v,d) would be present and (v,u) absent, and in the other half, the opposite would be true. Furthermore, we shall have a one-to-one mapping from points in the first half to that in the second half. The two points that are mapped to each other shall carry the same probability. In addition, we shall have the property that if for a particular point ω in say, the first half, $\phi(E',\omega) - \phi(E'',\omega) = 1$, then for the sample point ω' in the other half that ω maps to, $\phi(E',\omega') - \phi(E'',\omega') = -1$. Hence, in expectation over this sub-sample space, the size of matching matching of E' will be no more than that of E''.

   We now show the construction of the two halves and the mapping between them. Fix the outcome ω' of all edges in E but for (v,d) and (v,u). Let ω_u be ω' with (v,u) present and (v,d) absent, and let ω_d be ω' with (v,d) present and (v,u) absent. Consider the set A of points ω_d that we generate while enumerating over ω'. Similarly, consider the set B, that consists of points ω_u, again while enumerating over all possible ω'. It is easy to see that A and B are disjoint, and that their union captures the sub-sample space where exactly one of (v,d) and (v,u) is present. Also, again it is easy to verify that $|A| = |B|$. A and B shall constitute our two halves.

   We now explain the mapping from A to B. It will be the natural mapping, where $\omega_1 \in A$ and $\omega_2 \in B$ are mapped to each other, in case the outcome of all edges but for (v,d) and (v,u) is the same in ω_1 and ω_2. It is easy to see that this is a well defined one-to-one map and that both points that are mapped to each other carry the same probability weight.

   Consider a $\omega_d \in A$ and $\omega_u \in B$ that are mapped to each other. Consider the set S of all maximum matchings for E' in outcome space ω_d.

   (a) Either there exists a maximum matching M in set S that does not use the edge (v,d).

   In this case, $\phi(E'',\omega_d) \geq \phi(E',\omega_d) = |M|$ because M is also *a matching* in E'' since it does not use the edge (v,d) which is the only edge in $E'' \setminus E'$.

Moreover, $\phi(E'', \omega_u) \geq \phi(E', \omega_u)$ since under outcome ω_u, edge (v, d) is absent and hence, the maximum matching M for E' under ω_u shall also be a matching in E'' under ω_u.

(b) Or every maximum matching in S uses the edge (v, d).

As we have claimed earlier, that since $|E'' \setminus E'| = 1$, hence $\phi(E', \omega_d) - \phi(E'', \omega_d) \leq 1$.

Moreover, we have that $\phi(E', \omega_u) - \phi(E'', \omega_u) \leq -1$. Why is this the case? Well, consider any maximum matching M for E' under ω_d. We know that M uses edge (v, d). Construct M' which has all the edges as in M but has edge (v, u) replacing (v, d). M' is a legal matching for E'' under ω_u. Hence, $\phi(E'', \omega_u) \geq |M'| = |M| = \phi(E', \omega_d)$. Moreover, it is the case that $\phi(E', \omega_u) \leq \phi(E', \omega_d)$ for under ω_d, E' has strictly a superset of edges present as compared to in ω_u. Not only that, it is also the case that $\phi(E', \omega_u) \leq |M| - 1$, for if it were the case that $\phi(E', \omega_u) = |M|$, then it means that there exists a matching in E' that has cardinality equal to $|M|$ and does not use the edge (v, d) contradicting the assumption of this subcase that every maximum matching in S uses the edge (v, d).

In summary, for this subcase we have, $\phi(E', \omega_d) - \phi(E'', \omega_d) \leq 1$ and $\phi(E', \omega_u) - \phi(E'', \omega_u) \leq -1$.

Hence, in each one of the above cases, we have that in expectation over the sub-sample space considered in the case, $\phi(E', \omega) - \phi(E'', \omega) \leq 0$. And hence, in expectation over all of sample space, $\phi(E', \omega) - \phi(E'', \omega) \leq 0$. ■

3.8.4 Complete Kidney Exchange Graphs

In this section, we will deal with a kidney exchange graph where every pair of vertices that are blood-type compatible share an edge. In our results in this section we implicitly assume that tissue typing tests are always successful; this assumption is relaxed in Section 3.8.5.

There are four blood types A, B, AB, and O. For blood-type compatibility the patient should have as many types of antigens as the donor. Blood type O indicates absence of antigens and hence a donor of blood type O is blood-type compatible with all other blood groups. Blood groups A, B, and AB indicate presence of antigens A, B, and both A and B, respectively. Hence, a donor with blood type A is blood type compatible with a patient of either blood type A or AB. A patient with blood type AB is blood type compatible with a donor of any blood group.

Since every node in the graph represents a (patient, donor) pair, we can label each node by the blood-types of the patient and the donor. For instance, if the patient has blood type A and the donor blood type AB, then the label is $A - AB$.

We now borrow some definitions from Ashlagi and Roth [2011] that will help our presentation. In each definition $X, Y \in \{A, B, AB, O\}$.

Definition 3.8.8.

1. *A label $X - Y$ is* over-demanded *if $X \neq Y$ and Y is blood-compatible to donate to X.*
2. *A label $X - Y$ is* under-demanded *if $X \neq Y$ and X is blood-compatible to donate to Y.*
3. *All labels of the form $X - X$ are known as* self-demanded*.*
4. *The pair of labels $A - B$ and $B - A$ constitute* reciprocally-demanded *types.*

Note that if $X - Y$ is over-demanded, then $Y - X$ *must be* under-demanded. We will make the following assumption: For every $X - Y$ such that $X - Y$ is over-demanded and $Y - X$ is under-demanded, the number of nodes in the graph with label $X - Y$ is less than half the number of nodes with label $Y - X$. For instance, an implication of this assumption is that the number of nodes with blood type $AB - A$ is *less than half* of the number of nodes with blood-type $A - AB$.

Why might such an assumption be realistic? The justification stems from the way patient-donor pairs are formed in practice. Observe that every patient-donor pair that is not blood-type compatible has to enter the kidney exchange pool. On the other hand, if the donor is blood-type compatible to donate to the patient, then only pairs who fail a tissue typing or crossmatch test join the pool. Hence, a priori one has reason to believe that the number of pairs in the kidney exchange pool that have label $X - Y$ is significantly smaller than the number of pairs with label $Y - X$, so for example Roth et al. [Roth et al., 2007] assume that there is an endless pool of underdemanded pairs. Moreover, often the willing donor is a family member of the patient, and among family members there is a higher chance of the tissue typing and crossmatch tests being successful. In fact, the factor $1/2$ has been used by Ashlagi and Roth [Ashlagi and Roth, 2011], who based this assumption on real data Zenios et al. [2001].

Now, let us the consider the reciprocally demanded labels $A - B$ and $B - A$. Note that a donor with blood-type A cannot donate to a patient with blood-type B, and vice versa. Hence, every (patient, donor) pair of either of these types is forced to enter the

kidney exchange market. Moreover, the chances of (patient, donor) pair having blood type $A-B$ is the same as them having $B-A$, since there is no reason to believe that a person with blood type A has a higher or lower chance of kidney failure than a person of type B. Hence, in our complete kidney exchange graph, we assume that the number of nodes with label $A-B$ is *approximately* the same as those with label $B-A$.

With this we are ready to define our model of the complete kidney exchange graph, where for now (until Section 3.8.5) we only consider blood-type compatibility and ignore tissue-type compatibility.

Definition 3.8.9. *A* complete kidney graph *is a graph $G(V,E)$ with the following properties. The vertex set V can be partitioned into the sets V_{X-Y} where X and Y are the blood types of the patient and the donor respectively ($X,Y \in \{A,B,AB,O\}$). Furthermore,*

1. *Every pair of vertices in G that are* blood-type compatible *share an edge.*
2. *For each over-demanded label $X-Y$, $|V_{X-Y}| < \frac{1}{2}|V_{Y-X}|$.*
3. *The reciprocally demanded labels obey $\frac{1}{2}|V_{B-A}| \leq |V_{A-B}| \leq 2 \cdot |V_{B-A}|$.*

We define the term *an almost optimal subgraph* to denote a subgraph whose expected matching size is off from the optimal solution only by constant additive factors.

Definition 3.8.10. *An* almost optimal subgraph H *for a graph G is a solution to the* NONADAPTIVE$_2$ *problem for G, which has expected size of matching at least* NAOPT$_2(G) - O(1)$.

We now present the structure of an almost optimal solution for the complete kidney exchange graph (see Figure 3.6 for an illustration).

Theorem 3.8.11. *The subgraph $H(V,E')$ with the following description is an almost optimal subgraph for the complete kidney exchange graph $G(V,E)$.*

1. (Self-demanded form 4-cycles among themselves) *For every self-demanded label $X-X$, the edges of H constitute a 4-cycle cover of all (but for maybe $O(1)$) vertices of that label.*
2. (Each over-demanded pairs with two under-demanded) *For every pair of over-demanded $(X-Y)$ and under-demanded $(Y-X)$ labels, every node with label $X-Y$ has two edges incident to a unique pair of vertices with label $Y-X$.*

3. (Reciprocally demanded pair) *Every node in $A - B$ is involved in either a 4-cycle with one vertex of its own label and two nodes of the opposite label (i.e., $B - A$), or a path of length two using vertices of the opposite label and maybe of its own label. A similar statement holds for each node in $B - A$.*

The crucial result that helps us to prove the the optimality of the above solution is the following lemma. In a sense, it distills the core properties of kidney exchange graphs, and presents the structure of an optimal solution for all graphs that have these properties.

Definition 3.8.12. *An undirected graph $G(V, E)$ is said to be* lopsided-bipartite partitionable *if it has the following structure. The vertex set V can be partitioned into k pairs of sets (P_i, Q_i) ($1 \leq i \leq k$) and R for some k, such that $V = \bigcup_{i=1}^{k}(P_i \cup Q_i) \bigcup R$. Furthermore, for each $1 \leq i \leq k$,*

1. *$|Q_i| > 2 \cdot |P_i|$*
2. *P_i and Q_i form a complete bipartite graph.*
3. *No vertex $v \in Q_i$ has an edge incident to it from any vertex in $R \cup \bigcup_{j=1}^{k} Q_j$.*

All other possible edges may or may not be present in G.

Lemma 3.8.13. *For a lopsided-bipartite partitionable graph $G(V, E)$ with $V = \bigcup_{i=1}^{k}(P_i \cup Q_i) \bigcup R$, as in Definition 3.8.12, there exists an optimal subgraph $H(V, E')$ with the property that for every $1 \leq i \leq k$, all vertices $v \in P_i$ have two edges incident to a unique pair of vertices in $Q_i \times Q_i$. In particular, H does not have any edge between a vertex in P_i (for any i) and a vertex in R.*

Proof. Consider the optimal subgraph $H(V, E')$. If H does not already satisfy the stated property, we show how to convert it into one that satisfies the stated property and does not reduce the expected size of its maximum matching.

We can assume that subgraph H satisfies the properties stated in Corollary 3.8.7. We will now present the procedure to convert H into one that satisfies the properties stated in the statement of the theorem.

---

1. Let $S \leftarrow [k]$.
2. While S is non-empty
   - Pick a $j \in S$, such that there exists a vertex $u \in Q_j$ with no edges incident to it under E'.

- If for some $v \in P_j$ either of b or d are not members of Q_j, say it is b, we shall replace edge (v, b) by (v, u) in E'.
- If there does not exist a $v \in Q_j$ such that v has an edge incident to a vertex not in Q_j, remove j from S.

First we show that the above procedure is well-defined and that it terminates.

Claim 3.8.14. *(Well-defined) In each iteration of the while loop, in the first step of the loop, we can find a $j \in S$ and a vertex $u \in Q_j$ such that no edges are incident to it under E'.*

Proof. Since

1. the total number of edges in E' that are incident to the vertices in the set $\cup_{i \in S} P_i$ can be at most $2 \cdot \sum_{i \in S} |P_i|$ ($\because \forall v \in V$, $\delta_{E'}(v) \leq 2$), and
2. E, and therefore E', does not contain any edge going between a vertex in R and a vertex in $\cup_{i=1}^{k} Q_i$ or an edge going between a vertex in Q_i and a vertex in Q_j for any $1 \leq i, j \leq k$,

hence the number of edges incident to vertices in $\cup_{i=1}^{k} Q_j$ under edge set E' is at most $2 \cdot \sum_{i \in S} |P_i|$. On the other hand, the cardinality of the set $\cup_{i=1}^{k} Q_j$ is strictly greater than $2 \cdot \sum_{i \in S} |P_i|$. Hence, there must exist a vertex $u \in Q_j$ for some $j \in S$, such that u does not have any edge incident to it. ■

Claim 3.8.15. *(Loop terminates) The while loop eventually terminates.*

Proof. After each iteration of the while loop, the number $|E' \cap \bigcup_{i=1}^{k} (P_i \times Q_i)|$ increases by one. And this number is upper bounded by $2 \cdot \sum_{i=1}^{k} |P_i|$. ■

The following claim states that the subgraph $H(V, E')$ satisfies the properties stated in Corollary 3.8.7 at all points of the execution of the procedure.

Claim 3.8.16. *At all points in the execution of the procedure (including the point when it terminates), the subgraph $H(V, E')$ satisfies the properties stated in Corollary 3.8.7.*

Proof. Before the start of the procedure, we had assumed that the subgraph H satisfies the properties stated in Corollary 3.8.7, and at no step in the above procedure, we make a move that can violate the properties stated in Corollary 3.8.7. ■

We now show that the expected size of matching does not change at any step of the procedure.

Claim 3.8.17. *(No change in solution quality) In each iteration of the while loop, the change made to E' in the second step of the loop, does not change the expected size of matching*

Proof. In any iteration, the pair (j, u) found in the first step of the iteration satisfy the property u has an edge to each vertex $v \in P_j$ in E. Hence by Corollary 3.8.7 and Claim 3.8.16, each $v \in P_j$ must be incident to two nodes b and d, such that $\delta_{E'}(b) = 1 = \delta_{E'}(d)$.

The second step changes E' only if there exists a $v \in P_j$ that has its edges incident to a b and d, such that at least one of b or d is outside Q_j. Suppose b is the vertex outside Q_j. In such a case, edge (v, b) is replaced with (v, u).

Since before replacement vertex b has only one edge incident to it and that was to v, by replacing edge (v, b) by (v, u) in E', from the point of view of matching, we have only switched the situation of b and u, and left the situation of v unchanged, and so the expected maximum matching size does not change. ■

The final claim shows that the procedures converts H into one that satisfies the properties stated in the statement of the theorem.

Claim 3.8.18. *At the end of the procedure, the subgraph $H(V, E)$ has the property that for every $1 \leq i \leq k$, all vertices $v \in P_i$ have, in E', two edges incident to a unique pair of vertices in $Q_i \times Q_i$. No other edges are present in E'.*

Proof. The procedure terminates when the set S becomes empty. Since $S = [k]$ at the beginning of the procedure, hence, for all elements $j \in [k]$, there is a point during the execution when j is removed from set S.

For any element j, consider the point it is removed from set S. That can occur only under the circumstance that all edges that are incident to vertices in P_j have their other ends in Q_j. Furthermore, since the subgraph H at all points in the execution of the procedure, satisfies the properties in Corollary 3.8.7, hence we have the property that all vertices $v \in P_i$ have, in E', two edges incident to a unique pair of vertices in $Q_i \times Q_i$.

Hence, by end of the procedure, we have the property that for every $1 \leq i \leq k$, all vertices $v \in P_i$ have, in E', two edges incident to a unique pair of vertices in $Q_i \times Q_i$. These edges exhaust the total number of edges that could have been incident to the set $\cup_{i=1}^{k} P_i$ since each vertex can have at most two edges incident to it in E'. Furthermore,

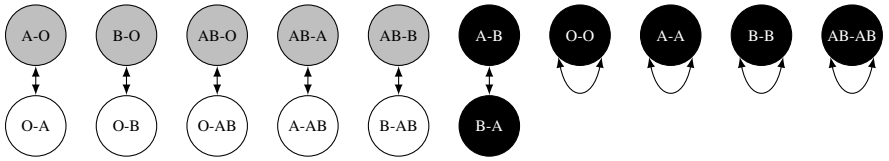


Figure 3.6: Edges chosen by Algorithm 3.8.11 in the kidney exchange graph. The grey circles are over-demanded labels, the white circles are under-demanded labels, and the black circles are reciprocally demanded and self-demanded labels.

note that the graph G does not contain any edges in the set $Q_i \times Q_j$ for any $1 \leq i, j \leq k$. Hence, no other edges can occur in H other than those already listed. ■

This completes the proof of the lemma. ■

We now complete the proof of the main result.

Proof of Theorem 3.8.11. We first set the stage for the application of Lemma 3.8.13. Consider the following settings of P_i's, Q_i's and R.

$$(P_1, Q_1) \triangleq (V_{AB-A}, V_{A-AB}), \quad (P_3, Q_3) \triangleq (V_{AB-O}, V_{O-AB}) \quad (P_5, Q_5) \triangleq (V_{B-O}, V_{O-B})$$
$$(P_2, Q_2) \triangleq (V_{AB-B}, V_{B-AB}), \quad (P_4, Q_4) \triangleq (V_{A-O}, V_{O-A}),$$
$$R \triangleq (V_{A-A}) \cup V_{B-B} \cup V_{O-O} \cup V_{AB-AB} \cup V_{A-B} \cup V_{B-A})$$

Every over-demanded label with the corresponding under-demanded label has been put in one of the (P_i, Q_i)'s with the over-demanded label taking the place of P_i. The set of self-demanded and reciprocally demanded labels have been put in R. Looking at Definition 3.8.9 and Table 3.8.4 to infer the edges present in G, we can see the graph G satisfies the condition to apply Lemma 3.8.13.

Hence, using Lemma 3.8.13, we know that there exists an optimal solution $K(V, E')$ for the complete kidney exchange graph G, that for every over-demanded/under-demanded pair of labels, satisfies the property that every vertex of the over-demanded label $(X - Y)$ has two edges incident to a unique pair of vertices of the under-demanded label $(Y - X)$.

Furthermore, in graph G (and hence in graph K), the vertices of the under-demanded types do not have an edge to any vertex in the set R as defined above.

From Table 3.8.4, it is easy to see that

1. For every self-demanded label $X - X$, a vertex of that label has edges in graph G to either vertices of of its own label or to an over-demanded label.

2. Vertices labeled $A - B$ (resp., $B - A$) share edges in graph G with vertices with either an overdemanded label or label $B - A$ (resp., $A - B$).

By Lemma 3.8.13, the optimal subgraph K does not have any edges between a vertex with an over-demanded label and a vertex with either a self-demanded or reciprocally demanded label. Hence,

1. For every self-demanded labeled $X - X$ vertex, graph K can only include edges that are incident to the vertex from other vertices of the same label.

2. For a vertex labeled $A - B$ (resp., $B - A$), graph K can only include edges that are incident to it from vertices with label $B - A$ (resp., $A - B$).

In other words, for each self-demanded label $X - X$, graph K might as well treat the complete graph formed by the vertices of that label in graph G as a separate entity and optimize on it. Similarly, graph K can optimize over the bipartite graph formed by the vertices of the reciprocally-demanded labels $A - B$ and $B - A$ separately.

For the complete graph formed by the vertices of a self-demanded labeled $X - X$, we know that if $|V_{X-X}|$ is divisible by 4, Lemma 3.7.1 states that the optimal solution is a 4-cycle cover of V_{X-X}. Otherwise, a set of vertex disjoint 4-cycles that cover all but $O(1)$ of the vertices is an almost optimal solution for the complete graph V_{X-X} (for sake of analysis, we can throw out $O(1)$ vertices to get a complete graph whose number of vertices is divisible by 4, and we know that for this remaining graph, the 4-cycle cover is optimal).

Similarly, applying Lemma 3.8.3 to the bipartite graph formed by vertices with the reciprocally demanded labels $A-B$ and $B-A$, we know that an optimal solution consists of a cover of the vertices by 4-cycles, paths of length two and at most one path of length 4 or an edge. If we throw out this one of path of length 4 or the edge, we get an almost optimal solution consisting purely of 4-cycles and paths of length two. If an $A - B$ vertex is in a 4-cycle, then it shares this 4-cycle with one $A - B$ vertex and two $B - A$ vertices. Moreover, depending on whether $|A - B| \geq |B - A|$ or the other way, any path of length two will contain two vertices of label $A - B$ and one of $B - A$ or vice-versa respectively.

Hence, the graph H as described in the statement of the theorem will have an expected size of matching at least that of K minus $O(1)$. We lose $O(1)$ terms if a 4-cycle cover for any of the complete graphs formed by vertices of a self-demanded label is not possible or if the bipartite graph formed by $A - B$ and $B - A$ cannot be covered using 4-cycles and paths of length 2. This completes the proof. ■

Patient-Donor	Com. Patient	Com. Donor	Patient-Donor	Com. Patient	Com. Donor
A-A	A/AB	O/A	A-AB	AB	O/A
B-B	B/AB	O/B	B-AB	AB	B/AB
O-O	O/A/B/AB	O	O-A	A/AB	O
AB-AB	AB	O/A/B/AB	O-B	B/AB	O
A-B	B/AB	O/A	O-AB	AB	O
B-A	A/AB	O/A			

Table 3.2: The set of compatible blood-types for all under-demanded, self-demanded and reciprocally-demanded type vertices.

An easy corollary of Theorem 3.8.11 is the following result.

Corollary 3.8.19. *There exists an almost optimal solution $H(V, E')$ for the complete kidney exchange graph G with the following properties:*

1. *For each self-demanded label, there are $\lfloor |V_{X-X}| \rfloor /4$ vertex-disjoint cycles of length 4 in the subgraph H.*
2. *For each over-demanded label $X - Y$, there are $|V_{X-Y}|$ vertex-disjoint paths of length 2, each path involving a vertex of label $X - Y$ with an edge incident to two unique vertices of label $Y - X$.*
3. *There are $\lfloor y \rfloor$ vertex disjoint paths of length 2 and $\lfloor z \rfloor$ vertex disjoint cycles of length 4 where x and y are given by the equations*

$$y + 2z = \min(|V_{A-B}|, |V_{B-A}|) \tag{3.5}$$
$$2y + 2z = \max(|V_{A-B}|, |V_{B-A}|) \tag{3.6}$$

In each such path of length 2, a vertex of label $\arg\min(|V_{A-B}|, |V_{B-A}|)$ has an edge each to two vertices of the other label. Every cycle of length 4 has two vertices of label $A - B$ that share an edge each with two vertices of label $B - A$.

3.8.5 Realistic Kidney Exchange Graphs

We now remove the assumption of successful tissue typing tests that we made in Section 3.8.4. In practice, if two pairs of donor-patient are blood-type compatible then the tissue-type test succeeds with some constant probability [Ashlagi and Roth, 2011]. This

probability depends on biological parameters of the patients and donors. Hence, a realistic kidney exchange graph can be seen as drawn from a distribution over graphs, where the distribution is defined as follows: The vertex set of each graph in the distribution is the same as the complete kidney exchange graph (Definition 3.8.9), and obeys the constraints imposed on its various vertex sets. Each edge of the complete kidney exchange graph exists independently in the randomly drawn graph with a constant probability c (which for our purposes can be thought of as a lower bound). The approach of drawing a realistic kidney exchange graph from a similar distribution has been taken before by Ashlagi and Roth [Ashlagi and Roth, 2011] and Toulis and Parkes [Toulis and Parkes, 2011].

input : A realistic kidney exchange graph G_r drawn from a distribution.
output: A subgraph H_r of G_r that is a solution to the NONADAPTIVE$_2$ problem for G_r.

1. For each of the complete graphs V_{A-A}, V_{B-B}, V_{AB-AB}, V_{O-O}, we run Algorithm 7 and add to H_r the edges it returns.
2. For each of the bipartite graphs (V_{AB-A}, V_{A-AB}), (V_{AB-B}, V_{B-AB}), (V_{AB-O}, V_{O-AB}), (V_{B-O}, V_{O-B}), (V_{AB-O}, V_{O-AB}), we run Algorithm 8 and add to H_r the edges it returns.
3. For the bipartite graph (V_{A-B}, V_{B-A}), we run Algorithm 9 and add to H_r the edges it returns.

Algorithm 6: Polynomial time algorithm for the NONADAPTIVE$_2$ problem for realistic kidney exchange graphs.

We now present our main result, building on most of the results presented above.

Theorem 3.8.20. *For a randomly drawn graph G_r from the kidney exchange graph G, we can algorithmically find in polynomial time a subgraph H_r that with probability at least $1 - o(\frac{1}{\text{NAOPT}_2(G)})$ has expected matching size at least $(1 - o(1))\text{NAOPT}_2(G) \geq (1 - o(1))\text{NAOPT}_2(G_r)$.*

Proof. From the characterization of an almost optimal subgraph H for the kidney graph G as mentioned in Lemma 3.8.19, we know that H will have the following:

1. $\alpha \triangleq |V_{AB-A}| + |V_{AB-B}| + |V_{AB-O}| + |V_{B-O}| + |V_{AB-O}| + \lfloor y \rfloor$ many paths of length 2

input : A random graph G_r drawn from a complete graph G.
output: A subgraph H_r of G_r with every node having at most incident edges.

1. Throw out $O(1)$ vertices from G_r to make the cardinality of the vertex-set divisible by 4.
2. Uniformly randomly partition the vertices of G_r into two sets A and B with $|A| = |B|$.
3. In A, pair up the vertices uniformly randomly to get a set A' which treats each pair as a vertex and hence $|A'| = |A|/2$. The vertices of A' can denoted as v_{xy} where x and y are the two vertices in A that were paired up.
   Do a similar operation with B to get B'.
4. Introduce an edge between a vertex v_{xy} in A' and a vertex $v_{x',y'}$ in B' if G_r contains all the edges $(x, x'), (x, y'), (y, x'), (y, y')$. Note that if G_r contains all these edges, then x, x', y, y' form a 4-cycle in G_r.
5. Compute the maximum matching M in the bipartite graph formed between A' and B'.
6. For each edge (v_{xy}, v_{st}) included in M, include the corresponding 4-cycle (x, s, y, t) in H_r.

Algorithm 7: Sub-module for complete graph.

input : A random graph G_r drawn from a lopsided complete bipartite graph $G(A \cup B, E)$, with $|A| < \frac{1}{2}|B|$.
output: A subgraph H_r of G_r where each vertex is incident to at most two edges.

1. Randomly pair up the vertices in B (if $|B|$ is not divisible by 2, throw out a vertex from B and then pair up the remaining vertices). Construct a new set B' by introducing a vertex v_{xy} in B for each pair (x, y) of vertices created from B.
2. Construct a bipartite graph G' between A and B'. Introduce an edge between a vertex $u \in A$ and a vertex $v_{xy} \in B'$, if the pair of edges (u, x) and (u, y) exist in G.
3. Find a maximum matching M in the bipartite graph G'.
4. For every matched edge (u, v_{xy}) in M, add the edges (u, x) and (u, y) to H_r.

Algorithm 8: Sub-module for lop-sided complete bipartite graph.

2. $\beta \triangleq \lfloor (|V_{A-A}| + |V_{B-B}| + |V_{AB-AB}| + |V_{O-O}|)/4 \rfloor + \lfloor z \rfloor$ many cycles of length 4

where y and z are given by the set of equations

$$y + 2z = \min(|V_{A-B}|, |V_{B-A}|) \tag{3.7}$$
$$2y + 2z = \max(|V_{A-B}|, |V_{B-A}|) \tag{3.8}$$

Hence the expected size of matching of H is given by $\alpha \cdot M_{2P} + \beta \cdot M_{4C}$ where M_{2P} and M_{4C} denote the expected size of matching in a path of length 2 and a cycle of length 4 respectively.

We will show that for a random graph G_r, with high probability, we can algorithmically find a subgraph H_r of G, that is composed of $\alpha - o(n)$ many paths of length 2 and $\beta - o(n)$ many cycles of length 4. Hence, with high probability, the expected matching size of H_r would be $(\alpha - o(n)) \cdot M_{2P} + (\beta - o(n)) \cdot M_{4C} = \text{NAOPT}_2(G) - o(n)$. Here $n = |V|$. It is easy to see that $\text{NAOPT}_2(G) \geq \text{NAOPT}_2(G_r$ for all graphs G_r since the edge set of G_r is a subset of that of G, and hence the optimal subgraph solution of G_r is also a subgraph of G. Hence, it also follows that the expected matching size of H_r is $\text{NAOPT}_2(G_r) - o(n)$.

All that is left to prove is that for a random graph G_r, with high probability, we can algorithmically find a subgraph H_r of G, that is composed of $\alpha - o(n)$ many paths of

input : A random graph G_r drawn from an almost balanced complete bipartite graph $G(L \cup R, E)$, with $|L| \leq |R| \leq 2|L|$.

output: A subgraph H_r of G_r where each vertex is incident to at most two edges.

1. With the given values of $|L|$ and $|R|$, solve for x and y in the equations $2 \cdot x + y = |L|$ and $2 \cdot x + 2 \cdot y = |R|$. Consider disjoint subsets L_1 and L_2 of L of sizes $2 \cdot \lfloor x \rfloor$ and y respectively. Similarly, consider disjoint subsets R_1 and R_2 of R of sizes $2 \cdot \lfloor x \rfloor$ and $2 \cdot y$ respectively.

2. Pair up the vertices in L_1 and for every such pair (s, t), introduce a vertex v_{st} in a new set L'_1. Similarly, pair up vertices in R_1 and R_2 to construct sets R'_1 and R'_2 respectively.

3. Construct bipartite graphs G_1 over $L'_1 \cup R'_1$, and introduce an edge between vertices $v_{st} \in L'_1$ and $v_{pq} \in R'_1$ in G_1, if each of the edges (s, p), (p, t), (t, q) and (q, s) are present in G_r (i.e., the vertices (s, p, t, q) form a 4-cycle in G_r).

4. Construct bipartite graph G_2 over $L_2 \cup R'_2$, and introduce an edge between vertices $u \in L_2$ and $v_{st} \in R'_2$ if the edges (u, s) and (u, t) exist (i.e., (s, u, t) form a path of length 2) in G_r.

5. Find a maximum-cardinality matching M_1 in G_1, and M_2 in G_2.

6. For every edge $(v_{st}, v_{pq}) \in M_1$, include the edges of the 4-cycle (s, p, t, q) in H_r. For every edge $(u, v_{st}) \in M_2$, include the edges of the path of length 2 formed by (s, u, t) in H_r.

Algorithm 9: Sub-module for an almost balanced complete bipartite graph.

length 2 and $\beta - o(n)$ many cycles of length 4. We claim Algorithm 6 has the desired properties.

The algorithm can be easily seen to run in polynomial since each of the sub-algorithms clearly runs in polynomial time. We now complete the analysis.

1. For each of the bipartite graphs $(V_{X-Y}, V_{Y-X}) \in \{(V_{AB-A}, V_{A-AB}), (V_{AB-B}, V_{B-AB}), (V_{AB-O}, V_{O-AB}), (V_{B-O}, V_{O-B}), (V_{AB-O}, V_{O-AB})\}$, using Claim 3.8.27, we add to H_r, with probability at least $1 - o(\frac{1}{|V_{X-Y}|})$, $|V_{X-Y}|$ many paths of length 2.

2. For each of the complete graphs $V_{X-X} \in \{V_{A-A}, V_{B-B}, V_{AB-AB}, V_{O-O}\}$, applying Claim 3.8.26, we add, with probability at least $1 - o(\frac{1}{V_{X-X}})$, $\lfloor(|V_{X-X}|/4\rfloor - O(1)$ many 4-cycles to H_r.

3. For the bipartite graph (V_{A-B}, V_{B-A}), we can infer from Claim 3.8.28, that we add to H_r, with probability at least $1 - o(\frac{1}{T})$, $\lfloor y \rfloor - o(T)$ many paths of length 2 and $\lfloor z \rfloor - o(T)$ many cycles of length 4, where $T = |V_{A-B} \cup V_{B-A}|$.

We now need to sum up over the probability of failure in each of the high probability statements given above. For each high probability statement given above, either the probability of failure is $o(\frac{1}{\text{NAOPT}_2(G)})$ or the contribution of that term to the size of optimal matching $\text{NAOPT}_2(G)$ is $o(\text{NAOPT}_2(G))$.

We only have a small number of sub-algorithms, hence using the union bound we can say that with probability at least $1 - o(\frac{1}{\text{NAOPT}_2(G)})$, the size of expected matching of the graph H_r returned by the algorithm is $(1 - o(1))\text{NAOPT}_2(G)$. ■

Distribution on graphs

Assume that for a particular graph $G(V, E)$ we have been able to characterize an optimal subgraph $H(V, E')$. Moreover, we can find the subgraph H algorithmically. However, what if we are not dealing with G, but rather a graph G_r which has the same vertex set as G and whose edges are drawn from the following distribution: every edge $e \in E$ is included in G_r with some constant probability c. Can we somehow use the fact that we have been able to solve the problem for G, and use its solution for G_r?

We would like to emphasize here that the aim of this section is to solve NONADAPTIVE$_2$ problem for *the graph* G_r. In solving it, we would like to leverage the fact that we know that it is drawn from the underlying graph G and have the knowledge of an optimal or almost optimal solution for NONADAPTIVE$_2$ problem for G.

Observation 3.8.21. *The expected matching size of an optimal solution for the complete graph G, denoted by $\text{NAOPT}_2(G)$, is at least as much as the expected matching size of an optimal solution for any graph G_r, denoted by $\text{NAOPT}_2(G_R)$.*

The reason for the above observation is that the edge set of G_r is a subset of the edge set of G, and hence any solution for G_r, i.e., a subgraph H_r of G_r, is also a subgraph of G. Hence, $\text{NAOPT}_2(G) \geq \text{NAOPT}_2(G_r)$. One corollary of the above observation is the following.

Corollary 3.8.22. *If for a graph G_r, drawn from G, we can algorithmically find a subgraph H_r, that has expected matching size within some additive loss of $\text{NAOPT}_2(G)$, then that implies that the expected matching size of H_r is at least $\text{NAOPT}_2(G_r)$ within the same additive loss.*

Hence, if we wish to prove that a particular subgraph H_r for a graph G_r has expected matching size close to $\text{NAOPT}_2(G_r)$, it suffices to show that the expected matching size of H_r is close to $\text{NAOPT}_2(G)$. In this section, we explore this question for various special cases of G.

Before we delve into the special cases, we would like to state a result on random bipartite graph.

Claim 3.8.23. *Consider a complete bipartite graph $G(P \cup Q, P \times Q)$. Draw a random bipartite graph $G_r(P \cup Q, E')$, where every edge in $P \times Q$ is included in E independently with probability c, for some constant c. There exists n_0 (a constant depending on c) such that if $n = \min(|P|, |Q|) \geq n_0$, then with probability at least $1 - o(\frac{1}{n})$, there exists a bipartite matching in G_r of size n.*

Proof. Consider the case when $|P| \leq |Q|$; the other case can be taken care of similarly. Let $|P| = n$, and let us consider an arbitrary subset $Q' \subseteq Q$, such that $|Q'| = n$. We shall show that the random bipartite graph $G'_r(P \cup Q', E' \cap (P \times Q'))$ has a perfect matching with probability at least $1 - o(\frac{1}{n})$.

We first show that with probability at least $1 - \frac{1}{n^2}$, every vertex in both sets P and Q' has degree at least 3 in graph G'_r. Consider a particular vertex $v \in P \cup Q'$. Consider the n independent random variables, each taking value in $\{0, 1\}$ and representing a possible edge between v and a vertex from the opposite side. Let these random variable be $X_1, \cdots, X_n$. Since each X_i takes value 1 with probability c, hence the expected degree of vertex v, $\mathbb{E}[\sum_{i=1}^{n} X_i] = cn$.

Let $n_0 \geq 6/c$. If $\sum_{i=1}^{n} X_i \leq 3$ (i.e., vertex v has degree at most 3), then in particular, $\sum_{i=1}^{n} X_i \leq \frac{cn}{2}$. By Chernoff bound, $\Pr[\sum_{i=1}^{n} X_i \leq \frac{cn}{2}] \leq \exp(-cn/8)$.

By union bound, the probability that at least one vertex in $P \cup Q'$ has degree less than 3 in graph G'_r is at most $2n \cdot \exp(-cn/8)$. Let n_0 be the minimum integer $\geq 6/c$, such that $2n_0 \cdot \exp(-cn_0/8) \leq \frac{1}{n_0^2}$. We then have that for all $n \geq n_0$, with probability at least $1 - \frac{1}{n^2}$, every vertex in both sets P and Q' has degree at least 3 in graph G'_r.

Let us condition the analysis from here on to each vertex in $P \cup Q'$ having degree at least 3 in graph G'_r. Clearly, once we condition, then each vertex in graph G'_r has at least three *random* neighbors from the opposite side. We can now apply Walkup's theorem [Walkup, 1980] to conclude that there exists a perfect matching in G'_r with probability at least $1 - o(\frac{1}{n})$.

Removing the conditioning, we get that with probability at least $(1 - o(\frac{1}{n})) \cdot (1 - \frac{1}{n^2}) = 1 - o(\frac{1}{n})$, there exists a perfect matching in G'_r, and hence a matching of size n in G_r. ■

Remark 3.8.24. *For all the graphs that we consider below, we shall assume that the number of vertices in the graph is large enough to apply Claim 3.8.23.*

Complete Graph Suppose that G is a complete graph. If $|V|$ is divisible by 4, then using Corollary 3.8.2, we know that the optimal subgraph H for G is a cover of the vertices of G through 4-cycles. We now use this result for the complete graph G to get the polynomial time Algorithm 7, which with high probability, gives an almost optimal solution for G_r.

Lemma 3.8.25. *Algorithm 7, in polynomial time, constructs a subgraph H_r whose expected size of matching, with probability at least $1 - o(\frac{1}{|V|})$, over the draw of random graph G_r from a complete graph G, is at least* $\text{NAOPT}_2(G) - O(1) \geq \text{NAOPT}_2(G_r) - O(1)$.

We first prove an important claim.

Claim 3.8.26. *Over the draw of G_r, with probability at least $1 - o(\frac{1}{|V|})$, the subgraph H_r computed using Algorithm 7 has $|V|/4 - O(1)$ vertex disjoint 4-cycles.*

Proof. Having thrown out $O(1)$ vertices in Step 1, consider any fixed partition (A, B) of the remaining vertices for Step 2 of Algorithm 7, with $|A| = |B|$ and fixed pairing up of vertices in A and in B to get A' and B'. Consider a particular pair of vertices $v_{xy} \in A$ and $v_{st} \in B$. Over the draw of G_r, what is the probability that an edge exists between v_{xy} and v_{uv}? For an edge to exist between these two vertices, the edges $(x, s), (s, y), (y, t), (t, x)$ must exist in G_r. Each of these edge exists independently with probability c in G_r, and hence all four exist with probability c^4.

Therefore, between any pair of vertices $v_{xy} \in A$ and $v_{st} \in B$, an edge exists with probability c^4. Using Claim 3.8.23 and Remark 3.8.24, with probability at least $1 - o(\frac{1}{|V|})$

over the draw of G_r, the bipartite graph between A and B admits a maximum matching of size $|A| = |B|$. This in turn implies that with probability at least $1 - o(\frac{1}{|V|})$, the subgraph H_r constructed for G_r has at least $|V|/4 - O(1)$ vertex disjoint 4-cycles. ■

Proof of Lemma 3.8.25. It is easy to see that Corollary 3.8.2 implies that the subgraph H for G with $\lfloor |V|/4 \rfloor$ vertex-disjoint 4-cycles has expected size at least $\text{NAOPT}_2(G) - O(1)$ where we lose $O(1)$ if the cardinality of the vertex set of G is not divisible by 4. The expected size of maximum matching in H is $\lfloor |V|/4 \rfloor \cdot M_{4C}$, where M_{4C} is the expected size of maximum matching in a single 4-cycle. In other words, $\text{NAOPT}_2(G) \leq \lfloor |V|/4 \rfloor \cdot M_{4C} + O(1)$.

Claim 3.8.26 shows that Algorithm 7 produces a subgraph H_r that with probability at least $1 - o(\frac{1}{|V|})$, has at least $|V|/4 - O(1)$ vertex disjoint 4-cycles. Hence, the with probability at least $1 - o(\frac{1}{|V|})$, the expected size of maximum matching in H_r is at least $\text{NAOPT}_2(G) - O(1)$. Moreover, from Observation 3.8.21, $\text{NAOPT}_2(G) \geq \text{NAOPT}_2(G_r)$. Hence the result. ■

Almost balanced bipartite graphs We now consider G that is a complete bipartite graph between the two sets of vertices L and R with $|L| \leq |R| \leq 2|L|$. From Lemma 3.8.3, we know that an optimal subgraph H for the graph G consists of 4-cycles and paths of length 2 (plus maybe a path of length 4 or an edge). Furthermore, by Lemma 3.8.3, it is easy to discern that a subgraph H which has $\lfloor x \rfloor$ 4-cycles and y paths of length 2 has expected matching size at least $\text{NAOPT}_2(G) - O(1)$ where x and y are given by $2 \cdot x + y = |L|$ and $2 \cdot x + 2 \cdot y = |R|$. We now utilize this knowledge to build Algorithm 9 for construct a subgraph H_r for a randomly drawn graph G_r from G. The guarantee of this algorithm can be easily inferred from the preceding discussion and the following claim.

Claim 3.8.27. *Given bipartite graph $G(L \cup R, L \times R)$ with $|L| \leq |R| \leq 2|L|$, Algorithm 9, in polynomial time, constructs a subgraph H_r of G_r, that, with probability at least $1 - o(\frac{1}{T})$, over the draw of G_r, has $x - o(T)$ 4-cycles and $y - o(T)$ paths of length 2, where x and y are given by $2 \cdot x + y = |L|$ and $2 \cdot x + 2 \cdot y = |R|$, and $T = |L \cup R|$.*

Proof. If both x and y are $\Omega(T)$, we can apply Claim 3.8.23 to each of the bipartite matchings M_1 and M_2, found in Step 5 of Algorithm 9, to infer that

1. with probability at least $1 - o(\frac{1}{x})$, M_1 has size $\lfloor x \rfloor$, and hence, the number of 4-cycles in H_r is $\lfloor x \rfloor$

2. with probability at least $1 - o(\frac{1}{y})$, M_2 has size y, and hence, the number of paths of length 2 in H_r is y

where H_r is the subgraph returned by Algorithm 9. Hence, we can infer that with probability at least $1 - o(\frac{1}{T})$, H_r has $\lfloor x \rfloor$ 4-cycles and y paths of length 2.

On the other hand, if one of x or y is $o(T)$, then we can ignore the contribution from that term, and applying Claim 3.8.23 solely to the other term, get that with probability at least $1 - o(\frac{1}{T})$, H_r has $x - o(T)$ 4-cycles and $y - o(T)$ paths of length 2. ■

Lopsided bipartite graphs Let G be a complete bipartite graph between the two sets L and R, but with $|L| < \frac{1}{2}|R|$. By Lemma 3.8.13, we know that the optimal subgraph H for G has each vertex in L having an edge each to distinct and unique vertices in R, and this implies a total of $|L|$ vertex disjoint paths of length 2 in H. Hence, $\text{NAOPT}_2(G) = |L| \cdot M_{2P}$, where M_{2P} is the expected size of maximum matching in a path of length 2. We build Algorithm 8 for such a bipartite graph, and the guarantee of the algorithm can be easily inferred from the following claim.

Claim 3.8.28. *Given a graph $G(L \cup R, L \times R)$, with $|L| < \frac{1}{2}|R|$, Algorithm 8, in polynomial time, constructs a subgraph H_r, which has expected matching size, with probability at least $1 - o(\frac{1}{|A|})$, over the draw of the graph G_r, has $|L|$ paths of length 2.*

Proof. Applying Claim 3.8.23 to the matching found in Step 3, we can see that with probability at least $1 - o(\frac{1}{|A|})$, we find a perfect bipartite matching and hence the subgraph H_r returned by the algorithm has $|L|$ vertex disjoint paths of length 2. Hence the claim follows. ■

3.9 Directions for Future Research

Our *adaptive* algorithm for the matching setting achieves a $(1 - \epsilon)$-approximation in $O(1)$ rounds and using $O(1)$ queries per vertex. Is there a *non-adaptive algorithm* that achieves the same guarantee?

3.10 Acknowledgment

The results in this chapter are part of a joint work with Avrim Blum, Anupam Gupta, Nika Haghtalab and Ariel Procaccia [Blum et al., 2013, 2014].

Chapter 4

Spiteful Auctions

4.1 Introduction

In Chapters 2 and 3, we designed new allocation mechanisms to match the given objective function and acting under certain constraints. While we design new allocation mechanisms to achieve desired properties, in many settings, there are long-established allocation mechanisms in place, and these are usually so well accepted that they are hard to be replaced. In these situations, we would like to analyze the properties of these existing allocation mechanisms and provide guarantees about their performance. In this chapter, we study the widely prevalent auction mechanisms and specifically, some of its most common formats – English, Dutch, first- and second-price sealed bid auctions. While these auction formats are well studied in literature in cases where bidders care only for what they receive from the auction, here we study them in the case where bidders are *negatively affected by the happiness of other bidders*. We will call these agents *'spiteful'* and try to understand how these agents bid in these auction formats, and how the revenue and the winner of the auction are affected by their presence.

Auctions have emerged as effective ways of allocating resources and tasks among human and software agents. Most of the auction literature assumes that each agent only cares about her own surplus: what goods she gets and how much she has to pay. However, in reality agents often have *other-regarding* preferences where they care about others' surpluses too. This can take the form of altruism, or more commonly in auctions and similar settings, spite. The spite motive, which is the preference to make others worse off, stems from mainly two reasons. The first reason is strategic. The agent might benefit in the long run by weakening her competitors, for example, driving competitors' market share down

or causing them to have to pay more for a given allocation in the auction (as has been observed in spectrum auctions [Grimm et al., 2001] and sponsored search auctions [Zhou and Lukose, 2006]). Furthermore, in certain competitions such as the Trading Agents Competition, agents might give more weight to relative rankings rather than absolute performance. The second reason is psychological. There is ample evidence from experimental economics and psychology that people behave against their self-interest in strategic settings, and that this can be explained as rational behavior among agents that inherently have other-regarding preferences [Saijo and Nakamura, 1995, Levine, 1998, Loewenstein et al., 1989].

Game-theoretic analysis of spiteful bidding in auctions was initiated relatively recently [Brandt and Weiß, 2001]. Spite can explain why people bid more aggressively in auctions than theory would predict among self-interested agents [Morgan et al., 2003]. Brandt et al. [2007] and Morgan et al. [2003] discuss the scenario where each bidder is equally spiteful, and give the equilibrium bidding functions for the first- and second-price one-item auctions. Vetsikas and Jennings [2007] extend the analysis of the symmetric-spite setting to multi-unit auctions.

This prior literature has assumed that all bidders are equally spiteful. A priori, however, there is no reason to believe that each bidder would be equally spiteful [Brainov, 2000]. Different bidders can care to a different extent about the surplus of other bidders. Moreover, a bidder might care more for the surplus of a particular bidder than for the surplus of some other bidder. In this chapter, we present, to our knowledge, the first auction analysis of the broader setting where bidders can be asymmetrically spiteful.

4.2 Model

In this chapter, we study 1-item auctions. Making the standard assumptions of quasilinear utility functions and that losers in the auction pay nothing, we have that in the absence of spite, the utility function of bidder X is

$$u_X = \begin{cases} v_X - p_X & \text{if } X \text{ wins the auction} \\ 0 & \text{if } X \text{ loses the auction} \end{cases}$$

where v_X is the bidder's valuation of the item and p_X is the amount the bidder has to pay.

In presence of spite, the utility function is

$$u_X = \begin{cases} v_X - p_X & \text{if } X \text{ wins the auction} \\ -\alpha_X^t \cdot (v_Y - p_Y) & \text{if } Y \neq X \text{ wins the auction} \end{cases}$$
$$(0 \leq \alpha_X^t \leq 1)$$

where α_X^t is a measure of the spite of agent X. The subscript X in α_X^t is to emphasize that the spite factor depends on the bidder X. The superscript t, where t stands for 'true', is there to distinguish α_X^t from symbol α_X^e which we will introduce later in this chapter. Higher α_X^t means greater spite. In the symmetric model of spite, which has been considered in the prior work, all agents are equally spiteful ($\forall X, \alpha_X^t = \alpha^t$).

The utility can also be expressed in terms of surplus:

$$u_X = \begin{cases} surplus(X) & \text{if } X \text{ wins the auction} \\ -\alpha_X^t \cdot (surplus(Y)) & \text{if } Y \neq X \text{ wins the auction} \end{cases}$$

So, conditional on losing the auction, the bidder would like to minimize the surplus of the winning bidder. We assume $0 \leq \alpha_X^t \leq 1$, that is, bidders care at least as much for their own surplus as the negation of anyone else's surplus.

Among self-interested agents, it is weakly dominant for each bidder to bid her true valuation in a second-price sealed-bid auction [Vickrey, 1961]. The following example shows that this ceases to be the case among (even symmetrically) spiteful bidders. Let there be two bidders, A and B. Let $v_A = 5$ and $v_B = 10$, and $\alpha_A^t = \alpha_B^t = 0.1$. If both bid truthfully, B wins and pays 5 (second highest bid). B's surplus is $10 - 5 = 5$, so $u_A = -0.1 \cdot 5 = -0.5$. But, for example, A can get higher utility $u_A = -0.1 \cdot 2 = -0.2$ by bidding 8, thus causing B to pay 8.

We consider the four common auction mechanisms: first-price sealed-bid, second-price sealed-bid, English, and Dutch. In the first-price (second-price) sealed-bid auction, all bidders submit their bid in a sealed envelope and the bidder with the highest bid wins the auction and pays her bid price (second-highest bid price). The English auction is modeled with a clock displaying the current bid price. The clock price increases continuously and each bidder has a button which she releases when she wants to drop out. When the second-to-last bidder releases her button, the auction ends and the remaining bidder wins at the current price. The Dutch auction is modeled with a clock where the price decreases continuously. The first bidder to release her button wins and pays the price on the clock at that point.

4.3 Spite in the discrete valuations setting

Before discussing spiteful bidding in the case where the bidders draw their valuation from a continuous distribution, we first discuss the discrete case which has some characteristics worth noting and which gives insight into the issues that arise in spiteful bidding.

4.3.1 Complete information setting

Consider the example above, but now in a first-price sealed-bid auction. If A were to bid too low, say 3, then B could bid 4 and win the item with surplus $10 - 4 = 6$. A's utility would be $-0.1 \cdot 6 = -0.6$. So A must bid higher in order to force B to bid higher, thereby reducing B's surplus. How high can A go? Once A's bid overshoots his true valuation of 5, there is the risk that B does not bid higher which means A wins the item at a price greater than his own valuation thereby ending up with a negative utility. So, let us try to calculate the bid price at which even if A wins the item, he would get the same utility as in the case he loses.

Let the winning price be ρ. If A wins, his utility would be $(5 - \rho)$ and if he loses, it would be $-0.1 \cdot (10 - \rho)$. Equating these two yields $\rho \approx 5.5$. Hence, A is indifferent between winning and losing at that bid price. At any bid price above that, A would prefer to lose. At any bid price below that, A would prefer to win. We call 5.5 the *crossover point* for A.

Similarly, we can calculate the crossover point for B, which is approximately 9.5. Above that price B would prefer to lose, and below that price, B would prefer to win.

There is a range of bid prices from 5.5 to 9.5 wherein A prefers to lose and B prefers to win. Each bid price between these two values constitutes an equilibrium. If A is adamant to bid at least 8, it is in B's best interest to bid at least $8 + \epsilon$ and win. Similarly, if B is adamant not to bid above 6, it is in A's best interest to lose by bidding $6 - \epsilon$. Conditioned on B winning, the closer the winning bid is to 5.5, the higher is the B's utility and the lower is A's utility. Similarly, the closer the winning bid is to 9.5, the higher is A's utility and the lower is B's utility. Hence, there is a 'bargaining problem' in equilibrium selection here in the case of asymmetric valuations where the two agents bargain for the equilibrium bidding price.

In the English and Dutch auction there is no such bargaining problem. In the English auction the equilibrium is at the higher crossover point (9.5), because the lower-valuation bidder can safely bid up to that point because the higher-valuation bidder would not want to drop out before then. Analogously, in the Dutch auction the equilibrium is at the lower

crossover point (5.5).

4.3.2 Incomplete information setting

We now discuss the more realistic setting where bidders have incomplete information about each others' valuations. Let bidders A and B have the joint distribution of their valuations given in Table 4.1. If A has valuation M, we say that A is of type M. In this exam-

Table 4.1: *Joint probability distribution over valuations.*

A's type	B's type	Probability	A's type	B's type	Probability
50	100	1/6	100	50	1/6
50	200	1/6	100	200	1/6
200	50	1/6	200	100	1/6

ple, we intentionally set the probability that both bidders have same type to zero because tied bids make the analysis more involved. Let the bidders have spite $\alpha_A^t = \alpha_B^t = 0.1$.

Clearly A and B are symmetric so we can look for symmetric bidding strategies that will constitute an equilibrium. Furthermore, we make the natural assumption that the equilibrium bids at type 50, 100, and 200 are in increasing order so a bidder of type 50 (if one exists) always loses and a bidder of type 200 (if one exists) always wins.

Let Q be the bid made by bidder of type 100 in equilibrium. Now a type-50 bidder can bid anywhere up to Q since she knows that the other bidder (regardless of whether he is of type 100 or 200) will bid at least Q. Similarly, a type-200 bidder would like to bid as close to (but higher than) Q as possible since that would maximize her utility. So, in equilibrium all types bid basically the same amount (but they prefer ties to be broken in favor of higher types).

Q clearly has to be such that a bidder of type 50 prefers to lose at that bid and a bidder of type 200 prefers to win at that bid. Let T be the crossover point of the type-50 bidder. Then, $50 - T = -0.1 \cdot (100 - T + 200 - T)/2$, where the left hand side is her utility if she wins at T and the right hand side is her expected utility if she loses. Solving this yields T=59.1. So, Q must be at least 59.1. Similarly, denoting by W the crossover point of the type-200 bidder, we must have $200 - W = -0.1 \cdot (50 - T + 100 - T)/2$. This yields $W = 188.6$, so Q must be at most 188.6. If Q is the bidding price, then the expected utility of a type-100 bidder is $((100 - Q) - 0.1 \cdot (200 - Q))/2$. (Here we have used the fact that a type-50 other bidder always loses and a type-200 other bidder always wins.) If Q is above 88.9, then the expected utility is negative. Hence, Q must be below 88.9. Q can therefore

lie between 59.1 and 88.9. So again, like in the complete information setting, there is a range of bid values Q that constitute an equilibrium. Thus there is a bargaining problem in this setting as well.

4.4 Prior results

Prior work has provided equilibrium analysis for settings where bidders draw their valuations from the same distribution and have *equal* spite values α^t ($\forall X, \alpha^t_X = \alpha^t$) [Morgan et al., 2003, Brandt et al., 2007]. We now summarize some of those prior results in order to provide a comparison point to the results we will derive. Table 4.2 summarizes the symmetric equilibrium bidding strategies for the settings where the bidder's valuation are drawn uniformly from $[0, 1]$.

Table 4.2: *Bidding functions under symmetric spite.*

Auction	2-bidders	n-bidders
First-price sealed-bid and Dutch	$\left(\frac{1+\alpha^t}{2+\alpha^t}\right) v$	$\left(\frac{n-1}{n-\frac{\alpha^t}{1+\alpha^t}}\right) v$
Second-price sealed-bid	$\left(\frac{1+\alpha^t}{1+2\alpha^t}\right) v + \frac{\alpha^t}{1+2\alpha^t}$	$\left(\frac{1+\alpha^t}{1+2\alpha^t}\right) v + \frac{\alpha^t}{1+2\alpha^t}$
English	$\left(\frac{1+\alpha^t}{1+2\alpha^t}\right) v + \frac{\alpha^t}{1+2\alpha^t}$	*text explains strategy*

So, in the first-price sealed-bid 2-bidder case , agents bid higher under spite than under self-interest: $\frac{2v}{3}$ when $\alpha^t = 1$ and $\frac{v}{2}$ when $\alpha^t = 0$. This is also the case in the second-price 2-bidder auction: as α^t varies from 0 to 1, the bid of the spiteful bidder varies from v to $\frac{2}{3}v + \frac{1}{3}$. Incidentally, in the first-price auction, the bidding function of the 2-bidder case can be transformed to the n-bidder case by replacing $(1+\alpha^t)$ by $(n-1)(1+\alpha^t)$. Furthermore, for the second-price sealed-bid auction, the bidding function is the same in 2-bidder and n-bidder settings.

In the English auction with n bidders, the bidding strategy differs from the 2-bidder case and is the following [Morgan et al., 2003].

- If three or more bidders are present, each bidder drops out as the price reaches her valuation.

- If only two bidders remain, each bidder drops out when the price reaches $b(v)$, where $b(v)$ is the bid she would have submitted in a 2-bidder second-price sealed-bid auction.

4.5 Asymmetric spite results

We now move to the setting where bidders can be spiteful to different extents. Throughout the rest of the chapter, we will assume that the bidder's valuations are drawn uniformly and independently from $[0, 1]$. Further, as in prior research, we assume that for a given set of spite factors of the bidders, the equilibrium bidding function is strictly increasing in the agent's valuation. As in prior research, we will focus on studying symmetric equilibria, that is, equilibria where the form of the bidding function is the same for every agent. That, of course, does not mean that the agents' bids are the same because they have different valuations and different spite factors.

4.5.1 The 2-bidder case

We first analyze the setting with two bidders, A and B. Denote by $b_B(\cdot)$ the equilibrium bidding function of bidder B, that is, if B has valuation v_B, she bids $b_B(v_B)$ in equilibrium. Similarly, denote by b_A the bid of bidder A when she has valuation v_A.

First-price sealed-bid auction and Dutch auction

In the first-price sealed-bid auction (and its strategic equivalent, the Dutch auction), the expected utility of bidder A is

$$\int_0^{b_B^{-1}(b_A)} [v_A - b_A] \mathrm{d}v_B - \alpha_A^t \int_{b_B^{-1}(b_A)}^1 [v_B - b_B(v_B)] \mathrm{d}v_B \tag{4.1}$$

The first term above is for the case where A wins and the second term is for the case where she loses. To solve this for A's bid b_A, our high-level approach is to differentiate the above equation with respect to b_A and solve for b_A by equating the differential to 0. We will now present the derivation in detail. We guess that the bidding function in symmetric equilibrium is a linear functions of the bidder's valuation (as it was in the symmetric-spite setting). Hence we can write $b_B(v_B) = r_B(\alpha_A^t, \alpha_B^t) \cdot v_B$ and differentiate (4.1) with respect

to b_A. Equating the differential to 0, we get

$$b_A = \left(\frac{1}{2 + \alpha_A^t (1 - 1/r_B)} \right) v_A$$

We observe that this bidding function for A is of the linear form we guessed. This confirms the guess that these bidding functions constitute a symmetric equilibrium.

This can be written as $b_A = r_A \cdot v_A$, where

$$r_A = \left(\frac{1}{2 + \alpha_A^t (1 - 1/r_B)} \right)$$

We have an exactly analogous equation for r_B. Here, r_A and r_B are written as functions of each other, while we would like to express them as functions of the spite coefficients only. Hence we solve the equations for r_A and r_B simultaneously to get

$$r_A = \frac{1 - \alpha_A^t \alpha_B^t}{2 - \alpha_A^t - \alpha_A^t \alpha_B^t}$$

This can be put in a better-looking form by introducing symbols α_A^e and α_B^e, so b_A becomes

$$b_A(v) = \left(\frac{1 + \alpha_A^e}{2 + \alpha_A^e} \right) v \tag{4.2}$$

where

$$\alpha_A^e = \frac{\alpha_A^t}{1 + \alpha_B^e - \alpha_A^t} = \alpha_A^t \left(\frac{1 - \alpha_B^t}{1 - \alpha_A^t} \right) \tag{4.3}$$

We get analogous equations for α_B^e.

With the above transformation, we can observe that the bidding function in this asymmetric-spite settings looks like the bidding function in the symmetric-spite setting (Table 4.2)—except that there is now the symbol α_A^e in place of α^t. Because of their close connection, we call α^t *(t)rue* α and α^e *(e)xpressed* α, though no semantics behind these names are intended here.

Note that the first expression for α_A^e in Equation 4.3 is in terms of α_B^e and α_A^t, while the second expression is in terms of α_B^t and α_A^t. Depending on the situation, either of these forms can be useful.

Second-price sealed-bid auction

In the second-price sealed-bid auction, the expected utility of bidder A is

$$\int_0^{b_B^{-1}(b_A)} [v_A - b_B(v_B)]\mathrm{d}v_B - \alpha_A^t \int_{b_B^{-1}(b_A)}^1 [v_B - b_A]\mathrm{d}v_B \tag{4.4}$$

We guess that b_B is of the form $r_B(\alpha_A^t, \alpha_B^t) \cdot v_B + s_B(\alpha_A^t, \alpha_B^t)$ just as it was in the symmetric-spite second-price sealed-bid setting. Note that the guess here also includes an additive term $s_B(\alpha_A^t, \alpha_B^t)$ unlike in the first-price sealed-bid setting. Using this form of b_B, we differentiate (4.4) with respect to b_A. We then equate the differential to 0 to get

$$b_A = \left(\frac{r_B}{r_B - \alpha_A^t(1-2r_B)}\right) v_A + \frac{\alpha_A^t(r_B^2 + (r_B-1)s_B)}{r_B - \alpha_A^t(1-2r_B)}$$

This bidding function is of the form we had guessed with

$$r_A = \frac{r_B}{r_B - \alpha_A^t(1-2r_B)}, \qquad s_A = \frac{\alpha_A^t(r_B^2 + (r_B-1)s_B)}{r_B - \alpha_A^t(1-2r_B)}$$

We have analogous equations for bidder B. This proves that the guess is correct, that is, these functions indeed constitute a symmetric equilibrium.

Solving the above equations for r_A and s_A simultaneously with the analogous ones for r_B and s_B, we get

$$r_A = \frac{1 - \alpha_A^t\alpha_B^t}{1 + \alpha_A^t - 2\alpha_A^t\alpha_B^t}, \qquad s_A = \frac{\alpha_A^t - \alpha_A^t\alpha_B^t}{1 + \alpha_A^t - 2\alpha_A^t\alpha_B^t}$$

These equations can be again put in a nice form by introducing symbols α_A^e and α_B^e. We get the equilibrium bidding function for A as

$$b_A(v) = \left(\frac{1+\alpha_A^e}{1+2\alpha_A^e}\right) v + \frac{\alpha_A^e}{1+2\alpha_A^e} \tag{4.5}$$

where

$$\alpha_A^e = \frac{\alpha_A^t}{1 + \alpha_B^e - \alpha_A^t} = \alpha_A^t\left(\frac{1-\alpha_B^t}{1-\alpha_A^t}\right) \tag{4.6}$$

We get analogous equations for bidder B.

We observe that the bidding function is exactly of the form as in the symmetric-spite case—except that α^e has replaced α^t. Furthermore, the expression of α^e is the same as in the asymmetric-spite *first*-price sealed-bid auction.

English Auction

In the English auction, when the clock price is z, the expected utility of bidder A as a function of her bid b_A is

$$\frac{1}{1-z}\Big(\int_{b_B^{-1}(z)}^{b_B^{-1}(b_A)}[v_A - b_B(v_B)]\mathrm{d}v_B - \alpha_A^t\int_{b_B^{-1}(b_A)}^{1}[v_B - b_A]\mathrm{d}v_B\Big) \tag{4.7}$$

The methodology to solve for the equilibrium bid function remains the same. We guess that b_B is of the form $r_B(\alpha_A^t, \alpha_B^t)\cdot v_B + s_B(\alpha_A^t, \alpha_B^t)$ just as it was in the symmetric-spite case, and it turns out that we get the same bidding function as in the second-price sealed-bid auction.

We summarize the results in Table 4.3. We observe that in all cases, the bidding functions in is of the same form as in the symmetric-spite setting (Table 4.2)—except with α^e occupying the place of α^t.

Table 4.3: *Equilibrium in the 2-bidder asymmetric-spite setting.*

Auction type	Bidding function for bidder A	Expression for α_A^e
First-price sealed-bid and Dutch	$\left(\frac{1+\alpha_A^e}{2+\alpha_A^e}\right)v$	$\frac{\alpha_A^t}{1+\alpha_B^e-\alpha_A^t}$
Second-price sealed-bid and English	$\left(\frac{1+\alpha_A^e}{1+2\alpha_A^e}\right)v + \frac{\alpha_A^e}{1+2\alpha_A^e}$	

Furthermore, these are the only linear (in case of first-price sealed-bid and Dutch auctions) and affine (in case of second-price sealed-bid and English auctions) equilibrium bidding functions. This is because guessing these forms for one bidder yielded unique bidding functions of the same form for the other bidder.

Comparison of α^t and α^e

In this section we discuss α^e as compared to α^t. Although α^t always lies between 0 and 1 (by assumption), α^e is bounded below by 0 and is unbounded from above. Table 4.4 lists values of α_A^e for some combinations of α_A^t and α_B^t.

From Equation 4.3 and Table 4.4 we see the following.

1. In the symmetric case, $\alpha^e = \alpha^t$ as we should expect from comparing the equations of the symmetric and asymmetric case.

Table 4.4: *The values in the table are bidder A's α^e. The rows correspond to various values of bidder A's α^t and the columns correspond to values of bidder B's α^t.*

	0	**0.1**	**0.3**	**0.5**	**0.7**	**0.9**
0	0	0	0	0	0	0
0.1	0.11	0.1	0.08	0.06	0.03	0.01
0.3	0.43	0.39	0.3	0.21	0.13	0.04
0.5	1	0.9	0.7	0.5	0.3	0.1
0.7	2.33	2.1	1.63	1.17	0.7	0.23
0.9	9	8.1	6.3	4.5	2.7	0.9

2. In the asymmetric case,

   - For a given α_A^t, the value of α_A^e decreases linearly with increasing α_B^t.
   - For a given α_B^t, the value of α_A^e increases with increasing α_A^t.

   This implies that for a fixed α_A^t, bidder A might bid higher or lower in the asymmetric-spite case than in the symmetric-spite case depending on α_B^t. In fact, the mapping between true and expressed spite factors is such that if $\alpha_A^t \neq \alpha_B^t$, then the difference between α_A^e and α_B^e is greater than the difference between α_A^t and α_B^t. So, in equilibrium, the more spiteful bidder expresses an over-exaggerated spite and the less spiteful bidder expresses an under-exaggerated spite.

Revenue

Brandt et al. [2007] show that in symmetric-spite settings, second-price auctions yield higher expected revenue than first-price auctions. In contrast, we show that in asymmetric-spite settings there is no revenue-dominant auction. For a given set of α^e's, we give in Table 4.5 the expected revenue for a first-price sealed-bid/Dutch auction and a second-price sealed-bid/English auction.

For $\alpha_A^e = 8.1$ and $\alpha_B^e = 0.01$, corresponding to $\alpha_A^t = 0.9$ and $\alpha_B^t = 0.1$, the first-price auction yields expected revenue 0.63 while the second-price auction yields expected revenue 0.49. For $\alpha_A^e = 0.39$ and $\alpha_B^e = 0.08$, corresponding to $\alpha_A^t = 0.3$ and $\alpha_B^t = 0.1$, the first-price auction yields expected revenue 0.37 while the second-price auction yields expected revenue 0.42. So here, neither auction mechanism beats the other in expected revenue in general. By substituting various values of spite into the formulas of Table 4.5, we found that when bidders have comparable true spite, the second-price auction yields

Table 4.5: *Expected revenue for given α^e's in 2-bidder setting.*

Auction type	Expected revenue	Notation
First-price sealed-bid and Dutch	$\frac{1}{3}(\frac{p_A^2}{p_B} + \frac{p_B^2}{p_A})$	$p_X = \frac{1+\alpha_X^e}{2+\alpha_X^e}$
Second-price sealed-bid and English	$\frac{1}{2}(\frac{q_B}{q_A} + \frac{q_A}{q_B}) - \frac{1}{3}(\frac{q_B^2}{q_A} + \frac{q_A^2}{q_B})$	$q_X = \frac{1+\alpha_X^e}{1+2\alpha_X^e}$

higher expected revenue while if the true spite values differ largely, the first-price auction yields higher revenue, as can be seen in Figure 4.1.

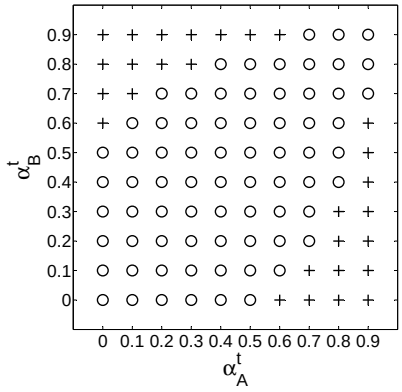


Figure 4.1: *The circles denote the points where the second-price auction yields higher expected revenue than the first-price auction. The pluses denote the points where the reverse occurs.*

Allocation

In symmetric-spite settings, the bidder with the highest valuation wins the item. In asymmetric-spite settings, this is no longer true in general. For example, in a first-price sealed-bid auction, consider the case where bidder A has valuation 0.9 and $\alpha_A^t = 0.1$ while bidder B has valuation 0.7 and $\alpha_B^t = 0.7$. For this pair of α^t's, we get $\alpha_A^e = 0.03$ and $\alpha_B^e = 2.1$, so A bids 0.46 and B bids 0.53. Hence, the lower-valuation bidder (B) wins.

Figures 4.2 and 4.3 plot a bidder's bid against her α^e for various valuations she may have in first-price and second-price auctions. For low valuations, $v = 0.1$ or 0.2, the curve is nearly flat in the first-price auction. This means that as α^e increases, her bid does not

increase much. However, in the second-price auction, for the same set of low valuations, the bid increases steeply as α^e increases from 0 to 2, and keeps increasing after that. In contrast, consider high valuations. For $v = 0.9$ or 0.8, in the first-price auction, the bid increases rapidly as α^e increases from 0 to 2, and keeps increasing steadily after that. In the second-price auction, however, for valuations 0.9 and 0.8, there is not much change in the bid as α^e increases from 0 to 10.

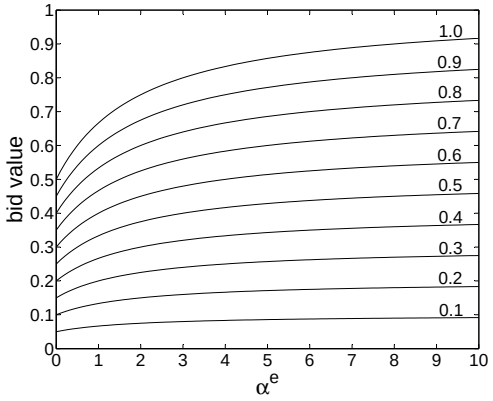


Figure 4.2: *The bid in a first-price auction varies with the expressed spite α^e. The curves are for different valuations v, with the lowest curve corresponding to $v = 0.1$ and the highest to $v = 1$.*

These observation imply that if A has a high valuation in the second-price auction, it is unlikely that B can win if B has a slightly lower valuation, say 0.8, no matter how high α^e_B is. In the first-price auction however, if A has a high valuation, say 0.9, but low α^e_A (< 1), then bidder B can win even with valuation 0.7 but with a high α^e_B, say 5. Similar statements can be made for the low valuation case, but with the first-price and second-price auction switching roles.

4.5.2 The n-bidder setting with directed spite

We now extend our analysis of asymmetric-spite auctions to n bidders. With more than two bidders, there is the possibility that some bidder(s) have different extents of spite toward different other bidders. We call this *directed* spite. For example, bidder A can have spite factor α^t_{AB} toward bidder B and spite factor α^t_{AC} toward bidder C, so her utility

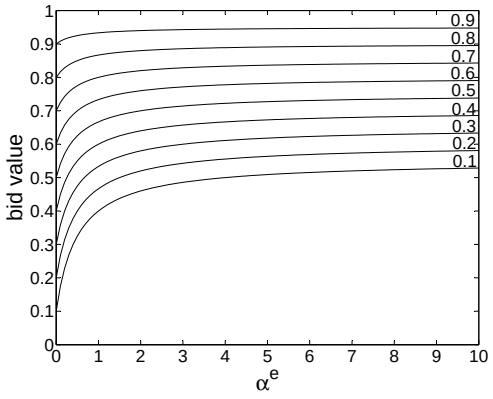


Figure 4.3: *The bid in a second-price auction varies with the expressed spite α^e. The curves are for different valuations v, with the lowest curve corresponding to $v = 0.1$ and the highest to $v = 0.9$. For $v = 1$, the bidder always bids 1.*

would be

$$u_A = \begin{cases} v_A - p_A & \text{if } A \text{ wins} \\ -\alpha^t_{AB} \cdot (v_B - p_B) & \text{if } B \text{ wins} \\ -\alpha^t_{AC} \cdot (v_C - p_C) & \text{if } C \text{ wins} \end{cases}$$

First-price sealed-bid and Dutch auction

The expected utility of A is

$$\begin{aligned} &\int_0^{b_C^{-1}(b_A)} \int_0^{b_B^{-1}(b_A)} [v_A - b_A] \mathrm{d}v_B \mathrm{d}v_C \\ &- \alpha^t_{AB} \int_{b_B^{-1}(b_A)}^1 \int_0^{b_C^{-1}(b_B(v_B))} [v_B - b_B(v_B)] \mathrm{d}v_C \mathrm{d}v_B \\ &- \alpha^t_{AC} \int_{b_C^{-1}(b_A)}^1 \int_0^{b_B^{-1}(b_C(v_C))} [v_C - b_C(v_C)] \mathrm{d}v_B \mathrm{d}v_C \end{aligned} \tag{4.8}$$

where b_B is the bidding function of B and b_C is the bidding function of C. The same formula extends to any number of agents in the obvious way. We again guess that b_A, b_B and b_C are linear functions of the valuation, and we indeed find an equilibrium, verifying the guess. The results for the n-agent case are stated in the last row of Table 4.6. In the row before that, we also show how these results specialize to the case of undirected spite, that is, where each agent X has a spite factor α_X toward each other bidder.

Table 4.6: *Bidding function for n-bidder asymmetric-spite first-price sealed-bid and Dutch auctions.*

Case	Bidding function for bidder A	Expression for $\frac{\alpha_A^e}{1+\alpha_A^e}$
Undirected spite	$\left(\frac{n-1}{n-\frac{\alpha_A^e}{1+\alpha_A^e}}\right) v$	$\frac{\alpha_A^t}{n-1}\left(\sum_{X:X\neq A}\frac{1}{1+\alpha_X^e}\right)$
Directed spite	$\left(\frac{n-1}{n-\frac{\alpha_A^e}{1+\alpha_A^e}}\right) v$	$\frac{1}{n-1}\left(\sum_{X:X\neq A}\frac{\alpha_{AX}^t}{1+\alpha_X^e}\right)$

English auction

The analysis of the English auction is a bit more intricate. We analyze directed spite; the undirected-spite setting is a special case.

Proposition 4.5.1. *In the English auction, an equilibrium strategy for any bidder A is to stay in if the clock price is lower than $\max\{v_A, \max_{X\in S} B(A,X)\}$ and to drop out otherwise. Here S is the set of other bidders who are still in, and $B(A,X)$ is the bidding function for A if X were the only other bidder (and we know this form from our analysis of the two-bidder case, Table 4.3).*

Proof. Any n-bidder auction would in the end reduce to a 2-bidder setting. Until what clock price should A stay in? As long as there is at least one bidder, Z, still in to whom A would bid higher in a 2-bidder setting than the current clock price, A should stay in. This is because if A leaves before the clock reaches that price and all other bidders except Z also exit before the clock reaches that price, then Z will win at a price lower than if A had stayed in. So, A would end up with a lower utility due to leaving early.

What if there is no such person Z still in? If the clock price has exceeded A's valuation v_A, then A no longer wants to win, so it is best for her to exit. If, on the other hand, the clock price has not yet reached v_A, then A should stay in until she wins or the price exceeds v_A. ■

4.6 Conclusions and future research

We game-theoretically analyzed the four common auction mechanisms when bidders have asymmetric spite. A noteworthy feature is that the symmetric equilibrium bidding function

continues to be the same as with symmetric spite—except that the true spite is replaced by 'expressed' spite. Unlike in the symmetric-spite setting, bidders express spites that are higher or lower than their true spite depending on others' spites. Moreover, the equation for expressed spite does not depend on the auction mechanism in the 2-bidder case. Furthermore, we found that the allocation can be inefficient and that the revenue ranking may reverse between first- and second-price auctions. We also studied the generalization in the n-bidders setting where agents can have different extents of spite toward different other bidders. We also showed that in sealed-bid auctions under asymmetric valuation distributions, there can be a "bargaining problem" in selecting bids.

Future work includes solving for the equilibrium of the second-price auction in the n-bidder case. We also plan to study valuation priors that are not uniform.

We assumed the bidders know each others' true spite. In settings where they do not know, we have to understand whether the equilibrium will be reached, and how. We conducted experiments that indicate that in a repeated-game setting the equilibrium can be learned as long as the bidders are able to infer each others' *expressed* spites. In our simulation, each bidder adjusts her own expressed spite given the others' expressed spites (using the expression for α^e in Table 4.3 or 4.8). The bidders rapidly converged to the equilibrium values of α^e's regardless of the initial values of the α^e's. Future work includes proving bounds on this convergence. Another interesting direction would be to solve for the equilibrium in settings where bidders do not know each others' true spite coefficients but have a joint prior over them.

4.7 Acknowledgment

The results in this chapter are part of a joint work with Tuomas Sandholm [Sharma and Sandholm, 2010].

Chapter 5

Understanding the complexity of Submodular Functions

5.1 Introduction

The previous chapters have either designed new allocation mechanisms or studied existing ones. We now turn our attention to valuation functions and try to understand the complexity of a particular class of valuations, namely, submodular functions. Valuation functions give the value that a claimant has for any set of resources. While Chapter 2 dealt with arbitrary valuation functions, usually designing and analyzing allocation mechanisms over arbitrary valuation functions is hard. One approach to helping with the mechanism design and analysis is to try to approximate a complex valuation function through simpler valuation functions, and then run known (or design new) allocation mechanisms for this simpler class. The overall solution guarantee of this approach is a function of the quality of approximation and the solution guarantee of the allocation mechanism for the simpler class. Other than this algorithmic reason, it is inherently important to understand the complexity of classes of valuation functions and compare them with other classes. This allows to relate and reason with the different classes of valuation functions. With this over-arching motivation, in this chapter, we study the class of submodular functions, and some of its better known sub-classes and analyze how well these sub-classes approximate each other and the general class of submodular functions.

Submodular function are an important class of valuation functions since they are characterized by the property of decreasing marginal return. Valuation functions of buyers in several settings exhibit the feature of decreasing marginal return. Roughly speaking, de-

creasing marginal return means that the additional value a resource brings is greater when it is added to a smaller set of resources than when it added to a larger set. Furthermore, submodular functions are ubiquitous in diverse disciplines, including economics, algorithmic game theory, machine learning, combinatorial optimization and combinatorics. While submodular function can be minimized efficiently, i.e., in polynomial time [Grötschel et al., 1981, Schrijver, 2000, Iwata et al., 2001], many natural optimization problems over submodular functions are NP-hard, e.g., Max-k-Coverage [Nemhauser et al., 1978], Max-Cut and Max-DiCut [Goemans and Williamson, 1995], and Max-Facility-Location [Cornuejols et al., 1977]. Consequently, many works, specifically in the setting of algorithmic game theory [Buchfuhrer et al., 2010, Dughmi, 2011, Dughmi et al., 2011, Hoefer and Kesselheim, 2012], have explored *simpler* subclasses of submodular functions for which the given algorithmic problem can still be well-approximated. Such subclasses of submodular functions have included cut functions of graphs, coverage functions of set systems, budgeted additive functions, matroid rank functions, etc.

Our work is motivated by the question, how complex can a submodular function be? Since this is such a fundamental question, it has been asked in different forms previously. Goemans et al. [2009] consider how many queries to a submodular function are sufficient to infer the value of the function, approximately, at every point in the domain. Balcan and Harvey [2011] focus on the problem of learning submodular functions in a probabilistic model; are few random queries enough to infer the value at almost all points in the domain? Badanidiyuru et al. [2012] ask whether an approximate *sketch* of a submodular function, or more generally a subadditive function, exists (i.e., can the function be represented in polynomial space)? Seshadhri and Vondrák [2011] consider the testability of submodular functions: how many queries does it take to check whether a function is *close* to being submodular?

We approach this question by noting that not all submodular functions are identically complex and some have been more amenable to optimization than others. Thus, one natural way to characterize the relative complexity of one class of submodular functions w.r.t another, is to ask how well can a function in the first class be approximated by a function in the second. Formally, we ask the following question. Given two classes of submodular functions $\mathcal{F}$ and $\mathcal{G}$ (typically $\mathcal{G} \subset \mathcal{F}$), what is the smallest θ, such that for every $f \in \mathcal{F}$, there exists a $g \in \mathcal{G}$ such that $f(S) \leq g(S) \leq \theta \cdot f(S)$ for each $S \subseteq U$? Here class $\mathcal{G}$ would represent the class of submodular functions which are easier to optimize for some problem and class $\mathcal{F}$ would represent a bigger class which we want to optimize over. We also note that this concept of approximation is not special to submodular functions and can be asked for any two classes of functions. We focus on submodular functions due to their ubiquitous nature in optimization.

Intuitively, this notion of approximation resembles the long and rich line of work that deals with the algorithmic applications of geometric embeddings, in which the goal is to embed hard metric spaces into simpler ones. Some successful examples include embedding general metrics into normed spaces [Bourgain, 1985], dimension reduction in a Euclidean space [Johnson and Lindenstauss, 1984] and the probabilistic embedding into ultrametrics [Bartal, 1996, Fakcharoenphol et al., 2004]. As in the metric case, a natural byproduct of the above approach is that if there exists an α-approximation algorithm for any submodular function in $\mathcal{G}$, then there exists a $(\theta \cdot \alpha)$-approximation algorithm for all functions in $\mathcal{F}$. As an application of our approach, we show how to obtain an algorithm for the online submodular function maximization problem, for general monotone submodular functions [Buchbinder et al., 2012]. Previously, results were known for only certain subclasses of submodular functions; see Section 5.6 for details.

5.1.1 Our Results and Techniques

We start by asking how well a general submodular function $f : 2^U \to \mathbb{R}_+$ (with the additional property that $f(\phi) = 0 = f(U)$) can be approximated by a function in the canonical simpler subfamily of non-symmetric submodular functions, cut function of a directed graph. We give matching upper and lower bounds for such an approximation (Theorem 5.1.4). Next, we ask the same question for *symmetric* submodular functions vis-a-vis its canonical simpler subfamily, cut functions of undirected graphs. In this case, we provide nearly matching upper and lower bounds (Theorem 5.1.5). We then move our attention to two subfamilies, budgeted additive functions and coverage functions, both of which, as already mentioned in the introduction, have received considerable interest in the algorithmic game theory setting. We show tight upper and lower bounds for approximating budgeted additive functions with coverage functions (Theorem 5.1.6). These results are summarized in Table 5.1. While previous works [Goemans et al., 2009, Balcan and Harvey, 2011, Badanidiyuru et al., 2012] studied the complexity of submodular functions from different perspectives, they do imply some additional results, both positive and negative, on the approximation of monotone submodular functions by simpler classes of submodular functions (as illustrated in Table 5.1 and discussed in detail in Section 5.7).

Let us now briefly discuss the main techniques that we use to obtain our results. In contrast to previous works [Goemans et al., 2009, Balcan and Harvey, 2011, Badanidiyuru et al., 2012], *arbitrary* submodular functions, as opposed to *monotone* submodular functions, present different challenges. As an illustration, for approximating a submodular function f via a cut function of a graph G, consider the case when there is a non-trivial set $\emptyset \neq S \neq U$ for which $f(S) = 0$. Then the weight of the cut $(S, \bar{S})$ in G is forced to

Input Class	Output Class	Approximation Upper Bound	Approximation Lower Bound
General Submodular	Cut Functions (directed)	$\frac{n^2}{4}$	$\frac{n^2}{4}$
Symmetric Submodular	Cut functions (undirected)	$n-1$	$\frac{n}{4}$
Budgeted Additive	Coverage	$\frac{e}{e-1}$	$\frac{e}{e-1}$
Monotone Submodular	Coverage/Budgeted Additive	$O(\sqrt{n}\log n)$ [Goemans et al., 2009]	$\Omega(\frac{n^{1/3}}{\log^2 n})$ [Balcan and Harvey, 2011, Badanidiyuru et al., 2012]

Table 5.1: Our results are described in the first three rows. The results in the last row are either implicit in the references or follow as a corollary (Section 5.7). When the output class is a cut function of a graph, we assume that the input function f satisfies $f(\emptyset) = f(U) = 0$, as every cut function must satisfy this constraint. Here, n denotes the size of the ground set.

be zero. Indeed, *all* sets S with $f(S) = 0$ must be in a correspondence with cuts in G of value zero. Thus, given a submodular function f, our construction of G optimizes for the minimizers of the submodular function f. Surprisingly, this can be shown to give the best possible approximation. For a *symmetric* submodular function f, we show that it suffices to use the cut function of a *tree* (as opposed to a general undirected graph) utilizing the Gomory-Hu tree representation [Gomory and Hu, 1961, Queyranne, 1993] of f.

For approximating budgeted additive functions by coverage functions, we first give a randomized construction achieving an approximation factor of $e/(e-1)$. We then show that this is the best possible approximation factor as characterized by a linear program. The proof of the lower bound of $e/(e-1)$ uses linear programming duality and proceeds by presenting a feasible dual solution to the linear program achieving an objective value of $e/(e-1)$ in the limit. We would like to point out that all our results are algorithmic and the claimed approximations can be found in polynomial time given a value oracle for the submodular function.

5.1.2 Related Work

Goemans et al. [2009] considered the problem of how well a given monotone submodular function f can be approximated when only polynomially many value oracle queries are

permitted. They presented an approximation of $O(\sqrt{n}\log n)$, and an improved guarantee of $\sqrt{n+1}$ in the case that f is a matroid rank function. Implicit in this algorithm and relevant to our setting is an approximation of all monotone submodular functions by budgeted additive functions (Section 5.7). The current best lower bound for the problem studied by Goemans et al. [2009] is given by Svitkina and Fleischer [2011] and is $\Omega(\sqrt{n/\log n})$.

Balcan and Harvey [2011] take the learning perspective to the study of the complexity of submodular functions. They study the problem of probabilistically learning a monotone submodular function, given the values the function takes on a polynomial sized sample of its domain. They provide a lower bound of $\Omega(n^{1/3})$ on the best possible approximation a learning algorithm can give to the submodular function, even when it knows the underlying sampling distribution, and the submodular function to be learned is Lipschitz. Another result with this perspective is by Balcan et al. [2012] who show that a symmetric non-monotone submodular function can be approximated to within $\sqrt{n}$ by the square root of a quadratic function. Furthermore, they show how to learn such submodular functions.

Badanidiyuru et al. [2012], motivated by the problem of communicating bidders' valuations in combinatorial auctions, study how well specific classes of set functions can be approximated given the constraint that the approximating function be representable in polynomially many bits. They named such an approximation a *sketch*, proving that coverage functions admit sketches with an arbitrarily good approximation factors. Additionally, for the larger class of monotone subadditive functions, they construct sketches that achieve an approximation of $\sqrt{n}\cdot polylog(n)$. Combining the results of Badanidiyuru et al. [2012] and Balcan and Harvey [2011], a lower bound of $\Omega(n^{1/3}/\log^2 n)$ follows for the approximation of monotone submodular functions by budgeted additive functions (Section 5.7).

Testing of submodular functions has been studied recently by Seshadhri and Vondrák [2011] for general monotone submodular functions and for coverage functions by Chakrabarty and Huang [2012]. The goal here is to query the function on few domain points and answer whether a function is *close* to being submodular or not. The measure of closeness is the fraction of the domain in which the function needs to be modified so as to make it submodular.

5.1.3 Preliminaries and Formal Statement of Results

Given a ground set U, a function $f : 2^U \to \mathbb{R}_+$ is called *submodular* if for all subsets $S, T \subseteq U$, we have $f(S)+f(T) \geq f(S\cup T)+f(S\cap T)$. A submodular function f is called non-negative if $f(S) \geq 0$ for each $S \subseteq U$. In this chapter, we only consider non-negative submodular functions. A submodular function f is called *symmetric* if $f(S) = f(U \setminus S)$

for each $S \subseteq U$, and *monotone* if $f(S) \leq f(T)$ for each $S \subseteq T \subseteq U$. We say that a class of functions $\mathcal{G}$ *θ-approximates* another class $\mathcal{F}$, if for every $f \in \mathcal{F}$ there exists a $g \in \mathcal{G}$ such that $f(S) \leq g(S) \leq \theta \cdot f(S)$ for any $S \subseteq U$. We denote by n the size of the ground set U. We now define certain subclasses of submodular functions that we consider in the chapter.

Definition 5.1.1 (Coverage function)**.** *A function f is a* coverage function *if there exists an auxiliary ground set Z, a weight function $w : Z \to \mathbb{R}_+$ and family of subsets $\{A_i : A_i \subseteq Z, i \in U\}$ such that $\forall S \subseteq U$, $f(S) = \sum_{z \in \cup_{i \in S} A_i} w(z)$.*

Definition 5.1.2 (Budgeted additive function)**.** *A function f is a* budgeted additive function *if there exist non-negative reals a_i for each $i \in U$, and a non-negative real B such that $\forall S \subseteq U$, $f(S) = \min\{B, \sum_{i \in S} a_i\}$.*

It is well known that coverage functions and budgeted additive functions are monotone submodular functions.

Definition 5.1.3 (Cut function)**.** *A function f is a* directed cut function *if there exists a directed graph $G = (U, A)$ with non-negative arc weights $w : A \to \mathbb{R}_+$, such that $\forall S \subseteq U$, $f(S) = w(\delta^+(S))$, where $\delta^+(S)$ denotes the set of outgoing arcs, with their tails in S and heads in $\bar{S}$, and $w(F) \triangleq \sum_{a \in F} w(a)$ for any subset $F \subseteq A$ of arcs.*

Similarly, one can define f to be the *undirected cut function* of an undirected graph by substituting $\delta^+(S)$ with $\delta(S)$, the set of edges with exactly one endpoint in S. It is well known that cut functions, whether directed or undirected, are submodular. Furthermore, clearly, undirected cut functions are symmetric.

Let us now formally state our main results:

Theorem 5.1.4. *Let $f : 2^U \to \mathbb{R}_+$ be a non-negative submodular function with $f(\emptyset) = f(U) = 0$. Then the class of directed cut functions $(n^2/4)$-approximates f. Moreover, there exists a non-negative submodular function $f : 2^U \to \mathbb{R}_+$ with $f(\emptyset) = f(U) = 0$ such that any directed cut function cannot approximate f within a factor better than $n^2/4$.*

Theorem 5.1.5. *Let $f : 2^U \to \mathbb{R}_+$ be a non-negative symmetric submodular function with $f(\emptyset) = 0$. Then the class of undirected cut functions $(n-1)$-approximates f. Moreover, there exists a symmetric submodular function $f : 2^U \to \mathbb{R}_+$ with $f(\emptyset) = 0$ such that any undirected cut function cannot approximate f within a factor better than $n/4$.*

Theorem 5.1.6. *Let $f : 2^U \to \mathbb{R}_+$ be a budgeted additive function. Then coverage functions $(e/(e-1))$-approximates f. Moreover, for every fixed $\varepsilon > 0$, there exists a budgeted additive function $f : 2^U \to \mathbb{R}_+$ such that any coverage function cannot approximate f within a factor better than $e/(e-1) - \varepsilon$.*

Theorems 5.1.4, 5.1.5 and 5.1.6, are proved in Sections 5.2, 5.3 and 5.4 respectively.

5.2 Approximating General Submodular Functions by Directed Cut Functions of Graphs

In this section we prove Theorem 5.1.4 which provides a tight approximation of a non-negative submodular function f using a directed cut function of a graph G. Before proving the main result of this section, we first state a technical lemma.

Lemma 5.2.1. *For every submodular function f, and any collection of sets A_1, A_2, ..., $A_n \subseteq U$: $f(\cap_{i=1}^{n} A_i) \leq \sum_{i=1}^{n} f(A_i)$.*

Proof of Lemma 5.2.1. The following inequalities are derived from the definition of submodularity:

$$\begin{aligned} f(A_1) + f(A_2) &\geq f(A_1 \cap A_2) + f(A_1 \cup A_2) \\ f(A_3) + f(A_1 \cap A_2) &\geq f(A_1 \cap A_2 \cap A_3) + f((A_1 \cap A_2) \cup A_3) \\ &\vdots \\ f(A_n) + f(\cap_{i=1}^{n-1} A_i) &\geq f(\cap_{i=1}^{n} A_i) + f((\cap_{i=1}^{n-1} A_i) \cup A_n) \end{aligned}$$

Summing up the above inequalities and canceling common terms on the two sides and using the fact that f is non-negative we obtain that $\sum_{i=1}^{n} f(A_i) \geq f(\cap_{i=1}^{n} A_i)$. ■

We are now ready to prove Theorem 5.1.4.

Proof of Theorem 5.1.4.
Upper Bound: Given a submodular function f, we construct a directed graph $G = (U, A)$ with non-negative weights w on the arcs such for every $S \subseteq U$, $f(S) \leq w(\delta^+(S)) \leq n^2/4 \cdot f(S)$. For every $(u, v) \in U \times U$ and $u \neq v$, introduce a directed arc from u to v with weight: $w_{uv} = f(T_{uv})$ where $T_{uv} = \text{argmin}\{f(R) : R \subseteq U, u \in R, v \notin R\}$. We start by proving that:

$$f(S) \leq w(\delta^+(S)) \quad \forall S \subseteq U. \tag{5.1}$$

If $S = U$ or $S = \phi$, then clearly $w(\delta^+(S)) = f(S) = 0$ and (5.1) holds. We now restrict our attention to the case where $S, \bar{S} \neq \emptyset$. For any $u \in S$ note that $u \in \cap_{v \in \bar{S}} T_{uv}$, since the

definition of T_{uv} implies that $u \in T_{uv}$ for all $v \in \bar{S}$. Additionally, for any $w \in \bar{S}$ note that $w \notin \cap_{v\in\bar{S}} T_{uv}$, since the definition of T_{uv} implies that $w \notin T_{uw}$. Thus, one can conclude that $\cup_{u\in S} \cap_{v\in\bar{S}} T_{uv} = S$ and therefore,

$$f(S) \overset{\text{(i)}}{\leq} \sum_{u\in S} f(\cap_{v\in\bar{S}} T_{uv}) \overset{\text{(ii)}}{\leq} \sum_{u\in S}\sum_{v\in\bar{S}} w_{uv} = w(\delta^+(S)).$$

Inequality (i) is derived from the fact that f is submodular and non-negative. Inequality (ii) is derived from Lemma 5.2.1 and the definition of w_{uv}. This concludes the proof of (5.1).

We continue by proving that:

$$w(\delta^+(S)) \leq \frac{n^2}{4} f(S) \quad \forall S \subseteq U. \tag{5.2}$$

If $S = U$ or $S = \phi$, then clearly $w(\delta^+(S)) = f(S) = 0$ and (5.2) holds. We now restrict our attention to the case where $S, \bar{S} \neq \emptyset$. Note that for any $u \in S$ and $v \in \bar{S}$, by the definition of T_{uv}: $f(T_{uv}) \leq f(S)$. Thus, one can conclude that:

$$w(\delta^+(S)) \overset{\text{(i)}}{=} \sum_{u\in S}\sum_{v\in\bar{S}} f(T_{uv}) \overset{\text{(ii)}}{\leq} \sum_{u\in S}\sum_{v\in\bar{S}} f(S) \overset{\text{(iii)}}{\leq} \frac{n^2}{4} f(S).$$

Equality (i) is by the definition of weights w_{uv}. Inequality (ii) is be the definition of T_{uv}. Inequality (iii) is by the fact that the number of pairs $(u, v) \in S \times \bar{S}$ is at most $n^2/4$. This concludes the proof of (5.2). Combining both (5.1) and (5.2) concludes the proof of the upper bound of the theorem.

Lower Bound: Assume that n is even and fix an arbitrary $A \subseteq U$ of size $|A| = n/2$. Consider the following function f

$$f(S) = \begin{cases} 1 & \text{if } S \cap A \neq \emptyset, \bar{A} \setminus S \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

Namely, $f(S)$ is the indicator function that S hits A but does not hit all of $\bar{A}$. A simple check shows that f is submodular. Let $G = (U, A)$ be a weighted graph with non-negative weights $w : A \to \mathbb{R}_+$ on the arcs whose directed cut function satisfies for each set $S \subseteq U$, $f(S) \leq w(\delta^+(S)) \leq \theta \cdot f(S)$ for some θ. We will show that $\theta \geq \frac{n^2}{4}$ proving the lower bound.

First, we prove that the arcs with non-zero weight must go from A to $\bar{A}$. We consider the following cases.

1. Consider an edge $(u,v) \in A \times A$. But $(u,v) \in \delta^+(U \setminus \{v\})$ and $w(\delta^+(U \setminus \{v\})) \leq \theta \cdot f(U \setminus \{v\}) = 0$ since $\bar{A} \setminus (U \setminus \{v\}) = \emptyset$. Thus, $w_{(u,v)} = 0$.
2. Consider an edge $(u,v) \in \bar{A} \times \bar{A}$. But $(u,v) \in \delta^+(\bar{A} \setminus \{v\})$ and $w(\delta^+(\bar{A} \setminus \{v\})) \leq \theta \cdot f(\bar{A} \setminus \{v\}) = 0$ since $A \cap (\bar{A} \setminus \{v\}) = \emptyset$. Thus, $w_{(u,v)} = 0$.
3. Consider an edge $(u,v) \in \bar{A} \times A$. But $(u,v) \in \delta^+(\bar{A})$ and , $w(\delta^+(\bar{A})) \leq \theta \cdot f(\bar{A}) = 0$ since $A \cap \bar{A} = \emptyset$. Hence, $w_{(u,v)} = 0$.

Therefore, all arcs with non-zero weight must go from A to $\bar{A}$. For any $u \in A$ and $v \in \bar{A}$ note that $w_{(u,v)} \geq 1$ since:

$$w_{(u,v)} = w(\delta^+\left(\{u\} \cup \left(\bar{A} \setminus \{v\}\right)\right)) \overset{\text{(i)}}{\geq} f\left(\{u\} \cup \left(\bar{A} \setminus \{v\}\right)\right) \overset{\text{(ii)}}{=} 1. \tag{5.3}$$

Inequality (i) is derived from the fact $w(\delta^+(S)) \geq f(S)$ for each set S. Equality (ii) is by the definition of f. Furthermore, note that:

$$\frac{n^2}{4} = |A| \cdot |\bar{A}| \overset{\text{(i)}}{\leq} w(\delta^+(A)) \overset{\text{(ii)}}{\leq} \theta \cdot f(A) \overset{\text{(iii)}}{=} \theta. \tag{5.4}$$

Inequality (i) is derived from inequality (5.3). Inequality (ii) is derived from the fact that $w(\delta^+(S)) \leq \theta f(S)$ for each set $S \subseteq U$. Equality (iii) is by the definition of f. Note that inequality (5.4) implies that $\theta \geq \frac{n^2}{4}$, thus, concluding the proof of the lower bound of the theorem. ■

5.3 Approximating Symmetric Submodular Functions by Undirected Cut Functions of Graphs

In this section, we prove Theorem 5.1.5 which provides upper and lower bounds on the approximation of a symmetric submodular function using an undirected cut function of a graph. For the upper bound, our algorithm uses Gomory-Hu trees of symmetric submodular functions [Gomory and Hu, 1961, Queyranne, 1993]. Given a symmetric non-negative submodular function f, a tree $T = (U, E_T)$ is a Gomory-Hu tree if for every edge $e = (u,v) \in E_T$: $f(R_e) = \min\{f(R) \ : \ R \subseteq U, u \in R, v \notin R\}$, where R_e is one of the two connected components obtained after removing e from T (since f is symmetric, it does not matter which one of the two connected components we choose). In other words, in a Gomory-Hu tree, the cut $e = (u,v)$ induced in T, corresponds to a minimum value subset that separates u and v. We prove that the cut function of the Gomory-Hu tree of f is a good approximation.

Proof of Theorem 5.1.5.
Upper Bound: Let f be a symmetric submodular function. We shall construct an undirected tree $T = (U, E_T)$ with non-negative weights $w : E \to \mathbb{R}_+$ on the edges such that for every $S \subseteq U$, $f(S) \leq w(\delta(S)) \leq (n-1) \cdot f(S)$. We set T to be a Gomory-Hu tree of f and let the weight of any edge $e = \{u, v\}$ to be $f(R_e)$ where R_e is the one of the two connected components obtained after removing edge e. As mentioned above, the weight of edge $e = \{u, v\}$ is the minimum of $f(R)$ over all R separating u and v.

Fix an arbitrary $S \subseteq U$ and denote by $\{e_1, \ldots, e_k\}$ all the edges crossing the cut that S defines in T. Let $T_1, \ldots, T_{k+1}$ denote the partition of U induced by deleting the edges $e_1, \ldots, e_k$ from T. Furthermore, denote by $\{S_1, \ldots, S_p\}$ the non-empty sets in $\{T_i \cap S : 1 \leq i \leq k+1\}$. Observe that $S_1, \ldots, S_p$ is a partition of S. Since each e_i, $1 \leq i \leq k$, has exactly one vertex in S and the other in $\bar{S}$, we can associate e_i with a *unique* set from $S_1, \ldots, S_p$, the set containing one of the endpoints of e. Additionally, let us denote F_i to be the edges which are associated with set S_i for each $1 \leq i \leq p$. Clearly $F_1, \ldots, F_p$ form a partition of $\{e_1, \ldots, e_k\}$.

We claim that for every $1 \leq i \leq p$:

$$f(S_i) \leq \sum_{f \in F_i} f(R_f). \tag{5.5}$$

Recall that R_f is one of the connected component after removing edge f from T. Since S_i is a subset of a connected component formed after removing all the edges $\{e_1, \ldots, e_k\}$ from T, it must be contained in one of the components formed after removing edge $f \in F_i$ from T. Without loss of generality, we assume that $R_f \cap S_i = \emptyset$ for each edge $f \in F_i$. It is straightforward to see that $\cap_{f \in F_i} \bar{R}_f = S_i$. Now, we have

$$\sum_{f \in F_i} f(R_f) \overset{\text{(i)}}{\geq} f\left(\cup_{e \in F_i} R_f\right) \overset{\text{(ii)}}{=} f\left(\overline{\cup_{f \in F_i} R_f}\right) = f\left(\cap_{e \in F_i} \bar{R}_f\right) \overset{\text{(iii)}}{=} f(S_i).$$

Inequality (i) is derived from the fact that f is submodular and non-negative. Equality (ii) is derived from the symmetry of f. Equality (iii) is derived from the fact that $\cap_{f \in F_i} \bar{R}_f = S_i$.

We start by proving that:

$$f(S) \leq w(\delta(S)) \quad \forall S \subseteq U. \tag{5.6}$$

This can be proved as follows:

$$w(\delta(S)) \overset{\text{(i)}}{=} \sum_{i=1}^{k} f(R_{e_i}) \overset{\text{(ii)}}{=} \sum_{i=1}^{p} \sum_{f \in F_i} f(R_f) \overset{\text{(iii)}}{\geq} \sum_{i=1}^{p} f(S_i) \overset{\text{(iv)}}{\geq} f\left(\cup_{i=1}^{p} S_i\right) = f(S).$$

Equality (i) is by the definition of edge weights in T. Equality (ii) is by the fact that $F_1, \ldots, F_p$ form a partition of $\{e_1, \ldots, e_k\}$. Inequality (iii) is derived from inequality (5.5). Inequality (iv) is derived from the fact that f is submodular and non-negative. This concludes the proof of (5.6).

We continue by proving that:

$$w(\delta(S)) \leq (n-1) \cdot f(S) \quad \forall S \subseteq U. \tag{5.7}$$

Let u and v be the endpoints of edge e_i and without loss of generality assume that $u \in S$ and $v \notin S$. Note that for every $1 \leq i \leq k$, $f(R_{e_i}) \leq f(S)$ since S is a candidate set separating u and v. Hence, one can conclude that:

$$w(\delta(S)) \overset{\text{(i)}}{=} \sum_{i=1}^{k} f(R_{e_i}) \overset{\text{(ii)}}{\leq} k \cdot f(S) \overset{\text{(iii)}}{\leq} (n-1) \cdot f(S).$$

Equality (i) is by the definition of edge weights in T. Inequality (ii) is what we proved above, and inequality (iii) is derived from the fact that T contains at most $n-1$ edges, thus, $k \leq n-1$. This concludes the proof of (5.7). Combining both (5.6) and (5.7) concludes the proof of the upper bound of the theorem.

Lower Bound: Consider the following symmetric submodular function f:

$$f(S) = \begin{cases} 1 & \text{if } S \neq \emptyset, U \\ 0 & \text{otherwise} \end{cases}$$

Let $G = (U, E)$ be an edge weighted graph with non-negative weights $w : E \to \mathbb{R}_+$ on the edges whose cut function satisfies $f(S) \leq w(\delta(S)) \leq \theta \cdot f(S)$ for each set $S \subseteq U$ for some θ. We will show that $\theta \geq \frac{n}{4}$.

For any vertex $v \in U$, $1 = f(\{v\}) \leq w(\delta(\{v\}))$. Thus, the total weight of edges in G is at least $\frac{1}{2}\sum_{v \in U} w(\delta(\{v\})) \geq \frac{n}{2}$. Every undirected graph has a non-trivial cut that contains at least half the total weight of edges in the graph, thus, there exists a cut $S \subseteq U$, $S \neq \emptyset, U$, where $w(\delta(S)) \geq \frac{n}{4}$. The existence of such a cut can be shown by picking a cut at random where each vertex is in S with probability $\frac{1}{2}$ independently. The expected weight of the cut will be exactly half the total weight of all edges. Now, we have $\frac{n}{4} \leq w(\delta(S)) \leq \theta \cdot f(S) = \theta$, concluding the proof of the lower bound of the theorem. ■

5.4 Approximating Budgeted Additive Functions by Coverage Functions

In this section, we present matching upper and lower bounds for approximating budgeted additive functions by coverage functions (Theorem 5.1.6)[1]. The following lemma from Chakrabarty and Huang [2012] provides the alternate representation of coverage functions used in our proof of the lower bound.

Lemma 5.4.1. *[Chakrabarty and Huang, 2012] A function $f : 2^U \to \mathbb{R}_+$ is a coverage function if and only if there exist reals $x_T \geq 0$ for each $T \subseteq U$ such that $f(S) = \sum_{T:T\cap S\neq\emptyset} x_T$ for each $S \subseteq U$.*

Proof of Theorem 5.1.6.
Upper Bound: Consider any budgeted additive function $f(\cdot)$ over some domain U, with budget B and the values of the elements be denoted by $v_1, v_2, \cdots v_n$ where $n = |U|$. Without loss of generality, we assume all these values to be integers. Take an auxiliary ground set G of size B. For each $i \in U$, construct a set $A_i \subseteq G$, formed by choosing v_i points (with replacement) at random from G. Consider function $g : 2^U \to \mathbb{Z}$, defined as $g(S) = |\cup_{i\in S} A_i|$ for all $S \subseteq U$.

By definition, $g(\cdot)$ is a coverage function. Furthermore, it is easy to see that for all $S \subseteq U$, $g(S) \leq f(S)$. We now show that $\mathbb{E}[g(S)] \geq (1 - 1/e) \cdot f(S)$, where the expectation is taken over the randomness of the procedure described to construct $g(S)$. Note that $g'(\cdot) = \mathbb{E}[g(\cdot)]$ is a coverage function. Consider any set $S \subseteq U$. Let $f(S) = V$, i.e., $\sum_{i\in S} v_i = V$. Consider the case when $V < B$. Consider any point in auxiliary ground set G. The probability that this point is not covered by any of the sets A_i for $i \in S$ is at most $(1-1/B)^V$. Hence, the expected value of $|\cup_{i\in S} A_i|$ is at least $B\cdot(1-(1-1/B)^V) \geq B \cdot (1 - e^{-V/B}) \geq (1 - 1/e) \cdot V$. Here we use the inequality $1 - e^{-x} \geq (1 - 1/e) \cdot x$. Hence, $|\cup_{i\in S} A_i| \geq (1-1/e) \cdot f(S)$. The proof for the case when $V = B$ is similar. Thus for each set $S \subseteq U$, we have

$$(1 - \frac{1}{e})f(S) \leq g'(S) \leq f(S).$$

Thus, we obtain the function $\frac{e}{e-1}g'(\cdot)$ approximates f within a factor of $\frac{e}{e-1}$.

Lower Bound: We will construct a budgeted additive function which cannot be approximated by coverage functions to factor better than $\frac{e}{e-1} - \epsilon$ for any $\epsilon > 0$. We will consider the family of budgeted additive function f_k, parameterized by the size of domain $|U| = n$

[1]It is easy to show that a coverage function can be written exactly as a sum of budgeted additive functions.

they are defined on, where $n = k^2$ for some integer k. Under f_k, all $n = k^2$ items have value one and the budget is k. Please note that these also constitute a family of uniform matroid rank functions. Therefore,

$$f_k(S) = \begin{cases} |S| & \text{if } |S| \leq k \\ k & \text{o.w.} \end{cases} \tag{5.8}$$

Let h_k be a coverage function that gives the maximum value of β such that $\forall S \subset [n],\ \ \beta \cdot f_k(S) \leq h_k(S) \leq f_k(S)$ and α_k be the value of β as given by h_k. Observe that here function h_k is always smaller than the function f_k. The function $\frac{h_k}{\beta}$ would give a $\frac{1}{\beta}$-approximation for approximating function f_k. This slight change in notation helps for exposition below. We shall show that as $k \to \infty$, α_k tends to a value that is at most $1 - 1/e$. This shall prove our claim.

Using Lemma 5.4.1, we note that α_k can be characterized by a solution to a linear problem (P) given below. Here, the variables are x_T, one for each set $T \subseteq U$. The dual (D) of this linear program is given alongside. We will construct a dual solution of value approaching $1 - \frac{1}{e}$ as $k \to \infty$. Since every feasible dual solution is an upper bound on α_k, the result follows.

$$\begin{aligned} & \max \alpha_k && \text{(P)} \\ & \text{subject to} \\ & \forall S \subseteq U, \quad \sum_{T \cap S \neq \phi} x_T \leq f_k(S) \\ & \forall S \subseteq U, \quad \alpha_k f_k(S) - \sum_{T \cap S \neq \phi} x_T \leq 0 \\ & \forall S \subseteq U, \quad x_S \geq 0 \end{aligned}$$

$$\begin{aligned} & \min \sum_{S \subseteq U} f_k(S) \cdot u_S && \text{(D)} \\ & \text{subject to} \\ & \forall S \subseteq U, \sum_{T \cap S \neq \phi} (u_T - v_T) \geq 0 \\ & \sum_{S \subseteq U} f_k(S) \cdot v_S \geq 1 \end{aligned}$$

Since, $f(\cdot)$ is symmetric across sets of the same cardinality, we can assume, without loss of generality, that the optimal dual solution is also symmetric. Specifically, the values of the dual variables u_T and v_T shall depend only the cardinality $|T|$. Let us write the symmetrized dual program.

$$\min \quad \sum_{j=1}^{k} j \cdot \binom{n}{j} \cdot u_j + \sum_{j=k+1}^{n} k \cdot \binom{n}{j} \cdot u_j \quad \text{Symmetrized Dual Program}$$

subject to

$$\forall j \in [n], \quad \sum_{i=1}^{n} \left(\binom{n}{i} - \binom{n-j}{i} \right) (u_i - v_i) \geq 0 \tag{5.9}$$

$$\sum_{j=1}^{k} j \cdot \binom{n}{j} \cdot v_j + \sum_{j=k+1}^{n} k \cdot \binom{n}{j} \cdot v_j \geq 1 \tag{5.10}$$

Let c_j denote the coefficient of v_k in the equation corresponding to set size j, i.e., $c_j = \binom{n}{k} - \binom{n-j}{k}$. Further, define $\Delta c_j = c_{j+1} - c_j$.

We give the following solution to the dual linear program. Let $v_k = \frac{1}{\binom{n}{k} \cdot k}$, $u_1 = \Delta c_k \cdot v_k$ and $u_n = (c_k - k \cdot \Delta c_k) \cdot v_k$. Rest of the variables are set to zero.

We first show that the above solution is feasible for the dual and has objective value that tends to $1 - 1/e$ as $k \to \infty$. It is easy to see that with the proposed setting of v_k, Equation (5.10) is satisfied. To show that Equation (5.9) is satisfied, we show that $\forall j \in [n]$, $j \cdot u_1 + u_n \geq (\binom{n}{k} - \binom{n-j}{k}) v_k$. Using our notation, it suffices to show that for all $j \in [n]$, $(j - k) \cdot \Delta c_k + c_k \geq c_j$.

Claim 5.4.2. *c_j is an increasing function of j and $\Delta c_j = c_{j+1} - c_j$ is a decreasing function of j.*

Proof. $\Delta c_j = c_{j+1} - c_j = (\binom{n}{k} - \binom{n-j-1}{k}) - (\binom{n}{k} - \binom{n-j}{k}) = \binom{n-j}{k} - \binom{n-j-1}{k} = \binom{n-j-1}{k-1}$. ■

Claim 5.4.3. *For all $j \in [n]$, $(j - k) \cdot \Delta c_k + c_k \geq c_j$.*

Proof.

- For $j = k + i$ such that $0 \leq i \leq n - k$: LHS = $i \cdot \Delta c_k + c_k \geq c_{k+i}$ = RHS
- For $j = k - i$ with $0 \leq i \leq k$: LHS = $c_k - i \cdot \Delta c_k \geq c_k - \sum_{q=k-i}^{k-1} \Delta c_q = c_{k-i} \cdot v_k$ = RHS

where the second inequality in both the cases follows because Δc_j is a decreasing function in j. ■

Let us now bound the value of the dual objective function. Look at the value that the dual objective function attains with this setting of variables. The function value is $n \cdot u_1 + k \cdot u_n = (n \cdot \Delta c_k + k \cdot (c_k - k \cdot \Delta c_k)) \cdot v_k$. Since $n = k^2$, the dual objective value is equal to

$$k \cdot c_k \cdot v_k = 1 - \frac{\binom{k^2-k}{k}}{\binom{k^2}{k}}.$$

This quantity tends to $1 - 1/e$ as $k \to \infty$. ■

5.5 Uniform Submodular and Matroid Rank Functions

Definition 5.5.1 (Uniform Submodular Function)**.** *A submodular function is said to be* uniform *if the value it takes on a set depends only on the cardinality of the set.*

Lemma 5.5.2. *Any non-negative, integer-valued, monotone, uniform, submodular function is* 1*-approximated by a sum of uniform matroid rank functions, and hence by a sum of budgeted additive functions.*

Proof. Consider an integer-valued, non-negative, monotone, uniform, submodular function $f(\cdot)$ over the universe $[n]$ and let f_k be the value f takes for sets S of cardinality k. Consider uniform matroid rank functions $g_1, g_2, \cdots g_n$ where

$$g_i(S) = \begin{cases} |S| & |S| \leq i \\ i & |S| > i \end{cases} \tag{5.11}$$

We claim that there exist a set of α_i's, such that $\alpha_i \geq 0$ for all $i \in [n]$ such that

$$\forall j \in [n], f_j = \sum_{i=1}^{k} \alpha_i \cdot j + \sum_{i=k+1}^{n} \alpha_i \cdot i \tag{5.12}$$

It is easy to see that the above claim implies that $f(S) = \sum_i \alpha_i \cdot g_i(S)$ for all $S \subseteq [n]$.

Now, we prove the claim. If, for every $j \in [n-1]$, we substract equation j from $j+1$, we get the following set of equations

$$\forall j \in [n-1], f_{j+1} - f_j = \sum_{i=j+1}^{n} \alpha_i \tag{5.13}$$

From here, we can see that the following assignments to α_i is a valid solution to the above equations. Set $\alpha_1 = 2 \cdot f_2 - f_1, \alpha_n = f_n - f_{n-1}$ and for $i \notin \{1, n\}$, set $\alpha_i = 2 \cdot f_i - f_{i+1} - f_{i-1}$. All the α_i's are positive since f is monotone and f is submodular.

Finally, it is easy to see that every uniform matroid rank function is also a budgeted additive function with all elements having value one, and the budget equal to the rank of the matroid. ■

5.6 Application to Online Submodular Function Maximization

We consider the problem of online submodular function maximization as studied by Buchbinder et al. [2012]. We are given a universe U and matroid $\mathcal{M} = (U, \mathcal{I})$. In an online manner, at each step for $1 \leq i \leq m$, we are given a monotone submodular function $f_i : 2^U \to \mathbb{R}_+$. The goal is to maintain an independent set $F_i \in \mathcal{I}$ at any step i such that $F_i \subseteq F_{i+1}$. The objective value to maximize is $\sum_{i=1}^{m} f_i(F_i)$. As in the notion of competitive analysis, any algorithm is compared to the best offline optimum $\max_{O \in \mathcal{I}} \sum_{i=1}^{m} f_i(O)$.

Buchbinder et al. [2012] give a $O(\log^2 n \log m \log f_{ratio})$-competitive algorithm when each of the submodular function is weighted matroid rank function where

$$f_{ratio} = \frac{\max_{i,a} f_i(\{a\})}{\min_{i,a:f_i(\{a\})>0} f_i(\{a\})}.$$

In particular, the result applies when each of the functions f_i is a coverage function.

Using the fact every monotone submodular function can be approximated by a coverage function to a factor of $O(\sqrt{n} \log n)$, we directly obtain the following corollary.

Corollary 5.6.1. *There is a $O(\sqrt{n} \log^3 n \log m f_{ratio})$-competitive online algorithm for the online submodular function maximization problem when each of the submodular functions is an arbitrary monotone submodular function.*

5.7 Approximating Monotone Submodular Functions by Coverage Functions and by Budgeted Additive Functions

The two main results of the section are the following.

Theorem 5.7.1. *Coverage functions can approximate every non-negative monotone submodular function to within a factor $O(\sqrt{n}\log n)$. Additionally, the class of coverage functions cannot approximate every non-negative monotone submodular function to a factor within $o\left(\frac{n^{1/3}}{\log^2 n}\right)$.*

Theorem 5.7.2. *The class of sum of budgeted additive functions can approximate every non-negative monotone submodular function to a factor within $(\sqrt{n}\log n)$. Additionally, the class of sum of budgeted additive functions cannot approximate every non-negative monotone submodular function to a factor within $o\left(\frac{n^{1/3}}{\log^2 n}\right)$.*

5.7.1 Upper Bound

For the upper-bound, we show that budgeted additive functions can $\sqrt{n}\log(n)$-approximate the class of non-negative, monotone, submodular functions. Then we use Theorem 5.1.6 to infer that the coverage functions too can give approximately the same guarantee.

Lemma 5.7.3. *The class of sum of budgeted additive functions can approximate every non-negative monotone submodular function to within a factor $(\sqrt{n}\log n)$.*

The following corollary follows easily from Lemma 5.7.3 and Theorem 5.1.6

Corollary 5.7.4. *The class of sum of coverage functions can approximate every non-negative monotone submodular function to factor $O(\sqrt{n}\log n)$.*

The proof of Lemma 5.7.3 follows from Lemmas 5.7.5 and 5.7.6. Lemma 5.7.5 [Goemans et al., 2009] gives a particular function that $\sqrt{n}\log(n)$-approximates a general monotone, non-negative, sub-modular function, and Lemma 5.7.6 implies that this approximating function can be written as a sum of budgeted additive functions.

Lemma 5.7.5. *[Goemans et al., 2009] For every monotone submodular function $f : 2^U \to R_+$, there exists positive reals a_e for each $e \in E$ such that $g : 2^U \to R_+$ defined as $g(S) = \sqrt{\sum_{e\in S} a_e}$, approximates f within factor $\sqrt{n}\log(n)$.*

Lemma 5.7.6. *Every submodular function $f : 2^{[n]} \to \mathbb{Z}^+$ of the form $f(S) = g(\sum_{i\in S} a_i)$, where $a_i \in \mathbb{Z}^+$ and g is a non-negative, monotone, concave and integer valued on integral inputs, can be written as a sum of budgeted additive functions.*

Proof. Let $m = \sum_{i=1}^{n} a_i$. Consider the function h, over the domain $[m]$, defined as $h(S) = g(|S|)$ for all $S \subseteq [m]$. Since $g(\cdot)$ is a non-negative, monotone, concave function, it is easy

to verify that $h(\cdot)$ is a non-negative, monotone, submodular function. Construct n mutually disjoint sets $A_i \subseteq [m]$ such that $|A_i| = a_i$. Clearly, $\forall S \subseteq [n], f(S) = h(\cup_{i \in S} A_i)$.

From Lemma 5.5.2, we know that $h(\cdot)$ can be expressed as $\sum_{i=1}^{m} \alpha_i \cdot t_i(\cdot)$ where each t_i is a uniform matroid rank function with rank i, over the domain $[m]$ and each $\alpha_i \geq 0$.

This implies that for all $S \subseteq [n]$, $f(S) = \sum_{i=1}^{m} \alpha_i \cdot t_i(\cup_{i \in S} A_i)$. Now, for every $i \in [m]$, construct the budget additive function t'_i, defined as $\forall S \subseteq [n], t_i(S) = \min\{\sum_{i \in S} v_{ij}, B_i\}$, where for $j \in [n]$, the value $v_{ij} = a_j$ and the budget B_i is i. Since t_i is a uniform matroid rank function of rank i and A_i's are mutually disjoint, we have for all $i \in [m]$,

$$\forall S \subseteq [n], t'_i(S) = t_i(\cup_{i \in S} A_i) \tag{5.14}$$

Therefore, we get for all sets $S \subseteq [n]$, $f(S) = \sum_{i=1}^{m} t'_i(S)$. ■

5.7.2 Lower Bound

For the lower bound, we first show that sum of coverage functions cannot approximate the class of monotone, submodular functions well, and then use Theorem 5.1.6, to infer that, therefore, even the class of sum of budgeted additive functions cannot approximate a monotone submodular function well.

Lemma 5.7.7. *The class of sum of coverage functions cannot approximate every non-negative monotone submodular function to a factor within* $o\left(\frac{n^{1/3}}{\log^2 n}\right)$.

An easy corollary of Lemma 5.7.7 that follows from Theorem 5.1.6 is the following.

Corollary 5.7.8. *The class of sum of budgeted additive functions cannot approximate every non-negative monotone submodular function to a factor within* $o\left(\frac{n^{1/3}}{\log^2 n}\right)$.

We now present the proof of Lemma 5.7.7. We will need to use results from Badanidiyuru et al. [2012] and Balcan and Harvey [2011], for which we first present a definition.

Definition 5.7.9. *A* β*-sketch of a function* $f : 2^U \to \mathbb{R}$ *is a polynomially sized (in* $|U|$ *and* $1/(1-\beta)$*) representable function* g *such that* $\forall S \subseteq U$, $\beta \cdot f(S) \leq g(S) \leq f(S)$.

The following result is from Badanidiyuru et al. [2012].

Lemma 5.7.10. *[Badanidiyuru et al., 2012] Coverage functions allow from arbitrary well sketches i.e., for any* $\epsilon > 0$*, there exists a* $1 - \epsilon$ *sketch.*

The following result is from Balcan and Harvey [2011]. It gives a 'large' family of matroid rank functions, such that any two functions in the class have at least one point where the values that they take differ by a 'significant' factor.

Lemma 5.7.11. *[Balcan and Harvey, 2011] For any $k = 2^{o(n^{1/3})}$, there exists a family of sets $\mathcal{A} \subseteq 2^{[n]}$ with $|\mathcal{A}| = k$ and a family of matroids $\mathcal{M} = \{M_B | B \subseteq \mathcal{A}\}$ such that for all $B \subseteq \mathcal{A}$, it is the case that*

$$\forall S \in \mathcal{A}, r_{M_B}(S) = \begin{cases} 8 \log k & \text{if } S \in B \\ n^{1/3} & \text{if } S \notin B \end{cases} \tag{5.15}$$

where r_{M_B} is the rank function of the matroid M_B

Proof of Lemma 5.7.7. Let the class of matroid rank functions on the domain of size n be α-approximable by coverage functions, for some α. That is, for a domain $[n]$, for all matroid rank function r, there exists a coverage function g such that $\forall S \subseteq [n], r(S) \leq g(S) \leq \alpha \cdot r(S)$.

By Lemma 5.7.10, for every $\epsilon > 0$ and every coverage function g, there exists a polynomially sized (polynomial in n and $1/\epsilon$) representable function h such that $\forall S \subseteq [n], (1-\epsilon) \cdot g(S) \leq h(S) \leq g(S)$. Hence, for all $\epsilon > 0$ and for all matroid rank functions r, there exists a polynomial sized representable function h such that $\forall S \subseteq [n], r(S) \leq h(S)/(1-\epsilon) \leq \frac{\alpha}{1-\epsilon} \cdot r(S)$. For any given $\epsilon > 0$, there are only $2^{O(n,1/\epsilon)}$ many different h functions.

From Lemma 5.7.11, for $k = 2^{\log^2(n)}$, there exists family of sets $\mathcal{A} \subseteq 2^{[n]}$ with $|\mathcal{A}| = k$, and a 2^k sized matroid family $\mathcal{M}_B$ such that for all sets $A \in \mathcal{A}$ and $\forall B \subseteq \mathcal{A}$,

$$\forall S \in \mathcal{A}, r_{M_B}(S) = \begin{cases} 8 \log^2 n & \text{if } S \in B \\ n^{1/3} & \text{if } S \notin B \end{cases} \tag{5.16}$$

Now while the number of different g' functions are $2^{O(n,1/\epsilon)}$, the number of different matroid rank functions in this family is $2^{n^{\log(n)}}$. Hence, by pigeon-hole principle, there must be two matroids B and B' ($B \neq B'$) such that the best coverage functions g and g' approximating B and B' respectively, have the same best polysized representation h. But since, for every set $S \in B \Delta B'$, r_{M_B} and $r_{M_{B'}}$, differ by a factor of $\Omega(n^{1/3}/\log^2(n))$, therefore, h cannot approximate at least one of these two to a factor better $\Omega(n^{1/3}/\log^2(n))$. Since the value of g and g' at any point in the domain is off from that of h by at most $1-\epsilon$, and hence it follows that $\alpha = \Omega(n^{1/3}/\log^2(n))$. ■

5.8 Future Directions

We mention here a couple of main research questions that are left open by the present work. The first is how well a non-negative monotone submodular function can be approximated by the sum of matroid rank functions. Dughmi et al. [2011] show that the Hessian matrix for a matroid rank sum has to be negative semi-definite, and it is easy to come with a budgeted additive function that does not obey this property. Hence, we cannot hope for the best approximation factor for a submodular function by a matroid rank sum to be 1; in fact we can show that the approximation factor cannot be better than some constant bounded away from 1. In terms of positive results, a $O(\sqrt(n))$ factor approximation follows from Goemans et al. [2009] and a $O(\frac{\max_{e \in U} f(e)}{\min_{e \in U} f(e)})$ follows from a result in Section 44.6(B) in Schrijver [2003].

The second is approximating a non-negative symmetric submodular function by a hypergraph cut function (in this chapter, we only considered graph cut functions). The lower bound example in the chapter for graph cut functions can be extended to show that a r-regular hypergraph cannot approximate to a factor better than $O(\frac{n}{r})$. In terms of positive results, we know no better than the ones mentioned in this chapter.

5.9 Acknowledgment

The results in this chapter are part of a joint work with Nikhil R. Devanur, Shaddin Dughmi, Roy Schwartz and Mohit Singh [Devanur et al., 2013].

Chapter 6

Directions for Future Research

6.1 Online resource allocation with preemption

In online resource allocation, we know the resources and their limitation upfront, and the claimants arrive online, and at the time a claimant arrives, we need to decide whether or not to allocate resources to her. An important dimension along which various settings of online resource allocation vary, and which will be the main topic for our study, is *whether or not the allocation decisions made are reversible*. Consider two practical instances of online resource allocation – online retail and online job scheduling, to understand the distinction between reversible and irreversible allocation decisions. Online retail websites (such as Amazon) allocate their inventory to an online stream of buyers. In this setting, once a piece of inventory has been sold (and hence allocated) to a buyer, the sold good cannot be reclaimed from the user in future i.e., allocation decisions made are *irreversible*. In online job scheduling, jobs arriving online are allocated the resources of the servers. In this setting, allocation decisions made are *reversible*, i.e., jobs that were allocated resources at an earlier time step, can be evicted, possibly at the cost of losing their value. Note that in reversible allocation settings, we allow only positive allocation decisions to be reversed; any negative allocation decisions made earlier cannot be reversed. In other words, the allocation mechanism can only take back the resources from earlier claimants, and it cannot give more resources to earlier claimants. Furthermore, once a positive allocation is nullified, it cannot be reversed any time in future.

The reason it is interesting to consider the dimension of reversibility/irreversibility of allocation decisions is because the freedom of reversing previous allocation decisions potentially allows the allocation mechanism to achieve a much better welfare. It is easy to

show that no allocation mechanism, regardless of computational complexity, even with reversible allocation decisions, can achieve the offline optimal welfare. With irreversible allocation decisions, in many settings of interest, we know, regardless of computational complexity, tight approximation ratios that can be achieved with respect to the offline optimal welfare. For instance, with k resources, and each claimant being a single-minded buyer with value 1 and claiming at most a small fraction of any resource, we know of an online allocation mechanism that achieves $\Omega(1/\log(k))$ fraction of offline optimal welfare [Awerbuch et al., 1993], and we have examples showing that no online randomized allocation mechanism can achieve in expectation a fraction better than $O(1/\log(k))$.

With reversible allocation decisions, however, the theoretical picture is much less understood. In many general settings, we do not know whether we can achieve a constant fraction of the optimal welfare or whether it is only possible to achieve a logarithmic fraction (as guaranteed by algorithms designed for the irreversible setting). While some limited progress has been made on specific settings [Adler and Azar, 1999, Azar et al., 2010, Alon et al., 1999], the general setting is much less understood.

6.2 Other directions for future research

In each of the previous chapters, we listed the main open research questions for future research. We summarize them here.

1. For allocation settings with procurement cost, our analysis is restricted to *online* settings with *adversarial* stream of *self-interested* buyers. Can we achieve better welfare guarantees by relaxing some of these constraints? For instance, is it possible to achieve better welfare guarantees in the *offline* setting with self-interested buyers, in the offline setting with *no game theoretic constraints*, and in the online setting with a *uniformly random buyer ordering* or with *buyers drawn from a known distribution*?

2. For resource allocation with stochastic valuations, can we design a *non-adaptive* query algorithm that can achieve, in expectation, a $(1-\epsilon)$ fraction of the omniscient optimum for stochastic matching while making only $O_\epsilon(1)$ queries?

3. In analyzing first- and second-price auctions, can we compute equilibrium bidding functions in settings where the spite factors of the different bidders are not public information? Rather, the spite factors are drawn from a publicly known, and potentially, bidder-specific distribution.

4. Is it possible, regardless of computational constraints, to approximate any non-negative monotone submodular function by a positive linear combination of matroid rank functions (each defined over the same universe of elements as the submodular function) to within a constant factor?

Bibliography

D. J. Abraham, A. Blum, and T. Sandholm. Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges. In *Proceedings of the 8th ACM Conference on Electronic Commerce (EC)*, pages 295–304, 2007. 3.5, 3.7.2

M. Adamczyk. Improved analysis of the greedy algorithm for stochastic matching. *Information Processing Letters*, 111(15):731–737, 2011. 1.3.3, 3.1.3

Ran Adler and Yossi Azar. Beating the logarithmic lower bound: Randomized preemptive disjoint paths and call control algorithms. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '99, pages 1–10, 1999. 6.1

Noga Alon, Uri Arad, and Yossi Azar. Independent sets in hypergraphs with applications to routing via fixed paths. In *Randomization, Approximation, and Combinatorial Optimization. Algorithms and Techniques*, volume 1671 of *Lecture Notes in Computer Science*, pages 16–27. Springer Berlin Heidelberg, 1999. 6.1

Aaron Archer, Christos Papadimitriou, Kunal Talwar, and Éva Tardos. An approximate truthful mechanism for combinatorial auctions with single parameter agents. *Internet Math.*, 1(2), 2004. 1.2.3

I. Ashlagi and A. Roth. Individual rationality and participation in large scale, multi-hospital kidney exchange. In *Proceedings of the 13th ACM Conference on Electronic Commerce (EC)*, pages 321–322, 2011. 3.8.4, 3.8.4, 3.8.5

B. Awerbuch, Y. Azar, and S. Plotkin. Throughput-competitive on-line routing. In *FOCS*, 1993. 2.8.5, 6.1

Baruch Awerbuch, Yossi Azar, and Adam Meyerson. Reducing truth-telling online mechanisms to online optimization. In *STOC*, 2003. 1.2.2, 1.2.3, 2.2.1, 2.9

Yossi Azar, Uriel Feige, and Daniel Glasner. A preemptive algorithm for maximizing disjoint paths on trees. *Algorithmica*, 57(3):517–537, 2010. ISSN 0178-4617. doi: 10.1007/s00453-009-9305-4. URL http://dx.doi.org/10.1007/s00453-009-9305-4. 6.1

Ashwinkumar Badanidiyuru, Shahar Dobzinski, Hu Fu, Robert Kleinberg, Noam Nisan, and Tim Roughgarden. Sketching valuation functions. In *SODA*, pages 1025–1035, 2012. 1.5.2, 1.5.3, 5.1, 5.1.1, 5.1.1, 5.1.2, 5.7.2, 5.7.2, 5.7.10

Maria-Florina Balcan and Nicholas J. A. Harvey. Learning submodular functions. In *STOC*, pages 793–802, 2011. 1.5.2, 1.5.3, 5.1, 5.1.1, 5.1.1, 5.1.2, 5.7.2, 5.7.2, 5.7.11

Maria-Florina Balcan, Avrim Blum, and Yishay Mansour. Item pricing for revenue maximization. In *EC*, 2008. 1.2.1, 2.2.1, 2, 6, 2

Maria-Florina Balcan, Nicholas J.A. Harvey, and Satoru Iwata. Learning symmetric non-monotone submodular functions. In *Discrete Optimization in Machine Learning (DISCML)*, 2012. 5.1.2

N. Bansal, A. Gupta, J. Li, J. Mestre, V. Nagarajan, and A. Rudra. When LP is the cure for your matching woes: Improved bounds for stochastic matchings. *Algorithmica*, 63 (4):733–762, 2012. 1.3.3, 3.1.3

Yair Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *FOCS*, pages 184–193, 1996. 5.1

Yair Bartal, Rica Gonen, and Noam Nisan. Incentive compatible multi unit combinatorial auctions. In *TARK*, 2003. 1.2, 1.2.1, 1.2.2, 1.2.3, 2.2.1, 2.7, 2.8.5

Avrim Blum, Anupam Gupta, Yishay Mansour, and Ankit Sharma. Welfare and profit maximization with production costs. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, FOCS '11, pages 77–86, 2011. 2.10

Avrim Blum, Anupam Gupta, Ariel Procaccia, and Ankit Sharma. Harnessing the power of two crossmatches. In *Proceedings of the fourteenth ACM conference on Electronic commerce*, EC '13, pages 123–140, 2013. 3.10

Avrim Blum, Nika Haghtalab, Ariel Procaccia, and Ankit Sharma. Ignorance is almost bliss: Near-optimal stochastic matching with few queries. *CoRR*, abs/1407.4094, 2014. 3.10

Liad Blumrosen and Noam Nisan. Combinatorial auctions. In *Algorithmic Game Theory*. Cambridge University Press, 2007. 1.2.3

J. Bourgain. On lipschitz embedding of finite metric spaces in Hilbert space. *Israel Journal of Mathematics*, 52(1):46–52, March 1985. 5.1

Sviatoslav Brainov. The role and the impact of preferences on multiagent interaction. In Nick Jennings and Yves Lespérance, editors, *Intelligent Agents VI, LNAI 1757*, pages 349–363. Springer-Verlag, 2000. 4.1

Felix Brandt and Gerhard Weiß. Antisocial agents and Vickrey auctions. In J.-J. Ch. Meyer and M. Tambe, editors, *Intelligent Agents VIII, LNAI 2333*, 2001. 4.1

Felix Brandt, Tuomas Sandholm, and Yoav Shoham. Spiteful bidding in sealed-bid auctions. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI)*, Hyderabad, India, 2007. Early version in GTDT-05. 4.1, 4.4, 4.5.1

Patrick Briest, Piotr Krysta, and Berthold Vöcking. Approximation techniques for utilitarian mechanism design. In *STOC*, 2005. 1.2.3

Niv Buchbinder, Joseph Naor, R. Ravi, and Mohit Singh. Approximation algorithms for online weighted rank function maximization under matroid constraints. In *ICALP (1)*, pages 145–156, 2012. 5.1, 5.6

David Buchfuhrer, Michael Schapira, and Yaron Singer. Computation and incentives in combinatorial public projects. In *ACM EC*, pages 33–42, 2010. 5.1

Deeparnab Chakrabarty and Zhiyi Huang. Testing coverage functions. In *ICALP (1)*, pages 170–181, 2012. 1.5.3, 5.1.2, 5.4, 5.4.1

N. Chen, N. Immorlica, A. R. Karlin, M. Mahdian, and A. Rudra. Approximating matches made in heaven. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 266–278, 2009. 1.3.3, 3.1.3

G. Cornuejols, M. L. Fisher, and G. L. Nemhauser. Location of bank accounts to optimize float: an analytic study of exact and approximate algorithms. *Management Sciences*, 23:789–810, 1977. 5.1

K. P. Costello, P. Tetali, and P. Tripathi. Matching with commitment. In *Proceedings of the 39th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 822–833, 2012. 3.1.3

Nikhil R. Devanur, Shaddin Dughmi, Roy Schwartz, Ankit Sharma, and Mohit Singh. On the approximation of submodular functions. *CoRR*, abs/1304.4948, 2013. 5.9

Shahar Dobzinski. Two randomized mechanisms for combinatorial auctions. In *APPROX/RANDOM*, 2007. 1.2.3

Shahar Dobzinski, Noam Nisan, and Michael Schapira. Approximation algorithms for combinatorial auctions with complement-free bidders. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 610–618, 2005. 1.2.3

Shaddin Dughmi. A truthful randomized mechanism for combinatorial public projects via convex optimization. In *ACM EC*, pages 263–272, 2011. 5.1

Shaddin Dughmi, Tim Roughgarden, and Qiqi Yan. From convex optimization to randomized mechanisms: toward optimal combinatorial auctions. In *STOC*, pages 149–158, 2011. 5.1, 5.8

Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, 2004. 5.1

M. Fürer and H. Yu. Approximate the k-set packing problem by local improvements. *CoRR*, abs/1307.2262, 2013. 1.3.1, 3.1.2

Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42 (6):1115–1145, 1995. 5.1

Michel X. Goemans, Nicholas J. A. Harvey, Satoru Iwata, and Vahab S. Mirrokni. Approximating submodular functions everywhere. In *SODA*, pages 535–544, 2009. 1.5.2, 1.5.3, 5.1, 5.1.1, 5.1.1, 5.1.2, 5.7.1, 5.7.5, 5.8

Ralph E Gomory and Tien Chung Hu. Multi-terminal network flows. *Journal of the Society for Industrial & Applied Mathematics*, 9(4):551–570, 1961. 1.5.2, 5.1.1, 5.3

Veronika Grimm, Frank Riedel, and Elmar Wolfstetter. The third generation (UMTS) spectrum auction in Germany. CESifo Working Paper Series CESifo Working Paper No. 584, CESifo Group Munich, 2001. 4.1

M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981. ISSN 0209-9683. doi: 10.1007/BF02579273. URL http://dx.doi.org/10.1007/BF02579273. 5.1

A. Gupta and V. Nagarajan. A stochastic probing problem with applications. In *Proc. of 16th IPCO*, 2013. 1.3.3

Jason Hartline and Anna Karlin. Profit maximization in mechanism design. In *Algorithmic Game Theory*. Cambridge University Press, 2007. 1.2.3

J. Hastad. Clique is hard to approximate within $n^{1-\epsilon}$. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, FOCS '96, pages 627–, 1996. 1.2.2

Martin Hoefer and Thomas Kesselheim. Secondary spectrum auctions for symmetric and submodular bidders. In *ACM EC*, pages 657–671, 2012. 5.1

C. A. J. Hurkens and A. Schrijver. On the size of systems of sets every t of which have an SDR, with an application to the worst-case ratio of heuristics for packing problems. *SIAM Journal on Discrete Mathematics*, 2(1):68–72, 1989. 1.3.2, 3.1.2, 3.5, 3.5.1, 3.5.1, 3.5.3, 2a

Satoru Iwata, Lisa Fleischer, and Satoru Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *J. ACM*, 48(4):761–777, 2001. 5.1

W. Johnson and J. Lindenstauss. Extensions of Lipschitz maps into a Hilbert space. *Contemporary Mathematics*, 1984. 5.1

Ron Lavi and Chaitanya Swamy. Truthful and near-optimal mechanism design via linear programming. In *FOCS*, 2005. 1.2.3

B. Lehmann, D. Lehmann, and N. Nisan. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, 2006. 1.2.3

David K Levine. Modeling altruism and spitefulness in experiments. *Review of Economic Dynamics*, 1:593–622, 1998. 1.4, 4.1

George F. Loewenstein, Leigh Thompson, and Max H. Bazerman. Social utility and decision making in interpersonal contexts. *Journal of Personality and Social Psychology*, 57(3):426–441, 1989. 1.4, 4.1

John Morgan, Kenneth Steiglitz, and George Reis. The spite motive and equilibrium behavior in auctions. *Contributions to Economic Analysis & Policy*, 2(1), 2003. 1.4, 1.4.2, 1.4.3, 4.1, 4.4, 4.4

G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Mathematical Programming*, 14:265–294, December 1978. 5.1

M. Queyranne. A gomory-hu tree for symmetric sub- modular functions. unpublished manuscript, Faculty of Commerce, University of British Columbia, 1993. 1.5.2, 5.1.1, 5.3

A. E. Roth, T. Sönmez, and M. U. Ünver. Efficient kidney exchange: Coincidence of wants in markets with compatibility-based preferences. *American Economic Review*, 97:828–851, 2007. 3.8.4

Tatsuyoshi Saijo and Hideki Nakamura. The "spite" dilemma in voluntary contribution mechanism experiments. *Journal of Conflict Resolution*, 39(3):535–560, 1995. 1.4, 4.1

A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer, 2003. 5.8

Alexander Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *J. Comb. Theory, Ser. B*, 80(2):346–355, 2000. 5.1

C. Seshadhri and Jan Vondrák. Is submodularity testable? In *ICS*, pages 195–210, 2011. 1.5.3, 5.1, 5.1.2

Ankit Sharma and Tuomas Sandholm. Asymmetric spite in auctions. In *AAAI Conference on Artificial Intelligence*, 2010. 4.7

Zoya Svitkina and Lisa Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. *SIAM J. Comput.*, 40(6):1715–1737, 2011. 5.1.2

P. Toulis and D. C. Parkes. A random graph model of kidney exchanges: efficiency, individual-rationality and incentives. In *Proceedings of the 12th ACM Conference on Electronic Commerce (EC)*, pages 323–332, 2011. 3.8.5

Ioannis Vetsikas and Nicholas Jennings. Outperforming the competition in multi-unit sealed bid auctions. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Honolulu, HI, 2007. 4.1

William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961. 4.2

David W. Walkup. Matchings in random regular bipartite digraphs. *Discrete Mathematics*, 1980. 3.8.5

S. Zenios, E. S. Woodle, and L. F. Ross. Primum non nocere: Avoiding harm to vulnerable candidates in an indirect kidney exchange. *Transplantation*, 72:648–654, 2001. 3.8.4

Yunhong Zhou and Rajan Lukose. Vindictive bidding in keyword auctions. In *Workshop on Sponsored Search Auctions*, 2006. 4.1