An Online Algorithm for Maximizing Submodular Functions

Matthew Streeter **Daniel Golovin**

December 20, 2007
CMU-CS-07-171

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This research was supported in part by NSF ITR grants CCR-0122581 and IIS-0121678 and by DARPA under Contract #FA8750-05-C-0033.

Keywords: online algorithms, submodular functions

Abstract

We consider the following two problems. We are given as input a set of activities and a set of jobs to complete. Our goal is to devise a schedule for allocating time to the various activities so as to achieve one of two objectives: minimizing the average time required to complete each job, or maximizing the number of jobs completed within a fixed time T. Formally, a schedule is a sequence $\langle (v_1, \tau_1), (v_2, \tau_2), \ldots \rangle$, where each pair (v, τ) represents investing time τ in activity v. We assume that the fraction of jobs completed, f, is a monotone submodular function of the sequence of pairs that appear in a schedule.

In the offline setting in which we have oracle access to f, these two objectives give us, respectively, what we call the MIN SUM SUBMODULAR COVER problem (which is a generalization of the MIN SUM SET COVER problem and the related PIPELINED SET COVER problem) and what we call BUDGETED MAXIMUM SUBMODULAR COVERAGE (which generalizes the problem of maximizing a monotone, submodular function subject to a knapsack constraint).

We consider these problems in the online setting, in which the jobs arrive one at a time and we must finish each job (via some schedule) before moving on to the next. We give an efficient online algorithm for this problem whose worst-case asymptotic performance is simultaneously optimal for both objectives (unless P = NP), in the sense that its performance ratio (with respect to the optimal static schedule) converges to the best approximation ratios for the corresponding offline problems. Finally, we evaluate this algorithm experimentally by using it to learn, online, a schedule for allocating CPU time to the solvers entered in the 2007 SAT solver competition.

1 Introduction

This paper presents algorithms for solving a specific class of online resource allocation problems. Our online algorithms can be applied in environments where abstract *jobs* arrive one at a time, and one can complete the jobs by investing time in a number of abstract *activities*. Provided that the jobs and activities satisfy certain technical conditions, our online algorithm is guaranteed to perform almost as well as any fixed schedule for investing time in the various activities, according to two natural measures of performance. As we discuss further in §1.5, our problem formulation captures a number of previously-studied problems, including selection of algorithm portfolios [12, 15], selection of restart schedules [14, 23], and database query optimization [5, 25].

1.1 Formal setup

The problem considered in this paper can be defined as follows. We are given as input a finite set $\mathcal{V}$ of *activities*. A pair $(v, \tau) \in \mathcal{V} \times \mathbb{R}_{>0}$ is called an *action*, and represents spending time τ performing activity v. A *schedule* is a sequence of actions. We use $\mathcal{S}$ to denote the set of all schedules. A *job* is a function $f : \mathcal{S} \to [0, 1]$, where for any schedule $S \in \mathcal{S}$, $f(S)$ represents the proportion of some task that is accomplished by performing the sequence of actions S. We require that a job f have the following properties (here $\oplus$ is the concatenation operator):

1. (monotonicity) for any schedules $S_1, S_2 \in \mathcal{S}$, we have $f(S_1) \leq f(S_1 \oplus S_2)$ and $f(S_2) \leq f(S_1 \oplus S_2)$.

2. (submodularity) for any schedules $S_1, S_2 \in \mathcal{S}$ and any action $a \in \mathcal{V} \times \mathbb{R}_{>0}$,

$$f(S_1 \oplus S_2 \oplus \langle a \rangle) - f(S_1 \oplus S_2) \leq f(S_1 \oplus \langle a \rangle) - f(S_1)\,. \tag{1.1}$$

We will evaluate schedules in terms of two objectives. The first objective is to maximize $f(S)$ subject to the constraint $\ell(S) \leq T$, for some fixed $T > 0$, where $\ell(S)$ equals the sum of the durations of the actions in S. For example if $S = \langle (v_1, 3), (v_2, 3) \rangle$, then $\ell(S) = 6$. We refer to this problem as BUDGETED MAXIMUM SUBMODULAR COVERAGE (the origin of this terminology is explained in §2).

The second objective is to minimize the *cost* of a schedule, which we define as

$$c(f, S) = \int_{t=0}^{\infty} 1 - f\left(S_{\langle t \rangle}\right) dt \tag{1.2}$$

where $S_{\langle t \rangle}$ is the schedule that results from truncating schedule S at time t. For example if $S = \langle (v_1, 3), (v_2, 3) \rangle$ then $S_{\langle 5 \rangle} = \langle (v_1, 3), (v_2, 2) \rangle$.[1] One way to interpret this objective is to imagine that $f(S)$ is the probability that some desired event occurs as a result of performing the actions in S. For any non-negative random variable X, we have $\mathbb{E}[X] = \int_{t=0}^{\infty} \mathbb{P}[X > t]\, dt$. Thus $c(f, S)$ is the expected time we must wait before the event occurs if we execute actions according to the

[1] More formally, if $S = \langle a_1, a_2, \ldots \rangle$, where $a_i = (v_i, \tau_i)$, then $S_{\langle t \rangle} = \langle a_1, a_2, \ldots, a_{k-1}, a_k, (v_{k+1}, \tau') \rangle$, where k is the largest integer such that $\sum_{i=1}^{k} \tau_i < t$ and $\tau' = t - \sum_{i=1}^{k} \tau_i$.

schedule S. We refer to the problem of computing a schedule that minimizes $c(f, S)$ as MIN-SUM SUBMODULAR COVER.

In the online setting, an arbitrary sequence $\langle f_1, f_2, \ldots, f_n \rangle$ of jobs arrive one at a time, and we must finish each job (via some schedule) before moving on to the next job. When selecting a schedule S_i to use to finish job f_i, we have knowledge of the previous jobs $f_1, f_2, \ldots, f_{i-1}$ but we have no knowledge of f_i itself or of any subsequent jobs. In this setting our goal is to develop schedule-selection strategies that minimize *regret*, which is a measure of the difference between the average cost (or average coverage) of the schedules produced by our online algorithm and that of the best single schedule (in hindsight) for the given sequence of jobs.

The following example illustrates these definitions.

Example 1. Let each activity v represent a randomized algorithm for solving some decision problem, and let the action (v, τ) represent running the algorithm (with a fresh random seed) for time τ. Fix some particular instance of the decision problem, and for any schedule S, let $f(S)$ be the probability that one (or more) of the runs in the sequence S yields a solution to that instance. So $f(S_{\langle T \rangle})$ is (by definition) the probability that performing the runs in schedule S yields a solution to the problem instance in time $\leq T$, while $c(f, S)$ is the *expected* time that elapses before a solution is obtained. It is clear that $f(S)$ satisfies the monotonicity condition required of a job, because adding runs to the sequence S can only increase the probability that one of the runs is successful. The fact that f is submodular can be seen as follows. For any schedule S and action a, $f(S \oplus \langle a \rangle) - f(S)$ equals the probability that action a succeeds after every action in S has failed, which can also be written as $(1 - f(S)) \cdot f(\langle a \rangle)$. This, together with the monotonicity of f, implies that for any schedules S_1, S_2 and any action a, we have

$$\begin{aligned} f(S_1 \oplus S_2 \oplus \langle a \rangle) - f(S_1 \oplus S_2) &= (1 - f(S_1 \oplus S_2)) \cdot f(\langle a \rangle) \\ &\leq (1 - f(S_1)) \cdot f(\langle a \rangle) \\ &= f(S_1 \oplus \langle a \rangle) - f(S_1) \end{aligned}$$

so f is submodular.

1.2 Sufficient conditions

In some cases of practical interest, f will not satisfy the submodularity condition but will still satisfy weaker conditions that are sufficient for our results to carry through.

In the offline setting, our results will hold for any function f that satisfies the monotonicity condition and, additionally, satisfies the following condition (we prove in §3 that any submodular function satisfies this weaker condition).

Condition 1. For any $S_1, S \in \mathcal{S}$,

$$\frac{f(S_1 \oplus S) - f(S_1)}{\ell(S)} \leq \max_{(v,\tau) \in \mathcal{V} \times \mathbb{R}_{>0}} \left\{ \frac{f(S_1 \oplus \langle (v, \tau) \rangle) - f(S_1)}{\tau} \right\} .$$

Recall that $\ell(S)$ equals the sum of the durations of the actions in S. Informally, Condition 1 says that the increase in f per unit time that results from performing a sequence of actions S is

always bounded by the maximum, over all actions (v, τ), of the increase in f per unit time that results from performing that action.

In the online setting, our results will apply if each function f_i in the sequence $\langle f_1, f_2, \ldots, f_n \rangle$ satisfies the monotonicity condition and, additionally, the sequence as a whole satisfies the following condition (we prove in §4 that if each f_i is a job, then this condition is satisfied).

Condition 2. For any sequence $S_1, S_2, \ldots, S_n$ of schedules and any schedule S,

$$\frac{\sum_{i=1}^{n} \left(f_i(S_i \oplus S) - f_i(S_i) \right)}{\ell(S)} \leq \max_{(v,\tau) \in \mathcal{V} \times \mathbb{R}_{>0}} \left\{ \frac{\sum_{i=1}^{n} \left(f_i(S_i \oplus \langle (v, \tau) \rangle) - f_i(S_i) \right)}{\tau} \right\}.$$

This generality allows us to handle jobs similar to the job defined in Example 1, but where an action (v, τ) may represent *continuing* a run of algorithm v for an additional τ time units (rather than running v with a fresh random seed). Note that the function f defined in Example 1 is no longer submodular when actions of this form are allowed.

1.3 Summary of results

We first consider the offline problems BUDGETED MAXIMUM SUBMODULAR COVERAGE and MIN-SUM SUBMODULAR COVER. As immediate consequences of existing results [10, 11], we find that, for any $\epsilon > 0$, (*i*) achieving an approximation ratio of $4 - \epsilon$ for MIN-SUM SUBMODULAR COVER is NP-hard and (*ii*) achieving an approximation ratio of $1 - \frac{1}{e} + \epsilon$ for BUDGETED MAXIMUM SUBMODULAR COVERAGE is NP-hard. We then present a greedy approximation algorithm that simultaneously achieves the optimal approximation ratio of 4 for MIN-SUM SUBMODULAR COVER and the optimal approximation ratio of $1 - \frac{1}{e}$ for BUDGETED MAXIMUM SUBMODULAR COVERAGE, building on and generalizing previous work on special cases of these two problems [11, 29].

The main contribution of this paper, however, is to address the online setting. In this setting we provide an online algorithm whose worst-case performance approaches that of the offline greedy approximation algorithm asymptotically (as the number of jobs approaches infinity). More specifically, we analyze the online algorithm's performance in terms of "α-regret". For the cost-minimization objective, α-regret is defined as the difference between the average cost of the schedules selected by the online algorithm and α times the average cost of the optimal schedule for the given sequence of jobs. For the coverage-maximization objective, α-regret is the difference between α times the average coverage of the optimal fixed schedule and the average coverage of the schedules selected by the online algorithm. For the objective of minimizing cost, the online algorithm's 4-regret approaches zero as $n \to \infty$, while for the objective of maximizing coverage, its $1 - \frac{1}{e}$ regret approaches zero as $n \to \infty$. Assuming P $\neq$ NP, these guarantees are essentially the best possible among online algorithms that make decisions in polynomial time.

Our online algorithms can be used in several different feedback settings. We first consider the feedback setting in which, after using schedule S_i to complete job f_i, we receive complete access to f_i. We then consider more limited feedback settings in which: (*i*) to receive access to f_i we must pay a price C, which is added to the regret, (*ii*) we only observe $f_i\left(S_{i\langle t \rangle}\right)$ for each $t \geq 0$, and (*iii*) we only observe $f_i(S_i)$.

We also prove tight information-theoretic lower bounds on 1-regret, and discuss exponential time online algorithms whose regret matches the lower bounds to within logarithmic factors. Interestingly, these lower bounds also match the upper bounds from our online greedy approximation algorithm up to logarithmic factors, although the latter apply to α-regret (for $\alpha = 4$ or $\alpha = 1 - \frac{1}{e}$) rather than 1-regret.

1.4 Problems that fit into this framework

We now discuss how a number of previously-studied problems fit into our framework.

1.4.1 Special cases of BUDGETED MAXIMUM SUBMODULAR COVERAGE

The BUDGETED MAXIMUM SUBMODULAR COVERAGE problem introduced in this paper is a slight generalization of the problem of maximizing a monotone submodular set function subject to a knapsack constraint [21, 29]. The only difference between the two problems is that, in the latter problem, $f(S)$ may only depend on the *set* of actions in the sequence S, and not on the order in which the actions appear. The problem of maximizing a monotone submodular set function subject to a knapsack constraint in turn generalizes BUDGETED MAXIMUM COVERAGE [19], which generalizes MAX k-COVERAGE [26].

1.4.2 Special cases of MIN-SUM SUBMODULAR COVER

The MIN-SUM SUBMODULAR COVER problem introduced in this paper generalizes several previously-studied problems, including MIN-SUM SET COVER [11], PIPELINED SET COVER [17, 25], the problem of constructing efficient sequences of trials [9], and the problem of constructing restart schedules [14, 23, 28]. Specifically, these problems can be represented in our framework by jobs of the form

$$f\left(\langle (v_1, \tau_1), (v_2, \tau_2), \ldots, (v_L, \tau_L) \rangle\right) = \frac{1}{n} \sum_{i=1}^{n} \left(1 - \prod_{l=1}^{L} \left(1 - p_i(v_l, \tau_l)\right)\right) . \tag{1.3}$$

This expression can be interpreted as follows: the job f consists of n subtasks, and $p_i(v, \tau)$ is the probability that investing time τ in activity v completes the i^{th} subtask. Thus, $f(S)$ is the expected fraction of subtasks that are finished after performing the sequence of actions in S. Assuming $p_i(v, \tau)$ is a non-decreasing function of τ for all i and v, it can be shown that any function f of this form satisfies the monotonicity and submodularity properties required of a job. In the special case $n = 1$, this follows from Example 1. In the general case $n > 1$, this follows from the fact (which follows immediately from the definitions) that any convex combination of jobs is a job.

The problem of computing restart schedules places no further restrictions on $p_i(v, \tau)$. PIPELINED SET COVER is the special case in which for each activity v there is an associated time τ_v, and $p_i(v, \tau) = 1$ if $\tau \geq \tau_v$ and $p_i(v, \tau) = 0$ otherwise. MIN-SUM SET COVER is the special case in which, additionally, $\tau_v = 1$ or $\tau_v = \infty$ for all $v \in \mathcal{V}$. The problem of constructing efficient sequences of trials corresponds to the case in which we are given a matrix q, and $p_i(v, \tau) = q_{v,i}$ if $\tau \geq 1$ and $p_i(v, \tau) = 0$ otherwise.

1.5 Applications

We now discuss applications of the results presented in this paper. The first application, "Combining multiple heuristics online", is evaluated experimentally in §6. Evaluating the remaining applications is an interesting area of future work.

1.5.1 Combining multiple heuristics online

An *algorithm portfolio* [15] is a schedule for interleaving the execution of multiple (randomized) algorithms and periodically restarting them with a fresh random seed. Previous work has shown that combining multiple heuristics for NP-hard problems into a portfolio can dramatically reduce average-case running time [12, 15, 27]. In particular, algorithms based on chronological backtracking often exhibit heavy-tailed run length distributions, and periodically restarting them with a fresh random seed can reduce the mean running time by orders of magnitude [13]. Our algorithms can be used to learn an effective algorithm portfolio online, in the course of solving a sequence of problem instances.

1.5.2 Database query optimization

In database query processing, one must extract all the records in a database that satisfy every predicate in a list of one or more predicates (the conjunction of predicates comprises the query). To process the query, each record is evaluated against the predicates one at a time until the record either fails to satisfy some predicate (in which case it does not match the query) or all predicates have been examined. The order in which the predicates are examined affects the time required to process the query. Munagala *et al.* [25] introduced and studied a problem called PIPELINED SET COVER, which entails finding an evaluation order for the predicates that minimizes the average time required to process a record. As discussed in §1.4, PIPELINED SET COVER is a special case of MIN-SUM SUBMODULAR COVER. In the online version of PIPELINED SET COVER, records arrive one at a time and one may select a different evaluation order for each record. In our terms, the records are jobs and predicates are activities.

1.5.3 Sensor placement

Sensor placement is the task of assigning locations to a set of sensors so as to maximize the value of the information obtained (e.g., to maximize the number of intrusions that are detected by the sensors). Many sensor placement problems can be optimally solved by maximizing a monotone submodular set function subject to a knapsack constraint [20]. As discussed in §1.4, this problem is a special case of BUDGETED MAXIMUM SUBMODULAR COVERAGE. Our online algorithms could be used to select sensor placements when the same set of sensors is repeatedly deployed in an unknown or adversarial environment.

1.5.4 Viral marketing

Viral marketing infects a set of agents (e.g., individuals or groups) with an advertisement which they may pass on to other potential customers. Under a standard model of social network dynamics, the total number of potential customers that are influenced by the advertisement is a submodular function of the set of agents that are initially infected [18]. Previous work [18] gave an algorithm for selecting a set of agents to initially infect so as to maximize the influence of an advertisement, assuming the dynamics of the social network are known. In theory, our online algorithms could be used to adapt a marketing campaign to unknown or time-varying social network dynamics.

2 Related Work

As discussed in §1.4, the MIN-SUM SUBMODULAR COVER problem introduced in this paper generalizes several previously-studied problems, including MIN-SUM SET COVER [11], PIPELINED SET COVER [17, 25], the problem of constructing efficient sequences of trials [9], and the problem of constructing restart schedules [23, 14, 28].

Several of these problems have been considered in the online setting. Munagala *et al.* [25] gave an online algorithm for PIPELINED SET COVER whose $O(\log|\mathcal{V}|)$-regret is $o(n)$, where n is the number of records (jobs). Babu *et al.* [5] and Kaplan *et al.* [17] gave online algorithms for PIPELINED SET COVER whose 4-regret is $o(n)$, but these bounds hold only in the special case where the jobs are drawn independently at random from a fixed probability distribution. The online setting in this paper, where the sequence of jobs may be arbitrary, is more challenging from a technical point of view.

As already mentioned, BUDGETED MAXIMUM SUBMODULAR COVERAGE generalizes the problem of maximizing a monotone submodular set function subject to a knapsack constraint. Previous work gave offline greedy approximation algorithms for this problem [21, 29], which generalized earlier algorithms for BUDGETED MAXIMUM COVERAGE [19] and MAX k-COVERAGE [26]. To our knowledge, none of these three problems have previously been studied in an online setting.

It is worth pointing out that the online problems we consider here are quite different from online set cover problems that require one to construct a *single* collection of sets that cover each element in a sequence of elements that arrive online [1, 3]. Likewise, our work is orthogonal to work on online facility location problems [24].

The main technical contribution of this paper is to convert some specific greedy approximation algorithms into online algorithms. Recently, Kakade *et al.* [16] gave a generic procedure for converting an α-approximation algorithm for a linear problem into an online algorithm whose α-regret is $o(n)$, and this procedure could be applied to the problems considered in this paper. However, both the running time of their algorithm and the resulting regret bounds depend on the dimension of the linear problem, and a straightforward application of their algorithm leads to running time and regret bounds that are exponential in $|\mathcal{V}|$.

3 Offline Algorithms

In this section we consider the offline problems BUDGETED MAXIMUM SUBMODULAR COVERAGE and MIN-SUM SUBMODULAR COVER. In the offline setting, we are given as input a job $f : \mathcal{S} \to [0, 1]$. Our goal is to compute a schedule S that achieves one of two objectives: for BUDGETED MAXIMUM SUBMODULAR COVERAGE, we wish to maximize $f(S)$ subject to the constraint $\ell(S) \leq T$ (for some fixed $T > 0$), while for MIN-SUM SUBMODULAR COVER, we wish to minimize the cost $c(f, S)$.

The offline algorithms presented in this section will serve as the basis for the online algorithms we develop in the next section.

Note that we have defined the offline problem in terms of optimizing a *single* job. However, given a set $\{f_1, f_2, \ldots, f_n\}$, we can optimize average schedule cost (or coverage) by applying our offline algorithm to the job $f = \frac{1}{n}\sum_{i=1}^{n} f_i$ (as already mentioned, any convex combination of jobs is a job).

3.1 Computational complexity

Both of the offline problems considered in this paper are NP-hard even to approximate. As discussed in §1.4, MIN-SUM SUBMODULAR COVER generalizes MIN-SUM SET COVER, and BUDGETED MAXIMUM SUBMODULAR COVERAGE generalizes MAX k-COVERAGE. In a classic paper, Feige proved that for any $\epsilon > 0$, acheiving an approximation ratio of $1 - \frac{1}{e} + \epsilon$ for MAX k-COVERAGE is NP-hard [10]. Recently, Feige, Lovász, and Tetali [11] introduced MIN-SUM SET COVER and proved that for any $\epsilon > 0$, achieving a $4 - \epsilon$ approximation ratio for MIN-SUM SET COVER is NP-hard. These observations immediately yield the following theorems.

Theorem 1. *For any $\epsilon > 0$, achieving a $1 - \frac{1}{e} + \epsilon$ approximation ratio for* BUDGETED MAXIMUM SUBMODULAR COVERAGE *is NP-hard.*

Theorem 2. *For any $\epsilon > 0$, achieving a $4 - \epsilon$ approximation ratio for* MIN-SUM SUBMODULAR COVER *is NP-hard.*

3.2 Greedy approximation algorithm

In this section we present a greedy approximation algorithm that can be used to achieve a 4 approximation for MIN-SUM SUBMODULAR COVER and a $1 - \frac{1}{e}$ approximation for BUDGETED MAXIMUM SUBMODULAR COVERAGE. By Theorems 1 and 2, achieving a better approximation ratio for either problem is NP-hard.

Consider the schedule defined by the following simple greedy rule. Let $G = \langle g_1, g_2, \ldots \rangle$ be the schedule defined inductively as follows: $G_1 = \langle\rangle$, $G_j = \langle g_1, g_2, \ldots, g_{j-1} \rangle$ for $j > 1$, and

$$g_j = \operatorname*{arg\,max}_{(v,\tau) \in \mathcal{V} \times \mathbb{R}_{>0}} \left\{ \frac{f\left(G_j \oplus \langle (v, \tau) \rangle\right) - f\left(G_j\right)}{\tau} \right\}. \tag{3.1}$$

That is, G is constructed by greedily appending an action (v, τ) to the schedule so as to maximize the resulting increase in f per unit time.

Once we reach a j such that $f(G_j) = 1$, we may stop adding actions to the schedule. In general, however, G may contain an infinite number of actions. For example, if each action (v, τ) represents running a Las Vegas algorithm v for time τ and $f(S)$ is the probability that any of the runs in S return a solution to some problem instance (see Example 1), it is possible that $f(S) < 1$ for any finite schedule S. The best way of dealing with this is application-dependent. In the case of Example 1, we might stop computing G when $f(G_j) \geq 1 - \delta$ for some small $\delta > 0$.

The time required to compute G is also application-dependent. In the applications of interest to us, evaluating the $\arg\max$ in (3.1) will only require us to consider a finite number of actions (v, τ). In some cases, the evaluation of the $\arg\max$ in (3.1) can be sped up using application-specific data structures.

As mentioned in §1.2, our analysis of the greedy approximation algorithm will only require that f is monotone and that f satisfies Condition 1. The following lemma shows that if f is a job, then f also satisfies these conditions.

Lemma 1. *If f satisfies* (1.1)*, then f satisfies Condition 1. That is, for any schedules $S_1, S \in \mathcal{S}$, we have*

$$\frac{f(S_1 \oplus S) - f(S_1)}{\ell(S)} \leq \max_{(v,\tau) \in \mathcal{V} \times \mathbb{R}_{>0}} \left\{ \frac{f(S_1 \oplus \langle (v, \tau) \rangle) - f(S_1)}{\tau} \right\} .$$

Proof. Let r denote the right hand side of the inequality. Let $S = \langle a_1, a_2, \ldots, a_L \rangle$, where $a_l = (v_l, \tau_l)$. Let

$$\Delta_l = f(S_1 \oplus \langle a_1, a_2, \ldots, a_l \rangle) - f(S_1 \oplus \langle a_1, a_2, \ldots, a_{l-1} \rangle) .$$

We have

$$\begin{aligned}
f(S_1 \oplus S) &= f(S_1) + \sum_{l=1}^{L} \Delta_l && \text{(telescoping series)} \\
&\leq f(S_1) + \sum_{l=1}^{L} \left(f(S_1 \oplus \langle a_l \rangle) - f(S_1) \right) && \text{(submodularity)} \\
&\leq f(S_1) + \sum_{l=1}^{L} r \cdot \tau_l && \text{(definition of } r\text{)} \\
&= f(S_1) + r \cdot \ell(S) \ .
\end{aligned}$$

Rearranging this inequality gives $\frac{f(S_1 \oplus S) - f(S_1)}{\ell(S)} \leq r$, as claimed. □

The key to the analysis of the greedy approximation algorithm is the following fact, which is the only property of G that we will use in our analysis.

Fact 1. *For any schedule S, any positive integer j, and any $t > 0$, we have*

$$f(S_{\langle t \rangle}) \leq f(G_j) + t \cdot s_j$$

where s_j is the j^{th} value of the maximum in (3.1)*.*

Fact 1 holds because $f(S_{\langle t \rangle}) \leq f(G_j \oplus S_{\langle t \rangle})$ by monotonicity, while $f(G_j \oplus S_{\langle t \rangle}) \leq f(G_j) + t \cdot s_j$ by Condition 1 and the definition of s_j.

3.2.1 Maximizing coverage

We first analyze the performance of the greedy algorithm on the BUDGETED MAXIMUM SUBMODULAR COVERAGE problem. The following theorem shows that, for certain values of T, the greedy schedule achieves the optimal approximation ratio of $1 - \frac{1}{e}$ for this problem. The proof of the theorem is similar to arguments in [21, 29].

Theorem 3. *Let L be a positive integer, and let $T = \sum_{j=1}^{L} \tau_j$, where $g_j = (v_j, \tau_j)$. Then $f\left(G_{\langle T \rangle}\right) > \left(1 - \frac{1}{e}\right) \max_{S \in \mathcal{S}} \left\{ f\left(S_{\langle T \rangle}\right) \right\}$.*

Proof. Let $C^* = \max_{S \in \mathcal{S}} \left\{ f\left(S_{\langle T \rangle}\right) \right\}$, and for any positive integer j, let $\Delta_j = C^* - f\left(G_j\right)$. By Fact 1, $C^* \leq f\left(G_j\right) + Ts_j$. Thus

$$\Delta_j \leq Ts_j = T\left(\frac{\Delta_j - \Delta_{j+1}}{\tau_j}\right) .$$

Rearranging this inequality gives $\Delta_{j+1} \leq \Delta_j \left(1 - \frac{\tau_j}{T}\right)$. Unrolling this inequality, we get

$$\Delta_{L+1} \leq \Delta_1 \left(\prod_{j=1}^{L} 1 - \frac{\tau_j}{T}\right) .$$

Subject to the constraint $\sum_{j=1}^{L} \tau_j = T$, the product series is maximized when $\tau_j = \frac{T}{L}$ for all j. Thus we have

$$C^* - f\left(G_{L+1}\right) = \Delta_{L+1} \leq \Delta_1 \left(1 - \frac{1}{L}\right)^L < \Delta_1 \frac{1}{e} \leq C^* \frac{1}{e} .$$

Thus $f\left(G_{L+1}\right) > (1 - \frac{1}{e})C^*$, as claimed. □

Theorem 3 shows that G gives a $1 - \frac{1}{e}$ approximation to the problem of maximizing coverage at time T, provided that T equals the sum of the durations of the actions in G_j for some positive integer j. Under the assumption that f is a job (as opposed to the weaker assumption that f satisfies Condition 1), the greedy algorithm can be combined with the partial enumeration approach of Khuller *et al.* [19] to achieve a $1 - \frac{1}{e}$ approximation ratio for any fixed T. The idea of this approach is to guess a sequence $Y = \langle a_1, a_2, a_3 \rangle$ of three actions, and then run the greedy algorithm on the job $f'(S) = f\left(Y \oplus S\right) - f\left(Y\right)$ with budget $T - T_0$, where T_0 is the total time consumed by the actions in Y. The arguments of [19, 29] show that, for some choice of Y, this yields a $\left(1 - \frac{1}{e}\right)$-approximation. In order for this approach to be feasible, actions must have discrete durations, so that the number of possible choices of Y is finite.

3.2.2 Minimizing cost

We next analyze the performance of the greedy algorithm on the MIN-SUM SUBMODULAR COVER problem. The following theorem uses the proof technique of [11] to show that the greedy schedule G has cost at most 4 times that of the optimal schedule, generalizing results of [11, 17, 25, 27, 28]. As already mentioned, achieving a better approximation ratio is NP-hard.

Theorem 4. $c\left(f,G\right) \leq 4\int_{t=0}^{\infty} 1-\max_{S\in\mathcal{S}}\left\{f\left(S_{\langle t\rangle}\right)\right\}\,dt \leq 4\min_{S\in\mathcal{S}} c\left(f,S\right)$.

Proof. Let $R_j = 1 - f\left(G_j\right)$; let $x_j = \frac{R_j}{2s_j}$; let $y_j = \frac{R_j}{2}$; and let $h(x) = 1 - \max_S\left\{f\left(S_{\langle x\rangle}\right)\right\}$. By Fact 1,

$$\max_S\left\{f(S_{\langle x_j\rangle})\right\} \leq f\left(G_j\right) + x_j s_j = f\left(G_j\right) + \frac{R_j}{2}\ .$$

Thus $h(x_j) \geq R_j - \frac{R_j}{2} = y_j$. The monotonicity of f implies that $h(x)$ is non-increasing and also that the sequence $\langle y_1, y_2, \ldots\rangle$ is non-increasing. As illustrated in Figure 1, these facts imply that $\int_{x=0}^{\infty} h(x)\,dx \geq \sum_{j\geq 1} x_j\left(y_j - y_{j+1}\right)$. Thus we have

$$\begin{aligned}
\int_{t=0}^{\infty} 1-\max_{S\in\mathcal{S}}\left\{f\left(S_{\langle t\rangle}\right)\right\}\,dt &= \int_{x=0}^{\infty} h(x)\,dx \\
&\geq \sum_{j\geq 1} x_j\left(y_j - y_{j+1}\right) && \text{(Figure 1)} \\
&= \frac{1}{4}\sum_{j\geq 1} R_j\frac{\left(R_j - R_{j+1}\right)}{s_j} \\
&= \frac{1}{4}\sum_{j\geq 1} R_j\tau_j \\
&\geq \frac{1}{4}c\left(f,G\right) && \text{(monotonicity of } f\text{)}
\end{aligned}$$

which proves the theorem. □

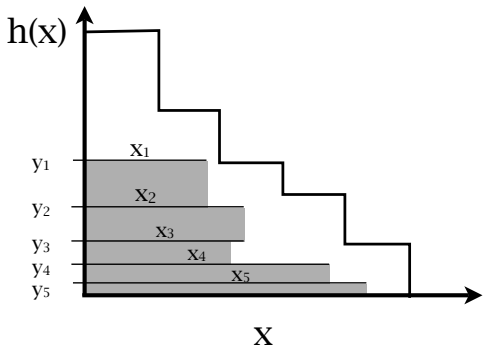

Figure 1: An illustration of the inequality $\int_{x=0}^{\infty} h(x)\,dx \geq \sum_{j\geq 1} x_j\left(y_j - y_{j+1}\right)$. The left hand side is the area under the curve, whereas the right hand side is the sum of the areas of the shaded rectangles.

3.2.3 A refined greedy approximation algorithm

A drawback of G is that it greedily chooses an action $g_j = (v, \tau)$ that maximizes the marginal increase in f divided by τ, whereas the contribution of (v, τ) to the cost of G is not τ but rather

$$\int_{t=0}^{\tau} 1 - f\left(G_j \oplus \langle (v,t) \rangle\right) dt \; .$$

This can lead G to perform suboptimally even in seemingly easy cases. To see this, let $\mathcal{V} = \{v_1, v_2\}$, let $S_t^1 = \langle (v_1, t) \rangle$, and let $S_t^2 = \langle (v_2, t) \rangle$. Let f be a job defined by

$$f(S_t^1) = \begin{cases} 1 & \text{if } t \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

whereas

$$f(S_t^2) = \min\{1, t\} \; .$$

For any schedule $S = \langle a_1, a_2, \ldots, a_L \rangle$ containing more than one action, let $f(S) = \max_{l=1}^{L} f(\langle a_l \rangle)$. It is straightforward to check that f satisfies the monotonicity and submodularity conditions required of a job.

Here the optimal schedule is $S^* = \langle (v_2, 1) \rangle$, with cost $c(f, S^*) = \int_{t=0}^{1} 1 - t \; dt = \frac{1}{2}$. However, if ties in the evaluation of the $\arg\max$ in (3.1) are broken appropriately, the greedy algorithm will choose the schedule $G = \langle (v_1, 1) \rangle$, with cost $c(f, G) = 1$.

To improve performance in cases such as this, it is natural to consider the schedule $G' = \langle g'_1, g'_2, \ldots \rangle$ defined inductively as follows: $G'_j = \left\{g'_1, g'_2, \ldots, g'_{j-1}\right\}$ and

$$g'_j = \underset{(v,\tau) \in \mathcal{V} \times \mathbb{R}_{>0}}{\arg\max} \left\{ \frac{f\left(G'_j \oplus \langle (v,\tau) \rangle\right) - f\left(G'_j\right)}{\int_{t=0}^{\tau} 1 - f\left(G'_j \oplus \langle (v,t) \rangle\right) dt} \right\} \; . \tag{3.2}$$

Theorem 5 shows that G' achieves the same approximation ratio as G. The proof is similar to the proof of Theorem 4, and is given in Appendix A.

Theorem 5. $c(f, G') \leq 4 \int_{t=0}^{\infty} 1 - \max_{S \in \mathcal{S}} \left\{ f\left(S_{\langle t \rangle}\right) \right\} dt \leq 4 \min_{S \in \mathcal{S}} \{c(f, S)\}$.

Furthermore, it can be shown that, in contrast to G, G' is *optimal* in the important special case when $\mathcal{V} = \{v\}$, action (v, τ) represents running a Las Vegas algorithm v (with a fresh random seed) for time τ, and $f(S)$ equals the probability that at least one of the runs in S returns a solution to some particular problem instance (as described in Example 1).

3.2.4 Handling non-uniform additive error

We now consider the case in which the j^{th} decision made by the greedy algorithm is performed with some additive error ϵ_j. This case is of interest for two reasons. First, in some cases it may not be practical to evaluate the $\arg\max$ in (3.1) exactly. Second, and more importantly, we will end up viewing our online algorithm as a version of the offline greedy algorithm in which each decision is made with some additive error. In this section we analyze the original greedy schedule

G as opposed to the refined schedule G' described in the previous section, because it is the original schedule G that will form the basis of our online algorithm (as we discuss further in §5, devising an online algorithm based on G' is an interesting open problem).

We denote by $\bar{G} = \langle \bar{g}_1, \bar{g}_2, \ldots \rangle$ a variant of the schedule G in which the j^{th} $\arg\max$ in (3.1) is evaluated with additive error ϵ_j. More formally, $\bar{G}$ is a schedule that, for any $j \geq 1$, satisfies

$$\frac{f\left(\bar{G}_j \oplus \bar{g}_j\right) - f\left(\bar{G}_j\right)}{\bar{\tau}_j} \geq \max_{(v,\tau)\in\mathcal{V}\times\mathbb{R}_{>0}} \left\{ \frac{f\left(\bar{G}_j \oplus \langle (v,\tau) \rangle\right) - f\left(\bar{G}_j\right)}{\tau} \right\} - \epsilon_j \tag{3.3}$$

where $\bar{G}_0 = \langle\rangle$, $\bar{G}_j = \langle \bar{g}_1, \bar{g}_2, \ldots, \bar{g}_{j-1} \rangle$ for $j > 1$, and $\bar{g}_j = (\bar{v}_j, \bar{\tau}_j)$.

The following two theorems summarize the performance of $\bar{G}$. The proofs are given in Appendix A, and are along the same lines as that those of theorems 3 and 4.

Theorem 6. *Let L be a positive integer, and let $T = \sum_{j=1}^{L} \bar{\tau}_j$, where $\bar{g}_j = (\bar{v}_j, \bar{\tau}_j)$. Then*

$$f\left(\bar{G}_{\langle T \rangle}\right) > \left(1 - \frac{1}{e}\right) \max_{S\in\mathcal{S}} \left\{ f\left(S_{\langle T \rangle}\right) \right\} - \sum_{j=1}^{L} \epsilon_j \bar{\tau}_j \,.$$

Theorem 7. *Let L be a positive integer, and let $T = \sum_{j=1}^{L} \bar{\tau}_j$, where $\bar{g}_j = (\bar{v}_j, \bar{\tau}_j)$. For any schedule S, define $c^T(f, S) \equiv \int_{t=0}^{T} 1 - f\left(S_{\langle t \rangle}\right) dt$. Then*

$$c^T\left(f, \bar{G}\right) \leq 4 \int_{t=0}^{\infty} 1 - \max_{S\in\mathcal{S}} \left\{ f\left(S_{\langle t \rangle}\right) \right\} dt + \sum_{j=1}^{L} E_j \bar{\tau}_j \,.$$

where $E_j = \sum_{l<j} \epsilon_l \bar{\tau}_l$.

4 Online Algorithms

In this section we consider the online versions of BUDGETED MAXIMUM SUBMODULAR COVERAGE and MIN-SUM SUBMODULAR COVER. In the online setting we are fed, one at a time, a sequence $\langle f_1, f_2, \ldots, f_n \rangle$ of jobs. Prior to receiving job f_i, we must specify a schedule S_i. We then receive complete access to the function f_i. We measure the performance of our online algorithm using two different notions of regret. For the cost objective, our goal is to minimize the 4-regret

$$R_{cost} \equiv \sum_{i=1}^{n} c^T\left(S_i, f_i\right) - 4 \cdot \min_{S\in\mathcal{S}} \left\{ \sum_{i=1}^{n} c\left(S, f_i\right) \right\}$$

for some fixed $T > 0$. Here, for any schedule S and job f, we define $c^T(S, f) = \int_{t=0}^{T} 1 - f\left(S_{\langle t \rangle}\right) dt$ to be the value of $c(S, f)$ when the integral is truncated at time T. Some form of truncation is necessary because $c(S_i, f_i)$ could be infinite, and without bounding it we could not prove any finite bound on regret (our regret bounds will be stated as a function of T).

For the objective of maximizing the coverage at time T, our goal is to minimize the $(1 - \frac{1}{e})$-regret

$$R_{coverage} \equiv \left(1 - \frac{1}{e}\right) \max_{S \in \mathcal{S}} \left\{ \sum_{i=1}^{n} f_i\left(S_{\langle T \rangle}\right) \right\} - \sum_{i=1}^{n} f_i\left(S_i\right)$$

where we require that $\mathbb{E}\left[\ell\left(S_i\right)\right] = T$, in expectation over the online algorithm's random bits. In other words, we allow the online algorithm to treat T as a budget in expectation, rather than a hard budget.

Our goal is to bound the expected values of R_{cost} (resp. $R_{coverage}$) on the worst-case sequence of n jobs. We consider the so-called *oblivious adversary model*, in which the sequence of jobs is fixed in advance and does not change in response to the decisions made by our online algorithm, although we believe our results can be readily extended to the case of adaptive adversaries. Note that the constant of 4 in the definition of R_{cost} and the constant of $1 - \frac{1}{e}$ in the definition of $R_{coverage}$ stem from the NP-hardness of the corresponding offline problems, as discussed in §3.1.

For the purposes of the results in this section, we confine our attention to schedules that consist of actions that come from some finite set $\mathcal{A}$, and we assume that the actions in $\mathcal{A}$ have integer durations (i.e. $\mathcal{A} \subseteq \mathcal{V} \times \mathbb{Z}_{>0}$). Note that this is not a serious limitation, because real-valued action durations can always be discretized at whatever level of granularity is desired.

As mentioned in §1.2, our results in the online setting will hold for any sequence $\langle f_1, f_2, \ldots, f_n \rangle$ of functions that satisfies Condition 2. The following lemma shows that any sequence of jobs satisfies this condition. The proof follows along the same lines as the proof of Lemma 1, and is given in Appendix A.

Lemma 2. *Any sequence $\langle f_1, f_2, \ldots, f_n \rangle$ of jobs satisfies Condition 2. That is, for any sequence $S_1, S_2, \ldots, S_n$ of schedules and any schedule S,*

$$\frac{\sum_{i=1}^{n} f_i(S_i \oplus S) - f_i(S_i)}{\ell\left(S\right)} \leq \max_{(v,\tau) \in \mathcal{V} \times \mathbb{R}_{>0}} \left\{ \frac{\sum_{i=1}^{n} f_i(S_i \oplus \langle (v,\tau) \rangle) - f_i(S_i)}{\tau} \right\}.$$

4.1 Background: the experts problem

In the experts problem, one has access to a set of k experts, each of whom gives out a piece of advice every day. On each day i, one must select an expert e_i whose advice to follow. Following the advice of expert j on day i yields a reward x_j^i. At the end of day i, the value of the reward x_j^i for each expert j is made public, and can be used as the basis for making choices on subsequent days. One's *regret* at the end of n days is equal to

$$\max_{1 \leq j \leq k} \left\{ \sum_{i=1}^{n} x_j^i \right\} - \sum_{i=1}^{n} x_{e_i}^i \ .$$

Note that the historical performance of an expert does not imply any guarantees about its future performance. Remarkably, randomized decision-making algorithms nevertheless exist whose regret grows sub-linearly in the number of days. By picking experts using such an algorithm, one can

guarantee to obtain (asymptotically as $n \to \infty$) an average reward that is as large as the maximum reward that could have been obtained by following the advice of any fixed expert for all n days.

In particular, for any fixed value of G_{max}, where $G_{max} = \max_{1 \le j \le k} \left\{ \sum_{i=1}^{n} x_j^i \right\}$, the randomized weighted majority algorithm (WMR) [22] can be used to achieve worst-case regret $O\left(\sqrt{G_{max} \ln k}\right)$. If G_{max} is not known in advance, a putative value can be guessed and doubled to achieve the same guarantee up to a constant factor.

4.2 Unit-cost actions

In the special case in which each action takes unit time (i.e., $\mathcal{A} \subseteq \mathcal{V} \times \{1\}$), our online algorithm $\mathbf{OG_{unit}}$ is very simple. $\mathbf{OG_{unit}}$ runs T experts algorithms:[2] $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_T$, where T is the number of time steps for which our schedule is defined. The set of experts is $\mathcal{A}$. Just before job f_i arrives, each experts algorithm $\mathcal{E}_t$ selects an action a_t^i. The schedule used by $\mathbf{OG_{unit}}$ on job f_i is $S_i = \langle a_1^i, a_2^i, \ldots, a_T^i \rangle$. The payoff that $\mathcal{E}_t$ associates with action a is $f_i\left(S_{i\langle t-1 \rangle} \oplus a\right) - f_i\left(S_{i\langle t-1 \rangle}\right)$.

Algorithm $\mathbf{OG_{unit}}$
Input: integer T, experts algorithms $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_T$.

For i from 1 to n:

1. For each t, $1 \le t \le T$, use $\mathcal{E}_t$ to select an action a_t^i.
2. Select the schedule $S_i = \langle a_1^i, a_2^i, \ldots, a_T^i \rangle$.
3. Receive the job f_i.
4. For each t, $1 \le t \le T$, and each action $a \in \mathcal{A}$, feed back $f_i\left(S_{i\langle t-1 \rangle} \oplus a\right) - f_i\left(S_{i\langle t-1 \rangle}\right)$ as the payoff $\mathcal{E}_t$ would have received by choosing action a.

Let r_t be the regret experienced by experts algorithm $\mathcal{E}_t$ when running $\mathbf{OG_{unit}}$, and let $R = \sum_{t=1}^{T} r_t$. The key to the analysis of $\mathbf{OG_{unit}}$ is the following lemma, which relates the regret experienced by the experts algorithms to the regret on the original online problem.

Lemma 3. *$R_{coverage} \le R$ and $R_{cost} \le TR$.*

Proof. We will view $\mathbf{OG_{unit}}$ as producing an approximate version of the offline greedy schedule for the function $f = \frac{1}{n} \sum_{i=1}^{n} f_i$. First, view the sequence of actions selected by $\mathcal{E}_t$ as a single "meta-action" $\tilde{a}_t$, and extend the domain of each f_i to include the meta-actions by defining $f_i(S \oplus \tilde{a}_t) = f_i(S \oplus a_t^i)$ for all $S \in \mathcal{S}$. Thus, the online algorithm produces a single schedule $\tilde{S} = \langle \tilde{a}_1, \tilde{a}_2, \ldots, \tilde{a}_T \rangle$ for all i. By construction,

$$\frac{r_t}{n} = \max_{a \in \mathcal{A}} \left\{ f\left(\tilde{S}_{\langle t-1 \rangle} \oplus a\right) - f\left(\tilde{S}_{\langle t-1 \rangle}\right) \right\} - \left(f\left(\tilde{S}_{\langle t-1 \rangle} \oplus \tilde{a}_t\right) - f\left(\tilde{S}_{\langle t-1 \rangle}\right) \right) .$$

[2] In general, $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_T$ will be T distinct copies of a single experts algorithm, such as randomized weighted majority.

Thus $\mathbf{OG_{unit}}$ behaves exactly like the greedy schedule $\bar{G}$ for the function f, where the t^{th} decision is made with additive error $\frac{r_t}{n}$.

Furthermore, the fact that the sequence $\langle f_1, f_2, \ldots, f_n \rangle$ satisfies Condition 2 implies that for any integer t ($1 \leq t \leq T$) and any schedule S, we have

$$\frac{f(\tilde{S}_{\langle t-1 \rangle} \oplus S) - f(\tilde{S}_{\langle t-1 \rangle})}{\ell(S)} \leq \max_{(v,\tau) \in \mathcal{V} \times \mathbb{R}_{>0}} \left\{ \frac{f(\tilde{S}_{\langle t-1 \rangle} \oplus \langle (v,\tau) \rangle) - f(\tilde{S}_{\langle t-1 \rangle})}{\tau} \right\}.$$

Thus the function f satisfies Condition 1, so the analysis of the greedy approximation algorithm in §3.2 applies to the schedule $\tilde{S}$. In particular, Theorem 6 implies that $R_{coverage} \leq \sum_{t=1}^{T} r_t = R$. Similarly, Theorem 7 implies that $R_{cost} \leq TR$. □

To complete the analysis, it remains to bound $\mathbb{E}[R]$. First, note that the payoffs to each experts algorithm $\mathcal{E}_t$ depend on the choices made by experts algorithms $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_{t-1}$, but not on the choices made by $\mathcal{E}_t$ itself. Thus, from the point of view of $\mathcal{E}_t$, the payoffs are generated by a non-adaptive adversary. Suppose that randomized weighted majority (WMR) is used as the subroutine experts algorithm. Because each payoff is at most 1 and there are n rounds, $\mathbb{E}[r_t] = O\left(\sqrt{G_{max} \ln |\mathcal{A}|}\right) = O\left(\sqrt{n \ln |\mathcal{A}|}\right)$, so a trivial bound is $\mathbb{E}[R] = O\left(T\sqrt{n \ln |\mathcal{A}|}\right)$. In fact, we can show that the worst case is when $G_{max} = \Theta\left(\frac{n}{T}\right)$ for all T experts algorithms, leading to the following improved bound. The proof is given in Appendix A.

Lemma 4. *Algorithm* $\mathbf{OG_{unit}}$*, run with WMR as the subroutine experts algorithm, has* $\mathbb{E}[R] = O\left(\sqrt{Tn \ln |\mathcal{A}|}\right)$ *in the worst case.*

Combining Lemmas 3 and 4 yields the following theorem.

Theorem 8. *Algorithm* $\mathbf{OG_{unit}}$*, run with WMR as the subroutine experts algorithm, has* $\mathbb{E}[R_{coverage}] = O\left(\sqrt{Tn \ln |\mathcal{A}|}\right)$ *and* $\mathbb{E}[R_{cost}] = O\left(T\sqrt{Tn \ln |\mathcal{A}|}\right)$ *in the worst case.*

4.3 From unit-cost actions to arbitrary actions

In this section we generalize the online greedy algorithm presented in the previous section to accommodate actions with arbitrary durations. Like $\mathbf{OG_{unit}}$, our generalized algorithm **OG** makes use of a series of experts algorithms $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_L$ (for L to be determined). On each round i, **OG** constructs a schedule S_i as follows: for $t = 1, 2, \ldots, L$, it uses $\mathcal{E}_t$ to choose an action $a_t^i = (v, \tau) \in \mathcal{A}$, and appends this action to S_i with probability $\frac{1}{\tau}$. The payoff that $\mathcal{E}_t$ associates with action a equals $\frac{1}{\tau}$ times the increase in f that would have resulted from appending a to the schedule-under-construction.

Algorithm OG
Input: integer L, experts algorithms $\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_L$.

For i from 1 to n:

1. Let $S_{i,0} = \langle\rangle$ be the empty schedule.
2. For each t, $1 \leq t \leq L$,
   (a) Use $\mathcal{E}_t$ to choose an action $a_t^i = (v, \tau) \in \mathcal{A}$.
   (b) With probability $\frac{1}{\tau}$, set $S_{i,t} = S_{i,t-1} \oplus \langle a \rangle$; else set $S_{i,t} = S_{i,t-1}$.
3. Select the schedule $S_i = S_{i,L}$.
4. Receive the job f_i.
5. For each t, $1 \leq t \leq L$, and each action $a \in \mathcal{A}$, feed back

$$x_{t,a}^i = \frac{1}{\tau}\left(f_i\left(S_{i,t-1} \oplus a\right) - f_i\left(S_{i,t-1}\right)\right)$$

as the payoff $\mathcal{E}_t$ would have received by choosing action a.

Our analysis of **OG** follows along the same lines as the analysis of $\mathbf{OG_{unit}}$ in the previous section. As in the previous section, we will view each experts algorithm $\mathcal{E}_t$ as selecting a single "meta-action" $\tilde{a}_t$. We extend the domain of each f_i to include the meta-actions by defining

$$f_i(S \oplus \tilde{a}_t) = \begin{cases} f_i(S \oplus a_t^i) & \text{if } a_t^i \text{ was appended to } S_i \\ f_i(S) & \text{otherwise.} \end{cases}$$

Thus, the online algorithm produces a single schedule $\tilde{S} = \langle \tilde{a}_1, \tilde{a}_2, \ldots, \tilde{a}_L \rangle$ for all i.

For the purposes of analysis, we will imagine that each meta-action $\tilde{a}_t$ *always* takes unit time (whereas in fact, $\tilde{a}_t$ takes unit time per job in expectation). We show later that this assumption does not invalidate any of our arguments.

Let $f = \frac{1}{n}\sum_{i=1}^n f_i$, and let $\tilde{S}_t = \langle \tilde{a}_1, \tilde{a}_2, \ldots, \tilde{a}_t \rangle$. As in the previous section, the fact that the sequence $\langle f_1, f_2, \ldots, f_n \rangle$ satisfies Condition 2 implies that f satisfies Condition 1 (even if the schedule S_1 in the statement of Condition 1 contains meta-actions). Thus $\tilde{S}$ can be viewed as a version of the greedy schedule in which the t^{th} decision is made with additive error (by definition) equal to

$$\epsilon_t = \max_{(v,\tau)\in\mathcal{A}} \left\{ \frac{1}{\tau}\left(f(\tilde{S}_{t-1} \oplus a) - f(\tilde{S}_{t-1})\right)\right\} - \left(f(\tilde{S}_{t-1} \oplus \tilde{a}_t) - f(\tilde{S}_{t-1})\right)$$

(where we have used the assumption that $\tilde{a}_t$ takes unit time).

As in the previous section, let r_t be the regret experienced by $\mathcal{E}_t$. In general, $\frac{r_t}{n} \neq \epsilon_t$. However, we claim that $\mathbb{E}[\epsilon_t] = \mathbb{E}\left[\frac{r_t}{n}\right]$. To see this, fix some integer t ($1 \leq t \leq L$), let $A_t = \langle a_t^1, a_t^2, \ldots, a_t^n \rangle$

be the sequence of actions selected by $\mathcal{E}_t$, and let y_t^i be the payoff received by $\mathcal{E}_t$ on round i (i.e., $y_t^i = x_{t,a_t^i}^i$). By construction,

$$y_t^i = \mathbb{E}\left[f_i(\tilde{S}_{t-1} \oplus \tilde{a}_t) - f_i(\tilde{S}_{t-1})|A_t, \tilde{S}_{t-1}\right] .$$

Thus,

$$\frac{r_t}{n} = \max_{(v,\tau)\in\mathcal{A}}\left\{\frac{1}{\tau}\left(f(\tilde{S}_{t-1} \oplus a) - f(\tilde{S}_{t-1})\right)\right\} - \mathbb{E}\left[f(\tilde{S}_{t-1} \oplus \tilde{a}_t) - f(\tilde{S}_{t-1})|A_t, \tilde{S}_{t-1}\right] .$$

Taking the expectation of both sides of the equations for ϵ_t and r_t then shows that $\mathbb{E}[\epsilon_t] = \mathbb{E}\left[\frac{r_t}{n}\right]$, as claimed.

We now prove a bound on $\mathbb{E}[R_{coverage}]$. As already mentioned, f satisfies Condition 1, so the greedy schedule's approximation guarantees apply to f. In particular, by Theorem 6, we have $R_{coverage} \leq \sum_{t=1}^{T} r_t$. Thus $\mathbb{E}[R_{coverage}] \leq \mathbb{E}[R]$, where $R = \sum_{t=1}^{T} r_t$.

To bound $\mathbb{E}[R_{coverage}]$, it remains to justify the assumption that each meta-action $\tilde{a}_t$ always takes unit time. Regardless of what actions are chosen by each experts algorithm, the schedule is defined for L time steps in expectation. Thus if we set $L = T$, the schedules S_i returned by **OG** satisfy the budget in expectation, as required in the definition of $R_{coverage}$. Thus, as far as $R_{coverage}$ is concerned, the meta-actions may as well take unit time (in which case $\ell(S_i) = T$ with probability 1). Combining the bound on $\mathbb{E}[R]$ stated in Lemma 4 with the fact that $\mathbb{E}[R_{coverage}] \leq \mathbb{E}[R]$ yields the following theorem.

Theorem 9. *Algorithm* **OG***, run with input $L = T$, has $\mathbb{E}[R_{coverage}] \leq \mathbb{E}[R]$. If WMR is used as the subroutine experts algorithm, then $\mathbb{E}[R] = O\left(\sqrt{Tn\ln|\mathcal{A}|}\right)$.*

The argument bounding $\mathbb{E}[R_{cost}]$ is similar, although somewhat more involved, and is given in Appendix A. Relative to the case of unit-cost actions addressed in the previous section, the additional complication here is that $\ell(S_i)$ is now a random variable, whereas in the definition of R_{cost} the cost of a schedule is always calculated up to time T. This complication can be overcome by making the probability that $\ell(S_i) < T$ sufficiently small, which can be accomplished by setting $L \gg T$ and applying concentration inequalities. However, $\mathbb{E}[R]$ grows as a function of L, so we do not want to make L too large. It turns out that the (approximately) best bound is obtained by setting $L = T\ln n$.

Theorem 10. *Algorithm* **OG***, run with input $L = T\ln n$, has $\mathbb{E}[R_{cost}] = O(T\ln n \cdot \mathbb{E}[R] + T\sqrt{n})$. In particular, $\mathbb{E}[R_{cost}] = O\left((\ln n)^{\frac{3}{2}} T\sqrt{Tn\ln|\mathcal{A}|}\right)$ if WMR is used as the subroutine experts algorithm.*

4.4 Dealing with limited feedback

Thus far we have assumed that, after specifying a schedule S_i, the online algorithm receives complete access to the job f_i. We now consider three more limited feedback settings that may arise in practice:

1. In the *priced feedback model*, to receive access to f_i we must pay a price C. Each time we do so, C is added to the regret $R_{coverage}$, and TC is added to the regret R_{cost}.

2. In the *partially transparent feedback model*, we only observe $f_i\left(S_{i\langle t\rangle}\right)$ for each $t > 0$.

3. In the *opaque feedback model*, we only observe $f_i\left(S_i\right)$.

The priced and partially transparent feedback models arise naturally in the case where action (v, τ) represents running a deterministic algorithm v for τ (additional) time units in order to solve some decision problem. Assuming we halt once some v returns an answer, we obtain exactly the information that is revealed in the partially transparent model. Alternatively, running each v until it terminates would completely reveal the function f_i, but incurs a computational cost.

Algorithm **OG** can be adapted to work in each of these three feedback settings. In all cases, the high-level idea is to replace the unknown quantities used by **OG** with (unbiased) estimates of those quantities. This technique has been used in a number of online algorithms (e.g., see [2, 4, 7]).

Specifically, for each day i and expert j, let $\hat{x}_j^i \in [0, 1]$ be an estimate of x_j^i, such that

$$\mathbb{E}\left[\hat{x}_j^i\right] = \gamma x_j^i + \delta^i$$

for some constant δ^i (which is independent of j). In order words, we require that $\frac{1}{\gamma}\left(\hat{x}_j^i - \delta^i\right)$ is an unbiased estimate of x_j^i. Furthermore, let $\hat{x}_j^i$ be independent of the choices made by the experts algorithm.

Let $\mathcal{E}$ be an experts algorithm, and let $\mathcal{E}'$ be the experts algorithm that results from feeding back $\hat{x}_j^i$ to $\mathcal{E}$ (in place of x_j^i) as the payoff $\mathcal{E}$ would have received by selecting expert j on day i. The following lemma relates the performance of $\mathcal{E}'$ to that of $\mathcal{E}$.

Lemma 5. *The worst-case expected regret that $\mathcal{E}'$ can incur over a sequence of n days is at most $\frac{R}{\gamma}$, where R is the worst-case expected regret that $\mathcal{E}$ can incur over a sequence of n days.*

Proof. Let $\hat{x} = \langle \hat{x}^1, \hat{x}^2, \ldots, \hat{x}^n \rangle$ be the sequence of estimated payoffs. Because the estimates $\hat{x}_j^i$ are independent of the choices made by $\mathcal{E}'$, we may imagine for the purposes of analysis that $\hat{x}$ is fixed in advance. Fix some expert j. By definition of R,

$$\mathbb{E}\left[\sum_{i=1}^{n} \hat{x}_{e_i}^i | \hat{x}\right] \geq \left(\sum_{i=1}^{n} \hat{x}_j^i\right) - R\,.$$

Taking the expectation of both sides with respect to the choice of $\hat{x}$ then yields

$$\mathbb{E}\left[\sum_{i=1}^{n} \left(\gamma x_{e_i}^i + \delta^i\right)\right] \geq \sum_{i=1}^{n} \left(\gamma x_j^i + \delta^i\right) - R$$

or rearranging,

$$\mathbb{E}\left[\sum_{i=1}^{n} x_{e_i}^i\right] \geq \left(\sum_{i=1}^{n} x_j^i\right) - \frac{R}{\gamma}\,.$$

Because j was arbitrary, it follows that $\mathcal{E}'$ has worst-case expected regret $\frac{R}{\gamma}$. □

4.4.1 The priced feedback model

In the priced feedback model, we use a technique similar to that of [7]. With probabiltiy γ, we will pay cost C in order to reveal f_i, and then feed the usual payoffs back to each experts algorithm $\mathcal{E}_t$. Otherwise, with probability $1-\gamma$, we feed back zero payoffs to each $\mathcal{E}_t$ (note that without paying cost C, we receive no information whatsoever about f_i, and thus we have no basis for assigning different payoffs to different actions). We refer to this algorithm as $\mathbf{OG^P}$. By Lemma 5, $\mathbb{E}[r_t]$ is bounded by $\frac{1}{\gamma}$ times the worst-case regret of $\mathcal{E}_t$. By bounding $\mathbb{E}[R_{coverage}]$ and $\mathbb{E}[R_{cost}]$ as a function of γ and then optimizing γ to minimize the bounds, we obtain the following theorem, a complete proof of which is given in Appendix A.

Theorem 11. *Algorithm* $\mathbf{OG^P}$*, run with WMR as the subroutine experts algorithm, has* $\mathbb{E}[R_{coverage}] = O\left((C\ln|\mathcal{A}|)^{\frac{1}{3}}(Tn)^{\frac{2}{3}}\right)$ *(when run with input* $L=T$*) and has* $\mathbb{E}[R_{cost}] = O\left((T\ln n)^{\frac{5}{3}}(C\ln|\mathcal{A}|)^{\frac{1}{3}}(n)^{\frac{2}{3}}\right)$ *(when run with input* $L = T\ln n$*) in the priced feedback model.*

4.4.2 The partially transparent feedback model

In the partially transparent feedback model, each $\mathcal{E}_t$ will run a copy of the **Exp3** algorithm [2], which is a randomized experts algorithm that only requires as feedback the payoff of the expert it actually selects. In the partially transparent feedback model, if $\mathcal{E}_t$ selects action $a_t^i = (v,\tau)$ on round i, it will receive feedback $f_i(S_{i,t-1}\oplus a_t^i) - f_i(S_{i,t-1})$ if a_t^i is appended to the schedule (with probability $\frac{1}{\tau}$), and will receive zero payoff otherwise. Observe that the information necessary to compute these payoffs is revealed in the partially transparent feedback model. Furthermore, the expected payoff that $\mathcal{E}_t$ receives if it selects action a is $x_{t,a}^i$, and the payoff that $\mathcal{E}_t$ receives from choosing action a on round i is independent from the choices made by $\mathcal{E}_t$ on previous rounds. Thus, by Lemma 5, the worst-case expected regret bounds of **Exp3** can be applied to the true payoffs $x_{t,a}^i$. The worst-case expected regret of **Exp3** is $O\left(\sqrt{n|\mathcal{A}|\ln|\mathcal{A}|}\right)$, so $\mathbb{E}[R] = O\left(L\sqrt{n|\mathcal{A}|\ln|\mathcal{A}|}\right)$. This bound, combined with Theorems 9 and 10, establishes the following theorem.

Theorem 12. *Algorithm* $\mathbf{OG}$*, run with* **Exp3** *as the subroutine experts algorithm, has* $\mathbb{E}[R_{coverage}] = O\left(T\sqrt{n|\mathcal{A}|\ln|\mathcal{A}|}\right)$ *(when run with input* $L=T$*) and has* $\mathbb{E}[R_{cost}] = O\left((T\ln n)^2\sqrt{n|\mathcal{A}|\ln|\mathcal{A}|}\right)$ *(when run with input* $L = T\ln n$*) in the partially transparent feedback model.*

4.4.3 The opaque feedback model

In the opaque feedback model, our algorithm and its analysis are similar to those of $\mathbf{OG^P}$. With probability $1-\gamma$, we feed back zero payoffs to each $\mathcal{E}_t$. Otherwise, with probability γ, we *explore* as follows. Pick t uniformly at random from $\{1,2,\ldots,L\}$, and pick an action $a=(v,\tau)$ uniformly at random from $\mathcal{A}$. Select the schedule $S_i = S_{i,t-1}\oplus a$. Observe $f_i(S_i)$, and feed $\frac{1}{\tau}$ times this value back to $\mathcal{E}_t$ as the payoff associated with action a. Finally, feed back zero for all other payoffs.

We refer to this algorithm as $\mathbf{OG^o}$. The key to its analysis is the following observation. Letting

$\hat{x}^i_{t,a}$ denote the payoff to experts algorithm $\mathcal{E}_t$ for choosing action $a = (v, \tau)$ on round i, we have

$$\mathbb{E}\left[\hat{x}^i_{t,a}\right] = \gamma \cdot \frac{1}{L} \cdot \frac{1}{|\mathcal{A}|} \cdot \frac{1}{\tau} \cdot f(S_{i,t-1} \oplus a) = \frac{\gamma}{L\,|\mathcal{A}|} x^i_{t,a} + \delta^i$$

where $x^i_{t,a} = \frac{1}{\tau}\left(f(S_{i,t-1} \oplus a) - f(S_{i,t-1})\right)$ and $\delta^i = \frac{\gamma}{L|\mathcal{A}|\tau} f(S_{i,t-1})$. Thus, $\hat{x}^i_{t,a}$ is a biased estimate of the correct payoff, and Lemma 5 implies that $\mathbb{E}[r_t]$ is at most $\frac{L|\mathcal{A}|}{\gamma}$ times the worst-case expected regret of $\mathcal{E}_t$.

The performance of $\mathbf{OG^o}$ is summarized in the following theorem, which we prove in Appendix A.

Theorem 13. *Algorithm* $\mathbf{OG^o}$*, run with WMR as the subroutine experts algorithm, has* $\mathbb{E}[R_{coverage}] = O\left(T(|\mathcal{A}| \ln |\mathcal{A}|)^{\frac{1}{3}} n^{\frac{2}{3}}\right)$ *(when run with input* $L = T$*) and has* $\mathbb{E}[R_{cost}] = O\left((T \ln n)^2 (|\mathcal{A}| \ln |\mathcal{A}|)^{\frac{1}{3}} n^{\frac{2}{3}}\right)$ *(when run with input* $L = T \ln n$*) in the opaque feedback model.*

4.5 Lower bounds on regret

In Appendix A we prove the following lower bounds on regret. The lower bounds apply to the online versions of two set-covering problems: MAX k-COVERAGE and MIN-SUM SET COVER. The offline versions of these two problems were defined in §1.4. The online versions are special cases of the online versions of BUDGETED MAXIMUM SUBMODULAR COVERAGE and MIN-SUM SUBMODULAR COVER, respectively. For a formal description of the online set covering problems, see the text leading up to the proofs of Theorems 14 and 15 in Appendix A.

It is worth pointing out that the lower bounds hold even in a *distributional* online setting in which the jobs $f_1, f_2, \ldots, f_n$ are drawn independently at random from a fixed distribution.

Theorem 14. *Any algorithm for online* MAX k-COVERAGE *has worst-case expected 1-regret* $\Omega\left(\sqrt{Tn \ln \frac{|\mathcal{V}|}{T}}\right)$*, where* $\mathcal{V}$ *is the collection of sets and* $T = k$ *is the number of sets selected by the online algorithm on each round.*

Theorem 15. *Any algorithm for online* MIN-SUM SET COVER *has worst-case expected 1-regret* $\Omega\left(T\sqrt{Tn \ln \frac{|\mathcal{V}|}{T}}\right)$*, where* $\mathcal{V}$ *is a collection of sets and* T *is the number of sets selected by the online algorithm on each round.*

In Appendix A we show that there exist exponential-time online algorithms for these online set covering problems whose regret matches the lower bounds in Theorem 14 (resp. Theorem 15) up to constant (resp. logarithmic) factors.

Note that the upper bounds in Theorem 8 match the lower bounds in Theorems 14 and 15 up to logarithmic factors, although the former apply to $(1 - \frac{1}{e})$-regret and 4-regret rather than 1-regret.

4.6 Refining the online greedy algorithm

We now discuss two simple modifications to **OG** that do not improve its worst-case guarantees, but that often improve its performance in practice (we make use of both of these modifications in our experimental evaluation).

4.6.1 Avoiding duplicate actions

In many practical applications, it is never worthwhile to perform the same action twice. As an example, suppose that an action $a = (v, \tau)$ represents performing a run of length τ of a deterministic algorithm v (and then removing the run from memory), and $f(S) = 1$ if performing the actions in S yields a solution to a problem instance, and $f(S) = 0$ otherwise. Clearly, performing a twice can never increase the value of f. In cases such as this, the online algorithm **OG** as currently defined may never "figure out" that it should avoid performing the same action twice, as the following example illustrates.

Example 2. Let $\mathcal{A} = \{a_1, a_2, \ldots, a_T\}$ be a set of T actions that each take unit time, and for all i, let $f_i(S)$ equal $\frac{1}{T}$ times the number of distinct actions that appear in S. Thus, the schedule $S^* = \langle a_1, a_2, \ldots, a_T \rangle$ has $f_i(S^*) = 1$ for all i, and is optimal in terms of coverage. Suppose we run **OG** on the sequence of jobs $\langle f_1, f_2, \ldots, f_n \rangle$. All actions yield equal payoff to $\mathcal{E}_1$. If $\mathcal{E}_1$ is a standard experts algorithm such as randomized weighted majority, it will choose actions uniformly at random. Given that $\mathcal{E}_1$ chooses actions uniformly at random, $\mathcal{E}_2$ will (asymptotically) choose actions uniformly at random as well. Inductively, all actions will be chosen at random. If so, the probability that any particular experts algorithm selects a unique action is $1 - (1 - \frac{1}{T})^T$ (which approaches $1 - \frac{1}{e}$ as $T \to \infty$). By linearity of expectation, the expected fraction of actions that are unique is exactly this quantity.

To improve performance on examples such as this one, we may force the online algorithm to return a schedule with no duplicate actions as follows. Just before job f_i arrives, obtain from each experts algorithm $\mathcal{E}_t$ a distribution over $\mathcal{A}$ (for experts algorithms such as randomized weighted majority, it is straightforward to obtain this distribution explicitly). We then sample from these distributions as follows. We first sample from $\mathcal{E}_1$ to obtain an action a_1^i. To obtain action a_t^i for $t > 1$, we repeatedly sample from the distribution returned by $\mathcal{E}_t$ until we obtain an action not in the set $\left\{a_1^i, a_2^i, \ldots, a_{t-1}^i\right\}$ (given the distribution, we can simulate this step without actually performing repeated sampling).

With this modification, **OG** always achieves coverage 1 for the job f described in example 2. Furthermore, this modification preserves the worst-case guarantees of the original version of **OG** (under the assumption performing the same action twice never increases the value of any function f_i). Informally, this follows from the fact that the expected payoff received by sampling from the modified distribution can *never* be smaller than the expected payoff received by sampling from the original distribution (because the payoffs associated with the experts corresponding to actions already in the schedule are all zero). For this reason, this modification never increases the worst-case regret of the experts algorithms, and our previous analysis carries through unchanged.

4.6.2 Independent versus dependent probabilities

Recall that in the case of arbitrary-cost actions, when an experts algorithm selects an action (v, τ) we add this action to the schedule independently with probability $\frac{1}{\tau}$. The fact that this addition is performed *independently* of the actions that are already in the schedule can lead to undesirable behavior, as the following example illustrates.

Example 3. Let $\mathcal{V} = \{v\}$ consist of a single activity, let $f(S) = 1$ if S contains the action (v, T), and let $f(S) = 0$ otherwise. Thus, the schedule $S^* = \langle (v, T) \rangle$ maximizes $f(S_{\langle T \rangle})$. However, $\mathbb{E}[f(S)] \leq 1 - (1 - \frac{1}{T})^T$ if S is a schedule returned by $\mathbf{OG}$. This is true because at most T experts algorithms can select the action (v, T), but in each case the action is only added to the schedule with probability $\frac{1}{T}$, so the probability that (v, T) is added to the schedule is at most $1 - (1 - \frac{1}{T})^T$, which approaches $1 - \frac{1}{e}$ as $T \to \infty$.

We can fix this problem as follows. When experts algorithm $\mathcal{E}_t$ selects an action $a_t = (v, \tau)$, we *increase* the probability that the action is in the schedule by $\frac{1}{\tau}$. In other words, if a_t has been picked by k experts algorithms so far but has still not been added to the schedule, then we add it to the schedule with probability $\frac{1}{\tau - k}$. Thus, if τ consecutive experts algorithms select the same action (v, τ), it will always be added to the schedule exactly once.

The schedules produced by this modified online algorithm still consume T time steps in expectation, and our previous analysis carries through to give same regret bounds on $R_{coverage}$ that were stated in Theorem 9. Unfortunately, the analysis for the bounds on R_{cost} stated in Theorem 10 depends critically on the use of independent probabilities, and does not carry through after having made this modification. Nevertheless, in our experiments in §6 we found that this modification was helpful in practice.

5 Open Problems

The results presented in this paper suggest several open problems:

1. *Avoiding discretization.* As currently defined, our online algorithm can only handle finite set of actions $\mathcal{A}$. Thus, to apply this online algorithm to a problem in which the actions have real-valued durations between 0 and 1, one might discretize the durations to be in the set $\left\{\frac{1}{T}, \frac{2}{T}, \ldots, 1\right\}$. To achieve the best performance, one would like to set T as large as possible, but the time and space required by the online algorithm grow linearly with T. It would be desirable to avoid discretization altogether, perhaps after making additional smoothness assumptions about the jobs f_i. A possible approach would be to consider the limiting behavior of our algorithm as $T \to \infty$, for some particular choice of subroutine experts algorithm.

2. *Lower bounds on 4-regret and* $1 - \frac{1}{e}$ *regret.* The lower bounds proved in §4.5 apply only to 1-regret, whereas our online algorithms optimize either 4 regret (in the case of R_{cost}) or $1 - \frac{1}{e}$ regret (in the case of $R_{coverage}$). It would be interesting to prove lower bounds on R_{cost} and $R_{coverage}$. Such lower bounds would hold for online algorithms that make decisions in polynomial time, under the assumption that P $\neq$ NP.

3. *An online version of the refined greedy approximation algorithm G'.* Recall that in §3.2 we showed that the offline greedy approximation algorithm is sub-optimal for a simple job involving two activities, and then considered an alternative greedy approximation algorithm that produces an optimal schedule for this job. The online algorithm presented in §4 is based on the original greedy approximation algorithm, and thus it also performs sub-optimally on this simple example. Although it appears non-trivial to do so, it would be interesting to develop an online version of the alternative greedy approximation algorithm that performed optimally on such examples.

6 Experimental Evaluation on SAT 2007 Competition Data

The annual SAT solver competition (`www.satcompetition.org`) is designed to encourage the development of efficient Boolean satisfiability solvers, which are used as subroutines in state-of-the-art model checkers, theorem provers, and planners. The competition consists of running each submitted solver on a number of benchmark instances, with a per-instance time limit. Solvers are ranked according to the number of instances they solve within each of three instance categories: *industrial*, *random*, and *hand-crafted*.

In this section we evaluate the online algorithm **OG** by using it to combine solvers from the 2007 SAT competition. To do so, we used data available on the competition web site[3] to construct a matrix t, where $t_{i,j}$ is the time that the j^{th} solver required on the i^{th} benchmark instance. We used this data to determine whether or not a given schedule would solve an instance within the time limit T (schedule S solves instance i if and only if, for some j, $S_{\langle T \rangle}$ contains actions $(h_j, \tau_1), (h_j, \tau_2), \ldots, (h_j, \tau_L)$ with $\sum_{l=1}^{L} \tau_l \geq t_{i,j}$). Within each instance category, we compared **OG** to the offline greedy schedule, to the individual solver that solved the most instances within the time limit, and to a schedule that ran each solver in parallel at equal strength. We ran **OG** in the full-information feedback model.

Table 1 summarizes the results. In each category, the offline greedy schedule and the online greedy algorithm solved more instances than any solver that was entered in the competition, and solve more instances than the naïve parallel schedule.

Table 1: Number of benchmark instances solved within time limit.

Category (#Instances)	**Offline**	**Online**	**Parallel**	**Top solver**
Industrial (234)	147	149	132	139
Random (511)	350	347	302	257
Hand-crafted (201)	114	107	95	98

[3]We use the data from phase 2 of the competition, available at `http://www.cril.univ-artois.fr/SAT07/`.

7 Conclusions

This paper considered an online resource allocation problem that generalizes several previously-studied online problems, and that has applications to algorithm portfolio design and the optimization of query processing in databases. The main contribution of this paper was an online version of a greedy approximation algorithm whose worst-case performance guarantees in the offline setting are the best possible assuming P $\neq$ NP.

References

[1] Noga Alon, Baruch Awerbuch, and Yossi Azar. The online set cover problem. In *Proceedings of the* 35^{th} *annual ACM Symposium on Theory of Computing*, pages 100–105, 2003.

[2] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[3] Giorgio Ausiello, Aristotelis Giannakos, and Vangelis Th. Paschos. Greedy algorithms for on-line set-covering and related problems. In *Twelfth Computing: The Australasian Theory Symposium (CATS2006)*, pages 145–151, 2006.

[4] Baruch Awerbuch and Robert Kleinberg. Adaptive routing with end-to-end feedback: Distributed learning and geometric approaches. In *Proceedings of the* 36^{th} *annual ACM Symposium on Theory of Computing*, pages 45–53, 2004.

[5] Shivnath Babu, Rajeev Motwani, Kamesh Munagala, Itaru Nishizawa, and Jennifer Widom. Adaptive ordering of pipelined stream filters. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pages 407–418, 2004.

[6] Nicolò Cesa-Bianchi, Yoav Freund, David Haussler, David Helmbold, Robert Schapire, and Manfred Warmuth. How to use expert advice. *Journal of the ACM*, 44(3):427–485, 1997.

[7] Nicolò Cesa-Bianchi, Gábor Lugosi, and Gilles Stoltz. Minimizing regret with label efficient prediction. *IEEE Transactions on Information Theory*, 51:2152–2162, 2005.

[8] Fan Chung and Linyuan Lu. Concentration inequalities and martingale inequalities – a survey. *Internet Math.*, 3(1):79–127, 2006.

[9] Edith Cohen, Amos Fiat, and Haim Kaplan. Efficient sequences of trials. In *Proceedings of the* 14^{th} *annual ACM-SIAM Symposium on Discrete Algorithms*, pages 737–746, 2003.

[10] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[11] Uriel Feige, László Lovász, and Prasad Tetali. Approximating min sum set cover. *Algorithmica*, 40(4):219–234, 2004.

[12] Carla P. Gomes and Bart Selman. Algorithm portfolios. *Artificial Intelligence*, 126:43–62, 2001.

[13] Carla P. Gomes, Bart Selman, Nuno Crato, and Henry Kautz. Heavy-tailed phenomena in satisfiability and constraint satisfications problems. *Journal of Automated Reasoning*, 24(1/2):67–100, 2000.

[14] Carla P. Gomes, Bart Selman, and Henry Kautz. Boosting combinatorial search through randomization. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 431–437, 1998.

[15] Bernardo A. Huberman, Rajan M. Lukose, and Tad Hogg. An economics approach to hard computational problems. *Science*, 275:51–54, 1997.

[16] Sham Kakade, Adam Kalai, and Katrina Ligett. Playing games with approximation algorithms. In *Proceedings of the* 39^{th} *annual ACM Symposium on Theory of Computing*, pages 546–555, 2007.

[17] Haim Kaplan, Eyal Kushilevitz, and Yishay Mansour. Learning with attribute costs. In *Proceedings of the* 37^{th} *annual ACM Symposium on Theory of Computing*, pages 356–365, 2005.

[18] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the* 9^{th} *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 137–146, 2003.

[19] Samir Khuller, Anna Moss, and Joseph (Seffi) Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.

[20] Andreas Krause and Carlos Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Proceedings of the* 21^{st} *Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 324–331, 2005.

[21] Andreas Krause and Carlos Guestrin. A note on the budgeted maximization of submodular functions. Technical Report CMU-CALD-05-103, Carnegie Mellon University, 2005.

[22] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

[23] Michael Luby, Alistair Sinclair, and David Zuckerman. Optimal speedup of Las Vegas algorithms. *Information Processing Letters*, 47:173–180, 1993.

[24] Adam Meyerson. Online facility location. In *FOCS '01: Proceedings of the* 42^{nd} *IEEE Symposium on Foundations of Computer Science*, pages 426–431, 2001.

[25] Kamesh Munagala, Shivnath Babu, Rajeev Motwani, Jennifer Widom, and Eiter Thomas. The pipelined set cover problem. In *Proceedings of the International Conference on Database Theory*, pages 83–98, 2005.

[26] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1):265–294, 1978.

[27] Matthew Streeter, Daniel Golovin, and Stephen F. Smith. Combining multiple heuristics online. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence*, pages 1197–1203, 2007.

[28] Matthew Streeter, Daniel Golovin, and Stephen F. Smith. Restart schedules for ensembles of problem instances. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence*, pages 1204–1210, 2007.

[29] Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32:41–43, 2004.

Appendix A: Additional Proofs

Theorem 5. $c\left(f, G'\right) \leq 4 \int_{t=0}^{\infty} 1 - \max_{S \in \mathcal{S}} \left\{ f\left(S_{\langle t \rangle}\right) \right\} dt \leq 4 \min_{S \in \mathcal{S}} \left\{ c\left(f, S\right) \right\}$.

Proof. Recall that $G' = \langle g'_1, g'_2, \ldots \rangle$, where $G'_j = \langle g'_1, g'_2, \ldots, g'_{j-1} \rangle$ and

$$g'_j = \operatorname*{arg\,max}_{(v,\tau) \in \mathcal{V} \times \mathbb{R}_{>0}} \frac{f\left(G'_j \oplus (v,\tau)\right) - f\left(G'_j\right)}{\int_{t'=0}^{\tau} 1 - f\left(G'_j + (v,t')\right) dt'} \,. \tag{7.1}$$

Let s'_j equal the j^{th} value of the $\arg\max$ in (7.1), multiplied by the quantity $1 - f(G'_j)$. We will make use of the following claim.

Claim 1. For any schedule S, any positive integer j, and any $t \geq 0$, $f\left(S_{\langle t \rangle}\right) \leq f\left(G'_j\right) + t s'_j$.

Proof. Fix an action $a = (v, \tau)$. By monotonicity of f, we have $\int_{t'=0}^{\tau} 1 - f\left(G'_j \oplus \langle (v,\tau) \rangle\right) dt' \leq \tau(1 - f\left(G'_j\right))$, or equivalently,

$$\frac{1}{\tau} \leq \frac{1 - f\left(G'_j\right)}{\int_{t'=0}^{\tau} 1 - f\left(G'_j + (v,\tau)\right) dt'} \,.$$

This and the definition of s'_j imply

$$\frac{f\left(G'_j \oplus \langle a \rangle\right) - f\left(G'_j\right)}{\tau} \leq \left(1 - f\left(G'_j\right)\right) \cdot \frac{f\left(G'_j \oplus \langle a \rangle\right) - f\left(G'_j\right)}{\int_{t'=0}^{\tau} 1 - f\left(G'_j \oplus \langle (v,t') \rangle\right) dt'} \leq s'_j \,.$$

The claim then follows by exactly the same argument that was used to prove Fact 1. □

The remainder of the proof parallels the proof of Theorem 4. Using Claim 1 and the argument in the proof of Theorem 4, we get that

$$\int_{t=0}^{\infty} 1 - \max_{S \in \mathcal{S}} \left\{ f\left(S_{\langle t \rangle}\right) \right\} dt \geq \sum_{j \geq 1} x_j (y_j - y_{j+1})$$

where $x_j = \frac{R_j}{2s'_j}$, $y_j = \frac{R_j}{2}$, and $R_j = 1 - f\left(G'_j\right)$. Letting $g'_j = (v_j, \tau_j)$, we have

$$\sum_{j \geq 1} x_j(y_j - y_{j+1}) = \frac{1}{4} \sum_{j \geq 1} \int_{t'=0}^{\tau_j} 1 - f\left(G'_j \oplus \langle (v_j, t') \rangle\right) dt' = \frac{1}{4} c\left(f, G'\right) .$$

which proves the theorem. □

We now prove the theorems concerning the performance of the greedy schedule $\bar{G}$, in which the j^{th} evaluation of the $\arg\max$ in (3.1) is performed with additive error ϵ_j. To ease notation, let $\bar{G} = \langle g_1, g_2, \ldots \rangle$, where $g_j = (v_j, \tau_j)$. Let $s_j = \frac{f(\bar{G}_{j+1}) - f(\bar{G}_j)}{\tau_j}$. To prove Theorems 6 and 7, we will make use of the following fact, which can be proved in exactly the same way as Fact 1.

Fact 2. *For any schedule S, any positive integer j, and any $t > 0$, we have $f(S_{\langle t \rangle}) \leq f(\bar{G}_j) + t \cdot (s_j + \epsilon_j)$.*

Theorem 6. *Let L be a positive integer, and let $T = \sum_{j=1}^{L} \tau_j$, where $g_j = (v_j, \tau_j)$. Then*

$$f\left(\bar{G}_{\langle T \rangle}\right) > \left(1 - \frac{1}{e}\right) \max_{S \in \mathcal{S}} \left\{ f\left(S_{\langle T \rangle}\right) \right\} - \sum_{j=1}^{L} \epsilon_j \tau_j .$$

Proof. Let $C^* = \max_{S \in \mathcal{S}} \left\{ f\left(S_{\langle T \rangle}\right) \right\}$, and for any positive integer j, let $\Delta_j = C^* - f\left(G_j\right)$. By Fact 2, $C^* \leq f\left(\bar{G}_j\right) + T(s_j + \epsilon_j)$. Thus

$$\Delta_j \leq T(s_j + \epsilon_j) = T\left(\frac{\Delta_j - \Delta_{j+1}}{\tau_j} + \epsilon_j\right) .$$

Rearranging this inequality gives $\Delta_{j+1} \leq \Delta_j \left(1 - \frac{\tau_j}{T}\right) + \tau_j \epsilon_j$. Unrolling this inequality (and using the fact that $1 - \frac{\tau_j}{T} < 1$ for all j), we get

$$\Delta_{L+1} \leq \Delta_1 \left(\prod_{j=1}^{L} 1 - \frac{\tau_j}{T} \right) + \sum_{j=1}^{L} \tau_j \epsilon_j .$$

Let $E = \sum_{j=1}^{L} \tau_j \epsilon_j$. Subject to the constraint $\sum_{j=1}^{L} \tau_j = T$, the product series is maximized when $\tau_j = \frac{T}{L}$ for all j. Thus we have

$$C^* - f\left(\bar{G}_{L+1}\right) = \Delta_{L+1} \leq \Delta_1 \left(1 - \frac{1}{L}\right)^L + E < \Delta_1 \frac{1}{e} + E \leq C^* \frac{1}{e} + E .$$

Thus $f\left(\bar{G}_{L+1}\right) > (1 - \frac{1}{e}) C^* - E$, as claimed. □

Theorem 7. *Let L be a positive integer, and let $T = \sum_{j=1}^{L} \tau_j$, where $g_j = (v_j, \tau_j)$. For any schedule S, define $c^T(f, S) \equiv \int_{t=0}^{T} 1 - f\left(S_{\langle t \rangle}\right) dt$. Then*

$$c^T\left(f, \bar{G}\right) \leq 4 \int_{t=0}^{\infty} 1 - \max_{S \in \mathcal{S}} \left\{ f\left(S_{\langle t \rangle}\right) \right\} dt + \sum_{j=1}^{L} E_j \tau_j \,.$$

where $E_j = \sum_{l<j} \epsilon_l \tau_l$.

Proof. Let $R_j = 1 - f(G_j)$, let $R'_j = R_j - E_j$. Assume for the moment that $R_L \geq E_L$, so that R'_j is non-negative for $j \leq L$. Let $s'_j = s_j + \epsilon_j$. By construction,

$$R'_j - R'_{j+1} = f\left(\bar{G}_{j+1}\right) - f\left(\bar{G}_j\right) + \epsilon_j \tau_j = \tau_j s'_j \,. \tag{7.2}$$

Let $x_j = \frac{R'_j}{2s'_j}$; let $y_j = \frac{R'_j}{2}$; and let $h(x) = 1 - \max_S \left\{ f\left(S_{\langle x \rangle}\right) \right\}$. By Fact 2,

$$\max_S \left\{ f(S_{\langle x_j \rangle}) \right\} \leq f(G_j) + x_j s'_j = f(G_j) + \frac{R'_j}{2} \,.$$

Thus $h(x_j) \geq R_j - \frac{R'_j}{2} = \frac{R_j + E_j}{2} \geq y_j$. The monotonicity of f implies that $h(x)$ is non-increasing and (together with the fact that E_j is non-decreasing as a function of j) implies that the sequence $\langle y_1, y_2, \ldots \rangle$ is non-increasing. As illustrated in Figure 1, these facts imply that $\int_{x=0}^{\infty} h(x)\, dx \geq \sum_{j=1}^{L} x_j (y_j - y_{j+1})$. Thus we have

$$\begin{aligned}
\int_{t=0}^{\infty} 1 - \max_{S \in \mathcal{S}} \left\{ f\left(S_{\langle t \rangle}\right) \right\} dt &= \int_{x=0}^{\infty} h(x)\, dx \\
&\geq \sum_{j=1}^{L} x_j (y_j - y_{j+1}) && \text{(Figure 1)} \\
&= \frac{1}{4} \sum_{j=1}^{L} R'_j \frac{\left(R'_j - R'_{j+1}\right)}{s'_j} \\
&= \frac{1}{4} \sum_{j=1}^{L} R'_j \tau_j && \text{(equation (7.2))} \\
&= \frac{1}{4} \left(\sum_{j=1}^{L} R_j \tau_j - \sum_{j \geq 1} E_j \tau_j \right) \\
&\geq \frac{1}{4} c^T(f, G) - \frac{1}{4} \sum_{j=1}^{L} E_j \tau_j && \text{(monotonicity of } f\text{)}
\end{aligned}$$

which proves the theorem, subject to the assumption that $R_L \geq E_L$.

Now suppose $R_L < E_L$. Let K be the largest integer such that $R_K \geq E_K$, and let $T_K = \sum_{j=1}^{K} \tau_j$. By the argument just given,

$$c^{T_K}(f, G) \leq 4 \int_{t=0}^{\infty} 1 - \max_{S \in \mathcal{S}} \left\{ f\left(S_{\langle t \rangle}\right) \right\} dt + \sum_{j=1}^{K} E_j \tau_j \, .$$

Thus to prove the theorem, it suffices to show that $c^T(f, G) \leq c^{T_K}(f, G) + \sum_{j=K+1}^{L} E_j \tau_j$. This holds because

$$\begin{aligned} c^T(f, G) - c^{T_K}(f, G) &= \int_{t=T_K}^{T} 1 - f\left(\bar{G}_{\langle t \rangle}\right) dt \\ &\leq (T - T_K)(1 - f(\bar{G}_{\langle T_K \rangle})) \\ &= (T - T_K) R_{K+1} \\ &< (T - T_K) E_{K+1} \\ &\leq \sum_{j=K+1}^{L} E_j \tau_j \, . \end{aligned}$$

□

Lemma 2. *Any sequence $\langle f_1, f_2, \ldots, f_n \rangle$ of jobs satisfies Condition 2. That is, for any sequence $S_1, S_2, \ldots, S_n$ of schedules and any schedule S,*

$$\frac{\sum_{i=1}^{n} f_i(S_i \oplus S) - f_i(S_i)}{\ell(S)} \leq \max_{(v,\tau) \in \mathcal{V} \times \mathbb{R}_{>0}} \left\{ \frac{\sum_{i=1}^{n} f_i(S_i \oplus \langle (v, \tau) \rangle) - f_i(S_i)}{\tau} \right\} .$$

Proof. Let r denote the right hand side of the inequality. Let $S = \langle a_1, a_2, \ldots, a_L \rangle$, where $a_l = (v_l, \tau_l)$. Let

$$\Delta_{i,l} = f_i(S_i \oplus \langle a_1, a_2, \ldots, a_l \rangle) - f(S_i \oplus \langle a_1, a_2, \ldots, a_{l-1} \rangle) \, .$$

We have

$$\begin{aligned}
\sum_{i=1}^{n} f_i(S_i \oplus S) &= \sum_{i=1}^{n} \left(f_i(S_i) + \sum_{l=1}^{L} \Delta_{i,l} \right) && \text{(telescoping series)} \\
&\leq \sum_{i=1}^{n} \left(f_i(S_i) + \sum_{l=1}^{L} \left(f_i(S_i \oplus \langle a_l \rangle) - f(S_i) \right) \right) && \text{(submodularity)} \\
&= \sum_{i=1}^{n} f_i(S_i) + \sum_{l=1}^{L} \sum_{i=1}^{n} \left(f_i(S_i \oplus \langle a_l \rangle) - f(S_i) \right) \\
&\leq \sum_{i=1}^{n} f_i(S_i) + \sum_{l=1}^{L} r \cdot \tau_l && \text{(definition of } r\text{)} \\
&= \sum_{i=1}^{n} f_i(S_i) + r \cdot \ell\left(S\right) \ .
\end{aligned}$$

Rearranging this inequality gives $\frac{\sum_{i=1}^{n} f_i(S_i \oplus S) - f_i(S_i)}{\ell(S)} \leq r$, as claimed. □

Lemma 4. *Algorithm* $\mathbf{OG_{unit}}$ *with randomized weighted majority as the subroutine experts algorithm has* $\mathbb{E}\left[R\right] = O\left(\sqrt{Tn \ln |\mathcal{A}|}\right)$ *in the worst case.*

Proof. Let $k = |\mathcal{A}|$. Let x_t be the total payoff received by $\mathcal{E}_t$, and let $g_t = x_t + r_t$ be the total payoff that could have been received by $\mathcal{E}_t$ in hindsight (had it been forced to choose a fixed expert each day). Because $\sum_{t=1}^{T} x_t \leq n$, we have $\sum_{t=1}^{T} g_t \leq n + R$. Using WMR, $\mathbb{E}\left[r_t\right] = O\left(\sqrt{g_t \ln k}\right)$. Using WMR, the actual value of r_t will be tightly concentrated about its expectation, as can be shown using Azuma's inequality. In particular, because $g_t \leq n$, the probability that $R > n$ is exponentially small. Assuming $R \leq n$, we have $\sum_{t=1}^{T} g_t \leq 2n$. Subject to this constraint, $\sum_{t=1}^{T} \sqrt{g_t}$ is maximized when $g_t = \frac{2n}{T}$ for all t. Thus in the worst case, $\mathbb{E}\left[R\right] = O\left(\sqrt{Tn \ln k}\right)$. □

In order to prove Theorem 10, we first prove the following lemma. The lemma relates the expected cost of the schedule S_i (selected by $\mathbf{OG}$ on round i) to the expected cost S_i would incur if, hypothetically, each of the "meta-actions" selected by each experts algorithm $\mathcal{E}_t$ consumed unit time on every job (require that this assumption was made in the analysis in the main text).

Lemma 6. *Fix a sequence of jobs* $\langle f_1, f_2, \ldots, f_n \rangle$ *and an integer* i $(1 \leq i \leq n)$*. Let* S_i *be the schedule produced by* $\mathbf{OG}$ *to use on job* f_i*, and let* $S_{i,t-1}$ *denote the partial schedule that exists after the first* $t-1$ *experts algorithms has selected actions. Then*

$$\mathbb{E}\left[c^{\ell(S_i)}\left(f_i, S_i\right)\right] \leq \mathbb{E}\left[\sum_{t=1}^{L} \left(1 - f_i(S_{i,t-1})\right)\right] \ .$$

Proof. Fix some t. Let $a_t^i = (v, \tau)$ be the action selected by $\mathcal{E}_t$ on round i, and define

$$c_t^i = \begin{cases} \int_{t'=0}^{\tau} 1 - f_i(S_{i,t-1} \oplus \langle (v, t') \rangle)\, dt' & \text{if } a_t^i \text{ is appended to } S_i \\ 0 & \text{otherwise.} \end{cases}$$

By construction, $c^{\ell(S_i)}(f_i, S_i) = \sum_{t=1}^{L} c_t^i$. Because a_t^i is appended to S_i with probability $\frac{1}{\tau}$, and because f_i is monotone, we have

$$\mathbb{E}\left[c_t^i | S_{i,t-1}\right] = \frac{1}{\tau} \int_{t'=0}^{\tau} 1 - f_i(S_{i,t-1} \oplus \langle (v, t') \rangle)\, dt' \leq 1 - f_i(S_{i,t-1})\,.$$

Taking the expectation of both sides yields $\mathbb{E}[c_t^i] \leq \mathbb{E}[1 - f_i(S_{i,t-1})]$. Then by linearity of expectation,

$$\mathbb{E}\left[c^{\ell(S_i)}(f_i, S_i)\right] = \mathbb{E}\left[\sum_{t=1}^{L} c_t^i\right] \leq \mathbb{E}\left[\sum_{t=1}^{L} (1 - f_i(S_{i,t-1}))\right]\,.$$

□

Theorem 10. *Algorithm* **OG***, run with input $L = T \ln n$, has $\mathbb{E}[R_{cost}] = O(T \ln n \cdot \mathbb{E}[R] + T\sqrt{n})$. In particular, $\mathbb{E}[R_{cost}] = O\left((\ln n)^{\frac{3}{2}} T \sqrt{Tn \ln |\mathcal{A}|}\right)$ if WMR is used as the subroutine experts algorithm.*

Proof. The arguments in the main text showed that **OG** can be viewed as a version of the greedy schedule for the function $f = \frac{1}{n}\sum_{i=1}^{n} f_i$, in which the t^{th} decision is made with additive error ϵ_t, under the assumption that all "meta-actions" $\tilde{a}_t^i$ require unit time on every job. Thus by Theorem 7, we have

$$\sum_{i=1}^{n}\sum_{t=1}^{L}(1 - f_i(S_{i,t-1})) \leq 4 \cdot \min_{S \in \mathcal{S}} \left\{\sum_{i=1}^{n} c(f_i, S)\right\} + nL\sum_{t=1}^{L}\epsilon_t\,. \tag{7.3}$$

Also recall from the main text that $\mathbb{E}[\epsilon_t] = \mathbb{E}\left[\frac{r_t}{n}\right]$, where r_t is the regret experienced by $\mathcal{E}_t$, and that we define $R = \sum_{t=1}^{L} r_t$. Thus, we have

$$\begin{aligned} \mathbb{E}\left[\sum_{i=1}^{n} c^{\ell(S_i)}(f_i, S_i)\right] &\leq \mathbb{E}\left[\sum_{i=1}^{n}\sum_{t=1}^{L}(1 - f_i(S_{i,t-1}))\right] && \text{(Lemma 6)} \\ &\leq 4 \cdot \min_{S \in \mathcal{S}} \left\{\sum_{i=1}^{n} c(f_i, S)\right\} + L \cdot \mathbb{E}[R]\,. && \text{(equation 7.3)} \end{aligned}$$

If it was always the case that $\ell(S_i) \geq T$, then we would have $c^T(f_i, S_i) \leq c^{\ell(S_i)}(f_i, S_i)$, and this inequality would imply $\mathbb{E}[R_{cost}] \leq L \cdot \mathbb{E}[R]$. In order to bound $\mathbb{E}[R_{cost}]$, we now address the possibility that $\ell(S_i) < T$. Letting $p_i = \mathbb{P}[\ell(S_i) < T]$, we have

$$\begin{aligned} \mathbb{E}\left[c^T(S_i, f_i)\right] &= (1 - p_i) \cdot \mathbb{E}\left[c^T(S_i, f_i) \,|\ell(S_i) \geq T\right] + p_i \cdot \mathbb{E}\left[c^T(S_i, f_i) \,|\ell(S_i) < T\right] \\ &\leq \mathbb{E}\left[c^{\ell(S_i)}(f_i, S_i)\right] + p_i \cdot T\,. \end{aligned}$$

Putting these inequalities together yields

$$\mathbb{E}\left[R_{cost}\right] \leq L \cdot \mathbb{E}\left[R\right] + T\sum_{i=1}^{n} p_i \,. \tag{7.4}$$

We now bound p_i. As already mentioned, $\mathbb{E}\left[\ell\left(S_i\right)\right] = L$ regardless of which actions are selected by the various experts algorithms. If $L \gg T$, then $\ell\left(S_i\right)$ will be sharply concentrated about its mean, as we can prove using standard concentration inequalities (e.g., Theorem 5 of [8]). In particular, for any $\lambda > 0$, we have

$$\mathbb{P}\left[\ell\left(S_i\right) \leq L - \lambda\right] <= \exp\left(-\frac{\lambda^2}{2LT}\right) \,.$$

Setting $\lambda = L - T$ and simplifying yields $p_i \leq \exp\left(-\frac{L}{2T} + 1\right)$. Setting $L = T\ln n$ then yields $p_i \leq \frac{e}{\sqrt{n}}$, so the right hand side of (7.4) is $O\left(T\sqrt{n}\right)$. Thus $\mathbb{E}\left[R_{cost}\right] = O\left(T\ln n \cdot \mathbb{E}\left[R\right] + T\sqrt{n}\right)$, as claimed. Substituting the bound on $\mathbb{E}\left[R\right]$ stated in Lemma 4 then proves the claim about WMR. □

Theorem 11. *Algorithm* $\mathbf{OG^P}$*, run with WMR as the subroutine experts algorithm, has* $\mathbb{E}\left[R_{coverage}\right] = O\left((C\ln|\mathcal{A}|)^{\frac{1}{3}}(Tn)^{\frac{2}{3}}\right)$ *(when run with input* $L = T$*) and has* $\mathbb{E}\left[R_{cost}\right] = O\left((T\ln n)^{\frac{5}{3}}(C\ln|\mathcal{A}|)^{\frac{1}{3}}(n)^{\frac{2}{3}}\right)$ *(when run with input* $L = T\ln n$*) in the priced feedback model.*

Proof. Let M be the number of exploration rounds (so $\mathbb{E}\left[M\right] = \gamma n$). The maximum payoff to any single expert cannot exceed M. Thus, by Lemma 5 and the regret bound of WMR, we have $\mathbb{E}\left[r_t|M\right] = O\left(\frac{1}{\gamma}\sqrt{M\ln|\mathcal{A}|}\right)$. Using the fact that $\mathbb{E}\left[\sqrt{X}\right] \leq \sqrt{\mathbb{E}\left[X\right]}$ for any non-negative random variable X, this implies

$$\mathbb{E}\left[r_t\right] = \mathbb{E}\left[\mathbb{E}\left[r_t|M\right]\right] = O\left(\frac{1}{\gamma}\sqrt{\mathbb{E}\left[M\right]\ln|\mathcal{A}|}\right) = O\left(\sqrt{\frac{n}{\gamma}\ln|\mathcal{A}|}\right) \,.$$

By Theorem 9, we have $\mathbb{E}\left[R_{coverage}\right] \leq \mathbb{E}\left[R\right] + C\gamma n = O\left(L\sqrt{\frac{n}{\gamma}\ln|\mathcal{A}|}\right) + C\gamma n$. Setting $\gamma = \left(\frac{L}{C}\sqrt{\frac{\ln|\mathcal{A}|}{n}}\right)^{\frac{2}{3}}$ then yields $\mathbb{E}\left[R_{coverage}\right] = O\left((C\ln|\mathcal{A}|)^{\frac{1}{3}}(Ln)^{\frac{2}{3}}\right)$, as claimed.

Similarly, by Theorem 10, we have $\mathbb{E}\left[R_{cost}\right] \leq L\cdot\mathbb{E}\left[R\right] + T\sqrt{n} + TC\gamma n = L\cdot O\left(L\sqrt{\frac{n}{\gamma}\ln|\mathcal{A}|} + C\gamma n\right)$, so the same setting of γ yields $\mathbb{E}\left[R_{cost}\right] = O\left(L^{\frac{5}{3}}(C\ln|\mathcal{A}|)^{\frac{1}{3}}n^{\frac{2}{3}}\right)$. □

Theorem 13. *Algorithm* $\mathbf{OG^O}$*, run with WMR as the subroutine experts algorithm, has* $\mathbb{E}\left[R_{coverage}\right] = O\left(T(|\mathcal{A}|\ln|\mathcal{A}|)^{\frac{1}{3}}n^{\frac{2}{3}}\right)$ *(when run with input* $L = T$*) and has* $\mathbb{E}\left[R_{cost}\right] = O\left((T\ln n)^2(|\mathcal{A}|\ln|\mathcal{A}|)^{\frac{1}{3}}n^{\frac{2}{3}}\right)$ *(when run with input* $L = T\ln n$*) in the opaque feedback model.*

Proof. We showed in the main text that $\mathbb{E}\left[\hat{x}^i_{t,a}\right] = \frac{\gamma}{L|\mathcal{A}|}x^i_{t,a} + \delta_i$, where $\hat{x}^i_{t,a}$ is the estimated payoff fed back by $\mathbf{OG^O}$ and $x^i_{t,a}$ is the true payoff. Thus by Lemma 5, $\mathbb{E}\left[r_t\right]$ is bounded by $\frac{|\mathcal{A}|L}{\gamma}$ times the worst-case regret of $\mathcal{E}_t$. Using the same argument we used in the proof of Theorem 11, we get

$\mathbb{E}[R] = O\left(L\sqrt{\frac{n}{\gamma'}\ln|\mathcal{A}|}\right)$, where $\gamma' = \frac{\gamma}{|\mathcal{A}|L}$. By Theorem 9, we have $\mathbb{E}[R_{coverage}] \leq \mathbb{E}[R] + \gamma n = O\left(L\sqrt{\frac{n}{\gamma'}\ln|\mathcal{A}|}\right) + C\gamma' n$, where $C = L|\mathcal{A}|$. As in the proof of Theorem 9, setting $\gamma' = \left(\frac{L}{C}\sqrt{\frac{\ln|\mathcal{A}|}{n}}\right)^{\frac{2}{3}}$ then yields $\mathbb{E}[R_{coverage}] = O\left((C\ln|\mathcal{A}|)^{\frac{1}{3}}(Ln)^{\frac{2}{3}}\right) = O\left(T(|\mathcal{A}|\ln|\mathcal{A}|)^{\frac{1}{3}}n^{\frac{2}{3}}\right)$, and the same setting of γ' yields $\mathbb{E}[R_{cost}] = O\left(L^{\frac{5}{3}}(C\ln|\mathcal{A}|)^{\frac{1}{3}}n^{\frac{2}{3}}\right) = O\left((T\ln n)^2(|\mathcal{A}|\ln|\mathcal{A}|)^{\frac{1}{3}}n^{\frac{2}{3}}\right)$. □

We now prove lower bounds on regret. As mentioned in the main text, our lower bounds will hold for the online versions of MAX k-COVERAGE and MIN-SUM SET COVER.

We consider the following online version of MAX k-COVERAGE. One is given a collection $\mathcal{C}$ of sets, where each set in $\mathcal{C}$ is a subset of a universe $E = \{e_1, e_2, \ldots, e_n\}$. One cannot examine the sets (or even determine their cardinalities) directly. On round i of the game, one must specify a subcollection $C \subset \mathcal{C}$, with $|C| = k$. One then receives a reward of 1 if element e_i belongs to some set in the collection, and receives a reward of zero otherwise. One then learns as feedback which sets e_i belonged to.

This problem is a special case of the online version of BUDGETED MAXIMUM SUBMODULAR COVERAGE. To see this, let $\mathcal{V} = \mathcal{C}$ be the set of activities, and think of the action (v, τ) as including the set v in the collection assuming $\tau \geq 1$, and having no effect otherwise. For any schedule S, let $f_i(S) = 1$ if one of the sets added to the collection by S contains e_i, and let $f_i(S) = 0$ otherwise. Then BUDGETED MAXIMUM SUBMODULAR COVERAGE on the sequence of jobs $\langle f_1, f_2, \ldots, f_n\rangle$, with time limit $T = k$, is exactly the problem just described.

The online version of MIN-SUM SET COVER is similar, except that instead of specifying a subcollection of cardinality k, one specifies a sequence of k sets from $\mathcal{C}$. One then incurs a loss equal to the number of sets one must look through in the sequence in order to find e_i, or a loss of k if e_i does not appear in the sequence at all. By the arguments just given, this is equivalent to online MIN-SUM SUBMODULAR COVER on the sequence of jobs $\langle f_1, f_2, \ldots, f_n\rangle$, where $T = k$ is the time at which schedule costs are truncated.

To prove lower bounds on regret, we will require the following technical lemma. The proof is a straightforward generalization of the proof of Lemma 3.2.1 of [6], which considered the special case $p = \frac{1}{2}$.

Lemma 7 ([6])**.** *Let $X_1, X_2, \ldots, X_s$ be s independent random variables, where X_i equals the number of heads in n flips of a coin with bias p. Let $\mu = np$ and let $\sigma = \sqrt{np(1-p)}$. Then*

$$\mathbb{E}[\max\{X_1, X_2, \ldots, X_s\}] = \mu + \Omega\left(\sigma\sqrt{\ln s}\right).$$

Theorem 14. *Any algorithm for online* MAX k-COVERAGE *has worst-case expected 1-regret $\Omega\left(\sqrt{Tn\ln\frac{|\mathcal{V}|}{T}}\right)$, where $\mathcal{V}$ is the collection of sets and $T = k$ is the number of sets selected by the online algorithm on each round.*

Proof. Let $\mathcal{V}$ be a collection of sets. On each round of the online game, whether or not a given set covers the element will be determined by flipping a coin of bias $p = \frac{1}{2T}$. Thus, regardless of which T sets are selected by the online algorithm, the probability that it covers the element is $q = 1 - \left(1 - \frac{1}{2T}\right)^T \in [\frac{1}{2}, \frac{1}{\sqrt{e}}]$, and the expected number of elements the online algorithm covers is nq.

We now consider the number of elements that could have been covered in hindsight. Let $R = \sqrt{\frac{n}{T} \ln \frac{|\mathcal{V}|}{T}}$. Partition $\mathcal{V}$ into T bins, each of size $s = \frac{|\mathcal{V}|}{T}$. Let S_i^* denote the set in the i^{th} bin which covers the largest number of elements, and let $C^* = \{S_1^*, S_2^*, \ldots, S_T^*\}$. To prove the theorem, it suffices to show that C^* covers $nq + \Omega(TR)$ elements in expectation.

Let a collection $C = \{S_1, S_2, \ldots, S_T\}$ consist of a *random* set drawn from each bin. In expectation C covers nq elements. Let $x_i := |S_i^*| - |S_i|$ and note that $x_i \geq 0$ and $\mathbb{E}[x_i] = \Omega(R)$ by Lemma 7. Randomly mark x_i elements of S_i^* and let M_i and U_i denote the marked and unmarked elements of S_i^*, respectively. Note that the collection $\{U_i : 1 \leq i \leq T\}$ covers nq elements in expectation. Let X denote the (random) number of additional elements covered by the collection $\{M_i : 1 \leq i \leq T\}$ (i.e., $X = |\cup_i M_i - \cup_i U_i|$). We claim that $\mathbb{E}[X] = \Omega(TR)$. To prove this, define ξ to be the event "for all $S \in \mathcal{C}$, $|S| \leq n/T$" and let Y be the number of marked elements covered exactly once in C^*. We will show that $\mathbb{E}[Y \mid \xi]\,\mathbb{P}[\xi] = \Omega(TR)$. Since $\mathbb{E}[Y \mid \xi] \cdot \mathbb{P}[\xi] \leq \mathbb{E}[Y] \leq \mathbb{E}[X]$, this is sufficient to complete the proof.

Fix i and any element $e \in M_i$. Then $\mathbb{P}[e \text{ uniquely covered} \mid \xi] = \prod_{j \neq i} \left(1 - |S_j^*|/n\right) \geq (1 - 1/T)^{T-1} \geq 1/e$. This implies $\mathbb{E}[Y \mid \xi] \geq \frac{1}{e}\mathbb{E}[\sum_i |M_i|] = \frac{1}{e}\Omega(TR)$, since, as mentioned, $\mathbb{E}[|M_i|] = \Omega(R)$ for all i. Finally, the Chernoff bound easily yields $\mathbb{P}[\xi] \geq (1 - |\mathcal{V}| \cdot \exp\{-n/8T\}) = 1 - o(1)$, and so $\mathbb{E}[Y \mid \xi] \cdot \mathbb{P}[\xi] = \Omega(TR)$ as claimed. □

The lower bound in Theorem 14 is optimal up to constant factors. To see this, observe that running randomized weighted majority with one expert for each of the $\binom{|\mathcal{V}|}{T}$ possible collections of T sets yields worst-case regret $O\left(\sqrt{n \ln \binom{|\mathcal{V}|}{T}}\right) = O\left(\sqrt{nT \ln \frac{|\mathcal{V}|}{T}}\right)$ for online MAX k-COVERAGE, using the fact that $\binom{|\mathcal{V}|}{T} \leq \left(\frac{|\mathcal{V}|e}{T}\right)^T$. Similarly, using a separate expert for each of the $O\left(|\mathcal{V}|^T\right)$ possible permutations of T sets yields regret $O\left(T\sqrt{Tn \ln |\mathcal{V}|}\right)$ for online MIN-SUM SET COVER, which shows that the lower bound in Theorem 15 is optimal up to logarithmic factors.

Theorem 15. *Any algorithm for online* MIN-SUM SET COVER *has worst-case expected 1-regret* $\Omega\left(T\sqrt{Tn \ln \frac{|\mathcal{V}|}{T}}\right)$*, where $\mathcal{V}$ is a collection of sets and T is the number of sets selected by the online algorithm on each round.*

Proof. We use the same construction as in the proof of Theorem 14. Define the *coverage time* of a schedule $S_i = \langle S_1^i, S_2^i, \ldots, S_T^i \rangle$ to be the smallest t such that S_t^i covers the i^{th} element, or T if no such t exists. As in the proof of Theorem 14, the probability that the online algorithm covers any particular element is q. Given that the online algorithm covers an element, the expected coverage time is zT for some $z < \frac{1}{2}$. Thus, any online algorithm has expected coverage time $\bar{t} = qzT + (1 - q)T$ for each element.

Now consider the schedule $S^* = \langle S_1^*, S_2^*, \ldots, S_T^* \rangle$, where $S_i^* = U_i \cup M_i$ was defined in the proof of Theorem 14, and let the sets be indexed in random order. The schedule $U = \langle U_1, U_2, \ldots, U_T \rangle$ is statistically equivalent to a random schedule, and thus has expected coverage time $\bar{t}$ per element. Using S^* in place of U causes X additional elements to be covered, where $\mathbb{E}[X] = \Omega\left(\sqrt{Tn \ln \frac{|\mathcal{V}|}{T}}\right)$. Because the sets in S^* are ordered randomly, the expected coverage time for each of the X additional elements is at most $\frac{T}{2}$. Thus, the total expected coverage time of S^* is smaller than that of U by at least $\frac{T}{2}\mathbb{E}[X] = \Omega\left(T\sqrt{Tn \ln \frac{|\mathcal{V}|}{T}}\right)$. □