The Power of Adaptivity for Decision-Making under Uncertainty

by

Rohan Ghuge

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Industrial and Operations Engineering)
in the University of Michigan
2023

Rohan Ghuge

rghuge@umich.edu

ORCID iD: 0000-0002-4681-9121

Dedication

Dedicated to my grandfather.

Acknowledgements

First of all, I would like to express my heartfelt gratitude to my advisor, Viswanath Nagarajan, for his unwavering support, guidance, and mentorship throughout my PhD. He always made himself available to discuss my ideas, answer my questions, and provide me with constructive feedback. His commitment to my success has been truly inspiring, and I am incredibly fortunate to have received the opportunity to work with him.

I am thankful to my collaborators Anupam Gupta and Arpit Agarwal for their time, expertise, and resources that led to successful research projects. I learnt a lot while working with Anupam: how to approach a hard problem, thinking about the right extensions, and lastly how to write/present research ideas well. Arpit introduced me to research in learning theory; specifically in dueling bandits. I am particularly thankful to Arpit for always being available (even on weekends) to discuss research ideas.

I also want to thank my thesis committee members, Nikhil Bansal, Xiuli Chao, Jon Lee, and Cong Shi. I am deeply grateful for the invaluable discussions I had with them at various stages of the PhD journey, which covered a wide range of topics related to research, the academic life, and beyond.

I am extremely grateful for my PhD cohort, which made my first two years at Ann Arbor feel like home: from Luke DeRoos' baking and Julia Coxen's stash of snacks to finding amazing friends in Daniel Otero-León and Kamolnat Tabattanon. I am especially grateful to Xubo Yue and Seokhyun Chung who, like me, were night owls; our frequent breaks and chats not only provided much-needed relief from the stress of research but also helped us form a lasting bond that I will deeply cherish.

I would also like to thank my friends for supporting me: Arpit Agarwal, Prathmesh Deshmukh, Eddie Kong, Prathamesh Patil, Dushyant Sahoo, Rahul Iyer, Tilak Vaidya, Yash Deshpande, Ria Pawar, and Tryambak Tope. Their kind words, encouragement, and willingness to listen have made all the difference.

Finally, I would like to thank my parents, brother, grandmother, and late grandfather for their unwavering support and encouragement throughout my PhD. I would also like to express my deepest gratitude to my partner, Yihan Li, for always being by my side. She has made numerous sacrifices, putting her own needs and aspirations on hold to support me in pursuing mine, and I will be forever grateful for her selflessness and dedication.

Contents

List of Tables

Abstract

In this thesis, we study the role of adaptivity in decision-making problems under uncertainty. The first part of the thesis focuses on combinatorial problems, while the second part of the thesis deals with the K-armed dueling bandits problem. Combinatorial optimization captures many natural decision-making problems such as matching, load balancing, assortment optimization, network design, and submodular optimization. In many practical settings, we have to solve such combinatorial problems under uncertainty; specifically when we only have partial knowledge about the input. Solutions to such problems are sequential decision processes that make decisions one by one "adaptively" (depending on prior observations). While such adaptive solutions achieve the best objective, the inherently sequential nature makes them undesirable in many applications. Specifically, we ask: *how well can solutions with only a few adaptive rounds approximate fully-adaptive solutions?*

- We study (and answer) the above question for the stochastic submodular cover and scenario submodular cover problems. These models capture many problems such as sensor placement with unreliable sensors, optimal decision tree, stochastic set cover, and correlated knapsack cover. We show how to obtain solutions that approximate fully-adaptive solutions using only a few "rounds" of adaptivity.

- We also study the stochastic score classification problem. We provide the first constant-factor approximation algorithm for this problem, which improves over the previously-known logarithmic approximation ratio. Moreover, our algorithm is non adaptive: it just involves performing tests in a fixed order until the class is identified.

- For both problems, we present experimental results demonstrating the practical efficacy of our algorithms. We find that a few rounds of adaptivity suffice to obtain high-quality solutions in practice.

In the second part of the thesis, we study the K-armed dueling bandits problem, which has applications in a wide-variety of domains like search ranking, recommendation systems and sports ranking where eliciting qualitative feedback is easy while real-valued feedback is not easily interpretable; thus, it has been a popular topic of research in the machine learning community. Previous works have only focused on the sequential setting where the policy adapts after every comparison. However, in many applications such as search ranking and recommendation systems, it is preferable to perform comparisons in a limited number of parallel batches. We introduce and study the batched dueling bandits problem under two standard settings: (i) existence of a Condorcet winner, and (ii) strong stochastic transitivity and stochastic triangle inequality. For both settings, we obtain algorithms with a smooth trade-off between the number of batches and regret. We complement our regret analysis with a nearly-matching lower bound. Finally, in experiments over a variety of real-world datasets, we observe that our algorithm using only a logarithmic number of rounds achieves almost the same performance as fully sequential algorithms (that use T rounds).

Chapter 1

Overview

1.1 Motivation

Many important problems involve decision making under uncertainty; for example, we may have to make decisions when the input is not fully-specified, or when our algorithms receive noisy observations. Typically, solutions for such problems involve sequential decision making: where all previous decisions and *feedback* are used to make the next decision. This makes the solutions inherently *adaptive* which may be undesirable in applications involving long wait-times for feedback (for example, waiting for test results in medical diagnosis). A central theme studied in this thesis is the following: *how well do solutions with only a few adaptive rounds compare to fully-adaptive solutions for a given problem?* Next, we motivate this question using a few applications.

A.1. Medical Diagnosis. Consider an application arising in medical diagnosis, where we know the possible conditions that a patient may suffer from, along with the priors on their occurrence, but not the actual underlying condition (see, for e.g., [49]). Under this uncertainty, our goal is to perform tests to identify the correct condition at minimum expected cost. This application involves uncertain data: the precise outcome of a medical test is not known until it is performed. Furthermore, performing such tests may involve long wait-times, making adaptive solutions undesirable. Therefore, solutions with only *few rounds of adaptivity* are preferable.

A.2. Sensor Placement. In a sensor placement application, we need to place a collection of sensors to monitor some phenomenon, such as air quality or traffic behavior (see [106]). The coverage area of each sensor is uncertain, and may depend on obstructions, the local geography, or even unexpected failures. The goal is to deploy the fewest sensors to cover some target region. The uncertainty in this application arises from the fact that the precise area covered by a sensor may not be known until the sensor is deployed. Additionally, physically deploying a sensor may take hours or days which, again, makes adaptive solutions undesirable. As in the previous application, solutions with a few adaptive rounds are preferable.

A.3. Diagnosing Complex Systems. Another important application of optimization under uncertainty is in diagnosing complex systems. A complex system comprises several components, and diagnosing it involves running a large number of tests for each component. The goal of testing is to determine what sort of intervention needs to be undertaken; for example, does simple maintenance suffice or should the entire system be replaced? Concretely, we consider a setting where the goal is to test various components, in order to assign a *risk class* (e.g., low/medium/high) to the system. One option to diagnose such systems is to perform tests on *all* components, which can be prohibitively expensive. Thus, we are interested in a policy that tests components one by one, and minimizes the average cost of testing. (See [107] for a survey.) Our focus is in designing *non-adaptive* solutions, which are simply described by a priority list: the components are tested in this fixed order until the class can be uniquely determined. Such solutions are simpler and faster to implement, compared to their adaptive counterparts. However, they are *weaker* than adaptive solutions, and our main research question is: *can we bound the multiplicative ratio (or, adaptivity gap) between the performance of a non-adaptive solution to that of the optimal adaptive one?*

A.4. Web-Search Ranking. A different stochastic optimization problem arises in web-search ranking where the goal is to provide a *ranked* list of documents to the user of the system in response to a query. Modern day search engines comprise hundreds of parameters which are used to output a ranked list in response to a query. However, manually tuning these parameters is infeasible, and online learning frameworks (based on user feedback) have been invaluable in automatically tuning these parameters. For example, given two rankings ℓ_1 and ℓ_2, they can be interleaved and presented to the user in such a way that clicks indicate which of the two rankings is more preferable to the user [96]. The availability of such pairwise comparison data led to the introduction of the dueling bandits problem [112], a variation of the traditional stochastic bandit problem in which feedback is obtained in the form of pairwise preferences. Previous learning algorithms for the dueling bandits problem have focused on a *fully adaptive* setting; in the web-ranking application this corresponds to the learning algorithm updating its parameters after each query. Such updates might be impractical in large systems due to limited computational resources or when a large amount of feedback is obtained simultaneously. Here we ask: *can we obtain low-regret learning algorithms that make a limited number of policy updates?*

1.2 Thesis Goal

In this thesis our goal is to, in a principled manner, quantify the power of adaptivity in problems involving uncertainty. The notion of adaptivity may change depending on the problem: for example, in the sensor placement problem, adapting to decisions involves waiting for feedback while in the web-search ranking problem, it translates to selecting an adaptive batch of queries to be executed. However, in either case, being adaptive leads to "better" solutions.

Concretely, this thesis makes an attempt to answer the following question for some specific problems: *can we give provable guarantees on the performance of solutions which use only a few adaptive rounds when compared to fully-adaptive solutions?*

Although we are able to understand the power of adaptivity for some problems, much more remains open. I believe that the techniques developed in this thesis will find use in proving guarantees for solutions with low adaptivity for other optimization problems under uncertainty.

1.3 Thesis Contributions

We conclude this chapter by giving an overview of our contributions.

Scenario Submodular Cover (`ScnSC`). This problem was introduced in [58] as a generalization of several problems including optimal decision tree, which models the medical diagnosis application (A.1). *In [51], we give the first r-round algorithm for* `ScnSC` that achieves an approximation guarantee with a smooth trade-off with r, the number of rounds of adaptivity available. Furthermore, the guarantee matches the one obtained in the fully adaptive setting using only a logarithmic number of rounds. We also provide a nearly matching lower bound on the approximation guarantee for any r-round algorithm (see Chapter 3).

Stochastic Submodular Cover (`SSC`). `SSC` formalizes the sensor deployment application (A.2). In [51], we study `SSC` under the limited adaptivity framework: *we designed an r-round algorithm with a nearly tight approximation guarantee that obtains a smooth trade-off with r.* We showed that a logarithmic number of adaptive rounds suffice to obtain solutions as good as fully adaptive algorithms. `SSC` is a generalization of the *stochastic set cover* problem studied by [54]: our result resolved (up to one logarithmic factor) the adaptivity gap for the stochastic set cover problem, an open question posed by [54]. See Chapter 2 for details.

Stochastic Score Classification (`SSClass`). This problem models the problem of diagnosing complex systems (A.3). This problem was introduced by [53], who showed that it can be formulated as an instance of stochastic submodular cover, and also obtained constant factor non-adaptive approximation algorithms for some special cases of the problem. A main open question from this work was the possibility of a constant factor approximation (even under full adaptivity) for the general `SSClass` problem. *We answer this question in the affirmative [52]* (Chapter 4). Moreover, *our algorithm is non-adaptive: so we also bound the adaptivity gap.*

Dueling Bandits (`DB`). The `DB` problem has been widely studied in machine learning due to its utility in applications like search ranking (A.4) (see [22] for a recent survey). We introduce the *batched dueling bandits* (`BatchedDB`) problem *where the learning algorithm can make a limited number of updates* [4]. The details regarding this appear in Chapter 5. We study `BatchedDB` under the strong stochastic transitivity (SST) and strong triangle inequality (STI) assumptions (these imply a linear order across the bandits), and obtain a *smooth trade-off between the expected regret and the number of batches.* Furthermore, we obtained regret bounds that *nearly match* those in the sequential setting using only a logarithmic number of batches. We also provided a nearly matching lower bound on the expected regret (see Chapter 6). In recent work [3], we study the `BatchedDB` under the more general *Condorcet setting*, which only assumes the existence of a "best" bandit. Again, we designed an algorithm which achieves a smooth trade-off between the expected regret and the number of batches. Surprisingly, in a logarithmic number of rounds, our regret bounds nearly match (up to logarithmic factors) the best regret bounds known in the fully sequential setting, showing that a *logarithmic number of rounds are sufficient to achieve asymptotically optimal regret* for the `DB` under the Condorcet setting (refer to Chapter 7 for details).

Chapter 2

Stochastic Submodular Cover

2.1 Introduction

Submodularity is a fundamental notion that arises in applications such as image segmentation, data summarization [102, 83, 103], hypothesis identification [17, 32], information gathering [95], and social networks [75] . The *submodular cover* optimization problem requires us to pick a minimum-cost subset S of items to cover a monotone submodular function f. Submodular cover has been extensively studied in machine learning, computer science and operations research [110, 55, 88, 18]: here is an example from sensor deployment.

In the sensor deployment setting, we consider the problem of deploying a collection of sensors to monitor some phenomenon [80, 87, 106], for example, we may wish to monitor air quality or traffic situations. The area each sensor can cover depends on its sensing range. The goal of the problem is to deploy as few sensors as possible to cover a desired region entirely. The area covered as a function of the sensors deployed is a submodular function, and we can cast the sensor deployment problem as a special case of submodular cover. Observe that this application involves uncertain data: the precise area covered by a sensor is not known before the sensor is actually setup. This uncertainty can be modeled using *stochastic submodular optimization*, where the items are stochastic. As a simple example of the stochastic nature, each item may be active or inactive (with known probabilities), and only active items contribute to the submodular function.

A solution for stochastic submodular cover is now a *sequential decision process*. At each step, an item is *probed* and its realization (e.g., active or inactive) is observed. The process is typically *adaptive*, where all the information from previously probed items is used to identify the next item to probe. This process continues until the submodular function is covered, and the goal is to minimize the expected cost of probed items. Such adaptive solutions are inherently fully sequential, which is undesirable if probing an item is time-consuming. E.g., in sensor deployment, probing a sensor corresponds to physically deploying a sensor and observing whether it functions as expected, which may take hours or days. Therefore, we prefer solutions with only few *rounds* of adaptivity.

Motivated by this, we ask: *how well do solutions with only a few adaptive rounds approximate fully-adaptive solutions for the stochastic submodular cover problem?* In this chapter, we consider the case where realizations of different items are independent. We give nearly tight answers, with smooth tradeoffs between the number r of adaptive rounds and the solution quality (relative to fully adaptive solutions).

Our main contribution is an r-round adaptive solution for stochastic submodular cover in the "set-based" model for adaptive rounds. In this model, a fixed subset of items is probed in parallel every round (and their total cost is incurred). The decisions in the current round can depend on the realizations seen in all previous rounds. However, as noted in [2], if we require function f to be covered with probability one then the r-round-adaptivity gap turns out to be very large. (See §2.8 for an example.)

Therefore, we focus on set-based solutions that are only required to cover the function with high probability.

In designing algorithms, it turns out to be more convenient to work with the "permutation" model for adaptive rounds, where the function is covered with probability one. This model was also used in prior literature [54, 2]. Here, every round of an r-round-adaptive solution specifies an ordering of all remaining items and probes them in this order until some stopping rule. See Definition 2.3.2 for a formal definition. Moreover, our r-round adaptive algorithm in the permutation model can be transformed into an r-round adaptive algorithm in the set-based model. We obtain algorithms in the set-based model that:

- for any $\eta \in (0, 1)$, finds an r-round adaptive solution that has expected cost at most $\frac{r\alpha}{\eta} \cdot \mathtt{OPT}$ and covers the function with probability at least $1 - \eta$.

- finds an $O(r)$-round adaptive solution that has expected cost at most $O(\alpha) \cdot \texttt{OPT}$ and covers the function with probability at least $1 - e^{-\Omega(r)}$.

Here $\texttt{OPT}$ is the cost of an optimal fully-adaptive solution and α is the approximation ratio of our algorithm in the permutation model. The first algorithm above is for the case where r, the number of rounds of adaptivity, is small (say, a constant). In this, we keep the number of rounds the same, but we lose a factor r in the expected cost. The second algorithm is for the case that r is large, e.g., more than a constant. Here, the number of set-based rounds increases by a factor 2, but we only lose a constant factor in expected cost. We formalize and prove these results in §2.8. For the rest of the paper, an r-round adaptive algorithm refers to an an r-round adaptive algorithm in the permutation model (unless specified otherwise).

2.1.1 Results and Techniques

Consider a monotone submodular function $f : 2^U \to \mathbb{Z}_{\geq 0}$ with $Q := f(U)$. There are m items, where each item i is a random variable $\mathcal{X}_i$ having cost c_i and corresponding to a random element of U. (Our results extend to the more general setting where each item realizes to a subset of U.) The goal is to select a set of items such that the union S of their corresponding realizations satisfies $f(S) = Q$, and the expected cost is minimized. We obtain the following result when items have *independent distributions.*

Theorem 2.1.1 (Independent Items). *For any integer $r \geq 1$, there is an r-round adaptive algorithm for the stochastic submodular cover problem with cost $O(Q^{1/r} \cdot \log Q)$ times the cost of an optimal adaptive algorithm.*

This improves over an $O(r\, Q^{1/r} \log Q \log(mc_{max}))$ bound from [2] by eliminating the dependence on the number of items m and the item costs (which could be much larger than Q). Moreover, our result nearly matches the lower bound of $\Omega(\frac{1}{r^3} Q^{1/r})$ from [2]. Setting $r = \log Q$ shows that $O(\log Q)$ adaptive rounds give an $O(\log Q)$-approximation. By transforming this algorithm into a set-based solution using Theorem 2.8.2, we get:

Corollary 2.1.2 (Independent Items: Set-Based Model). *There is an $O(\log Q)$ round algorithm for stochastic submodular cover in the set-based model that (i) has expected cost*

$O(\log Q)$ times the optimal (fully) adaptive cost, and (ii) covers the function with probability at least $1 - \frac{1}{Q}$.

This approximation ratio of $O(\log Q)$ is the best possible (unless P=NP) even with an arbitrary number ($r = m$) of adaptive rounds. Previously, one could only obtain an $O(\log^2 Q \log(mc_{max}))$-approximation in a logarithmic number of rounds [2], or an $O(\log(mQc_{\max}))$-approximation in $O(\log m \log(Qmc_{\max}))$ set-based rounds [44].

Moreover, Theorem 2.1.1 (with $r = 1$) implies an $O(Q \log Q)$ adaptivity gap for stochastic set cover (a special case of submodular cover), resolving an open question from [54] up to an $O(\log Q)$ factor, where Q is the number of objects in set cover.

Finally, we note that our algorithm is also easy to implement. We tested it on synthetic and real datasets that validate the practical performance of our algorithms. Specifically, we test our algorithm for the independent case (Theorem 2.1.1) on instances of stochastic set cover. For stochastic set cover, we use real-world datasets to generate instances with ≈ 1200 items. We observe a sharp improvement in performance within a few rounds of adaptivity, and that 6-7 rounds of adaptivity are nearly as good as fully adaptive solutions. We also compared our algorithm's cost to a information-theoretic lower bound: our costs are typically within 50% of these lower bounds.

Techniques. In each round the algorithm, we iteratively compute a "score" for each item and greedily select the item of maximum score. This results in a non-adaptive list of all remaining items, and the items are *probed* in this order until a stopping rule. The ParCA stopping rule corresponds to reducing the remaining target (on the function value) by a factor of $Q^{1/r}$.

The analysis for Theorem 2.1.1 is as follows. For each $i \geq 0$, we relate the "non-completion" probabilities of the algorithm after cost $\alpha \cdot 2^i$ to the optimal adaptive solution after cost 2^i. The "stretch" factor α corresponds to the approximation ratio. In order to relate these non-completion probabilities, we consider the total score G of items selected by the algorithm between cost $\alpha 2^{i-1}$ and $\alpha 2^i$. The crux of the analysis lies in giving lower and upper bounds on the total score G.

For the independent case, the score of any item $\mathcal{X}_e$ is an estimate of its relative marginal gain, where we take an expectation over all previous items as well as $\mathcal{X}_e$. We also normalize this gain by the item's cost. See Equation (2.1) for the definition. In lower bounding the total score G, we use a variant of a sampling lemma from [2] as well as the constant-factor adaptivity gap for submodular maximization [26]. We also need to partition the outcome space (of all previous realizations) into "good" and bad outcomes: conditional on a good outcome, $\texttt{OPT}$ has a high probability of completing before cost 2^i. Good outcomes are necessary in our proof of the sampling lemma, but luckily the total probability of bad outcomes is small (and they can be effectively ignored). In upper bounding G, we consider the total score as a sum over decision/sample paths and use the fact that the sum of relative gains corresponds to a harmonic series.

2.2 Related Work

A $(1 + \ln Q)$-approximation algorithm for the basic submodular cover problem was obtained in [110]. This is also the best possible (unless P=NP) as the set cover problem is a special case [45]. In the past several years, there have been many papers on *stochastic* variants of submodular cover as this framework captures many different applications; see [55, 67, 40, 58, 92]

The stochastic set cover problem was first studied by [54], who showed that the adaptivity gap lies between $\Omega(d)$ and $O(d^2)$ where d is the number of objects to be covered. Recently, [2] improved the upper bound to $O(d \log d \log(mc_{max}))$. As a corollary of Theorem 2.1.1, we obtain a tighter $O(d \log d)$ adaptivity gap. Importantly, we eliminate the dependence of the items m and maximum-cost, which may even be exponential in d.

An $O(\log Q)$ adaptive approximation algorithm for stochastic submodular cover (independent items) follows from [67]. Related results for special cases or with weaker bounds were obtained in [85, 55, 56, 40]. As mentioned earlier, [2] obtained r-round-adaptivity gaps for independent items. Theorem 2.1.1 improves their bound by an $O(r \cdot \log(mc_{max}))$ factor. Moreover, our algorithm is greedy-style and much simpler to implement. We bypass computationally expensive steps in prior work such as solving several instances of stochastic submodular maximization. Our analysis (outlined above) is also very different. While we use a sampling lemma similar to [2], this result is applied in different manner and it only affects the analysis (see Lemma 2.5.2). We note that our high-level approach of lower/upper bounding the total score is similar to the analysis in [67] for the fully adaptive problem. However, the details are very different because we need to handle issues of r-round-adaptivity gaps.

The framework of "adaptive submodularity", introduced by [56], models correlations in stochastic submodular cover. Adaptive submodularity is a combined condition on the goal function f (that needs to be covered) and the distribution $\mathcal{D}$ on items. While stochastic submodular cover with independent items satisfies adaptive-submodularity, scenario submodular cover does not. [56] gave a fully-adaptive $O(\log^2(mQ))$-approximation algorithm for adaptive-submodular cover ($\mathsf{AdSubCov}$) when all costs are unit. Recently, [44] improved this to an $O(\log(mQ))$-approximation in $O(\log m \log(Qm))$ rounds of adaptivity, still assuming unit costs. When costs are arbitrary, the result in [44] implies an $O(\log(mQc_{\max}))$-approximation in $O(\log m \log(Qmc_{\max}))$ rounds: this result also applies to stochastic submodular cover and can be compared to Corollary 2.1.2 that we obtain.

The role of adaptivity has been extensively studied for stochastic submodular *maximization*. A constant adaptivity gap under matroid constraints (on the probed items) was obtained in [8]. Later, [64] obtained a constant adaptivity gap for a very large class of "prefix closed" constraints; the constant factor was subsequently improved to 2 which is also the best possible [26]. We make use of this result in our analysis. More generally, the role of adaptivity has been extensively studied for various stochastic "maximization" problems such as knapsack [38, 23], matching [15, 19], probing [62] and orienteering [59, 61, 16].

[74, 43] and [47] study the role of adaptivity and batch processing in the online learning setting. Their work is motivated by noting that many real-world data observations are processed in batches. Recently, there have also been several results examining the role of adaptivity in (deterministic) submodular optimization(see, for example, [13, 11, 14, 12] and [30]). The motivation here was in parallelizing function queries (that are often expensive). In many settings, there are algorithms using (poly)logarithmic number of rounds that nearly match the best sequential (or fully adaptive) approximation algorithms. While our motivation is similar (in parallelizing the expensive probing steps), the techniques used are completely different.

2.3 Definitions

In the stochastic submodular cover problem, the input is a collection of m random variables (or *items*) $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_m\}$. Each item $\mathcal{X}_i$ has a cost $c_i \in \mathbb{R}_+$, and realizes to a random element of groundset U. Let the joint distribution of $\mathcal{X}$ be denoted by $\mathcal{D}$. Here, we assume that the random variables $\mathcal{X}_i$ are independent. The realization of item $\mathcal{X}_i$ is denoted by $X_i \in U$; this realization is only known when $\mathcal{X}_i$ is *probed* at a cost of c_i. Extending this notation, given a subset of items $\mathcal{S} \subseteq \mathcal{X}$, its realization is denoted $S = \{X_i : \mathcal{X}_i \in \mathcal{S}\} \subseteq U$.

In addition, we are given an integer-valued monotone submodular function $f : 2^U \to \mathbb{Z}_+$ with $f(U) = Q$. A realization S of items $\mathcal{S} \subseteq \mathcal{X}$ is *feasible* if and only if $f(S) = Q$ the maximal value; in this case, we also say that $\mathcal{S}$ *covers* f. The goal is to probe (possibly adaptively) a subset $\mathcal{S} \subseteq \mathcal{X}$ of items that gets realized to a feasible set. We use the shorthand $c(\mathcal{S}) := \sum_{i:\mathcal{X}_i \in \mathcal{S}} c_i$ to denote the total cost of items in $\mathcal{S} \subseteq \mathcal{X}$. The objective is to minimize the expected cost of probed items, where the expectation is taken over the randomness in $\mathcal{X}$. We consider the following types of solutions.

Definition 2.3.1. *For an integer $r \geq 1$, an r-**round-adaptive** solution in the **set-based model** proceeds as follows. For each round $k = 1, \ldots, r$, it specifies a subset $\mathcal{S}_k$ of items that is probed in parallel. The cost incurred in round k is $c(\mathcal{S}_k)$ the total cost of all probed items in that round. The subset $\mathcal{S}_k$ in round k can depend on realizations seen in all previous rounds $1, \ldots, k-1$.*

In the set-based model, we allow solutions to be infeasible (i.e., fail to cover f) with some small probability $\eta > 0$. As shown in Appendix 2.8, such a relaxed solution is necessary. In designing algorithms, we will work with the "permutation" model, as defined next.

Definition 2.3.2. *For an integer $r \geq 1$, an r-**round-adaptive** solution in the **permutation model** proceeds in r rounds of adaptivity. In each round $k \in \{1, \ldots, r\}$, the solution specifies an ordering of all remaining items and probes them in this order until some stopping rule. The decisions in round k can depend on the realizations seen in rounds $1, \ldots, k-1$.*

In the permutation model, solutions must be feasible with probability one. As shown in §2.8, our algorithms in the permutation model can be converted into algorithms in the set-based (with similar approximation ratios). Henceforth, an r-round adaptive algorithm refers to an an r-round adaptive algorithm in the permutation model (unless specified otherwise).

Setting $r = 1$ in Definition 2.3.2 gives us a *non-adaptive* solution as studied in [54, 2]. Setting $r = m$ gives us a *(fully) adaptive* solution. Having more rounds leads to a smaller objective value, so adaptive solutions have the least objective value. Our performance guarantees are relative to an optimal fully adaptive solution; let `OPT` denote this solution and its cost. The *r-round-adaptivity gap* is defined as follows:

$$\sup_{\text{instance } I} \frac{\mathbb{E}[\text{cost of best } r\text{-round adaptive solution on } I]}{\mathbb{E}[\text{cost of best adaptive solution on } I]}$$

Setting $r = 1$ gives the *adaptivity gap.*

2.4 Stochastic Submodular Cover

We now consider the (independent) stochastic submodular cover problem and prove Theorem 2.1.1. For simplicity, we assume that costs c_i are integers. Our results also hold for arbitrary costs (by replacing certain summations in the analysis by integrals).

We find it convenient to solve a *partial cover* version of the stochastic submodular cover problem. In this partial cover version, we are given a parameter $\delta \in [0, 1]$ and the goal is to probe some $\mathcal{R}$ that realizes to a set R achieving value $f(R) > Q(1-\delta)$. We are interested in a *non adaptive* algorithm for this problem. Clearly, if $\delta = 1/Q$, the integrality of the function f means that $f(R) = Q$, and we solve the original (full coverage) problem. Moreover, we can use this algorithm with different thresholds to also get the r-round version of the problem. The main result of this section is:

Theorem 2.4.1. *There is a non-adaptive algorithm for the partial cover version of stochastic submodular cover with cost $O\left(\frac{\ln 1/\delta}{\delta}\right)$ times the optimal adaptive cost for the (full) submodular cover.*

The algorithm first creates an ordering/list L of the items non-adaptively; that is, without knowing the realizations of the items. To do so, at each step we pick a new item that maximizes a carefully-defined score function (Equation (2.1)). The score of an item cannot depend on the realizations of previous items on the list (since we are non-adaptive). However, it depends on the *marginal relative increase* for a random draw from the previously chosen items. Once this ordering L is specified, the algorithm starts probing and realizing the items in this order, and does so until the realized value exceeds $(1-\delta)Q$.

Algorithm 1 PARtial Covering Algorithm $\textsc{ParCA}(\mathcal{X}, f, Q, \delta)$

```
1: $\mathcal{S} \leftarrow \emptyset$, list $L \leftarrow \langle\rangle$, $\tau \leftarrow Q(1-\delta)$
2: while $\mathcal{S} \neq \mathcal{X}$ do                                ▷ Building the list non-adaptively
3:     select an item $\mathcal{X}_e \in \mathcal{X} \setminus \mathcal{S}$ that maximizes:
           $$\text{score}(\mathcal{X}_e) := \frac{1}{c_e} \cdot \sum_{S \sim \mathcal{S}: f(S) \leq \tau} \mathbf{P}(\mathcal{S} = S) \cdot \mathbb{E}_{X_e \sim \mathcal{X}_e}\left[\frac{f(S \cup X_e) - f(S)}{Q - f(S)}\right] \qquad (2.1)$$
4:     $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathcal{X}_e\}$ and list $L \leftarrow L \circ \mathcal{X}_e$
5: $\mathcal{R} \leftarrow \emptyset$, $R \leftarrow \emptyset$
6: while $f(R) \leq \tau$ do                                          ▷ Probing items on the list
7:     $\mathcal{X}_e \leftarrow$ first r.v. in list $L$ not in $\mathcal{R}$, and let $X_e \in U$ be its realization.
8:     $R \leftarrow R \cup \{X_e\}, \mathcal{R} \leftarrow \mathcal{R} \cup \{\mathcal{X}_e\}$
9: return probed items $\mathcal{R}$ and their realizations $R$.
```

Given this partial covering algorithm we immediately get an algorithm for the r-round version of the problem, where we are allowed to make r rounds of adaptive decisions. Indeed, we can first set $\delta = Q^{-1/r}$ and solve the partial covering problem with this value of δ. Suppose we probe variables $\mathcal{R}$ and their realizations are given by the set $R \subseteq U$. Then we can condition on these values to get the marginal value function f_R (which is submodular), and inductively get an $r-1$-round solution for this problem. The following algorithm formalizes this.

Algorithm 2 r-round adaptive algorithm for stochastic submodular cover $\text{SSC}(r, \mathcal{X}, f)$

```
1: run $\textsc{ParCA}$ $(\mathcal{X}, f, Q, Q^{-1/r})$ for round #1.
2: let $\mathcal{R}$ (resp. $R$) denote the probed items (resp. their realizations) in $\textsc{ParCA}$.
3: define residual submodular function $\widehat{f} := f_R$.
4: recursively solve $\text{SSC}(r-1, \mathcal{X} \setminus \mathcal{R}, \widehat{f})$.
```

Theorem 2.4.2. *Algorithm 2 is an r-round adaptive algorithm for stochastic submodular cover with cost $O(Q^{1/r} \log Q)$ times the optimal adaptive cost.*

Proof. We proceed by induction on the number of rounds r. Let $\texttt{OPT}$ denote the cost of an optimal adaptive solution. The base case is $r = 1$, in which case $\delta = Q^{-1/r} = \frac{1}{Q}$. By Theorem 2.4.1, the partial cover algorithm $\textsc{ParCA}(\mathcal{X}, f, Q, Q^{-1/r})$ obtains a realization R with $f(R) > (1-\delta)Q = Q - 1$. As f is integer-valued, we must have $f(R) = Q$, which means the function is fully covered. So the algorithm's expected cost is $O(Q \log Q) \cdot \texttt{OPT}$.

We now consider $r > 1$ and assume (inductively) that Algorithm 2 finds an $r-1$-round $O(Q^{\frac{1}{r-1}} \log Q)$-approximation algorithm for any instance of stochastic submodular cover. Let $\delta = Q^{-1/r}$. By Theorem 2.4.1, the expected cost in round 1 (step 1 in Algorithm 2) is $O(\frac{1}{r} Q^{1/r} \log Q)$. Let $\widehat{Q} := Q - f(R) = \widehat{f}(U)$ denote the maximal value of the residual submodular function $\widehat{f} = f_R$. Note that $\widehat{Q} < \delta Q = Q^{(r-1)/r}$, by the definition of the partial covering algorithm. The optimal solution $\texttt{OPT}$ conditioned on the variables in $\mathcal{R}$ realizing to R gives a feasible adaptive solution to the residual problem of covering $\widehat{f}$; we denote this conditional solution by $\widehat{\texttt{OPT}}$. We inductively get that the cost of our $r-1$-round solution on $\widehat{f}$ is at most

$$O(\widehat{Q}^{\frac{1}{r-1}} \log \widehat{Q}) \cdot \widehat{\texttt{OPT}} \le O\left(\frac{r-1}{r} Q^{1/r} \log Q\right) \cdot \widehat{\texttt{OPT}},$$

where we used $\widehat{Q} < Q^{(r-1)/r}$. As this holds for every realization R, we can take expectations over R to get that the (unconditional) expected cost of the last $r-1$ rounds is $O\big(\frac{r-1}{r} Q^{1/r} \log Q\big) \cdot \texttt{OPT}$. Adding to this the cost of the first round, which is $O\big(\frac{1}{r} Q^{1/r} \log Q\big)$, completes the proof. $\square$

Remark: Assuming that the scores (2.1) can be computed in polynomial time, it is clear that our entire algorithm can be implemented in polynomial time. In particular, if T denotes the time taken to calculate the score of one item, then the overall algorithm runs in time $poly(m, T)$ where m is the number of items. We are not aware of a closed-form expression for the scores (for arbitrary submodular functions f). However, as discussed in the full version, we can use sampling to estimate these scores to within a constant factor. However, as discussed in §2.10, we can use sampling to estimate these scores to within a constant factor. Moreover, our analysis works even if we only choose an approximate maximizer for (2.1) at each step. It turns out that $T = poly(m, c_{max})$ many samples suffice to estimate these scores. So, the final runtime is $poly(m, c_{max})$; note that this does not depend on the number $|U|$ of elements. We note that even the previous algorithms [2, 44] have a polynomial dependence on c_{max} in their runtime. In the following analysis we will assume that the scores (2.1) are computed exactly. In the following analysis we will assume that the scores (2.1) are computed exactly; see §2.10 for the sampling details.

2.4.1 Analysis for a Call to ParCA

We now prove Theorem 2.4.1. We denote by `OPT` an optimal adaptive solution for the covering problem on f. Now we analyze the cost incurred by following the non-adaptive strategy (which we call `NA`): probe variables according to the order given by the list L generated by ParCA, and stop when the realized coverage exceeds $\tau := Q(1-\delta)$ (see Algorithm 1 for details). We consider the expected cost of this strategy, and relate it to the cost of `OPT`.

We refer to the cumulative cost incurred (either by `OPT` or by `NA`) until any point in a solution as *time* elapsing. We say that `OPT` is in phase i when it is in the time interval $[2^i, 2^{i+1})$ for $i \geq 0$. Let $\alpha := O\left(\frac{\ln 1/\delta}{\delta}\right)$. We say that `NA` is in *phase* i when it is in time interval $[\alpha \cdot 2^{i-1}, \alpha \cdot 2^i)$ for $i \geq 1$; phase 0 refers to the interval $[1, \alpha)$. Define

- u_i: probability that `NA` goes beyond phase i; that is, has cost at least $\alpha \cdot 2^i$.
- u_i^*: probability that `OPT` goes beyond phase $i-1$; that is, costs at least 2^i.

Since all costs are integers, $u_0^* = 1$. For ease of notation, we also use `OPT` and `NA` to denote the *random* cost incurred by `OPT` and `NA` respectively. The following lemma forms the crux of the analysis.

Lemma 2.4.3. *For any phase $i \geq 1$, we have $u_i \leq \frac{u_{i-1}}{4} + \frac{6}{5}u_i^*$.*

Before we prove this lemma, we use it to prove Theorem 2.4.1.

Proof of Theorem 2.4.1. With probability $(u_{i-1} - u_i)$, `NA` ends in phase i with cost at most $\alpha \cdot 2^i$. As a consequence of this, we have

$$\mathbb{E}[\texttt{NA}] \;\leq\; \alpha \cdot (1 - u_0) + \sum_{i \geq 1} \alpha \cdot 2^i \cdot (u_{i-1} - u_i) = \; \alpha + \alpha \cdot \sum_{i \geq 0} 2^i u_i. \tag{2.2}$$

Similarly, we can bound the cost of the optimal adaptive algorithm as

$$\mathbb{E}[\texttt{OPT}] \geq \sum_{i\geq 0} 2^i(u_i^* - u_{i+1}^*) \geq u_0^* + \frac{1}{2}\cdot\sum_{i\geq 1} 2^i u_i^* = 1 + \frac{1}{2}\cdot\sum_{i\geq 1} 2^i u_i^*, \tag{2.3}$$

where the final equality uses the fact that $u_0^* = 1$. Define $\Gamma := \sum_{i\geq 0} 2^i u_i$. We have

$$\begin{aligned}\Gamma := \sum_{i\geq 0} 2^i u_i &\leq u_0 + \frac{1}{4}\sum_{i\geq 1} 2^i \cdot u_{i-1} + \frac{6}{5}\sum_{i\geq 1} 2^i \cdot u_i^* \\ &\leq u_0 + \frac{1}{4}\sum_{i\geq 1} 2^i u_{i-1} + \frac{12}{5}\cdot(\mathbb{E}[\texttt{OPT}] - 1) \\ &= u_0 + \frac{1}{2}\Big(\sum_{i\geq 0} 2^i u_i\Big) + \frac{12}{5}\cdot(\mathbb{E}[\texttt{OPT}] - 1) \\ &\leq \frac{1}{2}\Gamma + \frac{12}{5}\,\mathbb{E}[\texttt{OPT}] - 1,\end{aligned}$$

where the first inequality follows from Lemma 2.4.3, the second inequality from (2.3), and the last inequality from the fact that $u_0 \leq 1$. Thus, $\Gamma \leq \frac{24}{5}\cdot\mathbb{E}[\texttt{OPT}] - 2$. From (2.2), we conclude $\mathbb{E}[\texttt{NA}] \leq \frac{24}{5}\alpha\cdot\mathbb{E}[\texttt{OPT}]$. Setting $\alpha = O\Big(\frac{\ln(1/\delta)}{\delta}\Big)$ completes the proof. □

2.5 Proof of the Key Lemma (Lemma 2.4.3)

Recall the setting of Lemma 2.4.3 and fix the phase $i \geq 1$. Consider the list L generated by ParCA$(\mathcal{X}, f, Q, \delta)$. Let NA denote both the non-adaptive strategy of Algorithm 1, as well as its cost. Note that NA probes items in the order given by L until a coverage value greater than $\tau := Q(1-\delta)$ is achieved.

We will assume (without loss of generality) that δ is a power-of-two, i.e., $\delta = 2^{-z}$ for some integer $z \geq 0$. Indeed, if this is not the case, we can use a power-of-two value δ' where $\frac{\delta}{2} \leq \delta' \leq \delta$: this only increases the approximation ratio $O(\frac{1}{\delta}\ln\frac{1}{\delta})$ in Theorem 2.4.1 by a constant factor.

For each time $t \geq 0$, let $\mathcal{X}_{e(t)}$ denote the item that would be selected at time t. In other words, this is the item which causes the cumulative cost of L to exceed t for the first time.

We define the *total gain* as the random variable

$$G := \sum_{t=\alpha 2^{i-1}}^{\alpha 2^i} \text{score}(\mathcal{X}_{e(t)}), \tag{2.4}$$

which corresponds to the sum of scores over the time interval $[\alpha \cdot 2^{i-1}, \alpha \cdot 2^i)$. The proof of Lemma 2.4.3 will be completed by giving upper- and lower-bounds on G, which we do next. The lower bound views G as a sum over time steps, whereas the upper bound views G as a sum over decision paths.

2.5.1 A Lower Bound for G

Fix some time $t \in [\alpha 2^{i-1}, \alpha 2^i)$ in phase i of our algorithm. Let $\mathcal{S}$ be the random variable denoting the set of chosen items (added to list L) until some time t, and let S denote its realization. We also need the following definitions:

1. For any power-of-two $\theta \in \{\frac{1}{2^j} : 0 \le j \le \log Q\}$, we say that a realization S of $\mathcal{S}$ belongs to *scale* θ iff $\frac{\theta}{2} < \frac{Q - f(S)}{Q} \le \theta$. We use $\mathcal{E}_\theta$ to denote all outcomes of $\mathcal{S}$ that belong to scale θ. We also use $S \sim \mathcal{E}_\theta$ to denote the conditional distribution of $\mathcal{S}$ corresponding to $\mathcal{E}_\theta$.

2. For any scale θ, let $r^*_{i\theta} := \mathbf{P}(\texttt{OPT}$ covers f with cost at most 2^i **and** $S \in \mathcal{E}_\theta)$.

3. Scale θ is called *good* if $\frac{r^*_{i\theta}}{\mathbf{P}(\mathcal{E}_\theta)} \ge \frac{1}{6}$ where $\mathbf{P}(\mathcal{E}_\theta) := \mathbf{P}_{S \sim \mathcal{S}}(S \in \mathcal{E}_\theta)$.

Lemma 2.5.1. `NA` *terminates by time t if and only if the realization of $\mathcal{S}$ is in a scale $\theta \le \delta$.*

Proof. Note that if the realization of $\mathcal{S}$ is in some scale $\theta \ge 2\delta$ then

$$f(S) < Q - \frac{\theta Q}{2} \le Q - \delta Q = \tau,$$

and `NA` would not terminate before time t. On the other hand, if the realization of $\mathcal{S}$ is in a scale $\theta \le \delta$ (note that all scales are powers-of-two) then $f(S) \ge Q - \theta Q \ge \tau$, and `NA` would terminate before time t. □

We now define a function

$$g(T) := \sum_{\theta > \delta} \mathbf{P}(\mathcal{E}_\theta) \cdot \mathbb{E}_{S \sim \mathcal{E}_\theta} \left[\frac{f(S \cup T) - f(S)}{Q - f(S)} \right] = \sum_{S \sim \mathcal{S}: f(S) < \tau} \mathbf{P}(\mathcal{S} = S) \cdot \frac{f_S(T)}{Q - f(S)}, \quad \forall T \subseteq U. \tag{2.5}$$

The second equality is by Lemma 2.5.1. The function g is monotone and submodular, since f_S is monotone and submodular for each $S \subseteq U$, and g is a non-negative linear combination of such functions. Moreover, for any item $\mathcal{X}_e$, we have $\text{score}(\mathcal{X}_e) = \frac{1}{c_e} \cdot \mathbb{E}_{\mathcal{X}_e}[g(X_e)]$.

Constrained stochastic submodular maximization. Our analysis makes use of some known results for stochastic submodular *maximization*. Here, we are given as input a non-negative monotone submodular function $g : 2^U \to \mathbb{R}_{\geq 0}$, independent stochastic items $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_m\}$ such that each $\mathcal{X}_i$ realizes to some element of the groundset U. There is a cost c_i associated with each item $\mathcal{X}_i$, and a budget B on the total cost. The goal is to select $\mathcal{S} \subseteq \mathcal{X}$ such that the total cost of $\mathcal{S}$ is at most B and it maximizes the expected value $\mathbb{E}_{S \sim \mathcal{S}}[g(S)]$. The adaptivity gap for stochastic submodular maximization is at most 2; see Theorem 1 in [26].

Lemma 2.5.2. *For any good scale θ, there exists a subset $\mathcal{T}_\theta \subseteq \mathcal{X} \setminus \mathcal{S}$ with $c(\mathcal{T}_\theta) \leq \frac{144}{\theta} \cdot 2^i$ such that:*

$$\mathbb{E}_{S \sim \mathcal{E}_\theta} \mathbb{E}_{T_\theta \sim \mathcal{T}_\theta} \Big[f(S \cup T_\theta) - f(S) \Big] \geq \frac{\theta Q}{6} \cdot \frac{r^*_{i\theta}}{\mathbf{P}(\mathcal{E}_\theta)}. \tag{2.6}$$

Proof. We construct set $\mathcal{T}_\theta$ as follows. Initially, $\mathcal{T}_\theta \leftarrow \emptyset$. For each $k = 1, 2, \ldots, \frac{144}{\theta}$:

1. Sample realization S of $\mathcal{S}$ from $\mathcal{E}_\theta$.
2. Let $\mathcal{T}_k \subseteq \mathcal{X} \setminus \mathcal{S}$ be an optimal *non-adaptive* solution to the stochastic submodular maximization instance with function f_S and cost budget 2^i.
3. Set $\mathcal{T}_\theta \leftarrow \mathcal{T}_\theta \cup \mathcal{T}_k$.

By construction, $c(\mathcal{T}_\theta) \leq \frac{144}{\theta} \cdot 2^i$. So we focus on the expected function value. Consider any $S \in \mathcal{E}_\theta$ as the realization of $\mathcal{S}$. Let $\mathcal{T}_S$ denote the non-adaptive solution obtained in step 2 above for realization S. Define $w^*_{i,S}$ as the probability that `OPT` covers f with cost at most 2^i given that $\mathcal{S}$ realizes to S, i.e.,

$$w^*_{i,S} := \mathbf{P}(\ \texttt{OPT} \text{ covers } f \text{ with cost at most } 2^i \mid \mathcal{S} = S\ ).$$

Note that $\sum_{S\in\mathcal{E}_\theta} w^*_{i,S} \cdot \mathbf{P}(\mathcal{S} = S) = r^*_{i\theta}$. Let `AD` denote `OPT` until time 2^i and restricted to items $\mathcal{X} \setminus \mathcal{S}$. Note that `AD` is a feasible adaptive solution to the stochastic submodular maximization instance with function f_S: every decision path has total cost at most 2^i. The expected value of `AD` is at least $(Q - f(S)) \cdot w^*_{i,S}$ by definition of $w^*_{i,S}$. As $S \in \mathcal{E}_\theta$, we have $Q - f(S) \geq \frac{\theta Q}{2}$, which implies `AD` has value at least $\frac{\theta Q}{2} \cdot w^*_{i,S}$. Now, using the factor 2 adaptivity gap for stochastic submodular *maximization* (discussed above), it follows that the non-adaptive solution $\mathcal{T}_S$ has expected value:

$$\mathbb{E}_{T_S\sim\mathcal{T}_S}[f(S\cup T_S) - f(S)] \geq \frac{\theta Q}{4} \cdot w^*_{i,S}, \qquad \forall S \in \mathcal{E}_\theta.$$

Taking expectations,

$$\begin{aligned}\mathbb{E}_S\mathbb{E}_{T_S}[f(S\cup T_S) - f(S) \mid S \in \mathcal{E}_\theta] &\geq \sum_{S\in\mathcal{E}_\theta} \frac{\theta Q}{4} \cdot w^*_{i,S} \cdot \mathbf{P}(\mathcal{S} = S \mid S \in \mathcal{E}_\theta) \\ &= \frac{\theta Q}{4} \cdot \sum_{S\in\mathcal{E}_\theta} w^*_{i,S} \cdot \frac{\mathbf{P}(\mathcal{S} = S)}{\mathbf{P}(\mathcal{E}_\theta)} = \frac{\theta Q}{4} \cdot \frac{r^*_{i\theta}}{\mathbf{P}(\mathcal{E}_\theta)}.\end{aligned}$$

The left-hand-side of the above relation can be rewritten to give

$$\mathbb{E}_{S\sim\mathcal{E}_\theta}\mathbb{E}_{T_S}[f(S\cup T_S) - f(S)] \geq \frac{\theta Q}{4} \cdot \frac{r^*_{i\theta}}{\mathbf{P}(\mathcal{E}_\theta)}. \tag{2.7}$$

For a contradiction to (2.6), suppose that

$$\mathbb{E}_{S\sim\mathcal{E}_\theta}\mathbb{E}_{T_\theta}[f(S\cup T_\theta) - f(S)] < \frac{\theta Q}{6} \cdot \frac{r^*_{i\theta}}{\mathbf{P}(\mathcal{E}_\theta)}. \tag{2.8}$$

Subtracting Equation (2.8) from Equation (2.7) gives the following:

$$\begin{aligned}\frac{\theta Q}{12}\cdot\frac{r^*_{i\theta}}{\mathbf{P}(\mathcal{E}_\theta)} &< \mathbb{E}_{S\sim\mathcal{E}_\theta}\mathbb{E}_{T_S}\mathbb{E}_{T_\theta}[f(S\cup T_S)-f(S\cup T_\theta)]\\ &\le \mathbb{E}_{S\sim\mathcal{E}_\theta}\mathbb{E}_{T_S}\mathbb{E}_{T_\theta}[f(S\cup T_\theta\cup T_S)-f(S\cup T_\theta)]\\ &\le \mathbb{E}_{S\sim\mathcal{E}_\theta}\mathbb{E}_{T_S}\mathbb{E}_{T_\theta}[f(T_\theta\cup T_S)-f(T_\theta)] \qquad (2.9)\end{aligned}$$

where the second inequality uses monotonicity of f, and the last one its submodularity.

Let $\mathcal{T}^{(k)} = \mathcal{T}_1\cup\mathcal{T}_2\ldots\cup\mathcal{T}_k$ denote the items selected until iteration k. We write a telescoping sum as follows

$$\mathbb{E}_{T_\theta}[f(T_\theta)] = \sum_k \mathbb{E}_{T^{(k)}}\left[f(T^{(k)})-f(T^{(k-1)})\right]$$

where we define $f(T^{(0)}) := 0$. Note that in each iteration k, the sample S is drawn independently and identically from $\mathcal{E}_\theta$, and items $\mathcal{T}_k = \mathcal{T}_S$ are added.

$$\begin{aligned}\mathbb{E}_{T^{(k)}}\left[f(T^{(k)})-f(T^{(k-1)})\right] &= \mathbb{E}_{S\sim\mathcal{E}_\theta}\mathbb{E}_{T_S}\mathbb{E}_{T^{k-1}}\left[f(T^{(k-1)}\cup T_S)-f(T^{(k-1)})\right]\\ &\ge \mathbb{E}_{S\sim\mathcal{E}_\theta}\mathbb{E}_{T_S}\mathbb{E}_{T_\theta}\left[f(T_\theta\cup T_S)-f(T_\theta)\right] \;>\; \frac{\theta Q}{12}\cdot\frac{r^*_{i\theta}}{\mathbf{P}(\mathcal{E}_\theta)}.\end{aligned}$$

The first inequality above uses $\mathcal{T}^{(k-1)}\subseteq\mathcal{T}_\theta$ and submodularity, and the second inequality uses (2.9). Adding over all iterations k,

$$\mathbb{E}_{T_\theta}[f(T_\theta)] \;>\; \sum_k \frac{\theta Q}{12}\cdot\frac{r^*_{i\theta}}{\mathbf{P}(\mathcal{E}_\theta)} \;=\; \frac{144}{\theta}\cdot\frac{\theta Q}{12}\cdot\frac{r^*_{i\theta}}{\mathbf{P}(\mathcal{E}_\theta)} \;\ge\; 2Q,$$

where the last inequality uses the fact that θ is a good scale. This is a contradiction since the maximum function value is Q. This completes the proof of (2.6). $\square$

Lemma 2.5.3. *For any $\mathcal{S}\subseteq\mathcal{X}$, there exists a subset $\mathcal{T}\subseteq\mathcal{X}\setminus\mathcal{S}$ of total cost at most $\frac{144}{\delta}\cdot 2^i$ with*

$$\mathbb{E}_{T\sim\mathcal{T}}[g(T)] \quad\ge\quad \frac{1}{6}\cdot\sum_{\theta>\delta,\, good} r^*_{i\theta}.$$

Proof. Let $\mathcal{B}$ denote the set of *good* scales θ with $\theta>\delta$. Note that $\sum_{\theta\in\mathcal{B}}\frac{1}{\theta}\le\frac{1}{\delta}$ as the scales are powers of two. From Lemma 2.5.2, let $\mathcal{T}_\theta$ denote the items satisfying (2.6) for each scale $\theta \in \mathcal{B}$. Define $\mathcal{T} = \cup_{\theta\in\mathcal{B}}\mathcal{T}_\theta$. As claimed, the total cost is

$$c(\mathcal{T}) \le \sum_{\theta\in\mathcal{B}} c(\mathcal{T}_\theta) \le 2^i \sum_{\theta\in\mathcal{B}} \frac{144}{\theta} \le \frac{144}{\delta}\cdot 2^i.$$

Next, we bound the expected increase in the value of f. By (2.6) and $\mathcal{T} \supseteq \mathcal{T}_\theta$, it follows that

$$\mathbb{E}_{S\sim\mathcal{E}_\theta}\mathbb{E}_{T\sim\mathcal{T}}[f(S\cup T) - f(S)] \ge \frac{\theta Q}{6}\cdot\frac{r^*_{i\theta}}{\mathbf{P}(\mathcal{E}_\theta)}$$

for each $\theta \in \mathcal{B}$. Using the fact that $\frac{\theta Q}{2} < Q - f(S) \le \theta Q$ for all $S \in \mathcal{E}_\theta$, we get:

$$\mathbb{E}_{S\sim\mathcal{E}_\theta}\mathbb{E}_{T\sim\mathcal{T}}\left[\frac{f(S\cup T) - f(S)}{Q - f(S)}\right] \ge \frac{1}{6}\cdot\frac{r^*_{i\theta}}{\mathbf{P}(\mathcal{E}_\theta)}, \qquad \forall\theta\in\mathcal{B}.$$

By definition of function g (see (2.5)),

$$\begin{aligned}\mathbb{E}_{T\sim\mathcal{T}}[g(T)] &= \sum_{\theta>\delta}\mathbf{P}(\mathcal{E}_\theta)\cdot\mathbb{E}_{S\sim\mathcal{E}_\theta}\mathbb{E}_{T\sim\mathcal{T}}\left[\frac{f(S\cup T) - f(S)}{Q - f(S)}\right]\\ &\ge \sum_{\theta\in\mathcal{B}}\mathbf{P}(\mathcal{E}_\theta)\cdot\mathbb{E}_{S\sim\mathcal{E}_\theta}\,\mathbb{E}_{T\sim\mathcal{T}}\left[\frac{f(S\cup T) - f(S)}{Q - f(S)}\right] \ge \frac{1}{6}\cdot\sum_{\theta\in\mathcal{B}} r^*_{i\theta},\end{aligned}$$

which completes the proof of the lemma. □

Using Lemma 2.5.3 and averaging,

$$\begin{aligned}\max_{\mathcal{X}_e\in\mathcal{X}\setminus\mathcal{S}}\text{score}(\mathcal{X}_e) &\ge \max_{\mathcal{X}_e\in\mathcal{T}}\text{score}(\mathcal{X}_e) \ge \frac{\sum_{\mathcal{X}_e\in\mathcal{T}}\mathbb{E}[g(X_e)]}{c(\mathcal{T})} = \frac{\mathbb{E}_{T\sim\mathcal{T}}[\sum_{X_e\in T} g(X_e)]}{c(\mathcal{T})}\\ &\ge \frac{\mathbb{E}_{T\sim\mathcal{T}}[g(T)]}{c(\mathcal{T})} \quad\ge\quad \frac{\delta}{\beta\cdot 2^i}\cdot\sum_{\theta>\delta,\,\text{good}} r^*_{i\theta},\end{aligned} \tag{2.10}$$

where $\beta = O(1)$.

Lemma 2.5.4. *We have* $\sum_{\theta>\delta,\,good} r^*_{i\theta} \ge u_i - \frac{6u^*_i}{5}$.

Proof. First, we upper bound $\sum_{\theta\text{ not good}} r^*_{i\theta}$. Consider any scale θ that is not good. Then, $\frac{r^*_{i\theta}}{\mathbf{P}(\mathcal{E}_\theta)} < \frac{1}{6}$, i.e., $\mathbf{P}(\texttt{OPT} < 2^i \mid S\sim\mathcal{E}_\theta) < 1/6$, which implies $\mathbf{P}(\texttt{OPT} \ge 2^i \mid S\sim\mathcal{E}_\theta) > \frac{5}{6} >$

$5\frac{r^*_{i\theta}}{\mathbf{P}(\mathcal{E}_\theta)}$. So,

$$\sum_{\theta \text{ not good}} r^*_{i\theta} < \frac{1}{5} \sum_{\theta \text{ not good}} \mathbf{P}(\mathcal{E}_\theta) \cdot \mathbf{P}(\texttt{OPT} \geq 2^i \mid S \sim \mathcal{E}_\theta) \leq \frac{1}{5} \sum_\theta \mathbf{P}(\texttt{OPT} \geq 2^i \textit{ and } S \sim \mathcal{E}_\theta) \leq \frac{u^*_i}{5} \tag{2.11}$$

where the last inequality uses the fact that $\sum_\theta \mathbf{P}(\texttt{OPT} \geq 2^i \textit{ and } S \sim \mathcal{E}_\theta) = u^*_i$.

We now upper bound $\sum_{\theta \leq \delta} r^*_{i\theta}$. By Lemma 2.5.1, if the realization S is in scale $\theta \leq \delta$ then $\texttt{NA}$ ends before time $t \leq \alpha 2^i$, i.e., it does not go beyond phase i. Hence,

$$\sum_{\theta \leq \delta} r^*_{i\theta} \leq \sum_{\theta \leq \delta} \mathbf{P}(S \sim \mathcal{E}_\theta) \leq 1 - u_i. \tag{2.12}$$

We now use the fact that $1 - u^*_i = \sum_\theta r^*_{i\theta}$ where we sum over all scales. So,

$$\sum_{\theta > \delta, \text{good}} r^*_{i\theta} \;\geq\; \sum_\theta r^*_{i\theta} - \sum_{\theta \text{ not good}} r^*_{i\theta} - \sum_{\theta \leq \delta} r^*_{i\theta} \;\geq\; (1 - u^*_i) - \frac{u^*_i}{5} - (1 - u_i) \;=\; u_i - \frac{6u^*_i}{5},$$

where we used (2.11) and (2.12). □

Combining (2.10) and Lemma 2.5.4, and using the greedy choice in step 3 (Algorithm 1), the score at time t,

$$\text{score}(\mathcal{X}_{e(t)}) \quad \geq \quad \frac{\delta}{\beta 2^i} \left(u_i - \frac{6}{5} u^*_i \right).$$

We note that this inequality continues to hold (with a larger constant β) even if we choose an item that only maximizes the score (2.1) within a constant factor.

Using the above inequality for each time t during phase i, we have

$$G \quad \geq \quad \alpha 2^{i-1} \cdot \frac{\delta}{\beta 2^i} \left(u_i - \frac{6}{5} u^*_i \right) \quad = \quad \frac{\alpha\delta}{2\beta} \cdot \left(u_i - \frac{6}{5} u^*_i \right). \tag{2.13}$$

We will use this lower bound for G in conjunction with an upper bound, which we prove next.

2.5.2 An Upper Bound for G

We now consider the implementation of the non-adaptive list L and calculate G as a sum of contributions over the observed decision path. Let Π denote the (random) decision path followed by the non-adaptive strategy `NA`: this consists of a prefix of L along with their realizations. Denote by $\langle X_1, X_2, \ldots, \rangle$ the sequence of realizations (each in U) observed on Π. So item $\mathcal{X}_j$ is selected between time $\sum_{\ell=1}^{j-1} c_\ell$ and $\sum_{\ell=1}^{j} c_\ell$. Let h (resp. p) index the first (resp. last) item in Π (if any) that is selected (even partially) during phase i, i.e., between time $\alpha 2^{i-1}$ and $\alpha 2^i$. For each index $h \le j \le p$, let t_j denote the duration of time that item $\mathcal{X}_j$ is selected during in phase i; so t_j is the width of interval $[\sum_{\ell=1}^{j-1} c_\ell,\ \sum_{\ell=1}^{j} c_\ell] \bigcap [\alpha 2^{i-1},\ \alpha 2^i]$. It follows that $t_j \le c_j$.

Define $G(\Pi) := 0$ if index h is undefined (i.e., Π terminates before phase i), and otherwise:

$$G(\Pi) := \sum_{j=h}^{p} \frac{t_j}{c_j} \cdot \frac{f(\{X_1, \ldots, X_j\}) - f(\{X_1, \ldots, X_{j-1}\})}{Q - f(\{X_1, \ldots, X_{j-1}\})} \le \sum_{j=h}^{p} \frac{f(\{X_1, \ldots, X_j\}) - f(\{X_1, \ldots, X_{j-1}\})}{Q - f(\{X_1, \ldots, X_{j-1}\})}. \tag{2.14}$$

By the stopping criterion for L, the f value *before* the end of Π remains at most $\tau = Q(1-\delta)$. So the denominator above, i.e., $Q - f(\{X_1, \ldots, X_{j-1}\})$ is at least δQ for all j.

Lemma 2.5.5. *For any decision path Π, we have $G(\Pi) \le 1 + \ln(1/\delta)$.*

Proof. For each $h \le j \le p$, let $V_j := f(\{X_1, \ldots, X_j\})$; also let $V_0 = 0$. For $j \le p-1$, as noted above, $V_j \le \tau$; as f is integer-valued, $V_j \in \{0, 1, \cdots, \lfloor \tau \rfloor\}$. We have:

$$\sum_{j=h}^{p-1} \frac{V_j - V_{j-1}}{Q - V_{j-1}} \le \sum_{j=h}^{p-1} \sum_{y=0}^{V_j - V_{j-1} - 1} \frac{1}{Q - V_{j-1} - y} \le \sum_{\ell=\delta Q+1}^{Q} \frac{1}{\ell} \le \ln\left(\frac{Q}{\delta Q}\right) = \ln(1/\delta).$$

The second inequality above uses $Q - V_{j-1} - y \ge Q - V_j + 1 \ge Q - \tau + 1 = \delta Q + 1$. The right-hand-side of (2.14) is then:

$$\sum_{j=h}^{p-1} \frac{V_j - V_{j-1}}{Q - V_{j-1}} + \frac{V_p - V_{p-1}}{Q - V_{p-1}} \quad \le \quad 1 + \ln(1/\delta),$$

where we used $V_p \le Q$. This completes the proof. □

Taking expectations over the various decision paths means $G = \mathbb{E}_\Pi[G(\Pi)]$, so Lemma 2.5.5 gives

$$G \;\leq\; (1+\ln(1/\delta)) \cdot \mathbf{P}(\Pi \text{ doesn't terminate before phase } i) \;=\; (1+\ln(1/\delta)) \cdot u_{i-1}. \quad (2.15)$$

2.5.3 Wrapping Up

To complete the proof of Lemma 2.4.3, we set $\alpha := \frac{8\beta}{\delta}(1+\ln(1/\delta)) = O\big(\frac{\ln(1/\delta)}{\delta}\big)$, and combine (2.13) and (2.15) to get

$$u_{i-1} \geq \frac{G}{1+\ln(1/\delta)} \geq \frac{\alpha\delta}{2\beta(1+\ln(1/\delta))} \cdot \left(u_i - \frac{6}{5}u_i^*\right) = 4\left(u_i - \frac{6}{5}u_i^*\right).$$

This completes the proof of Lemma 2.4.3 and hence Theorem 2.1.1.

2.6 Applications

2.6.1 Stochastic Set Cover

The stochastic set cover problem is a special case of stochastic submodular cover. The input is a universe E of d objects and a collection $\{\mathcal{X}_1, \ldots, \mathcal{X}_m\}$ of m items. Each item $\mathcal{X}_i$ has a cost $c_i \in \mathbb{R}_+$ and corresponds to a random subset of objects (with a known explicit distribution). Different items are independent of each other. The goal is to select a set of items such that the realized subsets cover E and the expected cost is minimized. This problem was first studied by [54], where it was shown that the *adaptivity gap* lies between $\Omega(d)$ and $O(d^2)$. The correct adaptivity gap for stochastic set cover was posed as an open question by [54]. Subsequently, [2] made significant progress, by showing that the adaptivity gap is $O(d \log d \cdot \log(mc_{max}))$. However, as a function of the natural parameter d (number of objects), the best adaptivity gap remained $O(d^2)$ because the number of stochastic sets m and maximum cost c_{max} may be arbitrarily larger than d.

As a corollary of Theorem 2.1.1, we obtain an $O(d \log d)$ adaptivity gap that nearly matches the $\Omega(d)$ lower bound. In fact, for any $r \geq 1$, we obtain an r-round adaptive algorithm that costs at most $O(d^{1/r}\log d)$ times the fully adaptive cost. This nearly matches the $\Omega(\frac{1}{r^3}d^{1/r})$ r-round-adaptivity gap shown in [2]. We note that when $r = \log d$, we obtain an $O(\log d)$-approximation algorithm with a logarithmic number of adaptive rounds; this approximation ratio is the best possible even for deterministic set cover.

2.6.2 Sensor Placement with Unreliable Sensors

In the sensor placement problem, we are concerned with deploying a collection of sensors for visual surveillance or to acquire information on air quality, temperature, humidity etc. One approach to model this problem (suitable for visual surveillance) is to assume that each sensor has a particular sensing region, and to minimize the number of sensors required to cover some target region. This corresponds to the art gallery problem (see [57]), which in turn is a special case of set cover. In the setting with unreliable sensors that we consider, each sensor may fail independently with some probability (in which case it does not cover its region), and the goal is to minimize the expected number of sensors so as to cover the target region. This can be modeled as an instance of stochastic set cover, discussed in §2.6.1.

An alternative approach (suitable for information acquisition) is to model sensor readings as Gaussian processes ([39]), where the goal is to place the minimum number of sensors so as to achieve a target level of "information gain". Formally, let $[m]$ denote the set of locations and $\mathcal{Z}_i$ be a random variable representing the information at location $i \in [m]$. Let $\mathcal{Z} = \langle \mathcal{Z}_i : i \in [m]\rangle$ denote the information at all locations. A sensor at location i makes observation $\mathcal{Y}_i = \mathcal{Z}_i + \epsilon_i$ where ϵ_i is an independent (random) noise term. For any subset $A \subseteq [m]$, let $\mathcal{Y}_A = \{\mathcal{Y}_i : i \in A\}$ denote the random observations at the locations in A. Given $\mathcal{Y}_A = Y_A$, we can use $\mathbb{E}[\mathcal{Z}_i \mid \mathcal{Y}_A = Y_A]$ to make predictions on the information at *any* location i. The *information gain* of the system if we place sensors at locations $A \subseteq [m]$ is defined as:

$$g(A) = \mathbb{H}(\mathcal{Z}) - \mathbb{H}(\mathcal{Z} \mid \mathcal{Y}_A)$$

where $\mathbb{H}(\mathcal{Z} \mid \mathcal{Y}_A) = \mathbb{E}_{Y_A}[\mathbb{H}(\mathcal{Z} \mid \mathcal{Y}_A = Y_A)]$ denotes the conditional entropy of $\mathcal{Z}$ given $\mathcal{Y}_A$.

[79] showed that the function $g(A)$ is monotone and submodular in A assuming that the observations $\mathcal{Y}_A$ are conditionally independent given $\mathcal{Z}$; see Corollary 4 in their paper. The

conditional independence assumption is satisfied in our setting because the noise terms ϵ_i are independent. Given a target $\widehat{Q}$ on the information gain, the goal in the deterministic problem is to find sensor locations $A \subseteq [m]$ so that $g(A) \geq \widehat{Q}$.

In the stochastic setting (with unreliable sensors), each sensor $i \in [m]$ is active independently with some probability p_i (and fails otherwise). The goal is to place sensors (possibly adapting based on the active/failure outcomes) so that the information gain from the active sensors is at least $\widehat{Q}$. This can be modeled as an instance of stochastic submodular cover as follows. The items correspond to the m locations. For each sensor at location $i \in [m]$, we define two elements T_i and F_i corresponding to active and failure outcomes respectively. The groundset $U = \{T_i : i \in [m]\} \cup \{F_i : i \in [m]\}$. For each $i \in [m]$, random variable $\mathcal{X}_i = T_i$ with probability p_i and $\mathcal{X}_i = F_i$ otherwise. Define function $f : 2^U \to \mathbb{R}_+$ as follows

$$f(S) = g(W(S)), \text{ where } W(S) = \{i \in [m] : T_i \in S\}, \qquad \forall S \subseteq U.$$

Observe that f is monotone and submodular because g is monotone and submodular (over $[m]$). Furthermore, let M be a large enough integer so that scaling f by a factor of M makes it integer-valued. Theorem 2.1.1 can be applied to the scaled function f with target $Q = M \cdot \widehat{Q}$. Then, we obtain an r-round adaptive algorithm of cost at most $O\left(Q^{1/r} \log Q\right)$ times an optimal fully adaptive algorithm for the unreliable sensor placement problem.

2.6.3 Shared Filter Evaluation

This problem was introduced by [90] in the context of executing multiple queries on a dataset with shared (boolean) filters. There are n independent "filters" $\mathcal{X}_1, \cdots, \mathcal{X}_n$, where each filter i has cost c_i and evaluates to *true* with probability p_i (and *false* otherwise). There are also m conjunctive queries, where each query $j \in [m]$ is the "and" of some subset $Q_i \subseteq [n]$ of the filters. So query j is true iff $X_i = \mathit{true}$ for all $i \in Q_j$. The goal is to evaluate all queries at the minimum expected cost. This can be modeled as an instance of stochastic submodular cover. The items correspond to filters. The groundset $U = \{T_i, F_i\}_{i=1}^n$ where T_i (resp. F_i)

corresponds to filter i evaluating to *true* (resp. *false*). The submodular function is:

$$f(S) := \sum_{j=1}^{m} \min\left\{|Q_j|\,,\, |Q_j| \cdot |S \cap \{F_i : i \in Q_j\}| + |S \cap \{T_i : i \in Q_j\}|\right\}.$$

Note that the term for query Q_j is $|Q_j|$ iff the query's value is determined: it is *false* when a single false filter is seen and it's *true* when all its filters are true. The maximal value of the function is $Q = \sum_{j=1}^{m} |Q_j| \leq mn$. Using Theorem 2.1.1, for any $r \geq 1$, we obtain an r-round adaptive algorithm with cost at most $O((mn)^{1/r} \cdot \log(mn))$ times the optimal adaptive cost.

2.6.4 Stochastic Score Classification

The stochastic score classification (`SSClass`) problem introduced by [53], models applications such as assessing disease risk-levels of patients and giving a quality ratings to products. Formally, there are n binary random variables $\mathcal{X}_1, ..., \mathcal{X}_n$, which are used to compute a *score* $r(X) = \sum_{i=1}^{n} a_i X_i$ where $a_i \in \mathbb{Z}_+$ for all i. The realization $X_i \in \{0, 1\}$ of variable $\mathcal{X}_i$ can be determined by performing a query of cost $c_i \in \mathbb{R}_+$. Additionally, there are $B + 1$ integers $\alpha_1 < \cdots < \alpha_{B+1}$ that partition the possible scores into classes. Realization X of $\mathcal{X}$ is in *class* $j \in [B]$ iff $\alpha_j \leq r(X) \leq \alpha_{j+1} - 1$. We can view the α_j values as "cutoffs" for the classes. The goal is to determine the correct class by querying variables at minimum expected cost. Note that it is not necessary to query all variables to determine the class.

[53] showed that any instance of `SSClass` can be converted into an instance of stochastic submodular cover as follows. The groundset $U = \{(i, 0), (i, 1) : i \in [n]\}$ corresponding to the possible realizations of the variables. Consider any index $j \in \{2, \cdots B\}$. Recall the cutoff α_j and let $\beta_j := \sum_{i=1}^{n} a_i - \alpha_j + 1$. Define two submodular functions:

$$R_j^1(S) := \min\left\{\alpha_j, \sum_{(i,1)\in S} a_i\right\} \quad \text{and} \quad R_j^0(S) := \min\left\{\beta_j, \sum_{(i,0)\in S} a_i\right\}, \qquad \forall S \subseteq U.$$

Observe that $R_j^1(S) = \alpha_j$ (i.e. R_j^1 is covered) iff the realizations in S imply that the score is *at least* α_j. Similarly, $R_j^0(S) = \beta_j$ iff the realizations in S imply that the score is *at most* $\alpha_{j+1} - 1$. We combine these two functions using the "OR construction" of submodular functions to get:

$$g_j(S) = \alpha_j\beta_j - (\alpha_j - R_j^1(S)) \cdot (\beta_j - R_j^0(S)).$$

Note that g_j is covered (i.e. has value $\alpha_j\beta_j$) if, and only if, either R_j^1 or R_j^0 is covered. Moreover, g_j is monotone and submodular. Finally, we combine all these B functions using the "AND" construction to obtain $f(S) := \sum_{j=2}^{B} g_j(S)$. Note that f is monotone and submodular, and it is covered if, and only if, each of the g_js is covered, which implies that the class is correctly identified. Moreover, the maximal value of f is $Q = \sum_{j=2}^{B} \alpha_j\beta_j \leq BW^2 \leq W^3$, where $W := \sum_{i=1}^{n} a_i$.

Using Theorems 2.1.1 we obtain an r-round adaptive algorithm for independent distributions with approximation ratio $O(W^{3/r} \log W)$ (relative to the fully adaptive optimum). See Chapter 4 for a non-adaptive (i.e., one round) algorithm that achieves an $O(1)$ approximation for stochastic score classification with independent distributions. Note that the result is not applicable for correlated distributions.

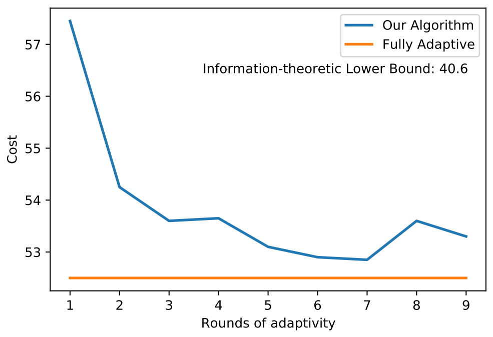
We can also extend our result to the *d-dimensional score classification problem*, where one needs to determine the classes of d different score functions $r_1, \cdots r_d$. For each score r_k, we define a submodular function f_k as above, and define a combined function $f(S) := \sum_{k=1}^{d} f_k(S)$. Now, the maximal value $Q \leq dW^3$. So we obtain r-round-adaptivity gaps of $O(d^{1/r}W^{3/r} \log(dW))$ for the independent case.

2.7 Computational Results

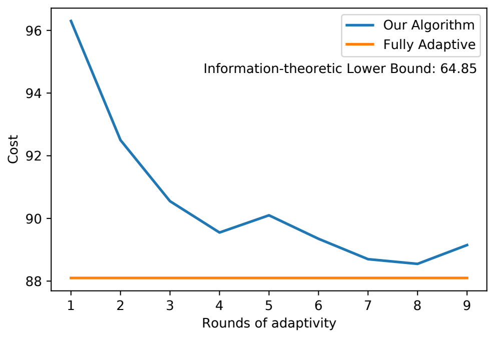
We provide a summary of computational results of our r-round adaptive algorithms for the stochastic set cover problem. We conducted all experiments using Python 3.8 and Gurobi 8.1 with a 2.3 Ghz Intel Core $i5$ processor and 16 GB 2133 MHz LPDDR3 memory.

2.7.1 Stochastic Set Cover

Instances. We use the Epinions network (`http://snap.stanford.edu/`) and a Facebook messages dataset described in [98] to generate instances of the stochastic set cover problem. The Epinions network consists of 75,879 nodes and 508,837 directed edges. We compute the subgraph induced by the top $1,000$ nodes with the highest out-degree (this subgraph has $57,092$ directed edges) and use this to generate the stochastic set cover instance. The Facebook messages dataset consists $1,266$ nodes and $6,451$ directed edges. Given an underlying directed graph, we generate an instance of the stochastic set cover problem as follows. We associate the ground set U with the set of nodes of the underlying graph. We associate an item $\mathcal{X}_u$ with each node u. Let $\Gamma(u)$ denote the out-neighbors of u. We sample a subset of $\Gamma(u)$ using the binomial distribution with $p = 0.1$ for 500 times. Let $S \subseteq \Gamma(u)$ be sampled α_S times: then $\mathcal{X}_u$ realizes to $S \cup \{u\}$ with probability $\alpha_S/500$. This ensures that $\mathcal{X}_u$ always covers u. We set f to be the coverage function. We interpret the realization of a node as the set of neighbors it influences, and so f computes the total number of people that are influenced. We set $Q = \delta n$ where n represents the number of nodes in the underlying network. We use $\delta = 0.5$ for the Epinions network. However, since the Facebook messages network is sparse, we set $\delta = 0.25$ in the second instance. All costs are one.



(a) Epinions network



(b) Facebook messages network

Figure 2.1: Computational results for Stochastic Set Cover

Results. We test our r-round adaptive algorithm on the two kinds of instances described above. We vary the number r of rounds over integers in the interval $[1, \log n]$, where $n \approx 1000$. We compare to our *fully-adaptive* algorithm which adapts after every probe: in each step, this algorithm probes the item that maximizes the score (2.1) with $\mathcal{S} = \emptyset$. We note that this also corresponds to the adaptive algorithm from [67]. To compute an estimate of the expected cost of any algorithm, we sample item realizations 20 times independently and take the average cost incurred. We also find an estimate of an information-theoretic lower bound by sampling item realizations 20 times and taking the average cost of an integer linear program (solved using the Gurobi solver) that computes the optimal *offline* cost to cover Q elements for a given realization. Note that no adaptive policy can do better than this lower bound. In fact, the gap between this information-theoretic lower bound and an optimal adaptive solution may be as large as $\Omega(Q)$ on worst-case instances. We observe that in both cases, the expected cost of our solution after only a few rounds of adaptivity is within 50% of the information-theoretic lower bound. Moreover, in $6-7$ rounds of adaptivity we notice a decrease of $\approx 8\%$ in the expected cost and these solutions are nearly as good as fully adaptive solutions. We plot this trend in Figure 2.1. Finally, note that the increase in expected cost with rounds of adaptivity (see Figure 2.1a) can be attributed to the the probabilistic nature of our algorithm (and the experimental setup). We also notice this in the next batch of experiments.

2.8 Set-based Model for Rounds

Here we discuss the "set based" model for adaptive rounds, where each round probes a fixed subset of items (and incurs their total cost). In this model, as noted in [2], we can no longer require the function f to be covered with probability one. We also provide an example below where the r-round-adaptivity gap is very large if we require coverage with probability one. Therefore, we consider solutions that are only required to cover the function with high probability. We will still compare to the (fully) adaptive optimum `OPT` that always covers the function.

Formally, an r-round adaptive solution in the set-based model proceeds as follows. For each round $i = 1, \dots, r$, it specifies a subset $\mathcal{S}_i$ of items (that depends on all realizations in rounds $1, \dots, i-1$) that are probed in parallel. The cost incurred in round i is $c(\mathcal{S}_i)$ the total cost of all probed items in that round.

Example: Consider an instance with $|U| = 1$, and function f with $f(U) = 1$ and $f(\emptyset) = 0$. So parameter $Q = 1$. There are m items, where each item $i \in \{1, \dots, m-1\}$ has cost 2^i and instantiates to U with probability $\frac{1}{2}$. The last item m costs 2^m and instantiates to U

with probability one. In the permutation model, there is a non-adaptive solution of cost $O(m)$, which just probes items in the order $1, 2 \ldots, m$. On the other hand, in the set-based model with $r < m$ rounds we claim that the optimal cost is $\Omega(2^{m/r})$ if we require f to be covered with probability one. Note that any solution $\mathcal{B}$ in the set-based model with r rounds is given by indices $0 = i(0) \leq i(1) \leq \ldots, i(r-1) \leq i(r) = m$ where the j^{th} round probes items $i(j-1)+1, \ldots, i(j)$ (if f is not already covered). The probability that round j is needed in $\mathcal{B}$ is $2^{-i(j-1)}$. So the expected cost of $\mathcal{B}$ is at least $\sum_{j=1}^{r} 2^{-i(j-1)} \cdot 2^{i(j)} \geq 2^{m/r}$ where we used $\sum_{j=1}^{r}(i(j) - i(j-1)) = i(r) - i(0) = m$. It follows that the r-round adaptivity gap in the set-based model is exponential in m (while the adaptivity gap is constant for the permutation model). This is the reason our set-based algorithms only cover f with high probability (rather than probability one).

2.8.1 Conversion Theorems

While we give our main technical results in the permutation model to make the analysis easier, we can translate our r-round algorithm in the permutation model into one in the set-based model, as we show next. Below, `OPT` is the cost of an optimal fully-adaptive solution.

Theorem 2.8.1. *For any $r \geq 1$ and $\eta > 0$, there is a set-based r-round adaptive solution for stochastic submodular cover (resp. scenario submodular cover) with expected cost at most $O(\frac{r}{\eta} Q^{1/r} \log Q) \cdot \mathtt{OPT}$ (resp. $O\big(\frac{r}{\eta}(s^{1/r} \log s + r \log Q) \cdot \mathtt{OPT}\big)$) and covers the function with probability at least $1 - \eta$.*

Proof. We will show the following "black box" reduction. Suppose there is an r-round adaptive algorithm in the permutation model with approximation ratio α. Then, there is an algorithm in the set-based model that for any $\eta \in (0, 1)$, finds an r-round adaptive solution that has expected cost at most $\frac{r\alpha}{\eta} \cdot \mathtt{OPT}$ and covers the function with probability at least $1 - \eta$. The theorem would then follow from Theorems 2.1.1 and 3.1.1.

Consider any solution $\mathcal{A}$ in the permutation-model that always covers the function. For each round $i = 1, \ldots, r$ in $\mathcal{A}$, let $\mathcal{L}_i$ be the permutation to probe items and let μ_i be the expected cost of probed items (after applying the stopping rule). Note that $\mathcal{L}_i$ and μ_i depend on realizations in all previous rounds. Let $\mathcal{S}_i$ denote the maximal prefix of $\mathcal{L}_i$ that has total cost at most $\frac{r}{\eta}\mu_i$. We define the set-based solution $\mathcal{B}$ as follows. In round i, we probe all items $\mathcal{S}_i$ in parallel. Note that the cost of solution $\mathcal{B}$ in round i is at most $\frac{r}{\eta}\mu_i$. Taking expectations and adding over all rounds, the expected cost of $\mathcal{B}$ is at most $\frac{r}{\eta}$ times that of $\mathcal{A}$. So the cost is at most $\frac{r\alpha}{\eta} \cdot \texttt{OPT}$.

We refer to round-i as a *failure* if the round-i stopping rule (in $\mathcal{A}$) *does not* apply by the end of $\mathcal{S}_i$. By Markov's inequality, the probability that round-i is a failure is at most $\frac{\eta}{r}$. Using a union bound over all r rounds, the probability of a failure in any round is at most η. Note that, if there is no failure then solution $\mathcal{B}$ indeed covers the function fully: as the items probed by $\mathcal{B}$ is a superset of those probed by $\mathcal{A}$. Hence, solution $\mathcal{B}$ covers the function with probability at least $1 - \eta$. □

Note that the approximation ratios for the set-based model are only a factor r more than in the permutation model (assuming that the "failure probability" η is any constant). So this is a good result for small (constant) r. When the number of rounds r is larger, we have a different approach that does not lose this factor r in the approximation ratio, but uses $O(r)$ set-based rounds.

Theorem 2.8.2. *For any $r \geq 1$, in the set-based model:*

- *there is a $2r$-round adaptive algorithm for stochastic submodular cover with expected cost at most $O(Q^{1/r} \log Q) \cdot \texttt{OPT}$*
- *there is a $4r$-round adaptive algorithm for scenario submodular cover with expected cost at most $O(s^{1/r} \log(sQ)) \cdot \texttt{OPT}$*

that covers the function with probability at least $1 - e^{-\Omega(r)}$.

Proof. We first consider stochastic submodular cover. We make use of the partial cover algorithm ParCA (Theorem 2.4.1) in an iterative manner (similar to the r-round algorithm in the permutation model). The difference is that we will run ParCA for $2r$ set-based rounds (instead of r). The items probed in each set-based round is a prefix of the permutation-based round. Algorithm 3 gives a formal description. The *state* at any round is the tuple $(\mathcal{T}, T)$ consisting of previously probed items and their realizations. For the analysis, we view the iterative algorithm as a $2r$ depth tree, where the nodes at depth i are the states in round i and the branches out of each node represent the observed realizations in that round. For any $i \in [2r]$, let Ω_i denote all the states in round i: note that these form a partition of the outcome space.

Algorithm 3 $2r$-round set-based algorithm for stochastic submodular cover

```
1: let $\mathcal{T} \leftarrow \emptyset$ denote the probed items and $T \leftarrow \emptyset$ their realization.
2: for $i = 1, 2, \dots, 2r$ do
3:     run ParCA $(\mathcal{X} \setminus \mathcal{T}, f_T, Q - f(T), Q^{-1/r})$.
4:     let $\mathcal{L}_i$ denote the permutation on items $\mathcal{X} \setminus \mathcal{T}$ from ParCA.
5:     let $\mu_i$ be the expected cost of probing $\mathcal{L}_i$ (in the permutation model).
6:     let $\mathcal{R}_i$ denote the maximal prefix of $\mathcal{L}_i$ of cost at most $4\mu_i$.
7:     probe items $\mathcal{R}_i$ in the set-based solution and let $R_i$ denote their realizations.
8:     update $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{R}_i$ and $T \leftarrow T \cup R_i$.
```

Expected cost. For any state $\omega \in \Omega_i$, let $\texttt{OPT}(\omega)$ denote the expected cost of the optimal adaptive policy conditioned on the realizations in ω. By Theorem 2.4.1, conditioned on ω, the expected cost $\mu_i(\omega)$ of the permutation $\mathcal{L}_i(\omega)$ is at most $\alpha \cdot \texttt{OPT}(\omega)$ where $\alpha = O(\frac{1}{r} Q^{1/r} \log Q)$. Note that the cost of the set-based round is at most $4 \cdot \mu_i(\omega)$. Hence, the (unconditional) expected cost in round i is at most

$$4 \sum_{\omega \in \Omega_i} \Pr[\omega] \cdot \mu_i(\omega) \leq 4\alpha \sum_{\omega \in \Omega_i} \Pr[\omega] \cdot \texttt{OPT}(\omega) = 4\alpha \cdot \texttt{OPT},$$

where we used that Ω_i is a partition of all outcomes. It follows that the expected cost of the set-based algorithm is at most $2r \cdot 4\alpha \cdot \texttt{OPT} = 8r\alpha \cdot \texttt{OPT}$ as claimed.

Completion probability. Consider any round i and state $\omega \in \Omega_i$. Recall that T denotes the realizations from prior rounds. The stopping rule for the permutation $\mathcal{L}_i(\omega)$ in ParCA is that the residual target drops by a factor $\delta = Q^{-1/r}$. In other words, if S denotes the realizations observed in permutation $\mathcal{L}_i(\omega)$ then the stopping rule is $Q - f(T \cup S) < \delta \cdot (Q - f(T))$. Recall that the set-based round-i involves probing the prefix $\mathcal{R}_i(\omega)$ of $\mathcal{L}_i(\omega)$ having cost $4 \cdot \mu_i(\omega)$. This round is said to be a *success* if the stopping rule applies by the end of $\mathcal{R}_i(\omega)$. By the choice of $\mathcal{R}_i$ and Markov's inequality, the probability that this round is a success is at least $\frac{3}{4}$. So the expected number of successful rounds is at least $2r \cdot \frac{3}{4} = \frac{3}{2}r$. Moreover, the success events in different rounds $i = 1, \ldots, 2r$ are independent. Hence, by a Chernoff bound, the probability of seeing less than r successes out of $2r$ rounds is at most $e^{-\Omega(r)}$. Finally, note that if there are at least r successes then the function f gets covered completely as each success reduces the residual target by a factor $Q^{-1/r}$. It follows that our algorithm covers f with probability $1 - e^{-\Omega(r)}$ as claimed.

The proof for scenario submodular cover is identical, except for the use of SParCA (Corollary 3.5.6) instead of ParCA. $\square$

2.9 Items realizing to subsets

Here, we consider a seemingly more general model for stochastic submodular cover, where each item realizes to a *subset* of elements. As before, there is a monotone submodular function $f : 2^U \to \mathbb{Z}_{\geq 0}$ with $Q := f(U)$. There are m items, where each item $\mathcal{X}_i$ has cost c_i and realizes to a random subset of U. (In the basic model, each item realizes to a single element.) The goal is to select a set of items such that the union S of their corresponding realizations satisfies $f(S) = Q$, and the expected cost is minimized. We can reduce this model to the usual one (considered in the paper) as follows. Let $W := 2^U$ denote an expanded groundset consisting of all subsets of U. Let function $g : 2^W \to \mathbb{Z}_{\geq 0}$ be defined as $g(R) = f(\cup_{r \in R} r)$ for any $R \subseteq W$. Note that g is monotone submodular as f is. Note that each item $\mathcal{X}_i$ realizes to a single element of the expanded set W. Moreover, covering function g is equivalent to covering f, and both have maximal value Q. So it suffices to solve the usual model with single-element realizations. However, the difficulty with this reduction is that the number of elements $|W|$ is exponential. Using the fact that our algorithm's runtime is independent of the size of the groundset (see remark after Algorithm 2), it follows that we obtain a polynomial (in $m, Q, c_{\max}$) time algorithm for this more general model as well.

2.10 Estimating Scores in Algorithm ParCA

The partial covering algorithm ParCA (Algorithm 1) relies on computing the maximum score according to (2.1). We are not aware of a closed-form expression to calculate the score for general functions f. Instead, we show that sampling can be used to estimate these scores, which suffices for the overall algorithm. At any point in algorithm ParCA, let $\mathcal{S} \subseteq \mathcal{X}$ denote the items already added to the non-adaptive list L. Then, we need to compute the maximum $\text{score}(\mathcal{X}_e) = \frac{g_e}{c_e}$ over $\mathcal{X}_e \in \mathcal{X} \setminus \mathcal{S}$, where

$$g_e := \sum_{S \sim \mathcal{S}: f(S) \le \tau} \mathbf{P}(\mathcal{S} = S) \cdot \mathbb{E}_{X_e \sim \mathcal{X}_e} \left[\frac{f(S \cup X_e) - f(S)}{Q - f(S)} \right] = \mathbb{E}_{\mathcal{S}, \mathcal{X}_e} \left[\mathbf{1}_{f(S) \le \tau} \cdot \frac{f(S \cup X_e) - f(S)}{Q - f(S)} \right].$$

To this end, we use $K = O(m^2 c_{\max} \log(m c_{\max}))$ independent samples to obtain an estimate $\bar{g}_e$ for each item e. Then, we choose the item that maximizes $\frac{\bar{g}_e}{c_e}$. Note that the time taken in each step is $O(m \cdot K)$, so the overall time taken by algorithm ParCA is $O(m^2 K) = \text{poly}(m, c_{\max})$.

Let L denote the list produced by the (sampling based) ParCA algorithm, and let `NA` be the resulting non-adaptive solution. We now show that the expected cost of `NA` is $O(\frac{1}{\delta} \log \frac{1}{\delta}) \cdot \texttt{OPT}$. As before, `OPT` is the cost of an optimal fully-adaptive solution. This would prove Theorem 2.4.1 even for the sampling-based ParCA algorithm.

Lemma 2.10.1. *At any step, with already added items $\mathcal{S}$, we have* $\Pr[f(S) \le \tau] \le \sum_{\mathcal{X}_e \in \mathcal{X} \setminus \mathcal{S}} g_e$.

Proof. We can write:

$$\sum_{\mathcal{X}_e \in \mathcal{X} \setminus \mathcal{S}} g_e = \sum_{\mathcal{X}_e \in \mathcal{X} \setminus \mathcal{S}} \mathbb{E}_{\mathcal{S}, \mathcal{X}_e} \left[\mathbf{1}_{f(S) \le \tau} \cdot \frac{f(S \cup X_e) - f(S)}{Q - f(S)} \right] = \mathbb{E}_{\mathcal{X}} \left[\mathbf{1}_{f(S) \le \tau} \cdot \sum_{\mathcal{X}_e \in \mathcal{X} \setminus \mathcal{S}} \frac{f(S \cup X_e) - f(S)}{Q - f(S)} \right]$$
$$\ge \mathbb{E}_{\mathcal{X}} \left[\mathbf{1}_{f(S) \le \tau} \cdot \frac{f(X) - f(S)}{Q - f(S)} \right] = \mathbb{E}_{\mathcal{X}} \left[\mathbf{1}_{f(S) \le \tau} \right] = \Pr[f(S) \le \tau].$$

Above, the inequality is by submodularity and the second-last equality uses the fact that $f(X) = Q$ with probability one. $\square$

Let L' denote the maximal prefix of list L where $\max_e g_e \ge \varepsilon := \frac{1}{m^2 c_{\max}}$ at each step. Let $\texttt{NA}'$ denote the cost incurred by the non-adaptive solution on items in L' and $\texttt{NA}''$ be the cost incurred on items $L \setminus L'$. Clearly, $\mathbb{E}[\mathtt{NA}] = \mathbb{E}[\mathtt{NA}'] + \mathbb{E}[\mathtt{NA}'']$.

By Lemma 2.10.1, it follows that after L' ends, we have $\Pr[f(S) \leq \tau] \leq m \cdot \max_e g_e \leq m\varepsilon \leq \frac{1}{mc_{\max}}$. In other words, $\Pr[\mathtt{NA}$ continues after $L'] \leq \frac{1}{mc_{\max}}$ and hence

$$\mathbb{E}[\mathtt{NA}''] \leq \Pr[\mathtt{NA} \text{ continues after } L'] \cdot mc_{\max} = O(1).$$

It now remains to bound $\mathbb{E}[\mathtt{NA}']$.

Let $\bar{\mathtt{NA}}$ denote any non-adaptive solution obtained by algorithm ParCA assuming that it always selects an item $\mathcal{X}_e$ with $\text{score}(\mathcal{X}_e) \geq \frac{1}{4}\max_i \text{score}(\mathcal{X}_i)$. In words, $\bar{\mathtt{NA}}$ is built using items having $\frac{1}{4}$-approximately maximum score. It can be easily verified that the analysis in §2.4.1 can be extended to prove $\mathbb{E}[\bar{\mathtt{NA}}] \leq O(\frac{1}{\delta}\log\frac{1}{\delta}) \cdot \mathtt{OPT}$. (The only change is an additional factor of $\frac{1}{4}$ in the right-hand-side of (2.13).) We now bound $\mathbb{E}[\mathtt{NA}']$ using $\mathbb{E}[\bar{\mathtt{NA}}]$.

Lemma 2.10.2. *Consider any item $\mathcal{X}_e$ in L' (with $\mathcal{S}$ denoting all previous items). Then, we have score$(\mathcal{X}_e) \geq \frac{1}{4}\max_{\mathcal{X}_i \in \mathcal{X}\setminus\mathcal{S}}$ score$(\mathcal{X}_i)$ with probability at least $1 - \frac{1}{m^2 c_{\max}}$.*

Proof. We partition the remaining items $\mathcal{X} \setminus \mathcal{S}$ into $I^+ = \{i : g_i \geq \varepsilon/4\}$ and $I^- = \{j : g_j < \varepsilon/4\}$.

Consider any item $i \in I^+$: so $g_i \geq \varepsilon/4$. As $\bar{g}_i$ is the average of K independent samples each of mean g_i, using a Chernoff bound we obtain $\Pr[\frac{1}{2}g_i \leq \bar{g}_i \leq 2g_i] \geq 1 - e^{-\Omega(\varepsilon K)} \geq 1 - \frac{1}{m^3 c_{\max}}$. Now consider an item $j \in I^-$: so $g_j < \varepsilon/4$. By a Chernoff bound again, $\Pr[\bar{g}_j > \varepsilon/2] \leq e^{-\Omega(\varepsilon K)} \leq \frac{1}{m^3 c_{\max}}$. So, with probability at least $1 - \frac{1}{m^2 c_{\max}}$,

$$\frac{1}{2}g_i \leq \bar{g}_i \leq 2g_i \text{ for all } i \in I^+, \text{ and } \bar{g}_j \leq \frac{\varepsilon}{2} \text{ for all } j \in I^-.$$

Below, we condition on this event.

As item e is in L', we know that $\max_i g_i \geq \varepsilon$. So $I^+ \neq \emptyset$. Recall that $\bar{g}_e = \max_i \bar{g}_i$. Combined with the above, it follows that $e \in I^+$. Moreover,

$$g_e \geq \frac{1}{2}\bar{g}_e = \frac{1}{2}\max_i \bar{g}_i = \frac{1}{2}\max_{i \in I^+} \bar{g}_i \geq \frac{1}{4}\max_{i \in I^+} g_i = \frac{1}{4}\max_i g_i.$$

This completes the proof. □

We say that the sampling was successful if the event in Lemma 2.10.2 occurs for every item in L'. As there are at most m items, sampling is successful with probability at least $1 - \frac{1}{mc_{\max}}$. Conditioned on the sampling being successful, L' is a prefix of some non-adaptive solution $\bar{\texttt{NA}}$ that always selects an item having $\frac{1}{4}$-approximately maximum score. Hence,

$$\mathbb{E}[\texttt{NA}' \,|\text{sampling successful}] \leq \mathbb{E}[\bar{\texttt{NA}}] \leq O(\frac{1}{\delta} \log \frac{1}{\delta}) \cdot \texttt{OPT}.$$

If sampling is not successful then $\texttt{NA}'$ costs at most $mc_{\max}$. So,

$$\mathbb{E}[\texttt{NA}'] \leq \mathbb{E}[\texttt{NA}' \,|\text{sampling successful}] + \Pr(\text{sampling not successful}) \cdot mc_{\max} \leq O\left(\frac{1}{\delta} \log \frac{1}{\delta}\right) \cdot \texttt{OPT}.$$

It now follows that $\mathbb{E}[\texttt{NA}] = \mathbb{E}[\texttt{NA}'] + \mathbb{E}[\texttt{NA}''] \leq O(\frac{1}{\delta} \log \frac{1}{\delta}) \cdot \texttt{OPT}$ as desired.

Chapter 3

Scenario Submodular Cover

3.1 Introduction

In this chapter we give algorithms for the scenario submodular cover problem (`ScnSC`). An instance of `ScnSC` is the same as an instance of `SSC` with the difference that the realizations of the random variables may be correlated. To keep the chapter self-contained, we repeat details from the previous chapter as required.

Recall that the *submodular cover* optimization problem requires us to pick a minimum-cost subset S of items to cover a monotone submodular function f. In the previous chapter, we discussed an application of submodular cover to sensor deployment. In this chapter, a more relevant application is in medical diagnosis as described next.

In the medical diagnosis example, we have s possible conditions (hypotheses) the patient may suffer from along with the priors on their occurrence, and our goal is to perform tests to identify the correct condition as quickly as possible [49, 78, 37, 35]. This can be cast as submodular cover by viewing each test as eliminating all inconsistent hypotheses. Hence we want a coverage value of $s-1$: once $s-1$ inconsistent hypotheses are eliminated, the remaining one must be correct. Observe that both this application involves uncertain data: the precise outcome (positive/negative) of a test is not known until the action has been taken. This uncertainty can be modeled using *stochastic submodular optimization*, where the items are stochastic, and their realizations are *correlated.*

As before, a solution for such a problem is a *sequential decision process.* At each step, an item is *probed* and its realization (e.g., active or inactive) is observed. The process is typically *adaptive*, where all the information from previously probed items is used to identify the next item to probe. This process continues until the submodular function is covered, and the goal is to minimize the expected cost of probed items. Such adaptive solutions are inherently fully sequential, which is undesirable if probing an item is time-consuming. E.g., probing/performing a test in medical diagnosis may involve a long wait for test results. Therefore, we prefer solutions with only few *rounds* of adaptivity.

Motivated by this, we ask: *how well do solutions with only a few adaptive rounds approximate fully-adaptive solutions for the stochastic submodular cover problem with correlated items?*

We give nearly tight answers, with smooth tradeoffs between the number r of adaptive rounds and the solution quality (relative to fully adaptive solutions).

The main contribution of our work is an r-round adaptive solution for stochastic submodular cover in the "set-based" model for adaptive rounds. In this model, a fixed subset of items is probed in parallel every round (and their total cost is incurred). The decisions in the current round can depend on the realizations seen in all previous rounds. However, as noted in [2], if we require function f to be covered with probability one then the r-round-adaptivity gap turns out to be very large. (See §2.8 for an example.)

Therefore, we focus on set-based solutions that are only required to cover the function with high probability.

In designing algorithms, it turns out to be more convenient to work with the "permutation" model for adaptive rounds, where the function is covered with probability one. This model was also used in prior literature [54, 2]. Here, every round of an r-round-adaptive solution specifies an ordering of all remaining items and probes them in this order until some stopping rule. See Definition 3.3.2 for a formal definition. Moreover, our r-round adaptive algorithm in the permutation model can be transformed into an r-round adaptive algorithm in the set-based model (see Chapter 2 for details). We obtain algorithms in the set-based model that:

- for any $\eta \in (0, 1)$, finds an r-round adaptive solution that has expected cost at most $\frac{r\alpha}{\eta} \cdot \mathtt{OPT}$ and covers the function with probability at least $1 - \eta$.

- finds an $O(r)$-round adaptive solution that has expected cost at most $O(\alpha) \cdot \texttt{OPT}$ and covers the function with probability at least $1 - e^{-\Omega(r)}$.

Here $\texttt{OPT}$ is the cost of an optimal fully-adaptive solution and α is the approximation ratio of our algorithm in the permutation model. The first algorithm above is for the case where r, the number of rounds of adaptivity, is small (say, a constant). In this, we keep the number of rounds the same, but we lose a factor r in the expected cost. The second algorithm is for the case that r is large, e.g., more than a constant. Here, the number of set-based rounds increases by a factor 2, but we only lose a constant factor in expected cost. We formalize and prove these results in §2.8. For the rest of the chapter, an r-round adaptive algorithm refers to an an r-round adaptive algorithm in the permutation model (unless specified otherwise).

3.1.1 Results and Techniques

Consider a monotone submodular function $f : 2^U \to \mathbb{Z}_{\geq 0}$ with $Q := f(U)$. There are m items, where each item i is a random variable $\mathcal{X}_i$ having cost c_i and corresponding to a random element of U. (Our results extend to the more general setting where each item realizes to a subset of U.) The goal is to select a set of items such that the union S of their corresponding realizations satisfies $f(S) = Q$, and the expected cost is minimized. Our main result is when the items have *correlated distributions*. Let s denote the support size of the joint distribution $\mathcal{D}$, i.e., the number of *scenarios*.

Theorem 3.1.1 (Correlated Items). *For any integer $r \geq 1$, there is an r-round adaptive algorithm for scenario submodular cover with cost $O\left(s^{1/r}(\log s + r \log Q)\right)$ times the cost of an optimal adaptive algorithm.*

We also obtain a $2r$-round adaptive algorithm with an better cost guarantee of $O\left(s^{1/r}\log(sQ)\right)$ times the cost of an optimal adaptive algorithm (see Corollary 3.5.6). Combined with the conversion to a set-based solution (Theorem 2.8.2) and setting $r = \log s$, we can then infer:

Corollary 3.1.2 (Correlated Items: Set-Based Model). *There is an $O(\log s)$ round algorithm for scenario submodular cover in the set-based model that (i) has expected cost $O(\log(sQ))$ times the optimal (fully) adaptive cost, and (ii) covers the function with probability at least $1 - \frac{1}{s}$.*

The above approximation guarantee is nearly the best possible, even with an arbitrary number of adaptive rounds: there is an $\Omega(\log s)$-factor hardness of approximation [27]. Scenario submodular cover generalizes the classic optimal decision tree problem [49, 66, 86, 78, 37, 1, 63]. A fully-adaptive $O(\log(sQ))$-approximation for scenario submodular cover was obtained in [58]; see also [92] for a more general result. In terms of rounds-of-adaptivity, an $O(\log(mQ\frac{c_{\max}}{p_{\min}}))$-approximation in $O(\log m \, \log(Qm\frac{c_{\max}}{p_{\min}}))$ set-based rounds follows from [58, 44]. Here $p_{\min} \le \frac{1}{s}$ is the minimum probability of any scenario.

We note that when the number of rounds is less than logarithmic, our result provides the first approximation guarantee even in the well-studied special case of optimal decision tree.

The results in Theorem 2.1.1 and Theorem 3.1.1 are incomparable: while the independent case has more structure in the distribution $\mathcal{D}$, its support size is exponential. Finally, the dependence on the support size s is necessary in the correlated setting, as our next result shows.

Theorem 3.1.3 (Lower Bound). *For any integer $r \ge 1$, there is an instance of scenario submodular cover with $Q = 1$ where the cost of any r-round adaptive solution is $\Omega\left(\frac{s^{1/r}}{\log s}\right)$ times the optimal adaptive cost.*

This lower bound is information-theoretic and does not depend on computational assumptions, whereas the upper bound of Theorem 3.1.1 is given by a polynomial algorithm.

Finally, we note that our algorithm is also easy to implement. We tested are algorithm on synthetic and real datasets that validate the practical performance of our algorithms. Specifically, we test our algorithm on instances of optimal decision tree. We use both real-world data and synthetic data. The real-world data has ≈ 400 scenarios and the synthetic data has $10,000$ scenarios. We find that about 6 rounds of adaptivity suffice to obtain solutions as good as fully adaptive ones. We also compared our algorithm's cost to information-theoretic lower bounds: our costs are typically within 50% of these lower bounds.

Techniques. In each round the algorithm, we iteratively compute a "score" for each item and greedily select the item of maximum score. This results in a non-adaptive list of all remaining items, and the items are *probed* in this order until a stopping rule. The SParCA rule involves reducing the number of "compatible scenarios" by an $s^{1/r}$ factor in the correlated case.

The analysis for Theorem 3.1.1 is as follows. For each $i \geq 0$, we relate the "non-completion" probabilities of the algorithm after cost $\alpha \cdot 2^i$ to the optimal adaptive solution after cost 2^i. The "stretch" factor α corresponds to the approximation ratio. In order to relate these non-completion probabilities, we consider the total score G of items selected by the algorithm between cost $\alpha 2^{i-1}$ and $\alpha 2^i$. The crux of the analysis lies in giving lower and upper bounds on the total score G (as in the independent case).

The score of any item $\mathcal{X}_e$ is the sum of two terms (i) its expected relative marginal gain as in the independent case, and (ii) an estimate of the probability on "eliminated" scenarios. Both terms are needed because the algorithm needs to balance (i) covering the function and (ii) identifying the realized scenario (after which it is trivial to cover f). Again, we normalize by the item's cost. See Equation (3.1). In lower bounding the total score G, we partition the outcome space into good/okay/bad outcomes that correspond to a high conditional probability of `OPT` (i) covering function f by cost 2^i, (ii) eliminating a constant fraction of scenarios by cost 2^i, or (iii) neither of the two cases. Further, by restricting to outcomes that have a "large" number of compatible scenarios (else, the algorithm's stopping rule would apply), we can bound the number of "relevant" outcomes by $s^{1/r}$. Then we consider `OPT` (up to cost 2^i) conditional on all good/okay outcomes and show that one of these items has a high score. To upper bound G, we again consider the total score as a sum over decision paths.

3.2 Related Work

We refer the reader to Chapter 2 for related work on the submodular cover problem, and on the stochastic submodular cover problem when item realizations are independent (a special case of this problem is the stochastic set cover problem).

The scenario submodular cover problem was introduced in [58] as a common generalization of several problems including optimal decision tree [49, 63], equivalence class determination [35] and decision region determination [69]. An $O(\log(sQ))$ fully adaptive algorithm was obtained in [58]. The same approximation ratio (in a more general setting) was also obtained in [92]. In the correlated setting, we are not aware of any prior work on limited rounds of adaptivity (when the number of rounds $r < \log s$) . Some aspects of our analysis (e.g., good/okay/bad outcomes) are similar to [92], but additional work is needed as we have to bound the r-round-adaptivity gap.

The framework of "adaptive submodularity", introduced by [56], models correlations in stochastic submodular cover in a different way. Adaptive submodularity is a combined condition on the goal function f (that needs to be covered) and the distribution $\mathcal{D}$ on items. While stochastic submodular cover with independent items satisfies adaptive-submodularity, scenario submodular cover does not. Although scenario submodular cover is *not* a special case of AdSubCov, [58] showed that scenario submodular cover can be re-formulated as AdSubCov with a different goal function that is adaptive-submodular. However, this new goal function has a larger "Q value" of $\frac{Q}{p_{min}}$. So, when the algorithm from [44] is applied to this new goal function, it only implies an $O(\log(mQ\frac{c_{\max}}{p_{\min}}))$-approximation in $O(\log m \, \log(Qm\frac{c_{\max}}{p_{\min}}))$ rounds (this can be compared to Corollary 3.1.2). To the best of our knowledge, there are no algorithms for AdSubCov using fewer than squared-logarithmic rounds of adaptivity.

3.3 Definitions

In the scenario submodular cover problem, the input is a collection of m random variables (or *items*) $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_m\}$. Each item $\mathcal{X}_i$ has a cost $c_i \in \mathbb{R}_+$, and realizes to a random element of groundset U. Let the joint distribution of $\mathcal{X}$ be denoted by $\mathcal{D}$ (this captures correlations). The realization of item $\mathcal{X}_i$ is denoted by $X_i \in U$; this realization is only known when $\mathcal{X}_i$ is *probed* at a cost of c_i. Extending this notation, given a subset of items $\mathcal{S} \subseteq \mathcal{X}$, its realization is denoted $S = \{X_i : \mathcal{X}_i \in \mathcal{S}\} \subseteq U$.

In addition, we are given an integer-valued monotone submodular function $f : 2^U \to \mathbb{Z}_+$ with $f(U) = Q$. A realization S of items $\mathcal{S} \subseteq \mathcal{X}$ is *feasible* if and only if $f(S) = Q$ the maximal value; in this case, we also say that $\mathcal{S}$ *covers* f. The goal is to probe (possibly adaptively) a subset $\mathcal{S} \subseteq \mathcal{X}$ of items that gets realized to a feasible set. We use the shorthand $c(\mathcal{S}) := \sum_{i:\mathcal{X}_i \in \mathcal{S}} c_i$ to denote the total cost of items in $\mathcal{S} \subseteq \mathcal{X}$. The objective is to minimize the expected cost of probed items, where the expectation is taken over the randomness in $\mathcal{X}$. We consider the following types of solutions.

Definition 3.3.1. *For an integer $r \geq 1$, an r-***round-adaptive** *solution in the* **set-based model** *proceeds as follows. For each round $k = 1, \ldots, r$, it specifies a subset $\mathcal{S}_k$ of items that is probed in parallel. The cost incurred in round k is $c(\mathcal{S}_k)$ the total cost of all probed items in that round. The subset $\mathcal{S}_k$ in round k can depend on realizations seen in all previous rounds $1, \ldots, k-1$.*

In the set-based model, we allow solutions to be infeasible (i.e., fail to cover f) with some small probability $\eta > 0$. As shown in Appendix 2.8, such a relaxed solution is necessary. In designing algorithms, we will work with the "permutation" model, as defined next.

Definition 3.3.2. *For an integer $r \geq 1$, an r-***round-adaptive** *solution in the* **permutation model** *proceeds in r rounds of adaptivity. In each round $k \in \{1, \ldots, r\}$, the solution specifies an ordering of all remaining items and probes them in this order until some stopping rule. The decisions in round k can depend on the realizations seen in rounds $1, \ldots, k-1$.*

In the permutation model, solutions must be feasible with probability one. As shown in Appendix 2.8, our algorithms in the permutation model can be converted into algorithms in the set-based (with similar approximation ratios). Henceforth, an r-round adaptive algorithm refers to an an r-round adaptive algorithm in the permutation model (unless specified otherwise).

Setting $r = 1$ in Definition 3.3.2 gives us a *non-adaptive* solution as studied in [54, 2]. Setting $r = m$ gives us a *(fully) adaptive* solution. Having more rounds leads to a smaller objective value, so adaptive solutions have the least objective value. Our performance guarantees are relative to an optimal fully adaptive solution; let `OPT` denote this solution and its cost. The *r-round-adaptivity gap* is defined as follows:

$$\sup_{\text{instance } I} \frac{\mathbb{E}[\text{cost of best } r\text{-round adaptive solution on } I]}{\mathbb{E}[\text{cost of best adaptive solution on } I]}$$

Setting $r = 1$ gives the *adaptivity gap*.

3.4 Scenario Submodular Cover

In this section, we describe an r-round adaptive algorithm for the *scenario submodular cover* problem. As before, we have a collection of m stochastic items $\mathcal{X} = \{\mathcal{X}_1, ..., \mathcal{X}_m\}$ with costs c_i. In contrast to the independent case, the stochastic items here are correlated, and their joint distribution $\mathcal{D}$ is given as input. The goal is to minimize the expected cost of a solution $\mathcal{S} \subseteq \mathcal{X}$ that realizes to a feasible set (i.e., $f(S) = Q$).

The joint distribution $\mathcal{D}$ specifies the (joint) probability that $\mathcal{X}$ realizes to any outcome $X \in U^m$. We refer to the realizations $X \in U^m$ that have a non-zero probability of occurrence as *scenarios*. Let $s = |\mathcal{D}|$ denote the number of scenarios in $\mathcal{D}$. The set of scenarios is denoted $M = \{1, \cdots, s\}$ and p_ω denotes the probability of each scenario $\omega \in M$. Note that $\sum_{\omega=1}^{s} p_\omega = 1$. For each scenario $\omega \in M$ and item $\mathcal{X}_e$, we denote by $X_e(\omega) \in U$ the realization of $\mathcal{X}_e$ in scenario ω. The distribution $\mathcal{D}$ can be viewed as selecting a random *realized scenario* $\omega^* \in M$ according to the probabilities $\{p_\omega\}$, after which the item realizations are deterministically set to $\langle X_1(\omega^*), \cdots, X_m(\omega^*)\rangle$. However, an algorithm does not know the realized scenario ω^*: it only knows the realizations of the probed items (using which it can infer a posterior distribution for ω^*). As stated in §4.2, our performance guarantee in this case depends on the support-size s. We will also show that such a dependence is necessary (even when Q is small).

For any subset $\mathcal{S} \subseteq \mathcal{X}$ of items, we denote by $S(\omega)$, the realizations for items in $\mathcal{S}$ under scenario ω. We say that scenario ω is *compatible* with a realization of $\mathcal{S} \subseteq \mathcal{X}$ if and only if, $\mathcal{X}_e$ realizes to $X_e(\omega)$ for all items $\mathcal{X}_e \in \mathcal{S}$.

3.4.1 The Algorithm

Similar to the algorithm for the independent case, it is convenient to solve a *partial cover* version of the scenario submodular cover problem. However, the notion of partial progress is different: we will use the *number of compatible scenarios* instead of function value. Formally, in the partial version, we are given a parameter $\delta \in [0, 1]$ and the goal is to probe some items $\mathcal{R}$ that realize to a set R such that either (i) the number of compatible scenarios is less than δs or (ii) the function f is fully covered. Clearly, if $\delta = 1/s$ then case (i) cannot happen (it corresponds to zero compatible scenarios), so the function f must be fully covered. We will use this algorithm with different parameters δ to solve the r-round version of the problem. The main result of this section is:

Theorem 3.4.1. *There is a non-adaptive algorithm for the partial cover version of scenario submodular cover with cost $O\left(\frac{1}{\delta}(\ln\frac{1}{\delta} + \log Q)\right)$ times the optimal adaptive cost for the (full) submodular cover.*

The algorithm first creates an ordering/list L of the items non-adaptively; that is, without knowing the realizations of the items. To do so, at each step we pick a new item that maximizes a carefully-defined score function (Equation (3.1)). The score of an item depends on an estimate of progress towards (i) eliminating scenarios and (ii) covering function f. Before we state this score formally, we need some definitions.

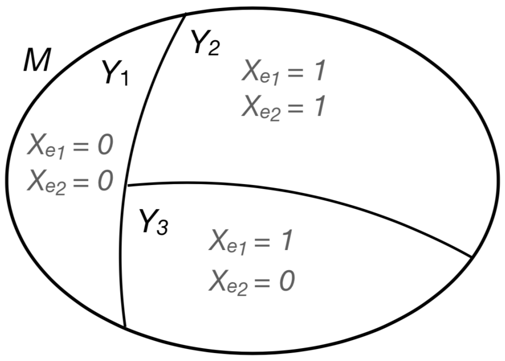
Definition 3.4.1. *For any $\mathcal{S} \subseteq \mathcal{X}$ let $\mathcal{H}(\mathcal{S})$ denote the partition $\{Y_1, \cdots, Y_\ell\}$ of the scenarios M where all scenarios in a part have the same realization for items in $\mathcal{S}$. Let $\mathcal{Z} := \{Y \in \mathcal{H}(\mathcal{S}) : |Y| \geq \delta s\}$ be the set of "large" parts having size at least δs.*

In other words, scenarios ω and σ lie in the same part of $\mathcal{H}(\mathcal{S})$ if and only if $S(\omega) = S(\sigma)$. Note that partition $\mathcal{H}(\mathcal{S})$ does not depend on the realization of $\mathcal{S}$. Moreover, after probing and realizing items $\mathcal{S}$, the set of compatible scenarios must be one of the parts in $\mathcal{H}(\mathcal{S})$. Also, the number of "large" parts $|\mathcal{Z}| \leq \frac{s}{\delta s} = \frac{1}{\delta}$ as the number of scenarios $|M| = s$. See Figure 3.1a for an example.

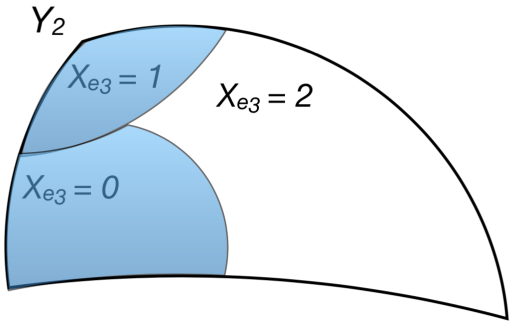
Definition 3.4.2. *For any $\mathcal{X}_e \in \mathcal{X}$ and subset $Z \subseteq M$ of scenarios, consider the partition of Z based on the realization of $\mathcal{X}_e$. Let $B_e(Z) \subseteq Z$ be the largest cardinality part, and define $L_e(Z) := Z \setminus B_e(Z)$.*

Note that $L_e(Z)$ is comprised of several parts of the above partition of Z. If the realized scenario $\omega^* \in L_e(Z)$ and $\mathcal{X}_e$ is selected then, at least half the scenarios in Z will be eliminated (as being incompatible with X_e). Figure 3.1b illustrates these definitions.

For any $Z \in \mathcal{H}(\mathcal{S})$, note that the realizations $S(\omega)$ are identical for all $\omega \in Z$: we use $S(Z) \subseteq U$ to denote the realization of $\mathcal{S}$ under each scenario in Z.



(a) We have $\mathcal{S} = \{\mathcal{X}_{e_1}, \mathcal{X}_{e_2}\}$, and we partition the set of scenarios M based on outcomes of $\mathcal{S}$ to get $\mathcal{H}(S) = \{Y_1, Y_2, Y_3\}$.



(b) We further partition scenarios Y_2 based on realizations of $\mathcal{X}_{e_3}$.The part of Y_2 compatible with outcome $X_{e_3} = 2$ is the largest cardinality part, that is, $B_{e_3}(Y_2)$. The shaded region represents $L_{e_3}(Y_2)$.

Figure 3.1: Illustrations of Key Definitions

If $\mathcal{S}$ denotes the previously added items in list L, the score (3.1) involves a term for each part $Z \in \mathcal{Z}$, which itself comes from two sources:

- *Information gain* $\sum_{\omega \in L_e(Z)} p_\omega$, the total probability of scenarios in $L_e(Z)$.
- *Relative function gain* $\sum_{\omega \in Z} p_\omega \cdot \frac{f(S(Z) \cup X_e(\omega)) - f(S(Z))}{Q - f(S(Z))}$, the expected relative gain obtained by including element X_e, where the expectation is over the scenarios in part Z.

The overall score of item $\mathcal{X}_e$ is the sum of these terms (over all parts in $\mathcal{Z}$) normalized by the cost c_e of item $\mathcal{X}_e$. In defining the score, we only focus on the "large" parts $\mathcal{Z}$. If the realization of $\mathcal{S}$ corresponds to any other part then the number of compatible scenarios would be less than δs (and the partial-cover algorithm would have terminated).

Once the list L is specified, the algorithm starts probing and realizing the items in this order, and does so until either (i) the number of compatible scenarios drops below δs, or (ii) the realized function value equals Q. Note that in case (ii), the function is fully covered. See Algorithm 4 for a formal description of the non-adaptive partial-cover algorithm.

Given this partial covering algorithm we immediately get an algorithm for the r-round version of the problem, where we are allowed to make r rounds of adaptive decisions. Indeed, we can first set $\delta = s^{-1/r}$ and solve the partial covering problem. Suppose we probe the items

Algorithm 4 Scenario PARtial Covering Algorithm SParCA$(\mathcal{X}, M, f, Q, \delta)$

```
1: $\mathcal{S} \leftarrow \emptyset$ and list $L \leftarrow \langle\rangle$.
2: while $\mathcal{S} \neq \mathcal{X}$ do                    ▷ Building the list non-adaptively
3:     define $\mathcal{Z}$ and $L_e(Z)$ as in Definitions 3.4.1 and 3.4.2.
4:     select an item $\mathcal{X}_e \in \mathcal{X} \setminus \mathcal{S}$ that maximizes:

       $$\text{score}(\mathcal{X}_e) = \frac{1}{c_e} \cdot \sum_{Z \in \mathcal{Z}} \left( \sum_{\omega \in L_e(Z)} p_\omega + \sum_{\omega \in Z} p_\omega \cdot \frac{f(S(Z) \cup X_e(\omega)) - f(S(Z))}{Q - f(S(Z))} \right) \qquad (3.1)$$

5:     $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathcal{X}_e\}$ and list $L \leftarrow L \circ \mathcal{X}_e$
6: $\mathcal{R} \leftarrow \emptyset$, $R \leftarrow \emptyset$, $H \leftarrow M$.
7: while $|H| \geq \delta|M|$ and $f(R) < Q$ do               ▷ Probing items on the list
8:     $\mathcal{X}_e \leftarrow$ first r.v. in list $L$ not in $\mathcal{R}$
9:     $X_e = v \in U$ be the realization of $\mathcal{X}_e$.
10:    $R \leftarrow R \cup \{v\}, \mathcal{R} \leftarrow \mathcal{R} \cup \{\mathcal{X}_e\}$
11:    $H \leftarrow \{\omega \in H : X_e(\omega) = v\}$
12: return probed items $\mathcal{R}$, realizations $R$ and compatible scenarios $H$.
```

$\mathcal{R}$ (with realizations $R \subseteq U$) and are left with compatible scenarios $H \subseteq M$. Then we can *condition* on scenarios H and the marginal value function f_R (which is submodular), and inductively get an $r-1$-round solution for this problem. The following algorithm and result formalizes this.

Algorithm 5 r-round adaptive algorithm for scenario submodular cover NSC$(r, \mathcal{X}, M, f)$

```
1: run SParCA$(\mathcal{X}, M, f, Q, |M|^{-1/r})$ for round one. Let $\mathcal{R}$ denote the probed items, $R$
   their realizations, and $H \subseteq M$ the compatible scenarios returned by SParCA.
2: define residual submodular function $\widehat{f} := f_R$.
3: recursively solve NSC$(r-1, \mathcal{X} \setminus \mathcal{R}, H, \widehat{f})$.
```

Theorem 3.4.2. *Algorithm 5 is an r-round adaptive algorithm for scenario submodular cover with cost $O\left(s^{1/r}(\log s + r \log Q)\right)$ times the optimal adaptive cost, where s is the number of scenarios.*

Proof. We proceed by induction on the number of rounds r. Let `OPT` denote the cost of an optimal fully adaptive solution. The base case is $r = 1$, in which case $\delta = s^{-1/r} = \frac{1}{s}$. By Theorem 3.4.1, the partial cover algorithm SParCA$(\mathcal{X}, M, f, Q, s^{-1/r})$ obtains a realization R and compatible scenarios $H \subseteq M$ where either (i) $|H| < \delta s = 1$ or (ii) $f(R) = Q$. Clearly, we cannot have case (i) as there is always at least one compatible scenario. So f is fully covered. Moreover, the algorithm's expected cost is $O(s(\log s + \log Q)) \cdot \texttt{OPT}$, as claimed.

We now consider $r > 1$ and assume (inductively) that Algorithm 5 finds an $r-1$-round $O\left(s^{1/r}(\log s + r \log Q)\right)$-approximation algorithm for any instance of scenario submodular cover. Let $\delta = s^{-1/r}$. By Theorem 3.4.1, the expected cost in round 1 (step 1 in Algorithm 5) is $O(s^{1/r}(\frac{1}{r}\log s + \log Q)) \cdot \texttt{OPT}$. Let $\widehat{s} = |H|$ denote the number of scenarios in the residual instance. Let $\widehat{Q} := Q - f(R) = \widehat{f}(U)$ denote the maximal value of the residual submodular function $\widehat{f} = f_R$. By definition of the partial covering problem, we have either $\widehat{s} < \delta s$ or $\widehat{Q} = 0$. If $\widehat{Q} = 0$ then our algorithm incurs no further cost and the inductive statement follows. We now assume $\widehat{s} < \delta s = s^{\frac{r-1}{r}}$. The optimal solution $\texttt{OPT}$ conditioned on the scenarios H gives a feasible adaptive solution to the residual instance of covering $\widehat{f}$; we denote this conditional solution by $\widehat{\texttt{OPT}}$. We inductively get that the cost of our $r-1$-round solution on $\widehat{f}$ is at most

$$O(\widehat{s}^{\frac{1}{r-1}}(\log \widehat{s} + (r-1)\log \widehat{Q})) \cdot \widehat{\texttt{OPT}} \le O\left(s^{1/r}\left(\frac{r-1}{r}\log s + (r-1)\log Q\right)\right) \cdot \widehat{\texttt{OPT}},$$

where we used $\widehat{s} < s^{(r-1)/r}$ and $\widehat{Q} \le Q$. As this holds for every subset of scenarios H, we can take expectations over H to get that the (unconditional) expected cost of the last $r-1$ rounds is $O\left(s^{1/r}\left(\frac{r-1}{r}\log s + (r-1)\log Q\right)\right) \cdot \texttt{OPT}$. Adding to this the cost of the first round, which is $O(s^{1/r}(\frac{1}{r}\log s + \log Q)) \cdot \texttt{OPT}$, completes the proof. □

3.4.2 Analysis for the partial covering algorithm

We now prove Theorem 3.4.1. Consider any call to SParCA. Let $s = |M|$ denote the number of scenarios in the instance. Recall that the goal is to probe items $\mathcal{R} \subseteq \mathcal{X}$ with some realization R and compatible scenarios $H \subseteq M$ such that (i) $|H| < \delta s$ or (ii) $f(R) = Q$. We denote by $\texttt{OPT}$ an optimal fully adaptive solution for the covering problem on f. Now we analyze the cost incurred by our algorithm's non-adaptive strategy (which we call $\texttt{NA}$). Note that $\texttt{NA}$ probes items in the order given by the list L (generated by SParCA) and stops when either condition (i) or (ii) above occurs. We consider the expected cost of this strategy, and relate it to the cost of $\texttt{OPT}$. The high-level approach is similar to that for the independent case: but the details are quite different.

We refer to the cumulative cost incurred until any point in a solution as *time*. We say that OPT is in phase i in the time interval $[2^i, 2^{i+1})$ for $i \geq 0$. We say that NA is in phase i in the time interval $[\beta \cdot 2^{i-1}, \beta \cdot 2^i)$ for $i \geq 1$. We use phase 0 to refer to the interval $[1, \beta)$. We set $\beta := \frac{16}{\delta} \log(Q/\delta)$; this choice will become clear in the proof of Lemma 3.4.3. The following notation is associated with any phase $i \geq 0$:

- u_i: probability that NA goes beyond phase i, i.e., costs at least $\beta \cdot 2^i$.
- u_i^*: probability that OPT goes beyond phase $i-1$, i.e., costs at least 2^i.

Since all costs are integers, $u_0^* = 1$. For ease of notation, we also use OPT and NA to denote the *random* cost incurred by OPT and NA respectively. The main result here is that $\mathbb{E}[\texttt{NA}] \leq O(\beta) \cdot \mathbb{E}[\texttt{OPT}]$. As in the independent case, it suffices to show:

Lemma 3.4.3. *For any phase $i \geq 1$, we have $u_i \leq \frac{u_{i-1}}{4} + 2u_i^*$.*

Using this lemma, we can immediately prove Theorem 3.4.1. This part of the analysis is identical to the one presented in §2.4 for Theorem 2.4.1.

3.5 Proof of the key lemma for Scenario Submodular Cover

We now prove Lemma 3.4.3. Let L be the list returned by SParCA. Recall that NA is the cost incurred by non-adaptively realizing items in the order of L until (i) the number of compatible scenarios $|H| < \delta s$ or (ii) f gets fully covered. For each time $t \geq 0$, let $\mathcal{X}_{e(t)}$ denote the item that would be selected at time t. In other words, this is the item which causes the cumulative cost of L to exceed t for the first time. We define the *total gain* as the random variable

$$G := \sum_{t=\beta 2^{i-1}}^{\beta 2^i} \text{score}(\mathcal{X}_{e(t)}),$$

which corresponds to the sum of scores over the time interval $[\beta \cdot 2^{i-1}, \beta \cdot 2^i)$. The proof will be completed by upper and lower bounding G, which we do next. The lower bound views G as a sum over time steps, whereas the upper bound views G as a sum over decision paths.

3.5.1 Lower bounding G

For the analysis, it will be convenient to view OPT as a decision tree where nodes correspond to probed items and branches correspond to item realizations. Fix some time $t \in [\beta \cdot 2^{i-1}, \beta \cdot 2^i)$ in phase i. Let $\mathcal{S}$ denote the items that have already been added to the list L: so $\mathcal{X} \setminus \mathcal{S}$ are the available items. The partition $\mathcal{H}(\mathcal{S})$ of scenarios M and the large parts $\mathcal{Z}$ are as in Definition 3.4.1. For any $Z \in \mathcal{Z}$, we define:

- $S(Z) \subseteq U$ is the realization from items $\mathcal{S}$ compatible with all scenarios in Z.
- $Q_Z := Q - f(S(Z))$ is the residual target (after selecting $\mathcal{S}$) under scenarios Z.
- residual submodular function $f_Z := f_{S(Z)}$ under scenarios Z.
- for any item $\mathcal{X}_e \in \mathcal{X} \setminus \mathcal{S}$, scenarios $L_e(Z) \subseteq Z$ are as in Definition 3.4.2. This is the complement of the largest part of Z (based on the realization of $\mathcal{X}_e$).
- $\texttt{OPT}_Z$ is the subtree of OPT until time 2^i, restricted to paths traced by scenarios in Z; this only includes items that are *completely* selected by time 2^i.
- $\texttt{Stem}_Z$ is the path in $\texttt{OPT}_Z$ that at each node $\mathcal{X}_e$ follows the branch corresponding to the realization compatible with scenarios $B_e(Z) = Z \setminus L_e(Z)$. We use $\texttt{Stem}_Z$ to also denote the set of items on this path.

See Figure 3.2 for an illustration of $\texttt{OPT}_Z$ and $\texttt{Stem}_Z$.

For each $Z \in \mathcal{Z}$, we partition scenarios Z based on the realizations on $\texttt{Stem}_Z$:

- $Z_{\text{good}} = \{\omega \in Z : \omega$ compatible at the end of $\texttt{Stem}_Z$ and f covered$\}$,
- $Z_{\text{bad}} = \{\omega \in Z : \omega$ compatible at the end of $\texttt{Stem}_Z$ and f uncovered$\}$,
- $Z_{\text{ok}} = \{\omega \in Z : \omega \in L_e(Z)$ for some $\mathcal{X}_e \in \texttt{Stem}_Z\}$; these are the scenarios that diverge from $\texttt{Stem}_Z$ at some item $\mathcal{X}_e$.

See Figure 3.2 for an illustration.

Definition 3.5.1. *We define part $Z \in \mathcal{Z}$ as **good** if $p(Z_{good}) \geq p(Z)/2$, **bad** if $p(Z_{bad}) \geq p(Z)/2$, or **okay** if $p(Z_{okay}) \geq p(Z)/2$.*

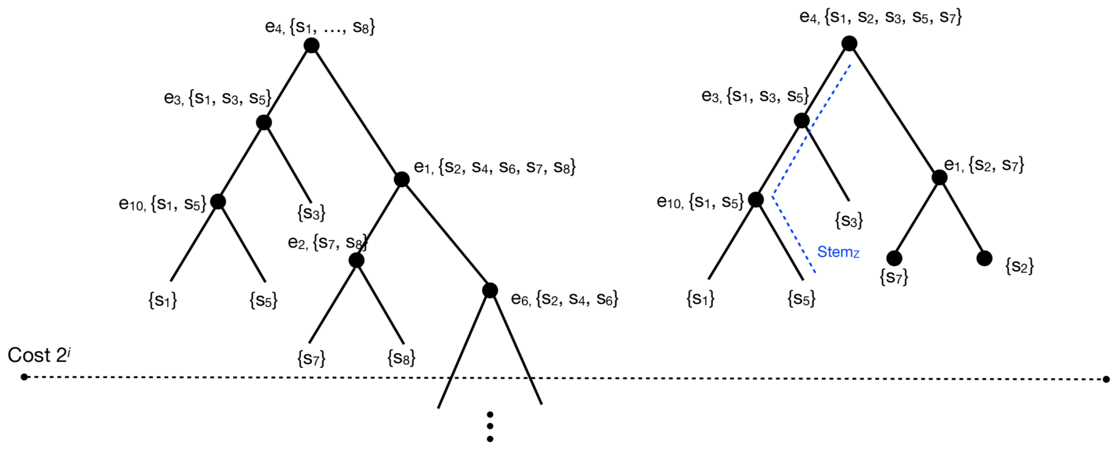


Figure 3.2: The tree on the left represents the decision tree `OPT` for scenarios $\{s_1, ..., s_8\}$. Each node represents the element chosen at the node, and the set of scenarios compatible until the node (before element at node is chosen). We redraw the decision tree for $Z = \{s_1, s_2, s_3, s_5, s_7\}$ and restrict attention to scenarios in Z. The decision tree on the right represents $\texttt{OPT}_Z$. Note that both trees are restricted to time 2^i. Furthermore, we add $\texttt{Stem}_Z$ (using dotted lines) in $\texttt{OPT}_Z$. Note that scenario s_5 is a *good* scenario since it is covered by the end of $\texttt{Stem}_Z$, and scenarios s_1, s_2, s_3, s_7 are all *okay* scenarios. Thus $Z_{\text{good}} = \{s_5\}$ and $Z_{\text{okay}} = \{s_1, s_2, s_3, s_7\}$. If s_5 was *not* covered by time 2^i, it would be a *bad* scenario.

If there are ties, they can be broken arbitrarily.

Lemma 3.5.1. *Each $Z \in \mathcal{Z}$ is either good, bad or okay.*

Proof. Observe that one of Z_{good} or Z_{bad} is always empty and the other equals all scenarios compatible at the end of $\texttt{Stem}_Z$. Moreover, Z_{okay} consists of all scenarios that diverge from $\texttt{Stem}_Z$. So $Z_{\text{good}} \cup Z_{\text{bad}} \cup Z_{\text{okay}} = Z$. It follows that $\max\{p(Z_{\text{good}}), p(Z_{\text{bad}}), p(Z_{\text{okay}})\} \geq p(Z)/2$, which proves the lemma. $\square$

For each $Z \in \mathcal{Z}$, let $\mathcal{T}_Z = \texttt{Stem}_Z \setminus \mathcal{S} \subseteq \mathcal{X} \setminus \mathcal{S}$ denote the *available* items on $\texttt{Stem}_Z$. In the next two lemmas, we lower bound the total score from items in $\mathcal{T}_Z$. We consider separately the terms corresponding to function gain and information gain. For any scenario $\omega \in Z$, we use $Stem_Z(\omega) \subseteq U$ denotes the realization of items $\texttt{Stem}_Z$ under scenario ω.

Lemma 3.5.2. *If Z is good, then $\sum_{\omega \in Z} p_\omega \cdot \frac{f_Z(T_Z(\omega))}{Q_Z} \geq \frac{p(Z)}{2}$.*

Proof. Consider any scenario $\omega \in Z_{\text{good}}$. If ω is realized then f is covered by the realization $Stem_Z(\omega)$ of $\texttt{Stem}_Z$. Thus, $f(Stem_Z(\omega)) = Q$. We have

$$f_Z(T_Z(\omega)) = f(T_Z(\omega) \cup S(Z)) - f(S(Z)) = f(Stem_Z(\omega) \cup S(Z)) - f(S(Z)) = Q - f(S(Z)) = Q_Z,$$

where the second equality uses that $T_Z(\omega) = Stem_Z(\omega) \setminus S(Z)$. As Z is good, $p(Z_{\text{good}}) \geq p(Z)/2$, and the lemma follows by summing over $\omega \in Z_{\text{good}}$. $\square$

Lemma 3.5.3. *If Z is okay, then $p\left(\bigcup_{\mathcal{X}_e \in \mathcal{T}_Z} L_e(Z)\right) \geq \frac{p(Z)}{2}$.*

Proof. Note that $Z_{\text{okay}} = \bigcup_{\mathcal{X}_e \in \texttt{Stem}_Z} L_e(Z)$. As Z is okay, we have $p(Z_{\text{okay}}) \geq p(Z)/2$. Recall that Z is a part in $\mathcal{H}(\mathcal{S})$, which implies that realizations $S(\omega)$ of $\mathcal{S}$ are identical under all scenarios $\omega \in Z$. Hence, $B_e(Z) = Z$ for each $\mathcal{X}_e \in \mathcal{S}$, i.e., $\cup_{\mathcal{X}_e \in \mathcal{S}} L_e(Z) = \emptyset$. So,

$$\bigcup_{\mathcal{X}_e \in \mathcal{T}_Z} L_e(Z) \quad \supseteq \quad \left(\bigcup_{\mathcal{X}_e \in \texttt{Stem}_Z} L_e(Z)\right) \setminus \left(\bigcup_{\mathcal{X}_e \in \mathcal{S}} L_e(Z)\right) \quad = \quad \bigcup_{\mathcal{X}_e \in \texttt{Stem}_Z} L_e(Z) \quad = \quad Z_{\text{okay}}.$$

Hence, $p\left(\bigcup_{\mathcal{X}_e \in \mathcal{T}_Z} L_e(Z)\right) \geq \frac{p(Z)}{2}$, which completes the proof. $\square$

We now prove a bound on the *overall* probability of parts that are either good or okay.

Lemma 3.5.4. *We have* $\sum_{Z:\textit{okay or good}} p(Z) \geq (u_i - 2u_i^*)$.

Proof. We first show that $\sum_{Z:\text{ bad}} p(Z) \leq 2 \cdot u_i^*$. For any $Z \in \mathcal{Z}$, if scenario $\omega \in Z_{\text{bad}}$ is realized then we know that `OPT` does not cover f at the end of $\texttt{Stem}_Z$, which implies that `OPT` costs at least 2^i. Thus $u_i^* \geq \sum_{Z\in\mathcal{Z}} \sum_{\omega \in Z_{\text{bad}}} p_\omega$. Moreover, $p(Z_{\text{bad}}) \geq p(Z)/2$ if Z is bad. So,

$$u_i^* \geq \sum_{Z\in\mathcal{Z}} \sum_{\omega\in Z_{\text{bad}}} p_\omega \geq \sum_{Z:\text{bad}} \sum_{\omega\in Z_{\text{bad}}} p_\omega \geq \sum_{Z:\text{ bad}} p(Z)/2. \tag{3.2}$$

Thus, $\sum_{Z:\text{bad}} p(Z) \leq 2u_i^*$ as claimed.

Consider any part $Y \in \mathcal{H}(\mathcal{S}) \setminus \mathcal{Z}$; recall Definition 3.4.1. Note that $|Y| < \delta s$. So, if $\mathcal{S}$ realizes to the set $S(Y)$ then `NA` would terminate by time $t \leq \beta 2^i$, which implies that `NA` does not go beyond phase i. Thus,

$$\sum_{Y\in\mathcal{H}(\mathcal{S})\setminus\mathcal{Z}} p(Y) \leq \Pr[\texttt{NA} \text{ terminates by phase } i] = 1 - u_i. \tag{3.3}$$

Finally, we have

$$\sum_{Z:\text{okay or good}} p(Z) = 1 - \sum_{Z:\text{ bad}} p(Z) - \sum_{Y\in\mathcal{H}(\mathcal{S})\setminus\mathcal{Z}} p(Y) \geq 1 - 2u_i^* - (1-u_i) = u_i - 2u_i^*,$$

where we used (3.2) and (3.3). This completes the proof. $\square$

We are now ready to prove the lower bound on $\text{score}(\mathcal{X}_{e(t)})$.

Lemma 3.5.5. *For any time t in phase i, we have* $\text{score}(\mathcal{X}_{e(t)}) \geq \delta \cdot \frac{1}{2^{i+1}} \cdot (u_i - 2u_i^*)$.

Proof. Recall the subsets $\mathcal{T}_Z \subseteq \mathcal{X} \setminus \mathcal{S}$ for $Z \in \mathcal{Z}$. Let $\mathcal{T} = \bigsqcup_{Z\in\mathcal{Z}} \mathcal{T}_Z$ denote the *multiset* that contains each item as many times as it occurs. By definition of $\texttt{Stem}_Z$, the cost $c(\mathcal{T}_Z) \leq 2^i$ for each Z. So, $c(\mathcal{T}) = \sum_{Z\in\mathcal{Z}} c(\mathcal{T}_Z) \leq |\mathcal{Z}| \cdot 2^i \leq \frac{1}{\delta} \cdot 2^i$. Moreover, $\mathcal{T} \subseteq \mathcal{X} \setminus \mathcal{S}$: so each item in $\mathcal{T}$ is available to be added to L at time t. We can lower bound $\text{score}(\mathcal{X}_{e(t)})$ by averaging over the multiset $\mathcal{T}$:

$$\begin{aligned}\text{score}(\mathcal{X}_{e(t)}) \geq \max_{\mathcal{X}_e \in \mathcal{T}} \text{score}(\mathcal{X}_e) &\geq \frac{1}{c(\mathcal{T})} \cdot \sum_{Z \in \mathcal{Z}} \sum_{\mathcal{X}_e \in \mathcal{T}_Z} \sum_{Y \in \mathcal{Z}} \left(p(L_e(Y)) + \sum_{\omega \in Y} p_\omega \cdot \frac{f_Y(X_e(\omega))}{Q_Y} \right) \\ &\geq \frac{1}{c(\mathcal{T})} \cdot \sum_{Z \in \mathcal{Z}} \sum_{\mathcal{X}_e \in \mathcal{T}_Z} \left(p(L_e(Z)) + \sum_{\omega \in Z} p_\omega \cdot \frac{f_Z(X_e(\omega))}{Q_Z} \right). \qquad (3.4)\end{aligned}$$

We now consider the two terms above (information and function gain) separately.

Bounding the information gain. We only consider *okay* sets Z.

$$\sum_{Z \in \mathcal{Z}} \sum_{\mathcal{X}_e \in \mathcal{T}_Z} p(L_e(Z)) \geq \sum_{Z \in \mathcal{Z}} p \left(\bigcup_{\mathcal{X}_e \in \mathcal{T}_Z} L_e(Z) \right) \geq \sum_{Z\text{: okay}} \frac{p(Z)}{2},$$

where the first inequality is by submodularity of the p-weighted coverage function and the second inequality is by Lemma 3.5.3.

Bounding the function gain. Consider any *good* set Z.

$$\sum_{\mathcal{X}_e \in \mathcal{T}_Z} \sum_{\omega \in Z} p_\omega \cdot \frac{f_Z(X_e(\omega))}{Q_Z} = \sum_{\omega \in Z} p_\omega \cdot \frac{\sum_{\mathcal{X}_e \in \mathcal{T}_Z} f_Z(X_e(\omega))}{Q_Z} \geq \sum_{\omega \in Z} p_\omega \cdot \frac{f_Z(T_Z(\omega))}{Q_Z} \geq \frac{p(Z)}{2},$$

where the first inequality is by submodularity of f_Z and the second inequality is by Lemma 3.5.2. Adding over all good sets Z,

$$\sum_{Z \in \mathcal{Z}} \sum_{\mathcal{X}_e \in \mathcal{T}_Z} \sum_{\omega \in Z} p_\omega \cdot \frac{f_Z(X_e(\omega))}{Q_Z} \geq \sum_{Z\text{: good}} \frac{p(Z)}{2}.$$

Combining the two bounds above,

$$\sum_{Z \in \mathcal{Z}} \sum_{\mathcal{X}_e \in \mathcal{T}_Z} \left(p(L_e(Z)) + \sum_{\omega \in Z} p_\omega \cdot \frac{f_Z(X_e(\omega))}{Q_Z} \right) \geq \sum_{Z\text{:okay or good}} \frac{p(Z)}{2} \geq \frac{u_i - 2u_i^*}{2},$$

where the last inequality is by Lemma 3.5.4. Combined with (3.4) and $c(\mathcal{T}) \leq \frac{1}{\delta} \cdot 2^i$, this completes the proof. $\square$

As Lemma 3.5.5 holds for each time t in phase i,

$$G = \sum_{t=\beta 2^{i-1}}^{\beta 2^i} \mathrm{score}(\mathcal{X}_{e(t)}) \geq \beta 2^{i-1} \cdot \delta \cdot \frac{1}{2^{i+1}} \cdot (u_i - 2u_i^*) \geq \frac{\beta\delta}{4} \cdot (u_i - 2u_i^*) \tag{3.5}$$

3.5.2 Upper bounding G

We now consider the implementation of the non-adaptive list L and calculate G as a sum of contributions over the decision path in the non-adaptive solution $\mathtt{NA}$. Let $\omega \in M$ denote the realized scenario, and consider the decision path under scenario ω. Let $\langle X_1, X_2, \ldots \rangle$ be the sequence of realizations (each in U) of items in L under scenario ω. So item $\mathcal{X}_j$ is selected between time $\sum_{\ell=1}^{j-1} c_\ell$ and $\sum_{\ell=1}^{j} c_\ell$. Let h (resp. p) index the first (resp. last) item (if any) that is selected (even partially) during phase i, i.e., between time $\beta 2^{i-1}$ and $\beta 2^i$. For each index $h \leq j \leq p$, let t_j denote the duration of time that item $\mathcal{X}_j$ is selected during in phase i; so t_j is the width of interval $[\sum_{\ell=1}^{j-1} c_\ell,\, \sum_{\ell=1}^{j} c_\ell] \bigcap [\beta \cdot 2^{i-1},\, \beta \cdot 2^i]$. Furthermore, for each index j, let $Z_j \subseteq M$ be the set of scenarios compatible with outcomes $\langle X_1, \cdots, X_{j-1} \rangle$. Note that $M = Z_1 \supseteq Z_2 \supseteq \cdots \supseteq Z_p \supseteq Z_{p+1}$. Define $G(\omega) := 0$ if index h is undefined (i.e., $\mathtt{NA}$ ends before phase i under scenario ω) and otherwise:

$$\begin{aligned} G(\omega) :=& \sum_{j=h}^{p} \frac{t_j}{c_j} \cdot \left(\mathbb{1}[\omega \in L_j(Z_j)] + \frac{f(\{X_1, \ldots, X_j\}) - f(\{X_1, \ldots, X_{j-1}\})}{Q - f(\{X_1, \ldots, X_{j-1}\})} \right) \\ \leq& \sum_{j=h}^{p} \left(\mathbb{1}[\omega \in L_j(Z_j)] + \frac{f(\{X_1, \ldots, X_j\}) - f(\{X_1, \ldots, X_{j-1}\})}{Q - f(\{X_1, \ldots, X_{j-1}\})} \right) \end{aligned} \tag{3.6}$$

By the stopping criterion for L, *before* the end of $\mathtt{NA}$ we must have (i) the number of compatible scenarios remains at least δs and (ii) the f value remains less than Q. In particular, we must have $|Z_p| \geq \delta s$.

We analyze the two quantities in the above expression separately. We begin by analyzing $\sum_{j=h}^{p} \mathbb{1}[\omega \in L_j(Z_j)]$. Fix any $h \leq j \leq p$. If the realized scenario $\omega \in L_j(Z_j)$, then the number of compatible scenarios drops by a factor of at least 2 upon observing X_j; that is,

$|Z_{j+1}| \leq \frac{1}{2}|Z_j|$. Using the fact that $|Z_1| = |M| = s$ and $|Z_p| \geq \delta s$, it follows that

$$\sum_{j=h}^{p} \mathbb{1}[\omega \in L_j(Z_j)] \leq \log_2 \frac{s}{\delta s} = \log_2(1/\delta).$$

Next, we analyze the second term in (3.6). Using the fact that f is integral, monotone and takes values in the range $[0, Q]$, we have:

$$\sum_{j=h}^{p} \frac{f(\{X_1, ..., X_j\}) - f(\{X_1, ..., X_{j-1}\})}{Q - f(\{X_1, ..., X_{j-1}\})} \ \leq \ \sum_{\ell=1}^{Q} \frac{1}{\ell} \ \leq \ \ln Q.$$

Thus, we have $G(\omega) \ \leq \ \log \frac{1}{\delta} + \ln Q \ \leq \ \log(Q/\delta)$.

Note that $G = \mathbb{E}_{\omega \sim M}[G(\omega)]$. It now follows that

$$G \ \leq \ \log(Q/\delta) \cdot \mathbf{P}(\texttt{NA} \text{ doesn't terminate before phase } i) \ = \ \log(Q/\delta) \cdot u_{i-1}. \qquad (3.7)$$

3.5.3 Completing proof of Lemma 3.4.3

Using (3.5), (3.7) and setting $\beta = \frac{16}{\delta} \log(Q/\delta)$, we get

$$\frac{16 \log(Q/\delta)}{4} \cdot (u_i - 2u_i^*) \ \leq \ \log(Q/\delta) \cdot u_{i-1}$$

which on rearranging gives $u_i \ \leq \ \frac{u_{i-1}}{4} + 2u_i^*$, as desired. This completes the proof of Theorem 3.1.1.

3.5.4 Tight approximation using more rounds

We now show that a better (and tight) approximation is achievable if we use $2r$ (instead of r) adaptive rounds. The main idea is to use the following variant of the partial covering problem, called *scenario submodular partial cover* (SSPC). The input is the same as scenario submodular cover: items $\mathcal{X}$, scenarios M with $|M| = s$ and submodular function f with maximal value Q. Given parameters $\delta, \varepsilon \in [0, 1]$, the goal in SSPC is to probe some items $\mathcal{R}$ that realize to set $R \subseteq U$ such that either (i) the number of compatible scenarios is less than

δs or (ii) the function value $f(R) > Q(1-\varepsilon)$. Unlike the partial version studied previously, we do not require f to be fully covered in case (ii). Note that setting $\varepsilon = \frac{1}{Q}$, we recover the previous partial covering problem; so SSPC is more general.

Corollary 3.5.6. *There is a non-adaptive algorithm for* SSPC *with cost* $O\left(\frac{1}{\delta}(\ln\frac{1}{\delta} + \ln\frac{1}{\varepsilon})\right)$ *times the optimal adaptive cost for the (full) submodular cover.*

Proof. The algorithm is nearly identical to SParCA (Algorithm 4). In Definition 3.4.1, we change

$$\mathcal{Z} := \{Y \in \mathcal{H}(\mathcal{S}) : |Y| \geq \delta s \textbf{ and } f(S(Y)) \leq Q(1-\varepsilon)\}.$$

In other words $\mathcal{Z}$ contains parts having size at least δs and for which the realized function value is at most the target $Q(1-\varepsilon)$. Note that scenarios in $\mathcal{Z}$ correspond to those under which the SSPC stopping rule does not already apply. After this, we use the same steps to produce list L in Algorithm 4. The only difference is in the stopping rule (when probing items from L) which reflects the definition of SSPC. In particular, the condition in the while-loop (step 7 of Algorithm 4) is now replaced by:

While $|H| \geq \delta|M|$ and $f(R) \leq Q(1-\varepsilon)$.

The analysis is also nearly identical: we prove Lemma 3.4.3 by lower/upper bounding the total gain G. The lower bound (3.5) remains the same. For the upper bound, the analysis for the first term in (3.6) remains the same, but its second term is now bounded as:

$$\sum_{j=h}^{p} \frac{V_j - V_{j-1}}{Q - V_{j-1}} \leq 1 + \sum_{j=h}^{p-1} \frac{V_j - V_{j-1}}{Q - V_{j-1}} \leq 1 + \sum_{\ell=\varepsilon Q}^{Q} \frac{1}{\ell} \leq 1 + \ln\frac{1}{\varepsilon},$$

where $V_j := f(\{X_1, ..., X_j\})$. The second inequality uses $V_1 \leq V_2 \leq \cdots V_{p-1} \leq Q(1-\varepsilon)$. This implies (just as before) that

$$G \leq \left(1 + \log\frac{1}{\delta} + \log\frac{1}{\varepsilon}\right) \cdot u_{i-1}.$$

Finally, choosing $\beta = \frac{16}{\delta}\left(1 + \log\frac{1}{\delta} + \log\frac{1}{\varepsilon}\right)$ and simplifying, we get $u_i \leq \frac{u_{i-1}}{4} + 2u_i^*$ (Lemma 3.4.3). This completes the proof. □

Next, we show how SSPC can be used iteratively to obtain a $2r$-round algorithm.

Theorem 3.5.7. *There is a $2r$-round adaptive algorithm for scenario submodular cover with cost $O\left(s^{1/r}\log(sQ)\right)$ times the optimal adaptive cost, where s is the number of scenarios.*

Proof. We use SSPC with $\delta = s^{-1/r}$ and $\varepsilon = Q^{-1/r}$ repeatedly for $2r$ adaptive rounds. Let $\rho = O(\frac{1}{r}s^{1/r}\log(sQ))$ denote the approximation ratio from Corollary 3.5.6 for these δ, ε values.

The *state* at any round consists of previously probed items and their realizations. For the analysis, we view the iterative algorithm as a $2r$ depth tree, where the nodes at depth i are the states in round i and the branches out of each node represent the observed realizations in that round. For any $i \in [2r]$, let Ω_i denote all the states in round i: note that these form a partition of the outcome space.

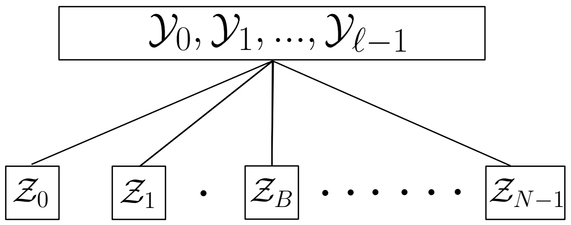
For any state $\omega \in \Omega_i$, let $\mathtt{OPT}(\omega)$ denote the expected cost of the optimal adaptive policy conditioned on the realizations in ω. By Corollary 3.5.6, conditioned on ω, the expected cost in round i is at most $\rho \cdot \mathtt{OPT}(\omega)$. Hence, the (unconditional) expected cost in round i is at most $\rho \sum_{\omega\in\Omega_i} \Pr[\omega] \cdot \mathtt{OPT}(\omega) = \rho \cdot \mathtt{OPT}$ where we used that Ω_i is a partition of all outcomes. So the expected cost of the $2r$-round algorithm is at most $2r \cdot \rho \cdot \mathtt{OPT}$ as claimed.

It remains to show that f is fully covered after $2r$ rounds. Note that after each round, we have one of the following (i) the number of compatible scenarios drops by factor $s^{1/r}$, or (ii) the residual target drops by factor $Q^{1/r}$. Clearly, case (i) can happen at most r times (as there is always some compatible scenario). So case (ii) occurs at least r times, which implies that the residual target is less than 1, i.e., f is fully covered. □

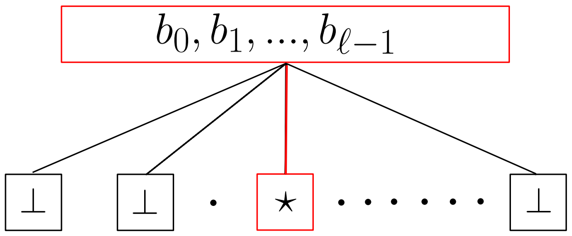
Setting $r = \log s$, we achieve a tight $O(\log(sQ))$ approximation in $O(\log s)$ rounds. Combined with the conversion to set-based rounds (Theorem 2.8.2), this proves Corollary 3.1.2.

3.6 Lower Bound for Scenario Submodular Cover

Recall that we obtained an r-round $O(s^{1/r} \cdot (\log s + r \log Q))$-approximation algorithm for scenario submodular cover. It is natural to ask if the dependence on the support-size s is necessary, given that the bound for the independent case only depends on Q. The main



(a) Set-up of the 1-round instance.



(b) We have $\mathcal{Y}_i = b_i$; pointing to item $\mathcal{Z}_B$.

Figure 3.3: The 1-round instance

result of this section is Theorem 3.1.3, which is a nearly matching lower bound on the r-round-adaptivity gap. In particular, it shows that the $s^{1/r}$ dependence is necessary even when $Q = 1$.

3.6.1 Hard Instances for Scenario Submodular Cover

1-round adaptivity gap. As a warm-up we first consider the case when $r = 1$. We design an instance to prove a 1-round adaptivity gap of $\Omega(s)$. The instances that we use to prove the adaptivity gap for $r > 1$ will build on this. The instance has groundset $U = \{0, 1, \star, \perp\}$ and function $f(S) = |S \cap \{\star\}|$ for $S \subseteq U$. Clearly, f is a $0-1$ valued monotone submodular function; and $Q = 1$. Let ℓ be an integer parameter and $N = 2^\ell$. We have two types of items: $\mathcal{Y}_0, ..., \mathcal{Y}_{\ell-1}$ each of which realizes to either 0 or 1 and costs 1, and N items $\mathcal{Z}_0, ..., \mathcal{Z}_{N-1}$ each costing ℓ and realizing to symbol $\star$ or $\perp$. The joint distribution $\mathcal{D}$ of these items has $s = N$ scenarios, each of uniform probability. For each $B \in \{0, 1, \cdots, N-1\}$, scenario B has the following realizations (below, $b_0 b_1 \cdots b_{\ell-1}$ is the binary representation of B).

$$Y_i(B) = b_i, \ \forall i = 0, \cdots, \ell-1 \quad \text{and} \quad Z_j(B) = \perp, \ \forall j \in \{0, \cdots, N-1\} \setminus \{B\} \text{ and } Z_B(B) = \star.$$

We say that item $\mathcal{Z}_k$ is *active* if, and only if, $\mathcal{Z}_k$ realizes to $\star$. Note that there is a unique active item in each scenario, and this item needs to be probed to cover f. See Figure 3.3 for a pictorial representation.

An adaptive solution is as follows. First, probe items $\mathcal{Y}_0, ..., \mathcal{Y}_{\ell-1}$, and based on their realizations calculate the number B represented by $Y_0, \cdots, Y_{\ell-1}$. Then, probe item $\mathcal{Z}_B$ (the active one) to cover f. This policy has cost 2ℓ. On the other hand, since each $\mathcal{Z}_k$ is

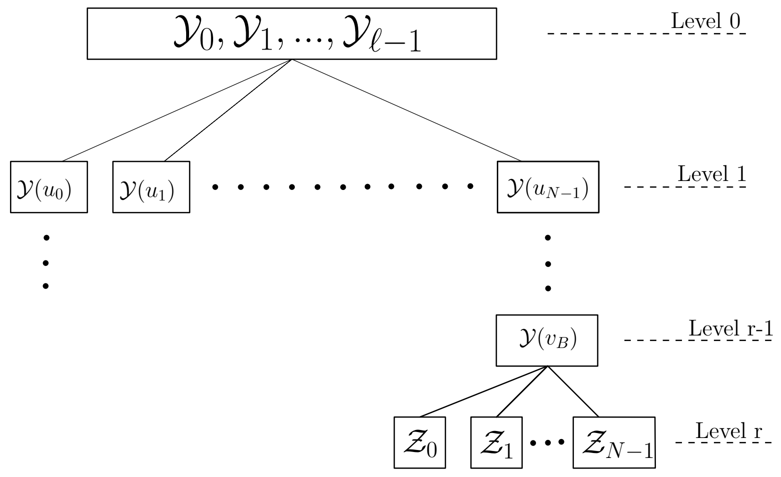


Figure 3.4: The r-round instance

equally likely to be active, any non-adaptive policy needs to pick $\frac{1}{2} \cdot 2^\ell = \Omega(2^\ell)$ $\mathcal{Z}$-items, in expectation, to cover f. This proves an $\Omega(2^\ell \ell/\ell) = \Omega(s)$ lower bound on the 1-round adaptivity gap for scenario submodular cover.

The r-round instance $\mathcal{I}_r$. Henceforth, we use $\mathcal{I}_r$ to denote the r-round instance. $\mathcal{I}_r$ is defined recursively by applying the above instance structure. Recall that ℓ is some parameter and $N = 2^\ell$. Moreover, the realizations of $\mathcal{Y}$-items above point to the unique $\mathcal{Z}$-item that is active. To extend this idea, we consider an N-ary tree $\mathcal{T}$ of depth r. Note that all leaves in $\mathcal{T}$ are at depth r. For each internal node, we index its N children by $\{0, \cdots, N-1\}$ in an arbitrary order. We define a collection of items using $\mathcal{T}$ as follows.

- For each depth $i = 0, \cdots, r-1$ and node $v \in \mathcal{T}$ at depth i, there are ℓ items $\mathcal{Y}_0(v), \cdots, \mathcal{Y}_{\ell-1}(v)$, each of unit cost and which realizes to 0 or 1.
- For each *leaf* node w of $\mathcal{T}$ (i.e., at depth r), there is an item $\mathcal{Z}_w$ of cost ℓ which realizes to one of the symbols $\star$ or $\perp$.

For any non-leaf node v, we use the shorthand $\mathcal{Y}(v) = \langle \mathcal{Y}_0(v), \cdots, \mathcal{Y}_{\ell-1}(v) \rangle$ to denote all its $\mathcal{Y}$-items. We illustrate the structure of $\mathcal{I}_r$ in Figure 3.4. The function f is as defined earlier: $f(S) = |S \cap \{\star\}|$. So $f(S) = 1$ if $\star \in S$ (else $f(S) = 0$). The distribution $\mathcal{D}$ involves $s = 2^{r\ell}$ scenarios with uniform probability. Each scenario is associated with a unique leaf node (i.e. $\mathcal{Z}$-item); note that the number of leaves is exactly s.

For each leaf w, we define its corresponding scenario as follows. Let $v_0, \cdots, v_{r-1}, v_r = w$ denote the path from the root of $\mathcal{T}$ to w. For each depth $i = 0, \cdots, r-1$, let node v_{i+1} be the B_i^{th} child of node v_i. Then, scenario w's realizations are:

- The realization $Y_0(v_i), \cdots, Y_{\ell-1}(v_i)$ of $\mathcal{Y}(v_i)$ equals the binary representation of B_i, for each depth $0 \le i \le r-1$. The realization of all other $\mathcal{Y}$-items is 0.
- The realization $Z_w = \star$ and $Z_u = \perp$ for all leaf $u \neq w$.

In this way, the realizations at the depth i node v_i point to the "correct" depth $i+1$ node v_{i+1} that should be probed next. This completes the description of the instance.

Adaptive solution. We now describe a good $(r+1)$-round adaptive solution OPT of cost $(r+1)\ell$. It is a natural top-down strategy. It starts by probing the ℓ items $\mathcal{Y}(v_0)$ at the root v_0, and based on their realization calculates the number B_0 represented by $Y_0(v_0), \cdots, Y_{\ell-1}(v_0)$; then it probes the $\mathcal{Y}$-items at the B_0^{th} child of v_0; and so on. After r rounds, it is able to discern the unique active item $\mathcal{Z}_w$ and probes it at cost ℓ (in the final round) to cover f.

3.6.2 Lower bound proof

In proving the lower bound, it is convenient to work with the *set-based* model of adaptivity. Recall that a set-based solution may fail (to cover function f) with some probability. Specifically, we will show that *any* set-based r-round solution for $\mathcal{I}_r$ which covers f with probability δ must cost at least $\delta \cdot \ell \cdot 2^\ell$. Then, using the contrapositive of Theorem 2.8.1 we obtain the desired lower bound in the *permutation-based* model.

We further restrict our attention to *batched* (set based) r-round solutions defined as follows.

Definition 3.6.1. *An r-round solution for $\mathcal{I}_r$ is a batched solution if in every round $k = 1, ..., r$, it has the property that for each node $v \in \mathcal{T}$, either all, or none, of the items at v are probed.*

A lower bound for batched r-round solutions implies a lower bound for *all* r-round solutions at a loss of a factor of ℓ (as each node contains at most ℓ items). The main result of this section is:

Lemma 3.6.1. *The expected cost of any batched r-round solution for $\mathcal{I}_r$ that covers f with probability $\delta > 0$ is at least $\delta \cdot \ell \cdot 2^\ell$.*

This implies that the expected cost of any (*unbatched*) r-round solution for the given instance is at least $\delta \cdot 2^\ell$. Setting $\delta = \frac{1}{2}$ and using Theorem 2.8.1, it follows that any permutation-based r-round solution must cost $\Omega\left(\frac{2^\ell}{r}\right)$. Combined with the fact that there is an adaptive solution of cost at most $2r\ell$, this implies an adaptivity gap of $\Omega\left(\frac{2^\ell}{r^2\ell}\right) = \Omega\left(\frac{s^{1/r}}{r\log s}\right)$ as $s = 2^{r\ell}$. This completes the proof of Theorem 3.1.3. Note that the above instance can also be viewed as one for optimal decision tree: by just ignoring the $\mathcal{Z}$-items, and with the goal of identifying the realized scenario.

Proof. Proof of Lemma 3.6.1 By Yao's minimax principle, it suffices to prove Lemma 3.6.1 for *deterministic* solutions. Let $\mathcal{A}$ be an arbitrary deterministic batched r-round solution for $\mathcal{I}_r$ that succeeds (i.e., covers f) with probability δ, and let $\texttt{NA}$ denote its (random) cost. We denote by random variables $\mathcal{S}_1, ..., \mathcal{S}_r$, the sets probed by $\mathcal{A}$ in each of its r rounds. We use $\sqcap_0, \cdots, \sqcap_r$ to denote the (random) nodes of $\mathcal{T}$ on the path from the root to the realized scenario $w^* = \sqcap_r$. We further assume that $\mathcal{A}$ satisfies the following properties:

1. At the beginning of each round k, $\mathcal{A}$ knows the identity of $\sqcap_{k-1}$.

2. In each round k, $\mathcal{A}$ probes $\sqcap_{k-1}$ for free and some subset of children of node $\sqcap_{k-1}$.

3. $\mathcal{A}$ is allowed to stop and claim "success" if it discerns $\sqcap_{k+1}$ at the end of any round k; note that this occurs exactly when $\mathcal{A}$ probes node $\sqcap_k$ in round k.

We formalize and justify these assumptions in Lemma 3.6.2, where we argue that the aforementioned assumptions can be ensured while neither increasing the cost of $\mathcal{A}$ nor decreasing $\mathcal{A}$'s success probability. We first complete the proof of Lemma 3.6.1 assuming properties 1-3.

We proceed by induction on r. When $r = 1$, *any* solution with success probability δ must probe at least δN leaf nodes, which implies its cost is at least $\delta \cdot \ell N$ (recall every leaf-item costs ℓ).

Now, suppose that $r > 1$. Note that since $\mathcal{A}$ is deterministic and has no information in the first round, $\mathcal{S}_1$ is a fixed subset, irrespective of the realized scenario. By property (2),

$\mathcal{S}_1$ only contains items from levels 0 and 1. Recall that $N = 2^\ell$. Let $\epsilon_1 N$ denote the number of depth-1 nodes in $\mathcal{S}_1$. Since $\sqcap_1$ is equally likely to be any of the 2^ℓ children of $\sqcap_0$, we have the following two cases:

- Suppose $\sqcap_1$ is in $\mathcal{S}_1$, i.e., it is probed in round 1. Then, $\mathcal{A}$ discerns $\sqcap_2$ and hence it claims "success" and stops. Note that the conditional success probability in this case is 1.

- Suppose $\sqcap_1$ is not in $\mathcal{S}_1$. Then, conditional on the realizations of $\mathcal{S}_1$, $\mathcal{A}$ discerns $\sqcap_1$ (but not $\sqcap_2$). In this case, the remaining $r-1$ rounds of $\mathcal{A}$ correspond to a solution for instance $\mathcal{I}_{r-1}$. Let γ be the (conditional) success probability in this case.

The first case above occurs w.p. ϵ_1 and the second case occurs w.p. $1-\epsilon_1$. So, the overall success probability of $\mathcal{A}$ is $\epsilon_1 + (1-\epsilon_1)\cdot\gamma = \delta$. It follows that $\gamma = \frac{\delta-\epsilon_1}{1-\epsilon_1}$. We are now ready to lower-bound the expected cost of $\mathcal{A}$. The cost of $\mathcal{A}$ in round 1 is $\ell \cdot |\mathcal{S}_1| = \epsilon_1 \ell N$ as the total cost at each node is ℓ. For the remaining $r-1$ rounds, the cost is 0 in the first case above. In the second case, using induction, the expected cost from the remaining $r-1$ rounds is at least $\gamma \cdot \ell N$. So, the expected cost of $\mathcal{A}$ is at least

$$\epsilon_1 \cdot \ell N + (1-\epsilon_1)\cdot\gamma\cdot\ell N \;=\; \delta\cdot\ell N,$$

which proves the desired result. Note that we assumed $\delta \geq \epsilon_1$ above; otherwise, the inductive statement trivially holds. $\square$

Lemma 3.6.2. *Given a batched r-round solution $\mathcal{A}$ for $\mathcal{I}_r$ that covers f with probability δ, there exists a batched r-round solution $\mathcal{A}'$ that succeeds with probability at least δ, has lower expected cost than $\mathcal{A}$, and has the following properties:*

1. *At the beginning of each round k, $\mathcal{A}'$ knows the identity of $\sqcap_{k-1}$.*

2. *In each round k, $\mathcal{A}'$ probes $\sqcap_{k-1}$ for free and some subset of children of node $\sqcap_{k-1}$.*

3. *$\mathcal{A}'$ stops and claims "success" if it discerns $\sqcap_{k+1}$ at the end of any round k; note that this occurs exactly when $\mathcal{A}$ probes node $\sqcap_k$ in round k.*

Proof. Proof of Lemma 3.6.2 Let $\mathcal{A}$ be an arbitrary batched solution. Recall the setting from the proof of Lemma 3.6.1. We denote by random variables $\mathcal{S}_1, ..., \mathcal{S}_r$, the sets probed by $\mathcal{A}$ in each of its r rounds. We use $\sqcap_0, \cdots, \sqcap_r$ to denote the (random) nodes of $\mathcal{T}$ on the path from the root to the realized scenario $w^* = \sqcap_r$. For any node v, we use $\Pi_k(v)$ to denote the depth-k ancestor of node v; if v is at depth k, then $\Pi_k(v) = v$. We convert $\mathcal{A}$ to $\mathcal{A}'$ inductively as follows.

Suppose that we have ensured $\mathcal{A}'$ satisfies the three properties for the first $k-1$ rounds. So, at the start of round k, $\mathcal{A}'$ knows the identity of $\sqcap_{k-1}$. The nodes probed by $\mathcal{A}'$ in round k are $\mathcal{S}'_k = \{\sqcap_{k-1}\} \cup \{\Pi_k(v) : v \in \mathcal{S}_k\}$. Clearly, A' is a batched r-round solution that satisfies the three properties for round $k+1$. Also, allowing $\mathcal{A}'$ to probe $\sqcap_{k-1}$ for free in round k only reduces the cost of $\mathcal{A}'$ compared to the cost of $\mathcal{A}$.

To complete the proof, we now argue that whenever $\mathcal{A}$ covers function f, solution $\mathcal{A}'$ claims success and stops. Suppose that $\mathcal{A}$ covers f is some round $k \in [r]$. Then, it must be the case that $\mathcal{A}$ probes $\sqcap_r$ in $\mathcal{S}_k$. Accordingly, $\mathcal{A}'$ would probe $\Pi_k(\sqcap_r) = \sqcap_k$, after which it discerns $\sqcap_{k+1}$. So $\mathcal{A}'$ succeeds after round k as well. Thus, $\mathcal{A}'$ succeeds with probability at least δ. □

3.7 Applications

3.7.1 Optimal Decision Tree

The optimal decision tree problem captures problems from a variety of fields such as learning theory, medical diagnosis, pattern recognition, and boolean logic; see the surveys [89] and [91]. In an instance of optimal decision tree (`ODT`) we are given s hypotheses with probabilities $\{p_i\}_{i=1}^s$. An unknown random hypothesis $y^* \in [s]$ is drawn from this distribution. There is a collection of m tests, where test e costs $c_e \in \mathbb{R}_+$ and returns a positive result if y^* lies in some subset $T_e \subseteq [s]$ and a negative result if $y^* \notin T_e$. The goal is to identify y^* using tests of minimum expected cost.

We can transform any `ODT` instance into an instance of scenario submodular cover. We associate scenarios with hypotheses and the distribution $\mathcal{D}$ is given by probabilities $\{p_i\}_{i=1}^s$.

For each test e, we have an item $\mathcal{X}_e$ of cost c_e. The groundset $U = \{e^+, e^- : e \text{ test}\}$ which corresponds to all possible (individual) test results. Item $\mathcal{X}_e$ realizes to e^+ (resp. e^-) if test e is positive (resp. negative). If y^* is the realized scenario then the item realizations correspond to the test outcomes for hypothesis y^*. For each test e, define subsets $T(e^-) = T_e$ and $T(e^+) = [s] \setminus T_e$. The submodular function is $f(S) = \min\left\{|\bigcup_{v\in S} T(v)|\,,\, s-1\right\}$ for $S \subseteq U$. Note that $f(S)$ is the number of incompatible hypotheses given the tests results S. Moreover, $Q = s-1$. Applying Theorem 3.1.1, for any $r \geq 1$, we obtain an r-round adaptive algorithm that costs at most $O(rs^{1/r}\log s)$ times the fully adaptive optimum. Our lower bound for scenario submodualar cover (Theorem 3.1.3) also implies an $\Omega(\frac{1}{r\log s}s^{1/r})$ bound on the r-round-adaptivity gap for `ODT`. So, for any constant $r \geq 1$, our r-round-adaptivity gap is tight up to an $O(\log^2 s)$ factor. Moreover, using Corollary 3.1.2, we obtain an $O(\log s)$ round algorithm of cost $O(\log s)$ times the adaptive optimum. As shown in [27], $O(\log s)$ is the best possible approximation ratio, even for fully adaptive algorithms.

Application to Medical Diagnosis. Recall our motivating application in medical diagnosis where we know s possible conditions a patient may suffer from (along with the priors of their occurrence), and our goal is to adaptively perform tests to identify the correct condition as quickly (and cheaply) as possible. This problem, known as automated diagnosis, can be directly cast as the optimal decision tree problem. See, e.g. [10] for applications of decision tree methods to diagnostic problems. See also [21] for an application of `ODT` in emergency response. Here, a first responder observes the symptoms of a victim of chemical exposure in order to identify the toxic chemical.

Application to Active Learning. In a typical (binary) classification problem, there is a X set of data points, each of which is associated with a $+$ or $-$ label. There is also a set $\mathcal{H}$ of hypotheses, where each hypothesis provides a $+/-$ labeling of the data points. It is assumed that the true classifier/hypothesis h^* belongs to $\mathcal{H}$. In the average-case setting that we consider, there is also a Bayesian prior $p_{\mathcal{H}}(\cdot)$ for the true hypothesis, i.e., $\Pr[h^* = h] = p_{\mathcal{H}}(h)$ for all $h \in \mathcal{H}$. The learner needs to identify the true hypothesis h^*. In *active learning*, the learner can query the label of any point $e \in X$ by incurring some cost c_e (which corresponds to some expert labeling e). The goal is to identify h^* by querying points (possibly adaptively) at the minimum expected cost. See, e.g. [37, 60, 35] for more details on this approach. This is exactly an instance of optimal decision tree, where the data points correspond to tests. Applying Theorem 3.1.1, for any $r \geq 1$, we obtain an r-round adaptive algorithm that costs at most $O(r|\mathcal{H}|^{1/r}\log(|\mathcal{H}|))$ times the fully adaptive optimum.

3.7.2 Correlated Knapsack Cover

There are n items, each of cost c_i and random (integer) reward $\mathcal{X}_i$. The rewards may be correlated and are given by a joint distribution $\mathcal{D}$ with s scenarios. The exact reward of an item is only known when it is probed. Given a target Q, the goal is to probe some items so that the total realized reward is at least Q and we minimize the expected cost. This is a special case of scenario submodular cover. The groundset $U = \{(i, v) : i \in [n], 0 \leq v \leq Q\}$ where element (i, v) represents item $\mathcal{X}_i$ realizing to v. Any realization of value at least Q is treated as equal to Q: this is because the target is itself Q. For each element $e = (i, v) \in U$ let $a_e = v$ denote its value. Then, the submodular function is $f(S) = \min\{\sum_{e\in S} a_e\, , Q\}$ for $S \subseteq U$. By Theorem 3.1.1, we obtain an r-round adaptive algorithm with cost at most $O(s^{1/r}(\log s + r\log Q))$ times the optimal adaptive cost.

3.8 Computational Results

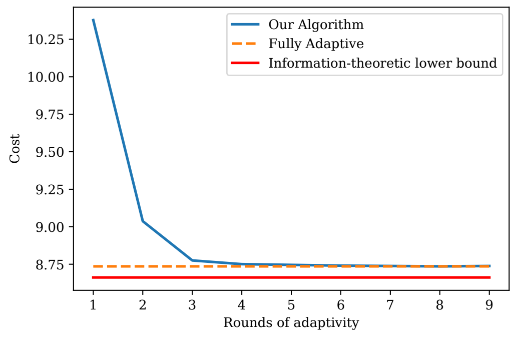
We provide a summary of computational results of our r-round adaptive algorithms for the stochastic set cover and optimal decision tree problems. We conducted all experiments using Python 3.8 and Gurobi 8.1 with a 2.3 Ghz Intel Core $i5$ processor and 16 GB 2133 MHz LPDDR3 memory.

3.8.1 Optimal Decision Tree

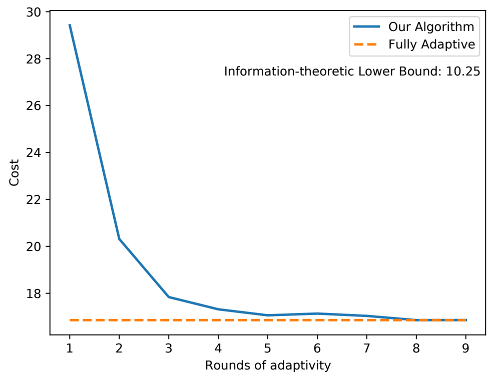
Instances. We use a real-world dataset called WISER (`http://wiser.nlm.nih.gov/`) for our experiments. The WISER dataset describes symptoms that one may suffer from after being exposed to certain chemicals. It contains data corresponding to 415 chemicals (scenarios for ODT) and 79 symptoms (elements with binary outcomes). Given a patient exhibiting certain symptoms, the goal is to identify the chemical that the patient has been exposed to (by testing as few symptoms as possible). This dataset has been used for evaluating algorithms for similar problems in other papers [24, 20, 21, 92]. For each symptom-chemical pair, the data specifies whether or not someone exposed to the chemical exhibits the given symptom. However, the WISER data has 'unknown' entries for some pairs. In order to obtain instances for ODT from this, we generate 10 different datasets by assigning random binary values to the 'unknown' entries. Then we remove all identical scenarios: to ensure that the ODT instance is feasible. We use the uniform probability distribution for the scenarios. Given these 10 datasets, we consider two cases. The first case (called $\mathtt{WISER-U}$) has all tests with unit costs. In the second case, we generate costs randomly for each test from $\{1, 4, 7, 10\}$ according to the probabilities $[0.1, 0.2, 0.4, 0.3]$; for example, with probability 0.4, a test is assigned cost 7. Varying cost captures the setting where tests may have different costs, and we may not want to schedule an expensive test without sufficient information. We refer to this case as $\mathtt{WISER-R}$.

We also test our algorithm on synthetic data which we generate as follows. We set $s = 10,000$ and $m = 100$. For each $y \in [s]$, we randomly generate a binary sequence of outcomes which corresponds to how y reacts to the tests. We do this in two ways: for test e, we set $y \in T_e$ with probability $p \in \{0.2, 0.5\}$. If a sequence of outcomes is repeated, we discard the scenario to ensure feasibility of the ODT instance. We assign equal probability to each scenario. We generate instances using (i) unit costs or (ii) random costs from $\{1, 4, 7, 10\}$ with respective probabilities $[0.1, 0.2, 0.4, 0.3]$. Thus, we generate 4 types of instances with synthetic data. We refer to the instance generated with $p = 0.2$ and unit costs as $\mathtt{SYN-U-}0.2$. Other instances are named similarly.

Results. We test our r-round adaptive algorithm on all of the above mentioned datasets. We vary r over integers in the interval $[1, \log s]$. Again, we compare to our *fully-adaptive* algorithm which adapts after every probe: in each step, this algorithm probes the item that maximizes the score (3.1) where $\mathcal{S} = \emptyset$ and $\mathcal{Z}$ consists of a single part with all scenarios. We also implemented the fully-adaptive "generalized binary search" algorithm due to [37]



(a) $\texttt{WISER} - \texttt{U}$



(b) $\texttt{WISER} - \texttt{R}$

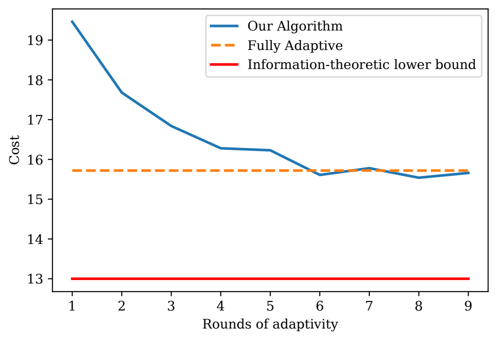
Figure 3.5: Computational results for $\texttt{ODT}$ on WISER dataset

as an algorithmic benchmark: on the instances considered, this algorithm performed nearly identically to our fully-adaptive algorithm; so, we exclude it from the plots.

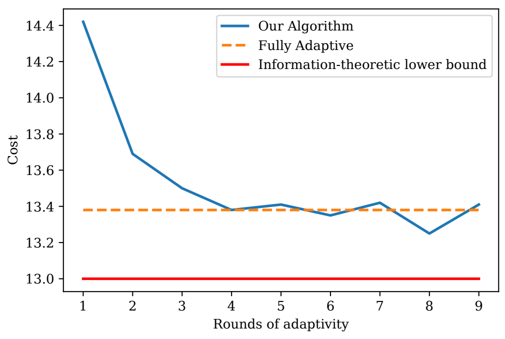
We also compare to information-theoretic lower bounds obtained as follows. For instances with unit costs ($\texttt{WISER} - \texttt{U}$, $\texttt{SYN} - \texttt{U} - 0.2$ and $\texttt{SYN} - \texttt{U} - 0.5$) this lower bound corresponds to the entropy which is $\log_2(s)$, where s is the number of scenarios. Recall that all instances have uniform probabilities across scenarios. For instances with non-uniform costs ($\texttt{WISER} - \texttt{R}$, $\texttt{SYN} - \texttt{R} - 0.2$ and $\texttt{SYN} - \texttt{R} - 0.5$) , the entropy is no longer a valid lower bound. Instead, we use an integer programming (IP) formulation to compute the optimal cost to identify a given scenario (see §3.9 for details) and then average over all scenarios. We note that there are instances where this information-theoretic lower bound is smaller than the optimal adaptive cost by an $O(s)$ factor.

For the WISER datasets, we compute averages using *all* scenarios. Figure 3.5a plots our algorithm's expected cost as a function of r for $\texttt{WISER} - \texttt{U}$. We observe that our algorithm gets very close to the information-theoretic lower bound in only 3 rounds of adaptivity. Figure 3.5b plots our algorithm's costs for $\texttt{WISER} - \texttt{R}$. Here, we observe a sharp decrease in costs within 4 rounds of adaptivity, after which our algorithm's cost is within $\approx 50\%$ of the information-theoretic lower bound. Note that we only plot results on our first dataset for $\texttt{WISER} - \texttt{U}$ and $\texttt{WISER} - \texttt{R}$. We include plots for all 10 WISER datasets in Appendix 3.10.

For the synthetic data, we compute averages by sampling scenarios over 100 trials (since $s = 10000$, computing expectation over all s would be very slow). We plot the results in



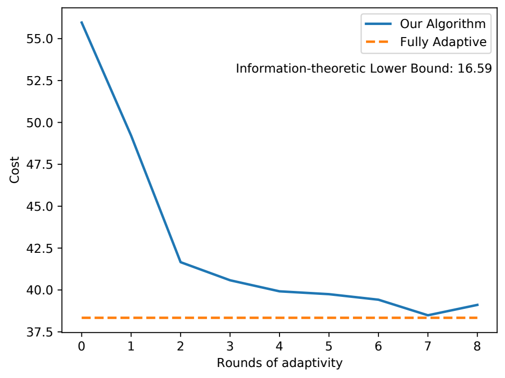
(a) $\texttt{SYN} - \texttt{U} - 0.2$



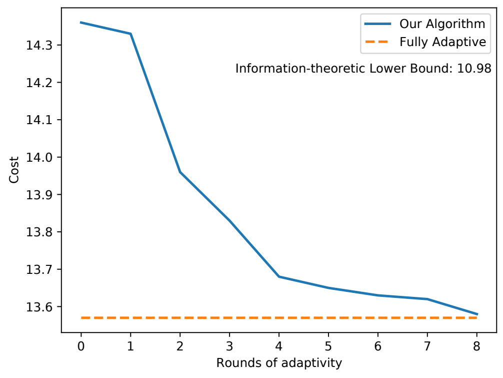
(b) $\texttt{SYN} - \texttt{U} - 0.5$

Figure 3.6: Computational results for ODT on synthetic data with unit costs

Figures 3.6 and 3.7. We observe that with 6 rounds of adaptivity, our algorithm's cost nearly matches that of the fully-adaptive algorithm.



(a) $\texttt{SYN} - \texttt{R} - 0.2$



(b) $\texttt{SYN} - \texttt{R} - 0.5$

Figure 3.7: Computational results for ODT on synthetic data with non-uniform costs

3.9 An Information-Theoretic Lower Bound for ODT

In this section, we present an information theoretic lower bound for ODT instances. In an instance of the ODT problem, we are given s hypotheses with probabilities $\{p_i\}_{i=1}^{s}$. An unknown random hypothesis $y^* \in [s]$ is drawn from this distribution (known as the true realization). Additionally, there is a collection of m tests where each test e costs $c_e \in \mathbb{R}_+$ and returns a positive result if $y^* \in T_e \subseteq [s]$, and a negative result otherwise. The goal is to identify y^* with minimum expected cost.

Let $\mathcal{Y}$ be a random variable denoting the true realization. We define set $S_{y^*}(y)$ as the set of tests that distinguish y from y^*. Formally, $S_{y^*}(y) = \{e : R_e(y) \neq R_e(y^*)\}$. where $R_e(y)$ denotes the result returned by test e under hypothesis y. Observe that to identify y^*, we must perform at least one test from $S_{y^*}(y)$ for all $y \neq y^*$. The following integer program (IP) computes the minimum cost needed to conclude that $\mathcal{Y} = y^*$.

$$\begin{aligned} \texttt{LB}(y^*) = \text{minimize} \quad & \sum_{e=1}^{m} c_e \cdot x_e \\ \text{subject to} \quad & \sum_{e \in S_{y^*}(y)} x_e \geq 1, \quad \text{for all } y \neq y^*, \qquad (3.8) \\ & x_e \in \{0,1\}, \quad e \in [m], \end{aligned}$$

where x_e, for $e \in [m]$, is a binary variable denoting whether test e is performed. The constraints denoted by (3.8) ensure that the selected tests simultaneously eliminate all hypotheses $y \neq y^*$.

Note that $\texttt{LB}(y^*)$ denotes the minimum cost needed to identify hypothesis y^*. Then, $\texttt{LB} = \mathbb{E}[\texttt{LB}(\mathcal{Y})] = \sum_{y^*} \texttt{LB}(y^*) \cdot \mathbf{P}_{y^*}$ is an information-theoretic lower bound for the given instance of ODT. We note that the above IP only provides a *lower bound* on the cost for identifying a fixed hypothesis; it is not a formulation for the given ODT problem itself. Also note that this gives us an information-theoretic lower bound for ODT instances with both non-uniform costs and non-uniform probabilities.

For the special case of uniform costs, we use $\log_2(s)$ as a lower bound. This lower bound follows from Shannon's source coding theorem (see, e.g., [101] or, [36], Chapter 5), and holds even when we have tests that allow us to distinguish between *any* partition of the hypotheses. In this case, the IP formulation will incur a cost of 1 to identify y^* (by performing the test that distinguishes y^* from the rest) leading to a trivial lower bound.

3.10 Additional Plots

In the main body of the paper, we only include plots related to the first dataset for the `WISER − U` and `WISER − R` cases. The trends observed in the other datasets are similar. We include plots for all 10 datasets here for completeness.

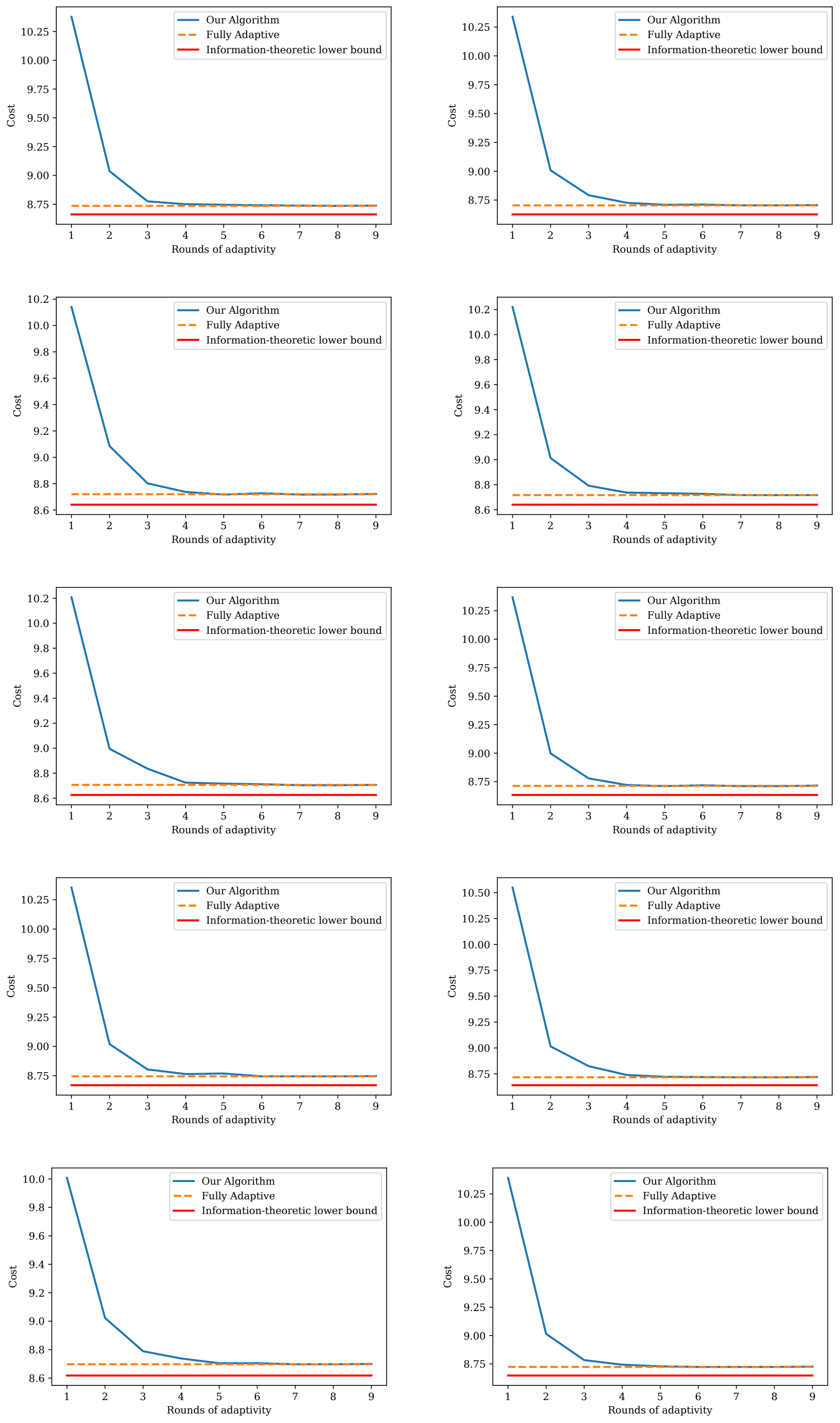


Figure 3.8: Computational results for $\texttt{ODT}$ on $\texttt{WISER} - \texttt{U}$

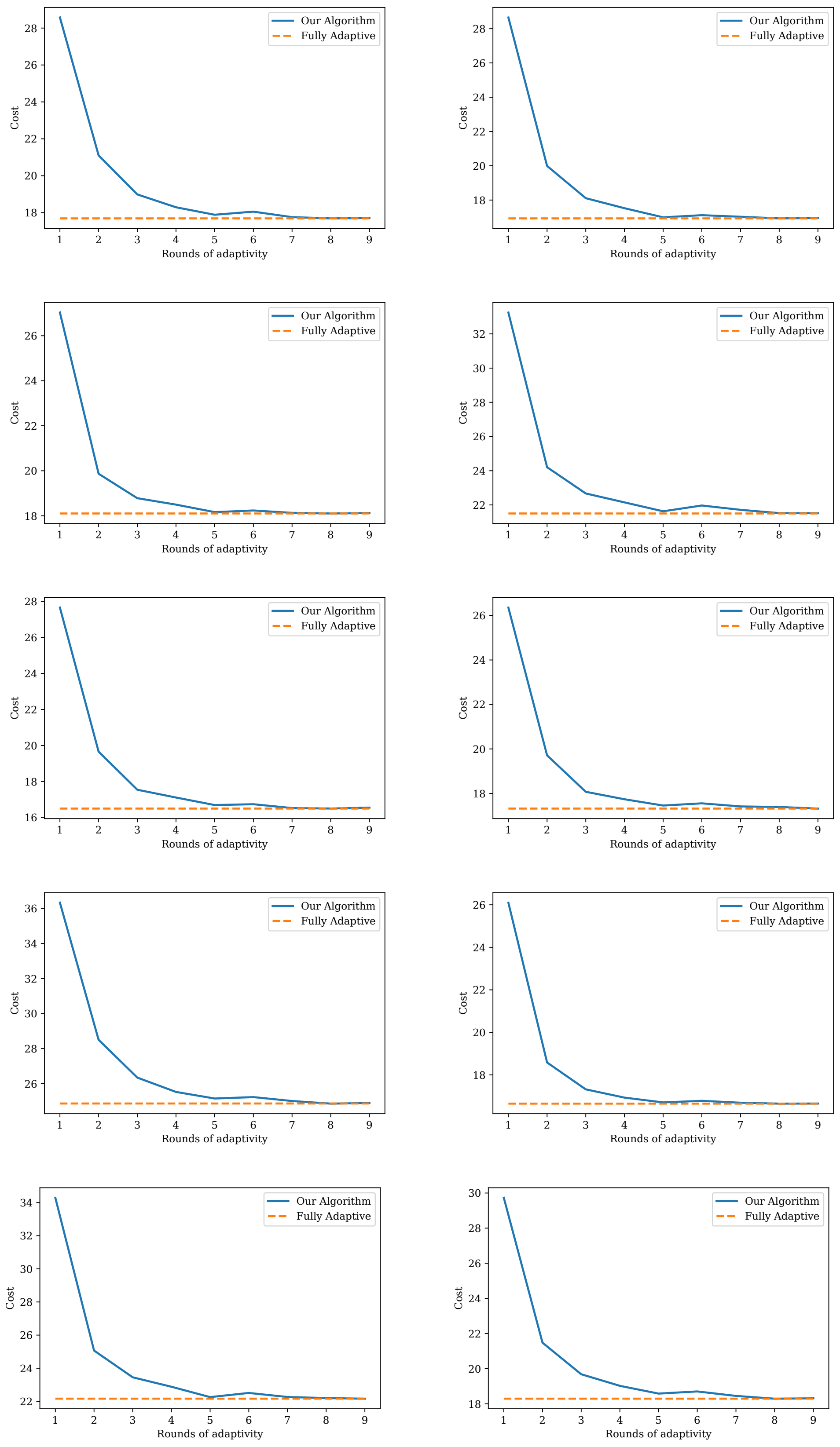


Figure 3.9: Computational results for ODT on WISER − R

Chapter 4

Stochastic Score Classification

4.1 Introduction

The problem of diagnosing complex systems often involves running a large number of tests for each component of such a system. One option to diagnose such systems is to perform tests on *all* components, which can be prohibitively expensive and slow. Therefore, we are interested in a policy that tests components one by one, and minimizes the average cost of testing. (See [107] for a survey.) Concretely, we consider a setting where the goal is to test various components, in order to assign a risk class to the system (e.g., whether the system has low/medium/high risk).

The *stochastic score classification* (`SSClass`) problem introduced by [53] models such situations. There are n components in a system, where each component i is "working" with independent probability p_i. While the probabilities p_i are known *a priori*, the random outcomes $X_i \in \{0, 1\}$ are initially unknown. The outcome X_i of each component i can be determined by performing a test of cost c_i: $X_i = 1$ if i is working and $X_i = 0$ otherwise. The overall status of the system is determined by a linear score $r(X) := \sum_{i=1}^{n} a_i X_i$, where the coefficients $a_i \in \mathbb{Z}$ are input parameters. We are also given a collection of intervals $I_1, I_2, \ldots, I_k$ that partition the real line (i.e., all possible scores). The goal is to determine the interval I_j (also called the *class*) that contains $r(X)$, while incurring minimum expected cost. A well-studied special case is when there are just two classes, which corresponds to evaluating a halfspace or linear-threshold-function [40].

Example: consider a system which must be assigned a risk class of *low*, *medium*, or *high*. Suppose there are five components in the system, each of which is working with probability $\frac{1}{2}$. The score for the entire system is the number of working components. A score of 5 corresponds to the "Low" risk class, scores between 2 and 4 correspond to "Medium" risk, and a score of at most 1 signifies "High" risk. Suppose that after testing components $\{1, 2, 3\}$, the system has score 2 (which occurs with probability $\frac{3}{8}$) then it will be classified as medium risk irrespective of the remaining two components: so testing can be stopped. Instead, if the system has score 3 after testing components $\{1, 2, 3\}$ (which occurs with probability $\frac{1}{8}$) then the class of the system cannot be determined with certainty (it may be either medium or low), and so further testing is needed.

A related problem is the *d-dimensional stochastic score classification problem* (d-`SSClass`), which models the situation when a system has d different functions, each with an associated linear score (as above). We must now perform tests on the underlying components to *simultaneously* assign a class to each of the d functions.

In another related problem, a system again has d different functions. Here, the status (working or failed) of each function is determined by some halfspace, and the overall system is considered operational if all d functions are working. The goal of a diagnosing policy is to decide whether the system is operational, and if not, to return at least one function that has failed (and therefore needs maintenance). This is a special case of a problem we call *explainable stochastic halfspace evaluation* (`EX-SFE`).

Solutions for all these problems (`SSClass`, d-`SSClass`, and `EX-SFE`) are *sequential decision processes.* At each step, a component is tested and its outcome (working or failed) is observed. The information from all previously tested components can then be used to decide on the next component to test; this makes the process *adaptive.* This process continues until the risk class can be determined with certainty from the tested components. One simple class of solutions are *non-adaptive* solutions, which are simply described by a priority list: we then test components in this fixed order until the class can be uniquely determined. Such solutions are simpler and faster to implement, compared to their adaptive counterparts: the non-adaptive testing sequence needs to be constructed just once, after which it can be used for all input realizations. However, non-adaptive solutions are weaker than adaptive ones, and our goal is to bound the *adaptivity gap*, the multiplicative ratio between the performance of the non-adaptive solution to that of the optimal adaptive one. Our main result shows that `SSClass` has a constant adaptivity gap, thereby answering an open question posed by [53]. Additionally, we show an adaptivity gap of $O(d^2 \log d)$ for both the d-`SSClass` and `EX-SFE` problems.

4.1.1 Problem Definitions

Before we present the results and techniques, let us formally define the problems. For any integer m, we use $[m] := \{1, 2, \ldots, m\}$.

Stochastic Score Classification. An instance of `SSClass` consists of n independent $\{0, 1\}$ random variables $X = X_1, \ldots, X_n$, where variable X_i has $\Pr[X_i = 1] = \mathbb{E}[X_i] = p_i$. The cost to probe/query X_i is $c_i \in \mathbb{R}_+$; both p_i, c_i are known to us. We are also given non-negative weights $a_i \in \mathbb{Z}_+$, and the *score* of the outcome $X = (X_1, \ldots, X_n)$ is $r(X) = \sum_{i=1}^n a_i X_i$. In addition, we are given $B+1$ integers $\alpha_1, \ldots, \alpha_{B+1}$ such that *class* j corresponds to the interval $I_j := \{\alpha_j, \ldots, \alpha_{j+1} - 1\}$. The *score classification function* $h : \{0, 1\}^n \to \{1, \ldots, B\}$ assigns $h(X) = j$ precisely when $r(X) \in I_j$. The goal is to determine $h(X)$ at minimum expected cost. We assume non-negative weights only for simplicity: any instance with positive and negative weights can be reduced to an equivalent instance with all positive weights (see Appendix 4.7). Let $W := \sum_{i=1}^n a_i$ denote the total weight. In our algorithm, we associate two numbers $(\beta_j^0, \beta_j^1) \in \mathbb{Z}_+^2$ with each class j, where $\beta_j^0 = W - \alpha_{j+1} + 1$ and $\beta_j^1 = \alpha_j$.

d-Dimensional Stochastic Score Classification. This is a natural generalization of `SSClass` to $d \geq 2$ score functions. An instance of d-`SSClass` is the same as that for `SSClass` but where the weights are given by vectors $\mathbf{a_i} \in \mathbb{Z}_+^d$. This results in a d-dimensional *score function* $r(X) = \sum_{i=1}^n \mathbf{a}_i X_i$; let $r_k(X)$ denote the k^{th} component of $r(X)$. The input also specifies intervals $\{I_j^{(k)}\}_j$ for each of these dimensions $k \in [d]$, which in turn define *score classification functions* $\{h_k\}_{k\in[d]}$, such that $h_k(X) = j$ if $r_k(X) \in I_j^{(k)}$. The goal is to design a strategy to *simultaneously* evaluate $h_1, \ldots, h_d$ with minimum expected cost.

Explainable Stochastic Halfspace Evaluation. Finally, an instance of EX-SFE is similar to that of d-SSClass, but where each dimension has only two intervals $I_0^{(k)}, I_1^{(k)}$ specified by a single threshold (i.e., $I_0^{(k)} = \{y < \alpha_k\}$ and $I_1^{(k)} = \{y \geq \alpha_k\}$). This means that the score classification functions $h_k : \{0,1\}^n \to \{0,1\}$ are Boolean-valued. Note that h_k corresponds to evaluating the *halfspace* $\sum_{i=1}^n a_{ki} X_i \geq \alpha_k$. Moreover, we are also given an *aggregation function* $f : \{0,1\}^d \to \{0,1\}$, and we want to evaluate $f(h_1(X), \ldots, h_d(X))$. In other words, if we define the composite score classification function $h(X) := (h_1(X), h_2(X), \ldots, h_d(X))$, we want to evaluate $(f \circ h)(X)$.

Moreover, the problem asks for an "explanation", or a witness of this evaluation. To define a witness, consider a tuple $(X', v(X'), T)$, where $X' \subseteq X$ is the subset of probed variables, $v(X')$ are the realizations for these variables, and $T \subseteq [d]$ is a subset of the dimensions. This tuple $(X', v(X'), T)$ is a *witness* for $f \circ h$ if the following conditions are satisfied:

- The realizations $v(X')$ of the probed variables X' determine $h_k(X)$ for all dimensions $k \in T$. In other words, we can evaluate all the halfspaces corresponding to subset $T \subseteq [d]$.
- The values of $\{h_k(X)\}_{k \in T}$ completely determine $(f \circ h)(X)$; i.e., all realizations of the remaining variables $X \setminus X'$ give the same value for $f \circ h$.

In other words, a witness is a proof of the function value that *only* makes use of probed variables. The goal is to design a probing strategy of minimum expected cost that determines $f \circ h$ along with a witness (as defined above). We assume that checking whether a given tuple is a witness can be done efficiently; see §4.5 for a discussion.

For example, when $f(y_1, \cdots y_d) = \bigwedge_{k=1}^d y_k$, EX-SFE corresponds to evaluating the intersection of d halfspaces; the witness either confirms that all halfspaces are satisfied or identifies some violated halfspace. EX-SFE also captures more general cases: given a target $\bar{d} \leq d$, we can check whether at least $\bar{d}$ out of d halfspaces are satisfied (using an appropriate f).

4.1.2 Results and Techniques

Our main result is the following algorithm (and adaptivity gap).

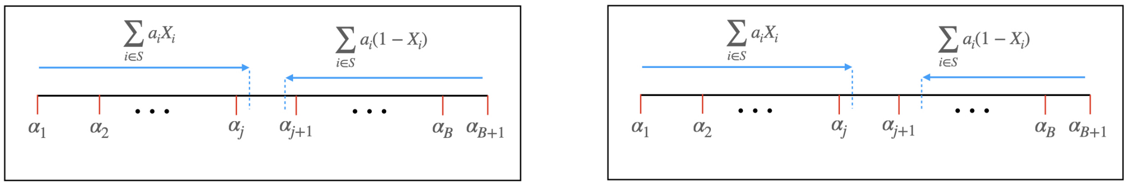
Theorem 4.1.1. *There is a polynomial-time non-adaptive algorithm (called* NaCl*) for stochastic score classification with expected cost at most a constant factor times that of the optimal adaptive policy.*

This result improves on the prior work from [53, 40] in several ways. Firstly, we get a constant-factor approximation, improving upon the previous $O(\log W)$ and $O(B)$ ratios, where W is the sum of weights, and B the number of classes. Secondly, our algorithm is non-adaptive in contrast to the previous adaptive ones. Finally, our algorithm has nearly-linear runtime, which is faster than the previous algorithms.

An added benefit of our approach is that we obtain a "universal" solution that is simultaneously $O(1)$-approximate for all class-partitions. Indeed, the non-adaptive list produced by NaCl only depends on the probabilities, costs, and weights, and not on the class boundaries $\{\alpha_j\}$; these α_j values are only needed in the stopping condition for probing.

Algorithm overview. To motivate our algorithm, suppose that we have probed a subset $S \subseteq [n]$ of variables, and there is a class j such that $\sum_{i\in S} a_i X_i \geq \alpha_j$ and $\sum_{i\in S} a_i(1-X_i) \geq W - \alpha_{j+1} + 1$. The latter condition can be rewritten as $\sum_{i\in S} a_i X_i + \sum_{i\notin S} a_i \leq \alpha_{j+1} - 1$. So, we can conclude that the final score lies in $\{\alpha_j, \dots, \alpha_{j+1} - 1\}$ irrespective of the outcomes of variables in $[n] \setminus S$. This means that $h(X) = j$. On the other hand, if the above condition is not satisfied for *any* class j, we must continue probing. Towards this end, we define two types of rewards for each variable $i \in [n]$: $R_0(i) = a_i \cdot (1 - X_i)$ and $R_1(i) = a_i \cdot X_i$. (See Figure 4.1.) The total R_0-reward and R_1-reward from the probed variables correspond to upper and lower bounds on the score, respectively. Our non-adaptive algorithm NaCl probes the variables in a predetermined order until $\sum_{i\in S} R_0(i) \geq \beta_j^0 = W - \alpha_{j+1} + 1$ and $\sum_{i\in S} R_1(i) \geq \beta_j^1 = \alpha_j$ for *some* class j (at which point it determines $h(X) = j$).

To get this ordering we first build two separate lists: list L_b for the R_b-rewards (for $b = 0, 1$) minimizes the cost required to cover some target amount of R_b-reward. Finally, interleaving lists L_0 and L_1 gives the final list. The idea behind list L_0 is as follows: if we only care about a *single* class j, we can set a target of β_j^0 and use the non-adaptive algorithm for stochastic knapsack cover [70]. Since the class j is unknown, so is the target β_j^0 on the R_0-reward. Interestingly, we show how to construct a "universal" non-adaptive list L_0 that



$\sum_{i\in S} a_i X_i$
$\sum_{i\in S} a_i(1-X_i)$
α_1 α_2 α_j α_{j+1} α_B α_{B+1}
$\sum_{i\in S} a_i X_i$
$\sum_{i\in S} a_i(1-X_i)$
α_1 α_2 α_j α_{j+1} α_B α_{B+1}

(a) In this case, the R_0 reward (upper bound) and R_1 reward (lower bound) lie in class j and hence probing can be stopped.

(b) Here $f(X)$ could be j or $j+1$, so probing must continue.

Figure 4.1: Illustration of non-adaptive approach

works for *all* targets simultaneously. The construction proceeds in phases: in each phase $\ell \geq 0$, the algorithm adds a subset of variables with cost $O(2^\ell)$ that (roughly) maximizes the *expected* R_0 reward. Naıvely using the expected rewards can lead to poor performance, so a natural idea is to use rewards *truncated* at logarithmically-many scales (corresponding to the residual target); see for example, [42]. Moreover, to get a constant-factor approximation, we use the *critical scaling* idea from [70]. Roughly speaking, this identifies a *single* scale κ such that with constant probability (1) the algorithm obtains large reward (truncated at scale κ), and (2) any subset of cost 2^ℓ has small reward.

Analysis overview. The analysis of Theorem 4.1.1 relates the "non-completion" probabilities of our algorithm after cost $\gamma\cdot 2^\ell$ to that of the optimal adaptive algorithm after cost 2^ℓ, for each phase $\ell \geq 0$. The factor γ corresponds to the approximation ratio of the algorithm. In order to relate these non-completion probabilities, we consider the R_0 and R_1-rewards obtained by an optimal adaptive algorithm, and argue that the non-adaptive algorithm obtains a higher R_0 *as well as* R_1 reward (with constant probability). Thus, if the optimal adaptive algorithm decides $h(X)$, so does the non-adaptive algorithm. Our algorithm/analysis also use various properties of the fractional knapsack problem.

Extensions. Next, we consider the d-dimensional score classification problem (d-`SSClass`) and obtain the following result:

Theorem 4.1.2. *There is a non-adaptive $O(d^2 \log d)$-approximation algorithm for d-dimensional stochastic score classification.*

We achieve this by extending the above approach (for $d = 1$). We now define two rewards (corresponding to R_0 and R_1) for *each* dimension d. Then, we apply the list building algorithm for each of these rewards, resulting in $2d$ separate lists. Finally, we interleave these lists to obtain the non-adaptive probing sequence. The analysis is also an extension of the $d = 1$ case. The main differences are as follows. Just accounting for the $2d$ lists results in an extra $O(d)$ factor in the approximation. Furthermore, we need to ensure (for each phase ℓ) that with constant probability, our non-adaptive algorithm achieves more reward than the optimum for *all* the $2d$ rewards. We incur another $O(d \log d)$ factor in the approximation in order to achieve this stronger property.

In a similar vein, our main result for `EX-SFE` is the following:

Theorem 4.1.3. *There is a non-adaptive $O(d^2 \log d)$-approximation algorithm for explainable stochastic halfspace evaluation.*

The non-adaptive list for `EX-SFE` is constructed in the same manner as for d-`SSClass`, but the stopping rule relies on the oracle for verifying witnesses of $f \circ h$. As a special case, we obtain:

Corollary 4.1.4. *There is a non-adaptive $O(d^2 \log d)$-approximation algorithm for the explainable stochastic intersection of half-spaces problem.*

The stochastic intersection of halfspaces problem (in a slightly different model) was studied previously by [25], where an $O(\sqrt{n \log d})$-approximation algorithm was obtained assuming all probabilities $p_i = \frac{1}{2}$. The main difference from our model is that [25] do not require a witness at the end. So their policy can stop if it concludes that there exists a violated halfspace (even without knowing which one), whereas our policy can only stop after it identifies a violated halfspace (or determines that all halfspaces are satisfied). We note that our approximation ratio is independent of the number of variables n and holds for arbitrary probabilities.

Computational Results. Finally, we evaluate the empirical performance of our algorithm for score classification. In these experiments, our non-adaptive algorithm performs nearly as well as the previous-best *adaptive* algorithms, while being an order of magnitude faster.

In fact, on many instances, our algorithm provides an *improvement* in both the cost as well as the running time. On most instances, the cost of our algorithm is within 50% of an information-theoretic lower bound on the optimal value.

4.1.3 Related Work

The special case of `SSClass` with $B = 2$ classes is the well-studied stochastic Boolean function evaluation for *linear threshold functions* ($\mathcal{SBFT}$). Here, the goal is to identify whether a single halfspace is satisfied (i.e., the score is above or below a threshold). [40] gave an elegant 3-approximation algorithm for $\mathcal{SBFT}$ using an adaptive dual greedy approach. Prior to their work, only an $O(\log W)$-approximation was known, based on the more general stochastic submodular cover problem [67, 56].

The general `SSClass` problem was introduced by [53], who showed that it can be formulated as an instance of stochastic submodular cover. Then, using general results such as [67, 56], they obtained an adaptive $O(\log W)$-approximation algorithm. Furthermore, [53] obtained an adaptive $3(B-1)$-approximation algorithm for `SSClass` by extending the approach of [40] for $\mathcal{SBFT}$; recall that B is the number of classes. [53] also studied the *unweighted* special case of `SSClass` (where all weights $a_i = 1$) and gave a slightly better $(B-1)$-approximation algorithm. For the further special case with unit weights and costs, they obtained a non-adaptive 4-approximation algorithm. A main open question from this work was the possibility of a constant approximation for the general `SSClass` problem. We answer this in the affirmative. Moreover, our algorithm is non-adaptive: so we also bound the adaptivity gap.

The stochastic knapsack cover problem ($\mathcal{SKC}$) is closely related to $\mathcal{SBFT}$. Given a set of items with random rewards and a target k, the goal is to (adaptively) select a subset of items having total reward at least k. The objective is to minimize the expected cost of selected items. [40] gave an adaptive 3-approximation algorithm for $\mathcal{SKC}$. Later, [70] gave a *non-adaptive* $O(1)$-approximation algorithm for $\mathcal{SKC}$. In fact, the result in [70] applied to the more general stochastic k-TSP problem [42]. Our algorithm and analysis use some ideas from [70, 42]. We use the notion of a "critical scale" from [70] to identify the correct reward truncation threshold. The approach of using non-completion probabilities in the analysis is similar to [42]. There are also a number of differences: we exploit additional structure in the (fractional) knapsack problem and obtain a simpler and nearly-linear time algorithm.

More generally, non-adaptive solutions (and adaptivity gaps) have been used in solving various other stochastic optimization problems such as max-knapsack [38, 23], matching [15, 19], probing [62, 64] and orienteering [59, 61, 16]. Our result shows that this approach is also useful for `SSClass`.

`SSClass` and `EX-SFE` also fall under the umbrella of designing query strategies for "priced information", where one wants to evaluate a function by sequentially querying variables (that have costs). There are two lines of work here: comparing to an optimal strategy (as in our model) [73, 7, 53, 25], and comparing to the min-cost solution in hindsight (i.e., competitive analysis) [29, 33, 34]. We note that the "explainable" requirement in the `EX-SFE` problem (that we solve) is similar to the requirement in [73].

Organization. The rest of the paper is organized as follows. We state some preliminaries necessary for our proofs in §4.2. In §4.3, we formally state our non-adaptive algorithm for stochastic score classification and prove Theorem 4.1.1. In §4.4 and §4.5, we consider the d-`SSClass` and `EX-SFE` problems. We present our computational results in §4.6.

4.2 Preliminaries

We first state some basic results for the deterministic knapsack problem. In an instance of the knapsack problem, we are given a set T of items with non-negative costs $\{c_i : i \in T\}$ and rewards $\{r_i : i \in T\}$, and a budget D on the total cost. The goal is to select a subset of items of total cost at most D that maximizes the total reward. The LP relaxation is the following:

$$g(D) = \max\left\{\textstyle\sum_{i\in T} r_i \cdot x_i \;\middle|\; \sum_{i\in T} c_i \cdot x_i \le D,\, x \in [0,1]^T\right\}, \qquad \forall D \ge 0.$$

The following algorithm $\mathcal{A}_{\mathcal{KS}}$ solves the *fractional* knapsack problem and also obtains an approximate integral solution. Assume that the items are ordered so that $\frac{r_1}{c_1} \ge \frac{r_2}{c_2} \ge \cdots$. Let t index the first item (if any) so that $\sum_{i=1}^{t} c_i \ge D$. Let $\psi := \frac{1}{c_t}(D - \sum_{i=1}^{t-1} c_i)$ which lies in $(0,1]$. Define

$$x_i = \begin{cases} 1 & \text{if } i \leq t-1 \\ \psi & \text{if } i = t \\ 0 & \text{if } i \geq t+1 \end{cases}.$$

Return x as the optimal fractional solution and $Q = \{1, \cdots, t\}$ as an integer solution. We prove the following fact in the appendix for completeness.

Theorem 4.2.1. *Consider algorithm $\mathcal{A}_{\mathcal{KS}}$ on any instance of the knapsack problem with budget D.*

1. *$\langle r, x\rangle = \sum_{i=1}^{t-1} r_i + \psi \cdot r_t = g(D)$ and so x is an optimal LP solution.*
2. *The derivative $g'(D) = \frac{r_t}{c_t}$.*
3. *Solution Q has cost $c(Q) \leq D + c_{max}$ and reward $r(Q) \geq g(D)$.*
4. *$g(D)$ is a concave function of D.*

4.3 The Stochastic Score Classification Algorithm

Our non-adaptive algorithm creates two lists L_0 and L_1 separately. These lists are based on the R_0 and R_1 rewards of the variables, where $R_0(i) = a_i(1 - X_i)$ and $R_1(i) = a_i X_i$. It interleaves lists L_0 and L_1 together (by power-of-2 costs) and then probes the variables in this non-adaptive order until the class is identified.

4.3.1 The Algorithm

We first explain how to build the lists L_0 and L_1. We only consider list L_0 below (the algorithm/analysis for L_1 are identical). The list building algorithm operates in phases. For each phase $\ell \geq 0$ it gets a budget of $O(2^\ell)$, and it solves several instances of the deterministic knapsack problem, where rewards are truncated expectations of R_0. We will use the following truncation values, also called *scales*.

$$\mathcal{G} := \left\{\theta^\ell \ : \ 0 \leq \ell \leq 1 + \log_\theta W\right\}, \text{ where } \theta > 1 \text{ is a constant.}$$

For each scale $\tau \in \mathcal{G}$, we find a deterministic knapsack solution with reward $\mathbb{E}[\min\{R_0/\tau, 1\}]$ (see Equation 4.1 for the formal definition) and budget $\approx C2^\ell$ (where $C > 1$ is a constant). Including solutions for each scale would lead to an $O(\log W)$ loss in the approximation factor. Instead, as in [70], we identify a "critical scale" and only include solutions based on the critical scale. To this end, each scale τ is classified as either *rich* or *poor*. Roughly, in a rich scale, the knapsack solution after budget $C2^\ell$ still has large "incremental" reward (formalized by the derivative of g being at least some constant δ). The *critical scale* is the smallest scale κ that is poor, and so represents a transition from rich to poor. For our analysis, we will choose constant parameters C, δ and θ so that $\frac{C\delta}{\theta} > 1$. We note however, that our algorithm achieves a constant approximation ratio for any constant values $C > 1$, $\delta \in (0, 1)$ and $\theta > 0$.

Algorithm 6 PickReps($\ell, \tau, \mathbf{r}$)

```
1: let $T \subseteq [n]$ denote the variables with non-zero reward and cost at most $2^\ell$
2: run algorithm $\mathcal{A}_{\mathcal{KS}}$ (Theorem 4.2.1) on the knapsack instance with items $T$ and budget
   $D = C2^\ell$
3: let $f = g'(D)$ be the derivative of the LP value and $Q \subseteq T$ the integral solution from
   $\mathcal{A}_{\mathcal{KS}}$
4: if $f > \delta 2^{-\ell}$ then
5:     scale $\tau$ is rich
6: else
7:     scale $\tau$ is poor
8: return $Q$
```

Subroutine PickReps (Algorithm 6) computes the knapsack solution for each scale τ, and classifies the scale as rich/poor. The subroutine BuildList(R) (Algorithm 7) builds the list for any set of random rewards $\{R(i) : i \in [n]\}$. List L_b (for $b = 0, 1$) is obtained by running BuildList(R_b). Finally, the non-adaptive algorithm NaCl involves interleaving the variables in lists L_0 and L_1; this is described in Algorithm 8. The resulting policy probes variables in the order given by NaCl until the observed upper and lower bounds on the score lie within the same class. Note that there are $O(\log(nc_{max}))$ phases and $O(\log W)$ scales: so the total number of deterministic knapsack instances solved is poly-logarithmic. Moreover, the knapsack algorithm $\mathcal{A}_{\mathcal{KS}}$ runs in $O(n \log n)$ time. So the overall runtime of our algorithm is nearly linear.

Algorithm 7 BuildList($\{R(i) : i \in [n]\}$)

```
1: list $\Pi \leftarrow \emptyset$
2: for phase $\ell = 0, 1, \ldots$ do
3:     for each scale $\tau \in \mathcal{G}$ do
4:         define truncated rewards
               $$r_i^\tau = \begin{cases} \mathbb{E}\left[\min\left\{\frac{R(i)}{\tau}, 1\right\}\right], & \text{if } i \notin \Pi \\ 0, & \text{otherwise} \end{cases} \qquad (4.1)$$
5:         $S_{\ell,\tau} \leftarrow$ PickReps($\ell, \tau, \mathbf{r}^\tau$)
6:     let $\kappa$ be the smallest poor scale in $\mathcal{G}$ (this is called the critical scale)
7:     $\Pi_\ell \leftarrow S_{\ell,\kappa}$ and $\Pi \leftarrow \Pi \circ \Pi_\ell$
8: return list $\Pi$
```

Algorithm 8 NaCl (Non-Adaptive Classifier)

```
1: list $L_b \leftarrow$ BuildList($\{R_b(i) : i \in [n]\}$) for $b = 0, 1$
2: let $L_b^\ell$ denote the variables in phase $\ell$ for list $L_b$
3: for each phase $\ell$, set $S_\ell \leftarrow L_0^\ell \cup L_1^\ell$
4: return list $S_0, S_1, \cdots, S_\ell \cdots$
```

4.3.2 The Analysis

Lemma 4.3.1. *The critical scale κ in Step 6 of Algorithm 7 is always well defined.*

Proof. To prove that there is a smallest poor scale, it suffices to show that not all scales can be rich. We claim that the last scale $\tau \geq W$ cannot be rich. Suppose (for a contradiction) that scale τ is rich. Then, by concavity of g (see property 4 in Theorem 4.2.1), we have $g(D) \geq D \cdot g'(D) > D \cdot \delta 2^{-\ell} = C\delta \geq 1$. On the other hand, the total deterministic reward at this scale, $\sum_{i=1}^n r_i^\tau \leq \frac{W}{\tau} \leq 1$. Thus, $g(D) \leq 1$, a contradiction. $\square$

Lemma 4.3.2. *The cost $c(S_{\ell,\tau}) \leq (C+1)2^\ell$ for any phase ℓ. Hence, the cost incurred in phase ℓ of* NaCl *is at most $(C+1)2^{\ell+1}$.*

Proof. Consider any call to PickReps in phase ℓ. We have $S_{\ell,\tau} = Q$ where Q is the integer solution from Theorem 4.2.1. It follows that $c(S_{\ell,\tau}) = c(Q) \leq C2^\ell + \max_{i \in T} c_i \leq (C+1)2^\ell$; note that we only consider variables of cost at most 2^ℓ (see Step 1 of Algorithm 6). Finally, the variables S_ℓ in phase ℓ of NaCl consist of the phase-ℓ variables of both L_0 and L_1. So the total cost of these variables is at most $(C+1)2^{\ell+1}$. $\square$

We now analyze the cost incurred by our non-adaptive strategy NaCl. We denote by `OPT` an optimal adaptive solution for `SSClass`. To analyze the algorithm, we use the following notation.

- u_ℓ: probability that NaCl is not complete by end of phase ℓ.
- u^*_ℓ: probability that `OPT` costs at least 2^ℓ.

We can assume by scaling that the minimum cost is 1. So $u^*_0 = 1$. For ease of notation, we use `OPT` and `NA` to denote the *random* cost incurred by `OPT` and NaCl respectively. We also divide `OPT` into phases: phase ℓ corresponds to variables in `OPT` after which the cumulative cost is between $2^{\ell-1}$ and 2^ℓ. The following lemma forms the crux of the analysis.

Lemma 4.3.3. *For any phase $\ell \geq 1$, we have $u_\ell \leq q \cdot u_{\ell-1} + u^*_\ell$ where $q \leq 0.3$.*

Before we prove this lemma, we use it to prove Theorem 4.1.1.

Proof of Theorem 4.1.1. This proof is fairly standard, see e.g., [42]. By the description of the non-adaptive algorithm, the cost until end of *phase* ℓ (in NaCl) is at most $(C+1)\sum_{j=0}^{\ell} 2^{j+1} \leq 4(C+1)\cdot 2^\ell$. Let $\gamma = 4(C+1)$ below. Moreover, NaCl ends in phase ℓ with probability $(u_{\ell-1} - u_\ell)$. As a consequence of this, we have

$$\mathbb{E}[\mathtt{NA}] \;\leq\; \gamma \cdot (1 - u_0) + \sum_{\ell \geq 1} \gamma \cdot 2^\ell \cdot (u_{\ell-1} - u_\ell) = \; \gamma + \gamma \cdot \sum_{\ell \geq 0} 2^\ell u_\ell. \tag{4.2}$$

Similarly, we can bound the cost of the optimal adaptive algorithm as

$$\mathbb{E}[\mathtt{OPT}] \geq \sum_{\ell \geq 0} 2^\ell (u^*_\ell - u^*_{\ell+1}) \geq u^*_0 + \frac{1}{2} \cdot \sum_{\ell \geq 1} 2^\ell u^*_\ell = 1 + \frac{1}{2} \cdot \sum_{\ell \geq 1} 2^\ell u^*_\ell, \tag{4.3}$$

where the final equality uses the fact that $u_0^* = 1$. Define $\Gamma := \sum_{\ell \geq 0} 2^\ell u_\ell$. We have

$$\begin{aligned}
\Gamma = \sum_{\ell \geq 0} 2^\ell u_\ell &\leq u_0 + q \cdot \sum_{\ell \geq 1} 2^\ell \cdot u_{\ell-1} + \sum_{\ell \geq 1} 2^\ell \cdot u_\ell^* \\
&\leq u_0 + q \cdot \sum_{\ell \geq 1} 2^\ell u_{\ell-1} + 2 \cdot (\mathbb{E}[\texttt{OPT}] - 1) \\
&= u_0 + 2q \cdot \Big(\sum_{\ell \geq 0} 2^\ell u_\ell \Big) + 2 \cdot (\mathbb{E}[\texttt{OPT}] - 1) \\
&\leq \; 2q \cdot \Gamma + 2\,\mathbb{E}[\texttt{OPT}] - 1,
\end{aligned}$$

where the first inequality follows from Lemma 4.3.3, the second inequality from (4.3), and the last inequality from the fact that $u_0 \leq 1$. Thus, $\Gamma \leq \frac{2}{1-2q} \cdot \mathbb{E}[\texttt{OPT}] - 1$. From (4.2), we conclude $\mathbb{E}[\texttt{NA}] \leq \frac{2\gamma}{1-2q} \cdot \mathbb{E}[\texttt{OPT}]$. Setting $C = O(1)$ and $q = 0.3$ completes the proof. □

4.3.3 Proof of Lemma 4.3.3

Recall that NaCl denotes the non-adaptive algorithm, and NA its random cost. Fix any phase $\ell \geq 1$, and let σ denote the realization of the variables probed in the first $\ell - 1$ phases of NaCl. We further define the following *conditioned on* σ:

- $u_\ell(\sigma)$: probability that NaCl is not complete by end of phase ℓ.
- $u_\ell^*(\sigma)$: probability that OPT costs at least 2^ℓ, i.e., OPT is not complete by end of phase ℓ.

If NaCl does not complete before phase ℓ then $u_{\ell-1}(\sigma) = 1$, and we will prove

$$u_\ell(\sigma) \leq u_\ell^*(\sigma) + 0.3. \tag{4.4}$$

We can complete the proof using this. Note that $u_{\ell-1}(\sigma)$ is either 0 or 1. If $u_{\ell-1}(\sigma) = 0$ then $u_\ell(\sigma) = 0$ as well. So, Equation (4.4) implies that $u_\ell(\sigma) \leq u_\ell^*(\sigma) + 0.3 u_{\ell-1}(\sigma)$ for all σ. Taking expectation over σ gives Lemma 4.3.3. It remains to prove Equation (4.4).

We denote by $\mathcal{R}_0$ and $\mathcal{R}_0^*$ the total R_0 reward obtained in the first ℓ phases by NaCl and OPT respectively. We similarly define $\mathcal{R}_1$ and $\mathcal{R}_1^*$. To prove Equation (4.4), we will show that the probabilities (conditioned on σ) $\mathbf{P}(\mathcal{R}_0^* > \mathcal{R}_0)$ and $\mathbf{P}(\mathcal{R}_1^* > \mathcal{R}_1)$ are small. Intuitively, this implies that with high probability, if `OPT` finishes in phase ℓ, then so does NaCl. Formally, we prove the following key lemma.

Lemma 4.3.4 (Key Lemma)**.** *For $b \in \{0,1\}$, we have $\mathbf{P}(\mathcal{R}_b < \mathcal{R}_b^* \mid \sigma) \leq 0.15$.*

Using these lemmas, we prove Equation (4.4).

Proof of Equation (4.4)*.* Recall that we associate a pair (β_j^0, β_j^1) with every class j. If `OPT` finishes in phase ℓ, then there exists some j such that $\mathcal{R}_0^* \geq \beta_j^0$ and $\mathcal{R}_1^* \geq \beta_j^1$. Thus,

$$\mathbf{P}(\texttt{OPT} \text{ finishes in phase } \ell \mid \sigma) = 1 - u_\ell^*(\sigma) = \mathbf{P}(\exists j : \mathcal{R}_0^* \geq \beta_j^0 \text{ and } \mathcal{R}_1^* \geq \beta_j^1 \mid \sigma).$$

From Lemma 4.3.4 and union bound, we have $\mathbf{P}(\mathcal{R}_0 < \mathcal{R}_0^* \text{ or } \mathcal{R}_1 < \mathcal{R}_1^* \mid \sigma) \leq 0.3$. Then, we have

$$\begin{aligned}
1 - u_\ell(\sigma) &= \mathbf{P}(\texttt{NA} \text{ finishes in phase } \ell \mid \sigma) \\
&\geq \mathbf{P}\left((\texttt{OPT} \text{ finishes in phase } \ell) \bigwedge \mathcal{R}_0 \geq \mathcal{R}_0^* \bigwedge \mathcal{R}_1 \geq \mathcal{R}_1^* \,\middle|\, \sigma\right) \\
&\geq \mathbf{P}(\texttt{OPT} \text{ finishes in phase } \ell \mid \sigma) - \mathbf{P}(\mathcal{R}_0 < \mathcal{R}_0^* \text{ or } \mathcal{R}_1 < \mathcal{R}_1^* \mid \sigma) \\
&\geq (1 - u_\ell^*(\sigma)) - 0.3
\end{aligned}$$

Upon rearranging, this gives $u_\ell(\sigma) \leq u_\ell^*(\sigma) + 0.3$ as desired. □

4.3.4 Proof of the Key Lemma

We now present the proof of Lemma 4.3.4 with $b = 0$ (the case $b = 1$ is identical). Henceforth, reward will only refer to R_0. For ease of notation, let σ also represent the set of variables probed in the first $\ell - 1$ phases. Observe that in phase ℓ of Algorithm 7, the previously probed variables $\Pi \subseteq \sigma$. (Note that Π only includes variables in L_0 whereas σ includes variables in both L_0 and L_1.)

Overview of proof. Recall that S_ℓ is the set of variables probed by NaCl in phase ℓ. Let O_ℓ be the variables probed by `OPT` in phase ℓ; so the total cost of O_ℓ is at most 2^ℓ. Note that

O_ℓ is a random subset as `OPT` is adaptive. On the other hand, S_ℓ is a deterministic subset as NACL is a non-adaptive list. We will show that (conditioned on σ) the probability that O_ℓ has more reward than S_ℓ is small. The key idea is to use the *critical scale* κ (in phase ℓ) to argue that the following hold with constant probability (1) reward of $O_\ell \setminus (S_\ell \cup \sigma)$ is at most κ, and (2) reward of $S_\ell \setminus O_\ell$ is at least κ. This would imply that with constant probability, NACL gets at least as much reward as `OPT` by the end of phase ℓ.

We note here that the probabilities and expectations in this proof are conditioned on σ. For ease of notation, we will drop this conditioning henceforth. For any subset $S \subseteq [n]$ of variables, we use $R_0(S) := \sum_{i \in S} R_0(i)$ to denote the total observed reward in S. We also use κ^- to be the scale immediately preceding the critical scale κ. (If $\kappa = 1$ then κ^- is undefined, and all steps involving κ^- can be ignored.)

Upper bounding reward of $O_\ell \setminus (S_\ell \cup \sigma)$. Define new rewards:

$$U_i = \begin{cases} \min\left\{\frac{R_0(i)}{\kappa}, 1\right\}, & \text{if } i \notin \sigma \cup S_\ell \\ 0, & \text{otherwise} \end{cases} \tag{4.5}$$

Note that $\mathbb{E}[U_i] = r_i^\kappa$ for all $i \notin \sigma \cup S_\ell$. Basically, we use the truncated rewards and ignore the variables probed up to phase ℓ. We now show that any *adaptive* policy that selects variables of cost 2^ℓ (outside $S_\ell \cup \sigma$) cannot get too much reward.

Lemma 4.3.5. *If $\mathcal{A}$ is any* adaptive *policy of selecting variables from $[n] \setminus (S_\ell \cup \sigma)$ with total cost $\leq 2^\ell$ then $\mathbf{P}[R_0(\mathcal{A}) < \kappa] \geq 1 - \delta$. Hence, $\mathbf{P}[R_0(O_\ell \setminus (S_\ell \cup \sigma)) \geq \kappa] \leq \delta$.*

Proof. Consider Algorithm 6 for scale κ; see also Theorem 4.2.1 for solving the knapsack problem. Subset $T \subseteq [n] \setminus \sigma$ consists of variables i with reward $r_i^\kappa > 0$ and cost $c_i \leq 2^\ell$. The variables in T are ordered in non-increasing reward-to-cost ratio. Subset $Q = \{1, 2, \cdots t\} \subseteq T$ is the minimal prefix of T with total cost $\geq D = C2^\ell$. As κ is a poor scale, the derivative $g'(D) = \frac{r_t^\kappa}{c_t} \leq \delta 2^{-\ell}$. We now claim:

$$\text{Any subset } O \subseteq [n] \setminus (Q \cup \sigma) \text{ with cost } c(O) \leq 2^\ell \text{ has reward } r^\kappa(O) \leq \delta. \tag{4.6}$$

Clearly, every variable in O has cost at most 2^ℓ: so $O \subseteq T \setminus Q$. Using the fact that Q is a *prefix* of T in decreasing reward-to-cost order, we have

$$\frac{r^\kappa(O)}{c(O)} \le \min_{i\in Q} \frac{r_i^\kappa}{c_i} = \frac{r_t^\kappa}{c_t} \le \delta 2^{-\ell}.$$

Using $c(O) \le 2^\ell$, it now follows that $r^\kappa(O) \le \delta$, proving (4.6).

Now consider adaptive policy $\mathcal{A}$. We also use $\mathcal{A}$ to denote the (random) subset selected. Note that $\mathcal{A} \subseteq [n] \setminus (S_\ell \cup \sigma) \subseteq [n] \setminus (Q \cup \sigma)$ as $S_\ell \supseteq S_{\ell,\kappa} = Q$. The expected U-reward is:

$$A^* = \mathbb{E}_{\mathcal{A},X}\left[\sum_{i\in\mathcal{A}} U_i\right] = \mathbb{E}_{\mathcal{A},X}\left[\sum_{i=1}^{n} \mathbf{1}_{i\in\mathcal{A}} \cdot U_i\right] = \sum_{i=1}^{n} \mathbf{P}_{\mathcal{A}}(i\in\mathcal{A}) \cdot \mathbb{E}_X[U_i] = \mathbb{E}_{\mathcal{A}}\left[\sum_{i\in\mathcal{A}} \mathbb{E}_X[U_i]\right].$$

The third equality above uses the fact that U_i are independent: so U_i is independent of $i \in \mathcal{A}$. Note that every outcome of $\mathcal{A}$ has total cost at most 2^ℓ. So, for all outcomes of $\mathcal{A}$, the total expected U-reward $\sum_{i\in\mathcal{A}} \mathbb{E}_X[U_i] = r^\kappa(\mathcal{A}) \le \delta$ by (4.6). Combined with the above, we get $A^* \le \delta$. By Markov's inequality, $\mathbf{P}\left[\sum_{i\in\mathcal{A}} U_i \ge 1\right] \le \delta$, which implies

$$\mathbf{P}\left[\sum_{i\in\mathcal{A}} U_i < 1\right] \ge 1 - \delta. \tag{4.7}$$

Now, observe that $\sum_{i\in\mathcal{A}} U_i < 1$ implies $\sum_{i\in\mathcal{A}} \min(R_0(i), \kappa) < \kappa$. Hence, $\sum_{i\in\mathcal{A}} R_0(X_i) < \kappa$. Combined with (4.7) this proves the first part of the lemma.

For the second part, consider $O_\ell \setminus (S_\ell \cup \sigma)$ as the adaptive policy $\mathcal{A}$. It follows that

$$\mathbf{P}\left[R_0(O_\ell \setminus (S_\ell \cup \sigma)) \ge \kappa\right] \le \delta,$$

as claimed. □

Lower bounding reward of $S_\ell \setminus O_\ell$. We first lower bound the expected reward.

Lemma 4.3.6. *If $\kappa > 1$ then $r^\kappa(S_\ell \setminus O_\ell) \ge \frac{\delta(C-1)}{\theta}$.*

Proof. As $\kappa > 1$, the rich scale κ^- exists. To reduce notation let $r_i^- = r_i^{\kappa^-}$ be the reward at scale κ^-. Recall that the budget $D = C2^\ell$. Also, let $h(D)$ and $g(D)$ denote the optimal LP values of the knapsack instances considered in scale κ^- and κ respectively. For any variable i, we have $r_i^- \geq r_i^\kappa \geq \frac{\kappa^-}{\kappa} \cdot r_i^- = \frac{1}{\theta} \cdot r_i^-$. Order the variables T according to κ^-, i.e., in decreasing order of $\frac{r_i^-}{c_i}$. Let t index the variable so that the derivative $h'(D) = \frac{r_t^-}{c_t}$. Note that we have:

$$\frac{r_i^\kappa}{c_i} \geq \frac{1}{\theta} \cdot \frac{r_i^-}{c_i} \geq \frac{1}{\theta} \cdot \frac{r_t^-}{c_t} \text{ for all } 1 \leq i \leq t, \quad \text{and} \quad \sum_{i=1}^{t} c_i \geq D.$$

This implies that the derivative $g'(D) \geq \frac{1}{\theta} \cdot \frac{r_t^-}{c_t} = \frac{h'(D)}{\theta}$. Combined with the fact that κ^- is rich, we have $g'(D) \geq \frac{1}{\theta}\delta 2^{-\ell}$. Now, using the concavity of g (see Theorem 4.2.1) and $D = C2^\ell$,

$$g(D) \geq g(2^\ell) + g'(D) \cdot (D - 2^\ell) \geq g(2^\ell) + \frac{1}{\theta}\delta 2^{-\ell}(C-1)2^\ell = g(2^\ell) + \frac{(C-1)\delta}{\theta}. \qquad (4.8)$$

Note that the variables in this phase are $S_\ell \supseteq Q$ where Q is obtained from Theorem 4.2.1 (for the knapsack instance at scale κ). Hence, $r^\kappa(S_\ell) \geq r^\kappa(Q) \geq g(D)$.

Now consider any outcome of O_ℓ (the variables up to cost 2^ℓ in `OPT`). The total cost $c(O_\ell) \leq 2^\ell$. So, O_ℓ is always a feasible solution to the knapsack instance with budget 2^ℓ. This implies $r^\kappa(O_\ell) \leq g(2^\ell)$; recall that $g(2^\ell)$ is the optimal LP value. Therefore,

$$r^\kappa(S_\ell \setminus O_\ell) \geq r^\kappa(S_\ell) - r^\kappa(O_\ell) \geq g(D) - g(2^\ell) \geq \frac{(C-1)\delta}{\theta}.$$

The last inequality uses (4.8). □

Let $\mu := (C-1)\delta/\theta$. We will ensure that $\mu > 1$. The following is a Chernoff-type bound.

Lemma 4.3.7. *We have* $\mathbf{P}\left(R_0(S_\ell \setminus O_\ell) < \kappa\right) \leq e^{-(\mu - \ln\mu - 1)}$.

Proof. Let $U_i := \min\left\{\frac{R_0(i)}{\kappa}, 1\right\}$ for $i \in S_\ell \setminus O_\ell$. Below, we drop the range $i \in S_\ell \setminus O_\ell$ to reduce clutter. By Lemma 4.3.6, $\sum_i \mathbb{E}[U_i] = r^\kappa(S_\ell \setminus O_\ell) \geq \mu$. Note that $R_0(S_\ell \setminus O_\ell) < \kappa$ implies $\sum_i U_i < 1$. So it suffices to upper bound $\mathbf{P}(\sum_i U_i < 1)$.

Let $t > 0$ be some parameter. We have:

$$\mathbf{P}(\sum_i U_i < 1) = \mathbf{P}\left(e^{-t\sum_i U_i} > e^{-t}\right) \leq \mathbb{E}\left[e^{-t\sum_i U_i}\right] \cdot e^t = e^t \prod_i \mathbb{E}\left[e^{-tU_i}\right].$$

By convexity of $g(u) = e^{-tu}$ we have $e^{-tu} \le 1-(1-e^{-t})\cdot u$ for all $u \in [0,1]$. Taking expectation over $U_i \in [0,1]$, it follows that $\mathbb{E}[e^{-tU_i}] \le 1-(1-e^{-t})\cdot \mathbb{E}[U_i] \le \exp\left(-(1-e^{-t})\cdot \mathbb{E}[U_i]\right)$. Combined with the above,

$$\mathbf{P}(\sum_i U_i < 1) \le e^t \prod_i e^{-(1-e^{-t})\cdot \mathbb{E}[U_i]} = e^{t-(1-e^{-t})\cdot \mu}.$$

Setting $t = \ln \mu > 0$, the right-hand-side above is $e^{-\mu+1+\ln \mu}$ which completes the proof. □

We are now ready to finish the proof.

Lemma 4.3.8. *Consider any parameters $\delta > 0$, $C > 1$ and $\theta > 1$ with $\mu = (C-1)\delta/\theta > 1$. Then, $\mathbf{P}\left(R_0(S_\ell \setminus O_\ell) \ge R_0(O_\ell \setminus (S_\ell \cup \sigma))\right) \ge 1 - \delta - e^{-\mu+1+\ln\mu}$.*

Proof. By Lemma 4.3.5, $\mathbf{P}\left[R_0(O_\ell \setminus (S_\ell \cup \sigma)) \ge \kappa\right] \le \delta$. By Lemma 4.3.7, $\mathbf{P}\left(R_0(S_\ell \setminus O_\ell) < \kappa\right) \le e^{-(\mu - \ln\mu - 1)}$. The lemma now follows by union bound. □

Setting $\delta = 0.1$, $C = 70$ and $\theta = 1.1$, and using Lemma 4.3.8, we get

$$\mathbf{P}\left[R_0(S_\ell \cup \sigma) \ge R_0(O_\ell)\right] = \mathbf{P}\left[R_0(S_\ell \setminus O_\ell) \ge R_0(O_\ell \setminus (S_\ell \cup \sigma))\right] \ge 0.85.$$

This completes the proof of Lemma 4.3.4.

4.4 d-Dimensional Stochastic Score Classification

In this section, we consider d-`SSClass` and prove Theorem 4.1.2. Recall that d-`SSClass` is a generalization of `SSClass` to d "dimensions" where $d \ge 2$. An instance of d-`SSClass` consists of n binary variables $X = X_1, \ldots, X_n$, and a d-dimensional *score function* $r(X) = \sum_{i=1}^n \mathbf{a}_i X_i$; $\mathbf{a_i} \in \mathbb{Z}_+^d$. We denote the k^{th} component of $r(X)$ by $r_k(X)$. The input also specifies intervals $\{I_j^{(k)}\}_j$ for each of these dimensions $k \in [d]$, which in turn define *score classification functions* $\{h_k\}_{k\in[d]}$, such that $h_k(X) = j$ if $r_k(X) \in I_j^{(k)}$. The goal is to design a strategy to *simultaneously* evaluate $h_1, \ldots, h_d$ with minimum expected cost. We let the pair $(\alpha_j^{(k)}, \alpha_{j+1}^{(k)} - 1)$ denote the lower and upper bound for interval $I_j^{(k)}$. For each dimension k and class j, define $\beta_{k,j}^0 = \sum_{i=1}^n a_{ik} - \alpha_{j+1}^{(k)} + 1$ and $\beta_{k,j}^1 = \alpha_j^{(k)}$. If $S \subseteq [n]$ denotes the set of variables selected at some point then, in order to conclude that the class in dimension k is j, we need $\sum_{i\in S} a_{ik}(1 - X_i) \geq \beta^0_{k,j}$ and $\sum_{i\in S} a_{ik}X_i \geq \beta^1_{k,j}$.

We now give an overview of our algorithm (see Algorithm 9 for a formal description). It is easy to state and follows the same overall idea as the algorithm for `SSClass` (see §4.3 for details).

1. The algorithm creates two lists $L_{k,0}$ and $L_{k,1}$ for each dimension $k \in [d]$. These lists are based on the rewards $R_{k,0}(i) := a_{ik}(1 - X_i)$ and $R_{k,1}(i) := a_{ik}X_i$.

2. These $2d$ lists are interleaved together by power-of-2 costs, and then probed non-adaptively until the algorithm can determine the correct class with respect to every dimension.

Algorithm 9 d-NaCl

```
list $L_{k,b} \leftarrow$ BuildList($R_{k,b}$) for $k \in [d]$, $b = 0, 1$
Let $L^\ell_{k,b}$ denote the variables in phase $\ell$ for list $L_{k,b}$
For each phase $\ell$, set $S_\ell \leftarrow \bigcup_{k\in[d]}(L^\ell_{k,0} \cup L^\ell_{k,1})$
**return** list $S_0$, $S_1$, ..., $S_\ell$, ...
```

Recall that the list building algorithm (BuildList) uses parameters C, δ and θ. For d-`SSClass`, we use $\delta \approx \frac{1}{d}$, $C \approx d \log d$ and $\theta = 2$. This algorithm operates in phases: in phase $\ell \geq 0$, the algorithm gets a budget $\approx C2^\ell$ to select variables to add to the list. From Lemma 4.3.2, the cost of the phase-ℓ variables for any list $L_{k,b}$ is at most $(C + 1)2^\ell$. Then, the cost of the variables in S_ℓ is at most $2d \cdot (C + 1)2^\ell$. We formalize this in the following lemma.

Lemma 4.4.1. *The cost incurred in phase ℓ of d-NaCl is at most $2d \cdot (C + 1)2^\ell$.*

4.4.1 The Analysis

The overall analysis is similar to the analysis of NaCl in §4.3. We denote by `OPT` an optimal adaptive solution for d-`SSClass`. The analysis relies on comparing the non-completion probabilities of `OPT` and d-NaCl. Towards this end, we define:

- u_ℓ: probability that d-NaCl is not complete by end of phase ℓ.

- u_ℓ^*: probability that `OPT` costs at least 2^ℓ.

We can assume by scaling that the minimum cost is 1. So $u_0^* = 1$. For ease of notation, we use `OPT` and `NA` to denote the *random* cost incurred by `OPT` and d-NaCl respectively. We also divide `OPT` into phases: phase ℓ corresponds to variables in `OPT` after which the cumulative cost is between $2^{\ell-1}$ and 2^ℓ. We set $C = O(d \log d)$, $\delta = \frac{0.05}{d}$ and $\theta = 2$. Recall that C, δ and θ are parameters of BuildList (Algorithm 7). The following lemma forms the crux of the analysis.

Lemma 4.4.2. *For any phase $\ell \geq 1$, we have $u_\ell \leq q \cdot u_{\ell-1} + u_\ell^*$ where $q \leq 0.3$*

Before we prove this lemma, we use it to prove Theorem 4.1.2.

Proof of Theorem 4.1.2. By the description of the non-adaptive algorithm, the cost until end of *phase* ℓ (in d-NaCl) is at most $2d \cdot (C+1) \sum_{j=0}^{\ell} 2^j \leq 4d \cdot (C+1) \cdot 2^\ell$. Let $\gamma = 4d(C+1)$ below. Following the proof of Theorem 4.1.1, we conclude $\mathbb{E}[\texttt{NA}] \leq \frac{2\gamma}{1-2q} \cdot \mathbb{E}[\texttt{OPT}]$. The proof follows since $C = O(d \log d)$ and $q \leq 0.3$. □

4.4.2 Proof of Lemma 4.4.2

Recall that d-NaCl denotes the non-adaptive algorithm, and `NA` its random cost. Fix any phase $\ell \geq 1$, and let σ denote the realization of the variables probed in the first $\ell - 1$ phases of d-NaCl. We further define the following *conditioned on* σ:

- $u_\ell(\sigma)$: probability that d-NaCl is not complete by end of phase ℓ.
- $u_\ell^*(\sigma)$: probability that `OPT` costs at least 2^ℓ, i.e., `OPT` is not complete by end of phase ℓ.

If d-NaCl does not complete before phase ℓ then $u_{\ell-1}(\sigma) = 1$, and we will prove

$$u_\ell(\sigma) \leq u_\ell^*(\sigma) + 0.3. \tag{4.9}$$

We can complete the proof using this. Note that $u_{\ell-1}(\sigma)$ is either 0 or 1. If $u_{\ell-1}(\sigma) = 0$ then $u_\ell(\sigma) = 0$ as well. So, Equation (4.9) implies that $u_\ell(\sigma) \leq u_\ell^*(\sigma) + 0.3u_{\ell-1}(\sigma)$ for all σ. Taking expectation over σ gives Lemma 4.4.2. It remains to prove Equation (4.9).

We denote by $\mathcal{R}_{k,0}$ and $\mathcal{R}^*_{k,0}$ the $R_{k,0}$ rewards obtained in dimension k in the first ℓ phases by d-NaCl and `OPT` respectively. We similarly define $\mathcal{R}_{k,1}$ and $\mathcal{R}^*_{k,1}$. To prove Equation (4.9), we will show that the probabilities (conditioned on σ) $\mathbf{P}(\mathcal{R}^*_{k,0} > \mathcal{R}_{k,0})$ and $\mathbf{P}(\mathcal{R}^*_{k,1} > \mathcal{R}_{k,1})$ over all dimensions k are small. This implies that with good probability, if `OPT` finishes in phase ℓ, then so does d-NaCl. Formally,

Lemma 4.4.3 (Key Lemma)**.** *For $k \in [d]$, and $b \in \{0,1\}$ we have $\mathbf{P}(\mathcal{R}_{k,b} < \mathcal{R}^*_{k,b} \mid \sigma) \leq \frac{0.15}{d}$.*

Proof of Equation (4.9). If `OPT` finishes in phase ℓ, then there exists some j_k for every dimension k such that $\mathcal{R}^*_{k,0} \geq \beta^0_{k,j_k}$ and $\mathcal{R}^*_{k,1} \geq \beta^1_{k,j_k}$. Thus,

$$\mathbf{P}(\texttt{OPT}\text{ finishes in phase } \ell \mid \sigma) = 1 - u^*_\ell(\sigma) = \mathbf{P}(\exists (j_k)_{k\in[d]} : \mathcal{R}^*_{k,0} \geq \beta^0_{k,j_k} \text{ and } \mathcal{R}^*_{k,1} \geq \beta^1_{k,j_k} \forall k \mid \sigma).$$

From Lemma 4.4.3 and union bound, we have $\mathbf{P}(\exists k, b : \mathcal{R}_{k,b} < \mathcal{R}^*_{k,b} \mid \sigma) \leq 0.3$. Then, we have

$$\begin{aligned} 1 - u_\ell(\sigma) = \mathbf{P}(\texttt{NA}\text{ finishes in phase } \ell \mid \sigma) &\geq \mathbf{P}\left((\texttt{OPT}\text{ finishes in phase } \ell) \bigwedge \wedge_{k,b}(\mathcal{R}_{k,b} \geq \mathcal{R}^*_{k,b}) \middle| \sigma\right) \\ &\geq \mathbf{P}(\texttt{OPT}\text{ finishes in phase } \ell \mid \sigma) - \mathbf{P}(\exists k, b : \mathcal{R}_{k,b} < \mathcal{R}^*_{k,b} \mid \sigma) \\ &\geq (1 - u^*_\ell(\sigma)) - 0.3 \end{aligned}$$

Upon rearranging, this gives $u_\ell(\sigma) \leq u^*_\ell(\sigma) + 0.3$ as desired. □

We now prove Lemma 4.4.3. Using Lemma 4.3.8,

Lemma 4.4.4. *We have $\mathbf{P}(\mathcal{R}_{k,b} < \mathcal{R}^*_{k,b} \mid \sigma) = \delta + e^{-\mu+1+\ln\mu}$ where $\mu := \frac{(C-1)\delta}{\theta}$.*

We set $C = 400d\ln d + 1 = O(d\log d)$, $\delta = \frac{0.05}{d}$, and $\theta = 2$. Then, we have $\mu = 10\ln d$, and from Lemma 4.4.4, we have

$$\begin{aligned} \mathbf{P}(\mathcal{R}_{k,b} < \mathcal{R}^*_{k,b} \mid \sigma) = \delta + e^{-\mu+1+\ln\mu} &= \frac{0.05}{d} + \exp\left(-10\ln d + 1 + \ln 10 \ln d\right) \\ &= \frac{0.05}{d} + \frac{1}{d^{10}} \cdot \exp(1 + \ln 10 + \ln\ln d) \;\leq\; \frac{0.15}{d}, \end{aligned}$$

where the last inequality uses $d \geq 2$.

4.5 Explainable Stochastic Halfspace Evaluation

We consider `EX-SFE` and prove Theorem 4.1.3 in this section. Recall that an instance of `EX-SFE` is similar to that of d-`SSClass`, but where each dimension k has only two intervals $I_0^{(k)}, I_1^{(k)}$, specified by a single threshold α_k. So, the score classification functions $h_k : \{0,1\}^n \to \{0,1\}$ are halfspaces. Additionally, there is an *aggregation function* $f : \{0,1\}^d \to \{0,1\}$, and we want to evaluate $f(h_1(X), \ldots, h_d(X))$. Letting $h(X) := (h_1(X), h_2(X), \ldots, h_d(X))$, we want to evaluate $(f \circ h)(X)$.

Moreover, the problem asks for a *witness* of this evaluation. Recall that a witness is a tuple $(X', v(X'), T)$, where $X' \subseteq X$ is the subset of probed variables, $v(X')$ are the realizations for these variables, and $T \subseteq [d]$ is a subset of the dimensions, satisfying the following conditions:

- The realizations $v(X')$ of the probed variables X' determine $h_k(X)$ for all dimensions $k \in T$. In other words, we can evaluate all the halfspaces corresponding to subset $T \subseteq [d]$.
- The values of $\{h_k(X)\}_{k \in T}$ completely determine $(f \circ h)(X)$; i.e., all realizations of the remaining variables $X \setminus X'$ give the same value for $f \circ h$.

The goal is to design a probing strategy of minimum expected cost that determines $f \circ h$ along with a witness. Before describing our algorithm, we highlight the role of a witness in stochastic halfspace evaluation (by comparing to a model without witnesses). We also discuss the complexity of verifying witnesses for $f \circ h$.

Solutions with/without witness. Solutions that are not required to provide a witness for their evaluation stop when they can infer that the function $f \circ h$ remains constant irrespective of the realizations of the remaining variables. For example, consider the stochastic intersection of halfspaces problem (as studied in [25]). A feasible solution without a witness can stop probing when it determines that, with probability one, either halfspace h_1 or halfspace h_2 is violated (though the solution does not know precisely which halfspace is violated). Such a "stopping rule" may not be useful in situations where one also wants to know the identity of a violated halfspace (say, in order to take some corrective action). In contrast, a solution with a witness (as required in our model) would provide one specific violated halfspace or conclude that all halfspaces are satisfied.

Verifying witnesses. We now address the issue of verifying whether a tuple $(X', v(X'), T)$ is a witness for $f \circ h$. Note that it is easy to check whether $v(X')$ determines $\{h_k(X)\}_{k \in T}$, and the challenge in verifying witnesses lies in confirming whether the values of $\{h_k(X)\}_{k \in T}$ completely determine $(f \circ h)(X)$. For certain special functions, such as intersection of half-spaces or p-of-d functions (where we require at least p of the d halfspaces to be satisfied), this can be done efficiently. However, for a general aggregation function f (without any structure to exploit), verifying a witness may require the evaluation of f at 2^d points (corresponding to all outcomes of the halfspaces $[d] \setminus T$). While our algorithm works for any aggregation function f, for a polynomial running time, we need to assume an efficient oracle O for verifying witnesses.

Our algorithm for `EX-SFE` is the same as for d-`SSClass` (Algorithm 9). However, there is a different stopping condition for probing variables on the non-adaptive list. We use Ex-NaCl to denote the non-adaptive algorithm for `EX-SFE`: this continues probing variables until the observed realizations form a witness for $f \circ h$ (which is verfied using the oracle O). The analysis is also essentially the same as in §4.4, except for the difference in the stopping rule.

Recall that the list building algorithm (BuildList) uses parameters C, δ and θ. As in d-`SSClass`, we use $\delta \approx \frac{1}{d}$, $C \approx d \log d$ and $\theta = 2$. Recall that Algorithm 9 operates in phases. By Lemma 4.4.1, the cost of variables in any phase ℓ is at most $2d \cdot (C+1)2^{\ell}$.

We denote by `OPT` an optimal adaptive solution for `EX-SFE`. As before, we define :

- u_ℓ: probability that Ex-NaCl is not complete by end of phase ℓ.
- u^*_ℓ: probability that `OPT` costs at least 2^{ℓ}.

We will show that Lemma 4.4.2 continues to hold for Ex-NaCl, which would imply Theorem 4.1.3.

In order to prove Lemma 4.4.2 for Ex-NaCl, fix any phase $\ell \geq 1$ and let σ denote the realization of the variables probed in the first $i-1$ phases of Ex-NaCl. Define the following *conditioned on* σ:

- $u_\ell(\sigma)$: probability that Ex-NaCl is not complete by end of phase ℓ.

- $u^*_\ell(\sigma)$: probability that `OPT` costs at least 2^ℓ, i.e., `OPT` is not complete by end of phase ℓ.

If Ex-NaCl does not complete before phase ℓ then $u_{\ell-1}(\sigma) = 1$, and we will prove

$$u_\ell(\sigma) \leq u^*_\ell(\sigma) + 0.3. \tag{4.10}$$

This would imply Lemma 4.4.2 (as shown in §4.4).

It just remains to prove Equation (4.10). We denote by $\mathcal{R}_{k,0}$ and $\mathcal{R}^*_{k,0}$ the $R_{k,0}$ rewards obtained in the first ℓ phases by Ex-NaCl and `OPT` respectively. We similarly define $\mathcal{R}_{k,1}$ and $\mathcal{R}^*_{k,1}$. We note that Lemma 4.4.3 continues to hold here.

Proof of Equation (4.10). If `OPT` finishes in phase ℓ, then there exists some witness $(X^\star, v(X^\star), T^\star)$ such that (i) the realizations $v(X^\star)$ of the variables $X^\star$ probed by `OPT` determine $h_k(X)$ for all dimensions $k \in T^\star$, and (ii) the values $\{h_k(X)\}_{k\in T^\star}$ completely determine $f \circ h$. We further partition $T^\star$ into $T^\star_0 = \{k \in T^\star : h_k(X) = 0\}$ and $T^\star_1 = \{k \in T^\star : h_k(X) = 1\}$. Using the definition of the thresholds $\beta^0_{k,j}$ and $\beta^1_{k,j}$ (see §4.4), we have:

C1. $\mathcal{R}^*_{k,0} \geq \beta^0_{k,0}$ and $\mathcal{R}^*_{k,1} \geq \beta^1_{k,0}$ for each $k \in T^\star_0$, and

C2. $\mathcal{R}^*_{k,0} \geq \beta^0_{k,1}$ and $\mathcal{R}^*_{k,1} \geq \beta^1_{k,1}$ for each $k \in T^\star_1$.

Let $\mathcal{T}$ denote the set of 3-way-partitions $(T_0, T_1, [d]\backslash T_0 \backslash T_1)$ where f is completely determined by setting the coordinates in T_0 (resp. T_1) to 0 (resp. 1). Note that for any witness as above, we have $(T^\star_0, T^\star_1, [d] \setminus T^\star_0 \setminus T^\star_1) \in \mathcal{T}$. Thus,

$$\begin{aligned} 1 - u^*_\ell(\sigma) \ &= \ \mathbf{P}(\texttt{OPT} \text{ finishes in phase } \ell \mid \sigma) \\ &= \mathbf{P}(\exists (T^\star_0, T^\star_1, [d] \setminus T^\star_0 \setminus T^\star_1) \in \mathcal{T} \ : \ \text{ conditions C1 and C2 hold } \mid \sigma). \end{aligned}$$

Note that if `OPT` finishes in phase ℓ and $\mathcal{R}_{k,b} \geq \mathcal{R}^*_{k,b}$ for all k and b, then we can conclude that Ex-NaCl also finishes in phase ℓ (with the same witness as `OPT`). From Lemma 4.4.3

and union bound, we have $\mathbf{P}(\exists k, b : \mathcal{R}_{k,b} < \mathcal{R}^*_{k,b} \mid \sigma) \leq 0.3$. Hence,

$$\begin{aligned}
1 - u_\ell(\sigma) = \mathbf{P}(\text{Ex-NaCl finishes in phase } \ell \mid \sigma) &\geq \mathbf{P}\left((\texttt{OPT} \text{ finishes in phase } \ell) \bigwedge \wedge_{k,b}(\mathcal{R}_{k,b} \geq \mathcal{R}^*_{k,b}) \middle| \sigma\right) \\
&\geq \mathbf{P}(\texttt{OPT} \text{ finishes in phase } \ell \mid \sigma) - \mathbf{P}(\exists k, b : \mathcal{R}_{k,b} < \mathcal{R}^*_{k,b} \mid \sigma) \\
&\geq (1 - u^*_\ell(\sigma)) - 0.3
\end{aligned}$$

Upon rearranging, this gives $u_\ell(\sigma) \leq u^*_\ell(\sigma) + 0.3$ as desired. □

4.6 Computational Results

We provide a summary of computational results of our non-adaptive algorithm for the stochastic score classification problem. We conducted all of our computational experiments using Python 3.8 with a 2.3 GHz Intel Core i5 processor and 16 GB 2133MHz LPDDR3 memory. We use synthetic data to generate instances of `SSClass` for our experiments.

Instance Generation. We test our algorithm on synthetic data generated as follows. We first set $n \in \{100, 200, \ldots, 1000\}$. Given n, we generate n Bernoulli variables, each with probability chosen uniformly from $(0, 1)$. We set the costs of each variable to be an integer in $[10, 100]$. To select cutoffs (when $B \neq 2$), we first select $B \in \{5, 10, 15\}$ and then select the cutoffs (based on the value of B) uniformly at random in the score interval. We provide more details and plots in the full version of the paper. For each n we generate 10 instances. For each instance, we sample 50 realizations in order to calculate the average cost and average runtime.

Algorithms. We compare our non-adaptive `SSClass` algorithm (Theorem 4.1.1) against a number of prior algorithms. For `SBFT` instances, we compare to the *adaptive* 3-approximation algorithm from [40]. For *unweighted* `SSClass` instances, we compare to the non-adaptive $2(B-1)$-approximation algorithm from [53]. For general `SSClass` instances, we compare to the *adaptive* $O(\log W)$-approximation algorithm from [53]. As a benchmark, we also compare to a naive non-adaptive algorithm that probes variables in a random order. We also compare to an information-theoretic lower bound (no adaptive policy can do better than this lower bound). We obtain this lower bound by using an integer linear program to compute the (offline) optimal probing cost for a given realization (see §4.6.1 for details), and then taking an average over 50 realizations.

Parameters C, δ, and θ. As noted in §4.3, our algorithm achieves a constant factor approximation guarantee for any constant $C > 1$, $\delta \in (0, 1)$ and $\theta > 1$. For our final computations, we (arbitrarily) choose values $C = 2$, $\delta = 0.01$, and $\theta = 2$.

Reported quantities. For every instance, we compute the cost and runtime of each algorithm by taking an average over 50 independent realizations. For the non-adaptive algorithms, note that we only need *one* probing sequence for each instance. On the other hand, adaptive algorithms need to find the probing sequence afresh for each realization. As seen in all the runtime plots, the non-adaptive algorithms are significantly faster.

For each instance type (`SBFT`, Unweighted `SSClass` and `SSClass`), the plots in Figures 4.2-4.5 show the averages (for both cost and runtime) against the number of variables n. Note that each point in these plots corresponds to an average over (i) the 10 instances of its type and (ii) the 50 sampled realizations for each instance.

In Table 4.1, we report the average performance ratio (cost of the algorithm divided by the information-theoretic lower bound) of the various algorithms. For each instance type (`SBFT`, Unweighted `SSClass` and `SSClass`), we report the performance ratio averaged over *all* values of n (10 choices) and all instances (10 each). Note that values closer to 1 demonstrate better performance.

Stochastic Boolean Function Evaluation for Linear Threshold Functions. To generate an instance of `SBFT` from synthetic data, we set $a_i \in [10]$ uniformly for all $i \in [n]$ and select a cutoff value uniformly in the score interval. We plot the results in Figure 4.2. Our cost is about 20% more than that of [40], but our runtime is over $100\times$ faster for large instances.

Unweighted Stochastic Score Classification. To generate instances of unweighted `SSClass` from synthetic data, we set $a_i = 1$ for all $i \in [n]$. We choose $B \in \{5, 10, 15\}$,

Instance Type	Our Alg.	GGHK Alg.	Random List
Unweighted `SSClass`, $B = 5$	1.50	1.48	1.80
Unweighted `SSClass`, $B = 10$	1.25	1.24	1.33
Unweighted `SSClass`, $B = 15$	1.13	1.13	1.19
`SSClass`, $B = 5$	1.59	1.94	2.43
`SSClass`, $B = 10$	1.34	1.45	1.73
`SSClass`, $B = 15$	1.22	1.39	1.47

Instance Type	Our Alg.	DHK Alg.	Random List
`SBFT`	2.18	1.74	5.63

Table 4.1: Average performance ratios relative to the lower bound.

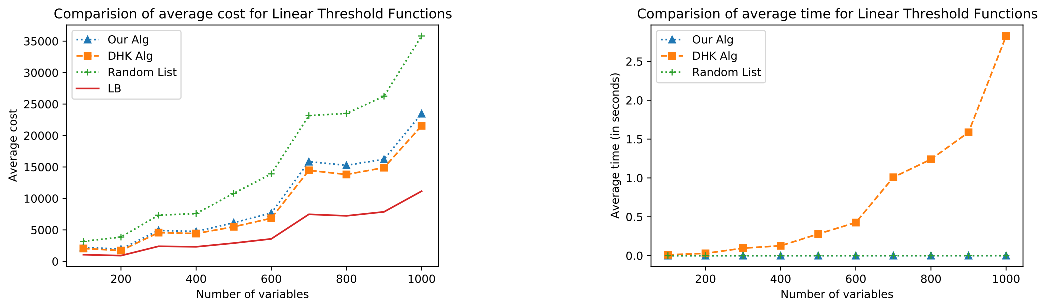


Figure 4.2: Results for `SBFT` (Cost and Runtime)

and then select the cutoffs (based on the value of B) uniformly at random in the score interval. We test our non-adaptive algorithm against the non-adaptive algorithm of [53] and a random query order. Since, all the algorithms are non-adaptive, there is no difference in their running time. So, we focus on the average cost comparison among the algorithms. We observe that the average cost incurred by our non-adaptive algorithm is comparable to that of the non-adaptive algorithm of [53], and both algorithms outperform a random query order. We plot the results in Figure 4.3

(Weighted) Stochastic Score Classification. We test on synthetic `SSClass` instances with $B \in \{5, 10, 15\}$. The cutoff values α_j are selected uniformly at random in the score interval. We plot results in Figures 4.4 and 4.5. In all cases, we observe that our non-adaptive algorithm beats the adaptive algorithm in both query cost and runtime; for e.g., when $B = 10$ and $n = 900$ our cost is 10% less and our runtime is about 100× faster.

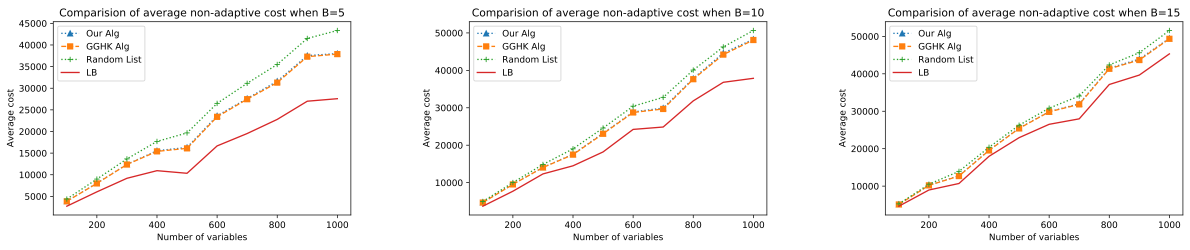


Figure 4.3: Results for Unweighted `SSClass` (Cost)

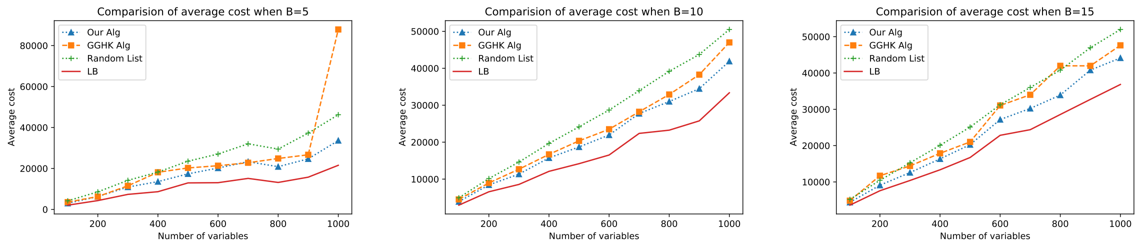


Figure 4.4: Results for `SSClass` (Cost)

4.6.1 An Information-Theoretic Lower Bound for `SSClass`

Here, we present an information theoretic lower bound for `SSClass`. Recall that an instance of `SSClass` consists of n independent Bernoulli random variables $X = X_1, \ldots, X_n$, where variable X_i is 1 with probability p_i, and its realization can be probed at cost $c_i \in \mathbb{R}_+$. The score of the outcome $X = (X_1, \ldots, X_n)$ is $r(X) = \sum_{i=1}^{n} a_i X_i$ where $a_i \in \mathbb{Z}_+$ for all $i \in [n]$. Additionally, we are given $B + 1$ thresholds $\alpha_1, \ldots, \alpha_{B+1}$ which define intervals $I_1, \ldots, I_B$ where $I_j = \{\alpha_j, \ldots, \alpha_{j+1} - 1\}$. These intervals define a *score classification function* $h : \{0,1\}^n \to \{1, \ldots, B\}$; $h(X) = j$ precisely when $r(X) \in I_j$. The goal is to determine $h(X)$ at minimum expected cost.

Let $\widehat{X} = (\widehat{X}_1, \ldots, \widehat{X}_n)$ correspond to a realization of the variables X. Furthermore, suppose that $h(\widehat{X}) = j$; that is, under realization $\widehat{X}$, the score $h(\widehat{X})$ lies in I_j. Let $S \subseteq [n]$ correspond to the set of probed variables. Recall that S is a feasible solution for `SSClass` under realization $\widehat{X}$ when the following conditions on the R_0 and R_1 rewards of S hold: $\sum_{i \in S} R_0(i) = \sum_{i \in S} a_i \cdot (1 - \widehat{X}_i) \geq \beta_j^0$ and $\sum_{i \in S} R_1(i) = \sum_{i \in S} a_i \cdot \widehat{X}_i \geq \beta_j^1$ where $\beta_j^0 = W - \alpha_{j+1} + 1$ and $\beta_j^1 = \alpha_j$. Thus, the following integer program computes a lower bound on

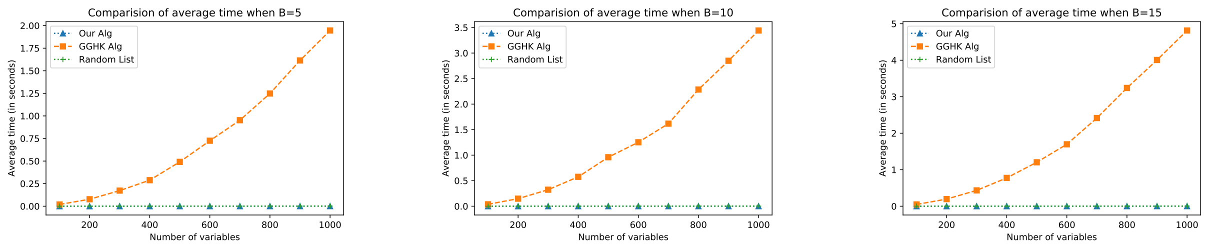


Figure 4.5: Results for `SSClass` (Runtime)

the probing cost needed to conclude that $h(\widehat{X}) = j$.

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{n} c_i \cdot z_i \\
\text{subject to} \quad & \sum_{i=1}^{n} a_i \widehat{X}_i \cdot z_i \geq \beta_j^1 \\
& \sum_{i=1}^{n} a_i (1 - \widehat{X}_i) \cdot z_i \geq \beta_j^0 \\
& z_i \in \{0,1\} \quad i \in [n]
\end{aligned}
\tag{4.11}
$$

where z_i, for $i \in [n]$, is a binary variable denoting whether X_i is probed. Let $\widehat{\texttt{LB}}$ denote the optimal value of (4.11). Then, $\texttt{LB} = \mathbb{E}[\widehat{\texttt{LB}}]$ is an information-theoretic lower bound for the given `SSClass` instance. We note that (4.11) only provides a lower bound on the probing cost for realization $\widehat{X}$, and is not a formulation for the given `SSClass` instance.

4.7 Handling Negative Weights

We now show how our results can be extended to the case where the weights a_i may be positive or negative. For the stochastic score classification problem (`SSClass`), we provide a reduction from any instance (with aribtrary weights) to one with positive weights. Let $\mathcal{I}$ be any instance of `SSClass`, and let $P \subseteq [n]$ (resp. $N \subseteq [n]$) denote the variables with positive (resp. negative) weight. Note that we can re-write the score as follows:

$$
r(X) = \sum_{i=1}^{n} a_i X_i = \sum_{i \in P} a_i X_i - \sum_{i \in N} |a_i| X_i = -\sum_{i \in N} |a_i| + \sum_{i \in P} a_i X_i + \sum_{i \in N} |a_i|(1 - X_i). \tag{4.12}
$$

Consider now a new SSClass instance with the n variables $X'_i = X_i$ for $i \in P$ and $X'_i = 1 - X_i$ for $i \in N$. The probabilities are $p'_i = p_i$ for $i \in P$ and $p'_i = 1 - p_i$ for $i \in N$. The weights are $a'_i = a_i$ for $i \in P$ and $a'_i = -a_i$ for $i \in N$; note that all weights are now positive. The costs remain the same as before. For any realization X, the new score is $\sum_{i=1}^n a'_i X'_i = \sum_{i \in N} |a_i| + r(X)$, using (4.12). Finally the class boundaries for the new instance are $\alpha'_j = \alpha_j + \sum_{i \in N} |a_i|$. It is easy to check that a realization X' in the new instance has class j if and only if the corresponding realization X has class j in the original instance $\mathcal{I}$.

d-Dimensional score classification and explainable stochastic halfspace evaluation. To handle negative weights in d-SSClass (and EX-SFE), we need to apply the above idea within the algorithm (it is not a black-box reduction to positive weights). At a high level, we use the fact that the algorithm simply interleaves the separate lists for each dimension.

As above, we introduce a new score for each dimension k that only has positive weights:

$$r'_k(X) = \sum_{i \in P_k} a_{ik} X_i + \sum_{i \in N_k} |a_{ik}|(1 - X_i), \text{ where } r_k(X) = r'_k(X) - \sum_{i \in N_k} |a_{ik}|. \tag{4.13}$$

Correspondingly, we redefine the two rewards as follows.

$$R_{k,0}(i) = \begin{cases} a_{ik}(1 - X_i) & \text{if } i \in P_k \\ -a_{ik} X_i & \text{if } i \in N_k \end{cases}, \qquad R_{k,1}(i) = \begin{cases} a_{ik} X_i & \text{if } i \in P_k \\ -a_{ik}(1 - X_i) & \text{if } i \in N_k \end{cases}$$

Note that after probing a subset $S \subseteq [n]$, the total $R_{k,0}$ (resp. $R_{k,1}$) reward corresponds to a lower (resp. upper) bound on the score $r'_k(X)$. This also implies lower/upper bounds on the original score $r_k(X)$ by a linear shift. The rest of the algorithm remains the same as Algorithm 9. The analysis also remains the same.

4.8 Proof of Theorem 4.2.1

Recall the setting of Theorem 4.2.1. We are given a set T of items with non-negative costs $\{c_k\}$ and rewards $\{r_k\}$, along with a budget D. The goal is to select a subset of items of total cost at most D that maximizes the total reward. The following is a natural LP relaxation of the knapsack problem where the objective is expressed as a function of the budget D.

$$g(D) = \max \left\{ \sum_{k \in T} r_k \cdot x_k \mid \sum_{k \in T} c_k \cdot x_k \leq D,\, x \in [0,1]^T \right\}, \qquad \forall D \geq 0.$$

The following algorithm $\mathcal{A}_{\mathcal{KS}}$ solves the *fractional* knapsack problem and also obtains an approximate integral solution. Assume that the items are ordered so that $\frac{r_1}{c_1} \geq \frac{r_2}{c_2} \geq \cdots$. Let t index the first item (if any) so that $\sum_{k=1}^{t} c_k \geq D$. Let $\psi := \frac{1}{c_t}(D - \sum_{k=1}^{t-1} c_k)$ which lies in $(0,1]$. Define

$$x_k = \begin{cases} 1 & \text{if } k \leq t-1 \\ \psi & \text{if } k = t \end{cases}.$$

Return x as the optimal fractional solution and $Q = \{1, \cdots, t\}$ as an integer solution. We restate Theorem 4.2.1 for completeness.

Theorem 4.8.1. *Consider algorithm $\mathcal{A}_{\mathcal{KS}}$ on any instance of the knapsack problem with budget D.*

1. *$g(D) = \sum_{k=1}^{t-1} r_k + \psi \cdot r_t = \langle r, x \rangle$ and so x is an optimal LP solution.*
2. *The derivative $g'(D) = \frac{r_t}{c_t}$.*
3. *Solution Q has cost $c(Q) \leq D + c_{max}$ and reward $r(Q) \geq g(D)$.*
4. *$g(D)$ is a concave function of D.*

Proof. Let x^* be an optimal LP solution. If $x^* = x$, we are done. Suppose that $x^* \neq x$, and assume without loss of generality that $\sum_{k=1}^{n} c_k \cdot x_k^* = D$; else we can obtain a greater reward by raising x^*. Let j be the smallest index such that $x_j^* < x_j$: note that $j \leq t$ is well defined by the definition of x. Let h be the largest index with $x_h^* > x_h$: note that such an index must exist as $\sum_{k=1}^{n} c_k \cdot x_k^* = \sum_{k=1}^{n} c_k \cdot x_k$. Moreover, $h \geq t$ by definition of our solution x. As $j \neq h$, we have $j < h$ from above. Define a new solution

$$x'_k = \begin{cases} x_k^* & \text{if } k \neq j, h \\ x_j^* + \delta & \text{if } k = j \\ x_h^* - \frac{c_j}{c_h}\delta & \text{if } k = h \end{cases}.$$

Above, $\delta = \min\{1 - x_j^*, \frac{c_h}{c_j} x_h^*\} > 0$. Intuitively, we are redistributing cost from item h to j. Note that the cost of the new solution $\sum_{k=1}^n c_k \cdot x_k' = \sum_{k=1}^n c_k \cdot x_k^* = D$. Moreover, the reward

$$\sum_{k=1}^{n} r_k \cdot x_k' = \sum_{k=1}^{n} r_k \cdot x_k^* + \delta r_j - \delta \frac{c_j}{c_h} r_h \geq \sum_{k=1}^{n} r_k \cdot x_k^*,$$

where we used $\frac{r_j}{c_j} \geq \frac{r_h}{c_h}$ which follows from $j < h$ and the ordering of items. Finally, by choice of δ, either $x_j' = 1$ or $x_h' = 0$. It follows that x' is also an optimal LP solution. Repeating this process, we obtain that x is also an optimal LP solution. This completes the proof of property (1).

For property (2), observe that

$$g'(D) = \lim_{\epsilon \to 0} \frac{g(D+\epsilon) - g(D)}{\epsilon} = \lim_{\epsilon \to 0} \frac{g(D) + \epsilon \cdot \frac{r_t}{c_t} - g(D)}{\epsilon} = \frac{r_t}{c_t},$$

as desired.

Since $\psi \in (0, 1]$, we have $c(Q) = D + (1-\psi) \cdot c_t \leq D + c_{\max}$ and $r(Q) = (1-\psi) \cdot r_t + g(D) \geq g(D)$, proving property (3).

Finally, using property (2) and the non-increasing $\frac{r_k}{c_k}$ order of the items, it follows that $g(D)$ is a concave function. This proves property (4). $\square$

Chapter 5

Introduction to Batched Dueling Bandits

5.1 Motivation

The K-armed dueling bandits problem has been widely studied in machine learning due to its applications in search ranking, recommendation systems, sports ranking, etc. [114, 112, 108, 6, 117, 115, 116, 41, 68, 76, 77, 97, 31]. It is a variation of the traditional stochastic bandit problem in which feedback is obtained in the form of pairwise preferences. This problem falls under the umbrella of *preference learning* [109], where the goal is to learn from relative feedback (in our case, given two alternatives, which of the two is preferred). Designing learning algorithms for such relative feedback becomes crucial in domains where qualitative feedback is easily obtained, but real-valued feedback would be arbitrary or not interpretable. We illustrate this using the web-search ranking application.

Web-search ranking is an example of a complex information retrieval system, where the goal is to provide a list (usually *ranked*) of candidate documents to the user of the system in response to a query [96, 71, 113, 65]. Modern day search engines comprise hundreds of parameters which are used to output a ranked list in response to a query. However, manually tuning these parameters can sometimes be infeasible, and online learning frameworks (based on user feedback) have been invaluable in automatically tuning these parameters [84]. These methods do not affect user experience, enable the system to continuously learn about user preferences, and thus continuously adapt to user behavior. For example, given two rankings ℓ_1 and ℓ_2, they can be interleaved and presented to the user in such a way that clicks indicate which of the two rankings is more preferable to the user [96]. The availability of such pairwise comparison data motivates the study of learning algorithms that exploit such relative feedback.

Previous learning algorithms have focused on a *fully adaptive* setting; in the web-ranking application this corresponds to the learning algorithm updating its parameters after each query. Such updates might be impractical in large systems for the following reasons. If the parameters are fine-tuned for each user and users make multiple queries in a short time, such continuous updates require a lot of computational power. Even if users are assigned to a small number of classes (and parameters are fine-tuned for each user-class), multiple users from the same class may simultaneously query the system, making it impractical to adapt after each interaction.

Motivated by this, we introduce the *batched K-armed dueling bandits problem* (or, *batched dueling bandits*), where the learning algorithm is only allowed to *adapt a limited number of times*. Specifically, the algorithm uses at most B *adaptive rounds* and in each round it commits to a fixed *batch* of pairwise comparisons. The feedback for a batch is received simultaneously, and the algorithm chooses the next batch based on this (and previous) feedback.

5.2 Preliminaries

The *K-armed dueling bandits* problem [112] is an online optimization problem, where the goal is to find the best among K bandits $\mathcal{B} = \{b_1, \ldots, b_K\}$ using noisy pairwise comparisons with low *regret*. In the traditional multi-armed bandit problem [9], an *arm* (or equivalently, bandit) b_j can be pulled at each time-step t, which generates a random reward from an unknown stationary distribution with expected value μ_j. However, in the K-armed dueling bandits problem, each iteration comprises a noisy comparison between two bandits (possibly the same), say (b_i, b_j). The outcome of the comparison is an independent random variable, and the probability of picking b_i over b_j is a constant denoted $P_{i,j} = \frac{1}{2} + \Delta_{i,j}$ where $\Delta_{i,j} \in$

$(-\frac{1}{2}, \frac{1}{2})$. Here $\Delta_{i,j}$ can be thought of as a measure of distinguishability between the two bandits, and we use $b_i \succ b_j$ when $\Delta_{i,j} > 0$. We also refer to $\Delta_{i,j}$ as the *gap* between b_i and b_j.

Throughout the paper, we let b_1 refer to the best bandit. To further simplify notation, we define $\Delta_j = \Delta_{1,j}$; that is, the gap between b_1 and b_j. We define the *regret* per time-step as follows: suppose bandits b_{t_1} and b_{t_2} are chosen in iteration t, then the regret $r(t) = \frac{\Delta_{t_1}+\Delta_{t_2}}{2}$. The cumulative regret up to time T is $R(T) = \sum_{t=1}^{T} r(t)$, where T is the time horizon, and it's assumed that $K \leq T$. The cumulative regret can be equivalently stated as $R(T) = \frac{1}{2}\sum_{j=1}^{K} T_j \Delta_j$, where T_j denotes the number comparisons involving b_j. We define $\Delta_{\min} = \min_{j:\Delta_j>0} \Delta_j$ to be the smallest non-zero gap of any bandit with b_1. We say that bandit b_i is a *Condorcet winner* if, and only if, $P_{i,j} \geq \frac{1}{2}$ for all $j \in \mathcal{B} \setminus \{i\}$. Furthermore, we say that the probabilistic comparisons exhibit *strong stochastic transitivity* (SST) if there exists an ordering, denoted by $\succeq$, over arms such that for every triple $b_i \succeq b_j \succeq b_k$, we have $\Delta_{i,k} \geq \max\{\Delta_{i,j}, \Delta_{j,k}\}$, and exhibits *stochastic triangle inequality* (STI) if for every triple $b_i \succeq b_j \succeq b_k$, $\Delta_{i,k} \leq \Delta_{i,j} + \Delta_{j,k}$.

5.2.1 Batch Policies

In traditional bandit settings, actions are performed *sequentially*, utilizing the results of *all prior actions* in determining the next action. In the batched setting, the algorithm must commit to a round (or *batch*) of actions to be performed *in parallel*, and can only observe the results after all actions in the batch have been performed. More formally, in round $r = 1, 2, \ldots$, the algorithm must decide the comparisons to be performed; afterwards *all* outcomes of the comparisons in batch r are received. The algorithm can then, *adaptively*, select the next batch of comparisons. However, it can use at most a given number, B, of batches.

The batch sizes can be chosen *non-adaptively* (fixed upfront) or *adaptively*. In an adaptive policy the batch sizes may even depend on previous observations of the algorithm. An adaptive policy is more powerful than a non-adaptive policy, and may suffer a smaller regret. In this paper, we focus on such adaptive policies. Furthermore, note that the total number of comparisons (across all batches) must sum to T. We assume that the values of T and B

are known. Observe that when $T = B$, we recover the fully sequential setting.

5.3 Overview of Results

We now provide a summary of our results. Our first result is as follows.

Theorem 5.3.1. *For any integer $B \geq 1$, there is an algorithm for batched dueling bandits that uses at most B rounds, and if the instance admits a Condorcet winner, the expected regret is bounded by*

$$\mathbb{E}[R(T)] \leq 3KT^{1/B} \log\left(2wTK^2B\right) \sum_{j:\Delta_j>0} \frac{1}{\Delta_j}.$$

The above bound is an instance-dependent bound. To obtain an instance-independent bound, recall that $\Delta_{\min} = \min_{j:\Delta_j>0} \Delta_j$. We get that the expected worst-case regret is bounded by

$$\mathbb{E}[R(T)] \leq \frac{3K^2T^{1/B}\log\left(2TK^2B)\right)}{\Delta_{\min}}.$$

In the sequential setting, [117, 76] achieve a bound of $O\left(K\frac{\log T}{\Delta_{\min}}\right) + O\left(\frac{K^2}{\Delta_{\min}}\right)$ on the expected regret in the worst-case. When $B = \log(T)$, our worst-case regret is at most

$$\mathbb{E}[R(T)] \leq 3K^2 \log(6TK^2B)/\Delta_{\min} = O(K^2 \log(T)/\Delta_{\min}),$$

which nearly matches the best-known bound in the sequential setting. Our algorithm in Theorem 5.3.1 proceeds by performing all pairwise comparisons in an *active set* of bandits, and gradually eliminating sub-optimal bandits. This algorithm is straightforward, and its analysis follows that of [43] for batched stochastic multi-armed bandits (see §5.5). Although this is a simple result, it is an important step for the next set of results which apply when the instance satisfies the SST and STI conditions. These conditions impose a structure on the pairwise preference probabilities, and we are able to exploit this additional structure to obtain improved bounds. We only state the results here: refer to Chapter 6 for details and proofs.

Theorem 5.3.2. *For any integer $B \geq 1$, there is an algorithm for batched dueling bandits that uses at most $B+1$ rounds, and if the instance satisfies the SST and STI assumptions, the expected regret is bounded by*

$$\mathbb{E}[R(T)] = \sum_{j:\Delta_j>0} O\left(\frac{\sqrt{K}T^{1/B}\log(T)}{\Delta_j}\right).$$

Using additional rounds of adaptivity, we get the following improved guarantee.

Theorem 5.3.3. *For any integer $B \geq 1$, there is an algorithm for batched dueling bandits that uses at most $2B+1$ rounds, and if the instance satisfies the SST and STI assumptions, the expected worst-case regret is bounded by*

$$\mathbb{E}[R(T)] = O\left(\frac{KBT^{1/B}\log(T)}{\Delta_{\min}}\right).$$

Thus, in $B = \log(T)$ rounds, our expected worst-case regret is bounded by $E[R(T)] \leq O\left(\frac{K\log^2(T)}{\Delta_{\min}}\right)$ matching the best known result in the sequential setting up to an additional logarithmic factor.

We also improve the instance-dependent regret bound in Theorem 5.3.2 by using a few additional rounds. In particular, using the approach in Theorem 5.3.2 along with recursion, we obtain:

Theorem 5.3.4. *For integers $B \geq 1$, $m \geq 0$ and parameter $\eta \in (0,1)$, there is an algorithm for batched dueling bandits that uses at most $B+m$ rounds, and if the instance satisfies the SST and STI assumptions, the expected regret is bounded by*

$$\mathbb{E}[R(T)] = O\left(m \cdot K^{\eta} + K^{(1-\eta)^m}\right) \cdot T^{1/B}\log(KTB) \sum_{j:\Delta_j>0} \frac{1}{\Delta_j}.$$

Thus, for any constant $\eta \in (0,1)$, setting $m = \frac{1}{\eta}\log\left(\frac{1}{\eta}\right)$, we obtain expected regret bounded by

$$\mathbb{E}[R(T)] = O\left(K^{\eta}\; T^{1/B}\;\; \log(T)\right) \sum_{j:\Delta_j>0} \frac{1}{\Delta_j}$$

Rounds $B+m$	Regret
$B+2$	$K^{0.39}\ T^{1/B}\ \log(T)\ E$
$B+3$	$K^{0.32}\ T^{1/B}\ \log(T)\ E$
$B+5$	$K^{0.25}\ T^{1/B}\ \log(T)\ E$
$B+10$	$K^{0.17}\ T^{1/B}\ \log(T)\ E$

Table 5.1: Instance-dependent regret bounds vs. rounds in Theorem 5.3.4; here, $E = \sum_{j:\Delta_j>0} \frac{1}{\Delta_j}$.

in at most $B + \frac{1}{\eta}\log\left(\frac{1}{\eta}\right)$ rounds. Conversely, given a value of m, we can appropriately select η to minimize the regret. Table 5.1 lists our instance-dependent regret bounds for some values of m.

Finally, we complement our upper bound results with a lower bound for the batched K-armed dueling bandits problem, even under the SST and STI assumptions.

Theorem 5.3.5. *Given an integer $B \geq 1$, and any algorithm that uses at most B batches, there exists an instance of the K-armed batched dueling bandit problem that satisfies the SST and STI condition such that the expected regret*

$$\mathbb{E}[R(T)] = \Omega\left(\frac{KT^{1/B}}{B^2\Delta_{\min}}\right).$$

The proofs of Theorems 5.3.2, 5.3.3, 5.3.4, and 5.3.5 appear in Chapter 6.

Lastly, we are able to obtained improved regret bounds for instances that satisfy the Condorcet condition using a novel technique that identifies the best arm in a small *expected* number of rounds, after which it uses this arm as an "anchor" to eliminate sub-optimal arms while incurring low regret. Formally, we show the following.

Theorem 5.3.6. *For any integer $B \geq 1$, there is an algorithm for the K-armed dueling bandit problem that uses at most B rounds with the following guarantee. For any $\delta > 0$, with* *probability at least* $1-\delta-\frac{1}{T}$*, its regret under the Condorcet condition is at most*

$$R(T) \le O\left(T^{1/B} \cdot \frac{K^2 \log(K)}{\Delta_{\min}^2} \cdot \log\left(\frac{\log K}{\Delta_{\min}}\right)\right) + O\left(T^{2/B} \cdot K^2 \cdot \sqrt{\frac{1}{\delta}}\right) + \sum_{j \neq a^*} O\left(\frac{T^{1/B} \cdot \log(KT)}{\Delta_j}\right).$$

We can convert the preceding high-probability bound to obtain an upper bound on expected regret as follows.

Theorem 5.3.7. *For any integer* $B \ge 1$*, there is an algorithm for the* K*-armed dueling bandit problem that uses at most* B *rounds, with expected regret under the Condorcet condition at most*

$$\mathbb{E}[R(T)] = O\left(T^{1/B} \cdot \frac{K^2 \log(K)}{\Delta_{\min}^2} \cdot \log\left(\frac{\log K}{\Delta_{\min}}\right)\right) + O\left(T^{2/B} \cdot K^2\right) + \sum_{j \neq a^*} O\left(\frac{T^{1/B} \cdot \log(KT)}{\Delta_j}\right).$$

When the number of rounds $B = \log(T)$, we obtain a batched algorithm that achieves the asymptotic optimality (in terms of T), even for sequential algorithms. We formalize this observation in the following corollary.

Corollary 5.3.8. *There is an algorithm for the* K*-armed dueling bandit problem that uses at most* $\log(T)$ *rounds, with expected regret under the Condorcet condition at most*

$$\mathbb{E}[R(T)] = O\left(\frac{K^2 \log(K)}{\Delta_{\min}^2} \cdot \log\left(\frac{\log K}{\Delta_{\min}}\right)\right) + \sum_{j \neq a^*} O\left(\frac{\log(KT)}{\Delta_j}\right).$$

As a consequence of Theorem 5.3.5, it follows that no algorithm can achieve $O(\frac{K}{\Delta_{\min}} \cdot poly(\log T))$ regret using $o(\frac{\log T}{\log\log T})$ rounds. So, the $O(\log T)$ rounds required to achieve asymptotic optimality in Corollary 5.3.8 is nearly the best possible. The proofs of Theorems 5.3.6 and 5.3.7 appear in Chapter 7.

5.4 Related Work

The K-armed dueling bandit problem has been widely studied in recent years (we refer the reader to [105] for a comprehensive survey). Here, we survey the works that are most closely related to our setting. This problem was first studied in [112] under the SST and STI setting.

The authors obtained a worst-case regret upper bound of $\widetilde{O}(K \log T/\Delta_{\min})$ and provided a matching lower bound. [114] considered a slightly more general version of the SST and STI setting and achieved an instance-wise optimal regret upper bound of $\sum_{j:\Delta_j>0} O\left(\log(T)/\Delta_j\right)$. Since, the SST+STI condition imposes a total order over the arms and might not hold for real-world datasets, [108] initiated the study of dueling bandits under the Condorcet winner condition. [108] proved a $O(K^2 \log T/\Delta_{\min})$ regret upper bound under the Condorcet condition, which was improved by [117] to $O(K^2/\Delta_{\min}^2) + \sum_{j:\Delta_j>0} O(\log T/\Delta_j^2)$. [76] achieved a similar but tighter KL divergence-based bound, which is shown to be *asymptotically instance-wise optimal* (even in terms constant factors). There are also other works that improve the dependence on K in the upper bound, but suffer a worse dependence on Δ_j's [116]. This problem has also been studied under other noise models such as utility based models [6] and other notions of regret [31]. Alternate notions of winners such as Borda winner [68], Copeland winner [115, 77, 111], and von Neumann winner [41] have also been considered. There are also several works on extensions of dueling bandits that allow multiple arms to be compared at once [104, 5, 100].

All of the aforementioned works on the dueling bandits problem are limited to the sequential setting. To the best of our knowledge, ours is the first work that considers the batched setting for dueling bandits. However, batched processing for the stochastic multi-armed bandit problem has been investigated in the past few years. A special case when there are two bandits was studied by [93]. They obtain a worst-case regret bound of $O\left(\left(\frac{T}{\log(T)}\right)^{1/B} \frac{\log(T)}{\Delta_{\min}}\right)$. [48] studied the general problem and obtained a worst-case regret bound of $O\left(\frac{K\log(K)T^{1/B}\log(T)}{\Delta_{\min}}\right)$, which was later improved by [43] to $O\left(\frac{KT^{1/B}\log(T)}{\Delta_{\min}}\right)$. Furthermore, [43] obtained an instance-dependent regret bound of $\sum_{j:\Delta_j>0} T^{1/B} O\left(\frac{\log(T)}{\Delta_j}\right)$. Our results for batched dueling bandits are of a similar flavor; that is, we get a similar dependence on T and B. [43] also give batched algorithms for stochastic linear bandits and adversarial multi-armed bandits.

Recently, [99] designed a fully adaptive algorithm achieving an optimal regret of $\sum_{j:\Delta_j>0} \frac{O(\log T)}{\Delta_j}$ for dueling bandits under the Condorcet setting. This algorithm is based on the idea of *dueling* two classical bandit (MAB) algorithms against each other in a repeated zero-sum game with carefully designed rewards. The reward for one algorithm depends on the actions of the other; hence, these algorithms need to achieve *best-of-both-worlds* guarantee for both stochastic and adversarial settings. However, the approach of [99] is not directly applicable to the *batched* setting that we consider. This is because, as shown by [43], any B-round algorithm for batched MAB in the adversarial setting has regret $\Omega(T/B)$.

Adaptivity and batch processing has been recently studied for stochastic submodular cover [56, 2, 44, 51], and for various stochastic "maximization" problems such as knapsack [38, 23], matching [15, 19], probing [62] and orienteering [59, 61, 16]. Recently, there have also been several results examining the role of adaptivity in (deterministic) submodular optimization; e.g. [13, 11, 14, 12, 30].

5.5 All Pairs Comparison Algorithm

In this section, we present a first algorithm, namely `PCOMP`, for the K-armed batched dueling bandits problem. Recall that given a set of K bandits (or arms) $\mathcal{B} = \{b_1, \ldots, b_K\}$, and a positive integer $B \leq T$, we wish to find a sequence of B batches of noisy comparisons with low regret. Given bandits b_i and b_j, $P_{i,j} = \frac{1}{2} + \Delta_{i,j}$ denotes the probability of b_i winning over b_j. Our pairwise comparison algorithm, `PCOMP`, proceeds by performing all-pairs comparisons amongst bandits in an *active* set, and gradually eliminating sub-optimal bandits.

Before describing our algorithm in detail we will set up some basic notation. We will denote by $\mathcal{A}$ the set of *active* arms, i.e. arms that have not been eliminated. We will use index r for rounds or batches. At the end of each round r, our algorithms compute a fresh estimate of the pairwise probabilities based on the feedback from comparisons in round r as:

$$\widehat{P}_{i,j} = \frac{\#b_i \text{ wins against } b_j \text{ in round } r}{\#\text{comparisons of } b_i \text{ and } b_j \text{ in round } r}\,. \tag{5.1}$$

If a pair (b_i, b_j) is compared in round r, it is compared $c_r = \lfloor q^r \rfloor$ times. In round r, the parameter $\gamma_r = \sqrt{\log\left(\frac{1}{\delta}\right)/2c_r}$ is used to eliminate bandits from the active set (the specific elimination criteria depends on the algorithm).

Algorithm 10 PCOMP(All Pairs Comparisons Algorithm)

```
1: Input: Bandits $\mathcal{B}$, time-horizon $T$, rounds $B$, comparison parameters $q$ and $\tau$
2: $K \leftarrow |\mathcal{B}|$, $\delta \leftarrow \frac{1}{2TK^2B}$, active bandits $\mathcal{A} \leftarrow \mathcal{B}$, $c_r \leftarrow \lfloor q^{r+\tau-1} \rfloor$, $\gamma_r \leftarrow \sqrt{\log(1/\delta)/2c_r}$, $r \leftarrow 1$
3: while number of comparisons $\leq T$ do
4:     for all $(b_i, b_j) \in \mathcal{A}^2$, perform $c_r$ comparisons and compute $\widehat{P}_{i,j}$ using Eq(5.1).
5:     if $\exists\, b_i, b_j$ such that $\widehat{P}_{i,j} > \frac{1}{2} + \gamma_r$ then
6:         $\mathcal{A} \leftarrow \mathcal{A} \setminus \{b_j\}$                ▷ delete $b_j$ from $\mathcal{A}$
7:     $r \leftarrow r + 1$
```

5.5.1 The Algorithm

We are now ready to describe the PCOMP algorithm. This algorithm takes as input the set of bandits $\mathcal{B}$, time-horizon T, rounds B and comparison parameters q and τ. We will set the parameters $q = T^{1/B}$ and $\tau = 1$, unless otherwise specified.[1] In round $r \in [B]$, this algorithm compares each pair $(b_i, b_j) \in \mathcal{A}^2$ for c_r times. It then computes fresh estimates of the pairwise probabilities $\widehat{P}_{i,j}$ for all $(b_i, b_j) \in \mathcal{A}^2$. If, for some bandit b_j, there exists bandit b_i such that $\widehat{P}_{i,j} > \frac{1}{2} + \gamma_r$, then bandit b_j is eliminated from $\mathcal{A}$. We provide the pseudo-code in Algorithm 10.

The following theorem (proved in §5.5.2) describes the regret bound obtained by PCOMP under the Condorcet assumption, and formalizes Theorem 5.3.1.

Theorem 5.5.1. *Given any set $\mathcal{B}$ of K bandits, time-horizon T, rounds B, parameters $q = T^{1/B}$ and $\tau = 1$, the expected regret of* PCOMP *for the batched K-armed dueling bandits problem under the Condorcet assumption is at most*

$$\mathbb{E}[R(T)] \leq 3KT^{1/B} \log\left(2TK^2B\right) \sum_{j:\Delta_j > 0} \frac{1}{\Delta_j}.$$

Setting $\Delta_{\min} := \min_{j:\Delta_j>0} \Delta_j$, we get

$$\mathbb{E}[R(T)] \leq \frac{3K^2T^{1/B} \log\left(2TK^2B\right)}{\Delta_{\min}}.$$

[1]We allow general parameters q and τ in order to allow PCOMP to be used in conjunction with other policies.

5.5.2 Regret Analysis

We present the regret analysis for `PCOMP` in this section. We first prove the following lemma which denotes a "good" event under which we will analyze the regret of the algorithm.

Lemma 5.5.2. *For any batch $r \in [B]$, and for any pair b_i, b_j that are compared c_r times, we have*

$$\mathbf{P}\left(|P_{i,j} - \widehat{P}_{i,j}| > \gamma_r\right) \leq 2\delta,$$

where $\gamma_r = \sqrt{\log(\frac{1}{\delta})/2c_r}$.

Proof. Note that $\mathbb{E}[\widehat{P}_{i,j}] = P_{i,j}$, and applying Hoeffding's inequality gives

$$\mathbf{P}\left(|\widehat{P}_{i,j} - P_{i,j}| > \gamma_r\right) \leq 2\exp\left(-2c_r \cdot \gamma_r^2\right) = 2\delta.$$

□

We analyze the regret of our algorithm under a *good* event, G. We show that the G occurs with high probability; in the event that G does not occur (denoted $\overline{G}$), we incur a regret of T. Towards defining G, we say that an estimate $\widehat{P}_{i,j}$ at the end of batch r is *correct* if $|\widehat{P}_{i,j} - P_{i,j}| \leq \gamma_r$. We say that G occurs if every estimate in every batch is correct.

Lemma 5.5.3. *The probability that every estimate in every batch of* `PCOMP` *is correct is at least $1 - 1/T$.*

Proof. Applying Lemma 5.5.2 and taking a union bound over all pairs and batches, we get that the probability that some estimate is incorrect is at most $K^2 \times B \times 2\delta = \frac{1}{T}$ where $\delta = 1/2K^2BT$. Thus, $\mathbf{P}(\overline{G}) \leq \frac{1}{T}$. □

Using Lemma 5.5.3, the expected regret (of *any* algorithm) can be written as follows:

$$\begin{aligned}\mathbb{E}[R(T)] &= \mathbb{E}[R(T) \mid G] \cdot \mathbf{P}(G) + \mathbb{E}[R(T) \mid \overline{G}] \cdot \mathbf{P}(\overline{G}) \\ &\leq \mathbb{E}[R(T) \mid G] + T \cdot \frac{1}{T} = \mathbb{E}[R(T) \mid G] + 1 \qquad (5.2)\end{aligned}$$

Proof of Theorem 5.5.1. First, recall that in each batch of `PCOMP` every pair of active arms is compared c_r times where $c_r = \lfloor q^r \rfloor$ with $q = T^{1/B}$. Since, $q^B = T$, `PCOMP` uses at most B batches.

Following Lemma 5.5.3 and (5.2), we only need to bound $\mathbb{E}[R(T) \mid G]$. Given G, whenever $P_{i,j} > \frac{1}{2} + 2\gamma_r$ (that is $\Delta_{i,j} > 2\gamma_r$), we have $\widehat{P}_{i,j} > \frac{1}{2} + \gamma_r$: so bandit b_j will be eliminated by b_i. Furthermore, given bandits b_i and b_j such that $b_i \succeq b_j$, b_i will never be eliminated by b_j under event G. This implies that b_1 is never eliminated: this is crucial as we use b_1 as an anchor to eliminate sub-optimal bandits. Recall that the regret can be written as follows:

$$R(T) = \frac{1}{2}\sum_{j=1}^{K} T_j \Delta_{1,j}$$

where T_j is the number of comparisons that b_j partakes in. We proceed by bounding T_j. Towards this end, let $T_{1,j}$ be a random variable denoting the number of comparisons performed between b_1 and b_j. As b_1 is never eliminated, $T_j \le K \cdot T_{1,j}$. Let r denote the last round such that b_j survives round r, i.e., $b_j \in \mathcal{A}$ at the end of round r. We can then conclude that $\Delta_j := \Delta_{1,j} \le 2\gamma_r$ (else b_1 would eliminate b_j in round r). We get

$$\Delta_j \le 2 \cdot \sqrt{\frac{\log(\frac{1}{\delta})}{2c_r}}$$

which on squaring and re-arranging gives:

$$c_r \le \frac{2\log\left(\frac{1}{\delta}\right)}{\Delta_j^2} \tag{5.3}$$

Now, note that b_j could have been played for at most one more round. Thus, we have

$$T_{1,j} = \sum_{\tau=1}^{r+1} c_\tau \le q \sum_{\tau=0}^{r} c_\tau \le 2q \cdot c_r$$

where the final inequality follows from summing up $\sum_{\tau=1}^{r-1} c_\tau$, and using $B \le \log(T)$. Then, we have $T_j \leq 2Kq \cdot c_r$. Using 5.3, and plugging in $q = T^{1/B}$ and $\delta = 1/6TK^2B$ we have

$$\begin{aligned} E[R(T) \mid G] &\leq \frac{1}{2}\sum_j \left(2KT^{1/B} \cdot \frac{2\log\left(6TK^2B\right)}{\Delta_j^2}\right) \cdot \Delta_j \\ &= \sum_{j:\Delta_j>0} \frac{KT^{1/B}\log\left(6TK^2B\right)}{\Delta_j} \\ &= 2KT^{1/B}\log\left(6TK^2B\right) \sum_{j:\Delta_j>0} \frac{1}{\Delta_j}. \end{aligned}$$

Note that when $\Delta_j = 0$ for $b_j \in \mathcal{B}$, we exclude the corresponding term in the regret bound. Combining this with (5.2) gives the first bound of Theorem 5.5.1. Plugging in $\Delta_{\min} = \min_{j:\Delta_j>0} \Delta_j$ completes the proof. □

Chapter 6

Algorithms using Seeded Comparisons for Batched Dueling Bandits

6.1 Overview

In this chapter we provide detailed proofs, and formalize Theorems 5.3.2,5.3.3, 5.3.4 and 5.3.5. The algorithms for the first three theorems are based on a seeded comparisons idea, and provide improved regret bounds when the dueling bandit instance satisfies *strong stochastic transitivity* and *stochastic triangle inequality*. The lower bound instance of Theorem 5.3.5 applies even under the SST and STI assumptions.

6.1.1 Results and Techniques

We restate the main theorems proved in this chapter, and provide the high-level ideas used in proving them.

Theorem 5.3.2. *For any integer $B \geq 1$, there is an algorithm for batched dueling bandits that uses at most $B+1$ rounds, and if the instance satisfies the SST and STI assumptions, the expected regret is bounded by*

$$\mathbb{E}[R(T)] = \sum_{j:\Delta_j>0} O\left(\frac{\sqrt{K}T^{1/B}\log(T)}{\Delta_j}\right).$$

The idea behind this algorithm is to first sample a "sufficiently small" *seed set*, and then to perform all pairwise comparisons between the seed set and the active set to eliminate sub-optimal arms. The idea is to exploit the structure of pairwise probabilities so that we do not need to perform *all* pairwise comparisons. Additionally, if the seed set is found to be sub-optimal, we can construct a *much smaller* active set; thus allowing us to switch to the pairwise comparison policy. In the sequential setting, [112] obtain instance-dependent regret bounded by $\sum_{j:\Delta_j>0} O\left(\frac{\log(T)}{\Delta_j}\right)$. Our result nearly matches this sequential bound (with an extra multiplicative factor of $\sqrt{K}$) when $B = \log(T)$. Observe that the worst-case regret of [114] in the sequential setting is bounded by $O\left(\frac{K\log(T)}{\Delta_{\min}}\right)$, while we obtain $\mathbb{E}[R(T)] \le O\left(\frac{K\sqrt{K}T^{1/B}\log(T)}{\Delta_{\min}}\right)$.

Next, we improve the worst-case regret by reducing the comparisons performed as follows. We first perform pairwise comparisons amongst bandits in the seed set, and pick a candidate bandit. This candidate bandit is used to eliminate sub-optimal arms from the active set. Although selecting a candidate bandit each time requires additional adaptivity, we get a better bound on the worst-case expected regret by exploiting the fact that there can be at most B candidate bandits.

Theorem 5.3.3. *For any integer $B \ge 1$, there is an algorithm for batched dueling bandits that uses at most $2B+1$ rounds, and if the instance satisfies the SST and STI assumptions, the expected worst-case regret is bounded by*

$$\mathbb{E}[R(T)] = O\left(\frac{KBT^{1/B}\log(T)}{\Delta_{\min}}\right).$$

Thus, in $B = \log(T)$ rounds, our expected worst-case regret is bounded by $E[R(T)] \le O\left(\frac{K\log^2(T)}{\Delta_{\min}}\right)$ matching the best known result in the sequential setting up to an additional logarithmic factor.

We also improve the instance-dependent regret bound in Theorem 5.3.2 by using a few additional rounds. In particular, using the approach in Theorem 5.3.2 along with recursion, we obtain:

Theorem 5.3.4. *For integers $B \ge 1$, $m \ge 0$ and parameter $\eta \in (0,1)$, there is an algorithm for batched dueling bandits that uses at most $B+m$ rounds, and if the instance satisfies the*

SST and STI assumptions, the expected regret is bounded by

$$\mathbb{E}[R(T)] = O\left(m \cdot K^{\eta} + K^{(1-\eta)^m}\right) \cdot T^{1/B} \log(KTB) \sum_{j:\Delta_j>0} \frac{1}{\Delta_j}.$$

Thus, for any constant $\eta \in (0,1)$, setting $m = \frac{1}{\eta}\log\left(\frac{1}{\eta}\right)$, we obtain expected regret bounded by

$$\mathbb{E}[R(T)] = O\left(K^{\eta}\ T^{1/B}\ \log(T)\right) \sum_{j:\Delta_j>0} \frac{1}{\Delta_j}$$

in at most $B + \frac{1}{\eta}\log\left(\frac{1}{\eta}\right)$ rounds. Conversely, given a value of m, we can appropriately select η to minimize the regret.

The idea behind this algorithm is to use a seed-set of size K^{η}, and to recurse when the seed-set is found to be sub-optimal. We bound the number of recursive calls by m (which ensures that there are at most $B + m$ rounds) and show that the active set shrinks by a shrinks by a power of $(1-\eta)$ in each recursive call (which is used to bound regret).

Finally, we complement our upper bound results with a lower bound for the batched K-armed dueling bandits problem, even under the SST and STI assumptions.

Theorem 5.3.5. *Given an integer $B \geq 1$, and any algorithm that uses at most B batches, there exists an instance of the K-armed batched dueling bandit problem that satisfies the SST and STI condition such that the expected regret*

$$\mathbb{E}[R(T)] = \Omega\left(\frac{KT^{1/B}}{B^2 \Delta_{\min}}\right).$$

The above lower bound shows that the $T^{1/B}$ dependence in our upper bounds is necessary. Note that the above lower bound also applies to the more general Condorcet winner setting. The proof is similar to the lower bound proof in [48] for batched multi-armed bandits. The main novelty in our proof is the design of a family of hard instances with different values of $\Delta_{\min}$'s that satisfy the SST and STI conditions.

Organization of the Chapter. The rest of the chapter is organized as follows. In §6.2 we provide details about two seeded comparisons algorithms that proves Theorems 5.3.2

and 5.3.3. In §6.3, we extend these ideas to prove a recursive algorithm that achieves the guarantees of Theorem 5.3.4. In §6.4, we provide computational results, and we conclude in §6.5 with a lower bound, proving Theorem 5.3.5.

6.2 The Algorithms

In this section, we present two algorithms, namely `SCOMP` and `SCOMP2`, for the K-armed batched dueling bandits problem. Recall that given a set of K bandits (or arms) $\mathcal{B} = \{b_1, \ldots, b_K\}$, and a positive integer $B \leq T$, we wish to find a sequence of B batches of noisy comparisons with low regret. Given bandits b_i and b_j, $P_{i,j} = \frac{1}{2} + \Delta_{i,j}$ denotes the probability of b_i winning over b_j. At a high0-level, both `SCOMP` and `SCOMP2`, first select a (sufficiently small) *seed* set $\mathcal{S} \subseteq \mathcal{B}$, and eliminate bandits in an *active* set by successively comparing them to (all or few) bandits in $\mathcal{S}$. If the seed set $\mathcal{S}$ is itself found to be *sub-optimal* in a subsequent round, then these algorithms call the all-pairs algorithm `PCOMP`(see §5.5 for details) over the remaining *active* arms.

Before describing our algorithms, we recall some relevant notation. We denote by $\mathcal{A}$ the set of *active* arms, i.e. arms that have not been eliminated. We use index r for rounds or batches. At the end of each round r, our algorithms compute a fresh estimate of the pairwise probabilities based on the feedback from comparisons in round r as:

$$\widehat{P}_{i,j} = \frac{\#b_i \text{ wins against } b_j \text{ in round } r}{\#\text{comparisons of } b_i \text{ and } b_j \text{ in round } r}\,. \tag{6.1}$$

If a pair (b_i, b_j) is compared in round r, it is compared $c_r = \lfloor q^r \rfloor$ times. In round r, the parameter $\gamma_r = \sqrt{\log\left(\frac{1}{\delta}\right)/2c_r}$ is used to eliminate bandits from the active set (the specific elimination criteria depends on the algorithm).

6.2.1 Seeded Comparisons Algorithms

In this section, we present two algorithms for the batched dueling bandits problem, namely `SCOMP` and `SCOMP2`. The algorithms work in *two phases*:

- In the first phase, the algorithms sample a *seed set* $\mathcal{S}$ by including each bandit from $\mathcal{B}$

independently with probability $1/\sqrt{K}$. This seed set is used to eliminate bandits from the active set $\mathcal{A}$.

- Under certain *switching* criteria, the algorithms enter the second phase which involves running algorithm PCOMP on some of the remaining bandits.

The algorithms differ in how the candidate set is used to eliminate active bandits in the first phase.

In SCOMP, *all* pairwise comparisons between $\mathcal{S}$ (seed set) and $\mathcal{A}$ (active bandits) are performed. Specifically, in round r, every active bandit is compared with every bandit in $\mathcal{S}$ for c_r times. If, for some bandit b_j, there exists bandit b_i such that $\widehat{P}_{i,j} > \frac{1}{2}+3\gamma_r$, then bandit b_j is eliminated (from $\mathcal{A}$ as well as $\mathcal{S}$); note that the elimination criteria here is stricter than in PCOMP. If, in some round r, there exists bandit b_j such that b_j eliminates *all* bandits $b_i \in \mathcal{S}$, then the algorithm constructs a set $\mathcal{A}^* = \{b_j \in \mathcal{A} \mid \widehat{P}_{j,i} > \frac{1}{2} + \gamma_r \text{ for all } b_i \in \mathcal{S}\}$, and invokes PCOMP on bandits $\mathcal{A}^*$ with starting batch r. This marks the beginning of the second phase, which continues until time T. We provide the pseudocode in Algorithm 11.

Algorithm 11 SCOMP(Seeded Comparisons Algorithm)

```
1: Input: Bandits $\mathcal{B}$, time-horizon $T$, rounds $B$
2: $q \leftarrow T^{1/B}$, $\delta \leftarrow \frac{1}{6TK^2B}$, active bandits $\mathcal{A} \leftarrow \mathcal{B}$, $c_r \leftarrow \lfloor q^r \rfloor$, $\gamma_r \leftarrow \sqrt{\log(1/\delta)/2c_r}$, $r \leftarrow 1$
3: $\mathcal{S} \leftarrow$ add elements from $\mathcal{B}$ into $\mathcal{S}$ w.p. $1/\sqrt{K}$
4: while number of comparisons $\leq T$ do                                        ▷ phase I
5:     for all $(b_i, b_j) \in \mathcal{S} \times \mathcal{A}$, compare $b_i$ and $b_j$ for $c_r$ times and compute $\widehat{P}_{i,j}$
6:     if $\exists b_i \in \mathcal{S}$, $b_j \in \mathcal{A}$, $\widehat{P}_{i,j} > \frac{1}{2} + 3\gamma_r$ then                     ▷ elimination
7:         $\mathcal{A} \leftarrow \mathcal{A} \setminus \{b_j\}$, $\mathcal{S} \leftarrow \mathcal{S} \setminus \{b_j\}$
8:     if $\exists b_j$ such that $\widehat{P}_{j,i} > \frac{1}{2} + 3\gamma_r$ for all $b_i \in \mathcal{S}$ then                ▷ switching
9:         construct set $\mathcal{A}^* = \{b_j \in \mathcal{A} \mid \widehat{P}_{j,i} > \frac{1}{2} + \gamma_r$ for all $b_i \in \mathcal{S}\}$
10:        $r^* \leftarrow r$, $T^* \leftarrow$ # comparisons until round $r^*$, break
11:    $r \leftarrow r + 1$
12: run PCOMP($\mathcal{A}^*, T - T^*, q, r^*$)                                   ▷ phase II
```

We obtain the following result (proved in §6.2.2) when the given instance satisfies SST and STI. This formalizes Theorem 5.3.2.

Theorem 6.2.1. *Given any set $\mathcal{B}$ of K bandits, time-horizon T, parameter B,* SCOMP *uses* *at most $B+1$ batches, and has expected regret bounded by*

$$\mathbb{E}[R(T)] = \sum_{j:\Delta_j>0} O\left(\frac{\sqrt{K}T^{1/B}\log(T)}{\Delta_j}\right)$$

under the strong stochastic transitivity and stochastic triangle inequality assumptions.

Observe that this gives a worst-case regret bound of $O\left(\frac{K\sqrt{K}T^{1/B}\log(T)}{\Delta_{\min}}\right)$ for `SCOMP` under SST and STI. We can improve this by sampling each bandit from $\mathcal{B}$ independently into the seed set with probability $K^{-2/3}$: this gives a worst-case regret bound of $O\left(\frac{K^{4/3}T^{1/B}\log(T)}{\Delta_{\min}}\right)$ in $B+1$ rounds. To further improve this worst-case bound, we *add more rounds of adaptivity* in `SCOMP` to obtain `SCOMP2`. Specifically, each round r in the first phase is divided into two rounds of adaptivity.

- In the first round $r^{(1)}$, pairwise comparisons among the bandits in $\mathcal{S}$ are performed, and an undefeated $b_{i_r^*}$ is selected as a *candidate*. We say that b_i *defeats* b_j if $\widehat{P}_{i,j} > \frac{1}{2} + \gamma_r$
- In the second round $r^{(2)}$, the candidate $b_{i_r^*}$ is used to eliminate active bandits. A bandit b_j is eliminated if $\widehat{P}_{i_r^*,j} > \frac{1}{2} + 5\gamma_r$.

The switching criterion in `SCOMP2` is different from that of `SCOMP`. Here, if in some round r, there is a bandit b_j such that b_j eliminates $b_{i_r^*}$, then the algorithm constructs set $\mathcal{A}^* = \{b_j \in \mathcal{A} \mid \widehat{P}_{j,i_r^*} > \frac{1}{2} + 3\gamma_r\}$, and invokes `PCOMP` on bandits $\mathcal{A}^*$ with starting batch r. See Algorithm 12 for a formal description.

In §6.2.2, we prove that `SCOMP2` obtains an improved worst-case regret bound (at the cost of additional adaptivity) over `SCOMP` when the given instance satisfies SST and STI, thus proving Theorem 5.3.3 (formalized in Theorem 6.2.2).

Theorem 6.2.2. *Given any set $\mathcal{B}$ of K bandits, time-horizon T and parameter B,* `SCOMP`*2 uses at most $2B+1$ batches, and has worst-case expected regret bounded by*

$$\mathbb{E}[R(T)] = O\left(\frac{KBT^{1/B}\log(T)}{\Delta_{\min}}\right)$$

under strong stochastic transitivity and stochastic triangle inequality, where $\Delta_{\min} := \min_{j:\Delta_j>0}\Delta_j$.

Algorithm 12 SCOMP2 (Seeded Comparisons Algorithm 2)

```
1: Input: Bandits $\mathcal{B}$, time-horizon $T$, rounds $B$
2: $q \leftarrow T^{1/B}$, $\delta \leftarrow \frac{1}{6TK^2B}$, active bandits $\mathcal{A} \leftarrow \mathcal{B}$, $c_r \leftarrow \lfloor q^r \rfloor$, $\gamma_r \leftarrow \sqrt{\log(1/\delta)/2c_r}$, $r \leftarrow 1$
3: $\mathcal{S} \leftarrow$ add elements from $\mathcal{B}$ into $\mathcal{S}$ with probability $1/\sqrt{K}$
4: while number of comparisons $\le T$ do                                        ▷ phase I
5:     $r^{(1)}$: compare all pairs in $\mathcal{S}$ for $c_r$ times and compute $\widehat{P}_{i,j}$.
6:     candidate $b_{i_r^*} \leftarrow$ any bandit $i \in \mathcal{S}$ with $\max_{j\in\mathcal{S}} \widehat{P}_{j,i} \le \frac{1}{2} + \gamma_r$.      ▷ extra batch
7:     $r^{(2)}$: for all $b_j \in \mathcal{A}$, compare $b_{i_r^*}$ and $b_j$ for $c_r$ times and compute $\widehat{P}_{i_r^*,j}$.
8:     if $\exists b_j \in \mathcal{A}$, $\widehat{P}_{i_r^*,j} > \frac{1}{2} + 5\gamma_r$ then                           ▷ elimination
9:         $\mathcal{A} \leftarrow \mathcal{A} \setminus \{b_j\}$, $\mathcal{S} \leftarrow \mathcal{S} \setminus \{b_j\}$
10:    if $\exists b_j$ such that $\widehat{P}_{j,i_r^*} > \frac{1}{2} + 5\gamma_r$ then                           ▷ switching
11:        construct set $\mathcal{A}^* = \{b_j \in \mathcal{A} \mid \widehat{P}_{j,i_r^*} > \frac{1}{2} + 3\gamma_r\}$
12:        $r^* \leftarrow r$, $T^* \leftarrow$ # comparisons until round $r^*$, break
13:    $r \leftarrow r + 1$
14: run PCOMP$(\mathcal{A}^*, T - T^*, q, r^*)$                                   ▷ phase II
```

6.2.2 Regret Analysis

We present the regret analysis for the algorithms described in §6.2 in this section. We first prove the following lemma which will be used in the analysis of both algorithms.

Lemma 6.2.3. *For any batch $r \in [B]$, and for any pair b_i, b_j that are compared c_r times, we have*

$$\mathbf{P}\left(|P_{i,j} - \widehat{P}_{i,j}| > \gamma_r\right) \le 2\delta,$$

where $\gamma_r = \sqrt{\log(\frac{1}{\delta})/2c_r}$.

Proof. Note that $\mathbb{E}[\widehat{P}_{i,j}] = P_{i,j}$, and applying Hoeffding's inequality gives

$$\mathbf{P}\left(|\widehat{P}_{i,j} - P_{i,j}| > \gamma_r\right) \le 2\exp\left(-2c_r \cdot \gamma_r^2\right) = 2\delta.$$

□

We analyze the regret of our algorithms under a *good* event, G. We show that the G occurs with high probability; in the event that G does not occur (denoted $\overline{G}$), we incur a regret of T. Towards defining G, we say that an estimate $\widehat{P}_{i,j}$ at the end of batch r is *correct* if $|\widehat{P}_{i,j} - P_{i,j}| \le \gamma_r$. We say that G occurs if every estimate in every batch is correct.

Lemma 6.2.4. *The probability that every estimate in every batch of* SCOMP *and* SCOMP*2 is correct is at least* $1 - 1/T$.

Proof. Applying Lemma 6.2.3 and taking a union bound over all pairs and batches (note SCOMP2 has at most $2B + 1 \leq 3B$ batches), we get that the probability that some estimate is incorrect is at most $K^2 \times 3B \times 2\delta = \frac{1}{T}$ where $\delta = 1/6K^2BT$. Thus, $\mathbf{P}(\overline{G}) \leq \frac{1}{T}$. □

Using Lemma 6.2.4, the expected regret (of *any* algorithm) can be written as follows:

$$\begin{aligned}\mathbb{E}[R(T)] &= \mathbb{E}[R(T) \mid G] \cdot \mathbf{P}(G) + \mathbb{E}[R(T) \mid \overline{G}] \cdot \mathbf{P}(\overline{G}) \\ &\leq \mathbb{E}[R(T) \mid G] + T \cdot \frac{1}{T} = \mathbb{E}[R(T) \mid G] + 1 \end{aligned} \tag{6.2}$$

6.2.3 Proofs of Theorems 6.2.1 and 6.2.2

In this section, we provide the proofs of Theorem 6.2.1 and Theorem 6.2.2. Henceforth, we assume the SST and STI properties. We need the following definition. For a bandit b_j, let $E_j = \{b_i \in \mathcal{B} : \Delta_{i,j} > 0\}$; that is, the set of bandits superior to bandit b_j. We define $rank(b_j) = |E_j|$. [1]

As before, we analyze the regret of SCOMP and SCOMP2 under event G. By Lemma 6.2.4 and (6.2), we only need to bound the expected regret under G; that is, we need to bound $\mathbb{E}[R(T) \mid G]$. Conditioned on event G, the following Lemmas 6.2.5,6.2.6 and 6.2.7 hold for both SCOMP and SCOMP2.

Lemma 6.2.5. *The best bandit* b_1 *is never deleted from* $\mathcal{A}$ *in the elimination step of phase I.*

Proof. In SCOMP, b_i deletes b_j in batch r if $\widehat{P}_{i,j} > \frac{1}{2} + 3\gamma_r$, and in SCOMP2 if $\widehat{P}_{i,j} > \frac{1}{2} + 5\gamma_r$. If b_1 is deleted due to some bandit b_j, then by applying Lemma 6.2.3 (in either case), we get $P_{j,1} > \frac{1}{2} + 2\gamma_r$, a contradiction. □

Lemma 6.2.6. *When the algorithm switches to* PCOMP *on set* $\mathcal{A}^*$*, we have* $b_1 \in \mathcal{A}^*$ *and* $|\mathcal{A}^*| \leq rank(b_{i^*_{\mathcal{S}}})$ *where* $b_{i^*_{\mathcal{S}}}$ *is the best bandit in* $\mathcal{S}$*.*

[1] Note that SST and STI imposes a linear ordering on the bandits. So, we can assume $b_1 \succeq b_2 \succeq \cdots \succeq b_K$. Thus, $rank(b_j) < j$; that is, it is at most the number of bandits strictly preferred over b_j.

Proof. We first consider algorithm SCOMP. Here, the switching occurs when, in some batch r, there exists $b_{j^*} \in \mathcal{A}$ such that $\widehat{P}_{j^*,i} > \frac{1}{2} + 3\gamma_r$ for all $b_i \in \mathcal{S}$, Moreover, $\mathcal{A}^* = \{b_j \in \mathcal{A} \mid \widehat{P}_{j,i} > \frac{1}{2} + \gamma_r \text{ for all } b_i \in \mathcal{S}\}$. Consider any $b_i \in \mathcal{S}$. Given G, $\widehat{P}_{j^*,i} > \frac{1}{2} + 3\gamma_r$ implies that $P_{j^*,i} > \frac{1}{2} + 2\gamma_r$. By SST, $P_{1,i} \geq P_{j^*,i}$, and again using event G, $\widehat{P}_{1,i} > \frac{1}{2} + \gamma_r$. Thus, $b_1 \in \mathcal{A}^*$. We now bound $|\mathcal{A}^*|$. Let $b_{i^*_\mathcal{S}}$ be the best bandit in $\mathcal{S}$, i.e., the bandit of smallest rank. Consider any bandit $b_j \in \mathcal{A}^*$. We have $\widehat{P}_{j,i^*_\mathcal{S}} > \frac{1}{2} + \gamma_r$, which implies (by event G) that $P_{j,i^*_\mathcal{S}} > \frac{1}{2}$. So, we must have $b_j \succ b_{i^*_\mathcal{S}}$. Consequently, $\mathcal{A}^* \subseteq \{b_j \in \mathcal{B} : b_j \succ b_{i^*_\mathcal{S}}\}$, which implies $|\mathcal{A}^*| \leq rank(b_{i^*_\mathcal{S}})$.

We now consider SCOMP2. Here, we select an *undefeated* candidate bandit $b_{i^*_r}$ in batch r, and the algorithm switches if there exists $b_{j^*} \in \mathcal{A}$ such that $\widehat{P}_{j^*,i^*_r} > \frac{1}{2} + 5\gamma_r$. Moreover, $\mathcal{A}^* = \{b_j \in \mathcal{A} \mid \widehat{P}_{j,i^*_r} > \frac{1}{2} + 3\gamma_r\}$. Given G, we have $P_{j^*,i^*_r} > \frac{1}{2} + 4\gamma_r$. By SST and again applying G, we obtain $\widehat{P}_{1,i^*_r} > \frac{1}{2} + 3\gamma_r$. So, $b_1 \in \mathcal{A}^*$. We now argue that $|\mathcal{A}^*| \leq rank(b_{i^*_\mathcal{S}})$. Again, let $b_{i^*_\mathcal{S}}$ be the best bandit in $\mathcal{S}$. As $b_{i^*_r}$ is undefeated after round $r^{(1)}$, we have $\widehat{P}_{i^*_\mathcal{S},i^*_r} \leq \frac{1}{2} + \gamma_r$, which implies $P_{i^*_\mathcal{S},i^*_r} \leq \frac{1}{2} + 2\gamma_r$ (by event G). Now, consider any bandit $b_j \in \mathcal{A}^*$. We have $\widehat{P}_{j,i^*_\mathcal{S}} > \frac{1}{2} + 3\gamma_r$, which implies (by event G) that $P_{j,i^*_\mathcal{S}} > \frac{1}{2} + 2\gamma_r$. It follows that $b_j \succ b_{i^*_\mathcal{S}}$ for all $b_j \in \mathcal{A}^*$. Hence, $|\mathcal{A}^*| \leq rank(b_{i^*_\mathcal{S}})$. □

Lemma 6.2.7. *We have* $\mathbb{E}[rank(b_{i^*_\mathcal{S}})] \leq \sqrt{K}$ *and* $\mathbb{E}[rank(b_{i^*_\mathcal{S}})^2] \leq 2K$.

Proof. Let R be a random variable denoting $rank(b_{i^*_\mathcal{S}})$. Note that $R = k$ if, and only if, the first $k-1$ bandits are not sampled into $\mathcal{S}$, and the k^{th} bandit is sampled into $\mathcal{S}$. Thus, R is a geometric random variable with success probability $p := \frac{1}{\sqrt{K}}$.[2] Recall that the mean and variance of a geometric random variable are $\frac{1}{p}$ and $\frac{1}{p^2} - \frac{1}{p}$ respectively. So, $\mathbb{E}[R] \leq \frac{1}{p} = \sqrt{K}$. Moreover, $\mathbb{E}[R^2] \leq \frac{2}{p^2} = 2K$. □

Using Lemmas 6.2.5, 6.2.6 and 6.2.7, we complete the proofs of Theorems 6.2.1 and 6.2.2.

Proof of Theorem 6.2.1. We bound the expected regret of SCOMP conditioned on G. Let R_1 and R_2 denote the regret incurred in phase I and II respectively.

[2]Strictly speaking, R is truncated at K.

Bounding R_1. Fix a bandit b_j. Let r denote the last round such that $b_j \in \mathcal{A}$ *and* switching does not occur (at the end of round r). Let $b_{i^*_{\mathcal{S}}}$ be the best bandit in $\mathcal{S}$. As b_j is not eliminated by $b_{i^*_{\mathcal{S}}}$, we have $\widehat{P}_{i^*_{\mathcal{S}},j} \le \frac{1}{2} + 3\gamma_r$, which implies (by event G) $P_{i^*_{\mathcal{S}},j} \le \frac{1}{2} + 4\gamma_r$. Moreover, as switching doesn't occur, we have $\min_{i\in\mathcal{S}} \widehat{P}_{1,i} \le \frac{1}{2} + 3\gamma_r$ (by Lemma 6.2.5, b_1 is never deleted from $\mathcal{A}$). We now claim that $P_{1,i^*_{\mathcal{S}}} \le \frac{1}{2} + 4\gamma_r$. Otherwise, by SST we have $\min_{i\in\mathcal{S}} P_{1,i} = P_{1,i^*_{\mathcal{S}}} > \frac{1}{2} + 4\gamma_r$, which (by event G) implies $\min_{i\in\mathcal{S}} \widehat{P}_{1,i} > \frac{1}{2} + 3\gamma_r$, a contradiction! It now follows that $\Delta_{i^*_{\mathcal{S}},j} \le 4\gamma_r$ and $\Delta_{1,i^*_{\mathcal{S}}} \le 4\gamma_r$. Consider now two cases:

1. $b_1 \succeq b_{i^*_{\mathcal{S}}} \succeq b_j$. Then, by STI, $\Delta_{1,j} \le 8\gamma_r$, and

2. $b_1 \succeq b_j \succeq b_{i^*_{\mathcal{S}}}$. Then, by SST $\Delta_{1,j} \le \Delta_{i^*_{\mathcal{S}},j} \le 4\gamma_r$.

In either case, we have $\Delta_j = \Delta_{1,j} \le 8\gamma_r$, which implies $c_r \le \frac{\log(1/\delta)}{2\gamma_r^2} \le \frac{32\log(1/\delta)}{\Delta_j^2}$.

Now, let T_j be a random variable denoting the number of comparisons of b_j with other bandits before switching. By definition of round r, bandit b_j will participate in at most one round after r (in phase I). So, we have

$$T_j \le \begin{cases} |\mathcal{S}| \cdot \sum_{\tau=1}^{r+1} c_\tau & \text{if } b_j \notin \mathcal{S} \\ K \cdot \sum_{\tau=1}^{r+1} c_\tau & \text{if } b_j \in \mathcal{S} \end{cases}$$

Taking expectation over $\mathcal{S}$, we get

$$\begin{aligned} \mathbb{E}[T_j] &\le \mathbb{E}\left[K\sum_{\tau=1}^{r+1} c_\tau \,|\, b_j \in \mathcal{S}\right] \cdot \mathbf{P}(b_j \in \mathcal{S}) + \mathbb{E}\left[|\mathcal{S}|\sum_{\tau=1}^{r+1} c_\tau \,|\, b_j \notin \mathcal{S}\right] \cdot \mathbf{P}(b_j \notin \mathcal{S}) \\ &\le \left(K\sum_{\tau=1}^{r+1} c_\tau\right) \cdot \frac{1}{\sqrt{K}} + \mathbb{E}[|\mathcal{S}| \,|\, b_j \notin \mathcal{S}] \cdot \sum_{\tau=1}^{r+1} c_\tau \;\le\; 2\sqrt{K}\sum_{\tau=1}^{r+1} c_\tau, \end{aligned}$$

where the third inequality uses $\mathbb{E}[|\mathcal{S}| \,|\, b_j \notin \mathcal{S}] \le \sqrt{K}$. Moreover,

$$\sum_{\tau=1}^{r+1} c_\tau \le 2T^{1/B} \cdot c_r = O\left(\frac{T^{1/B}\log(1/\delta)}{\Delta_j^2}\right).$$

Thus,

$$\mathbb{E}[R_1] = \sum_j \mathbb{E}[T_j] \cdot \Delta_j = \sum_{j:\Delta_j>0} O\left(\frac{T^{1/B}\sqrt{K}\log(6K^2TB)}{\Delta_j}\right) \tag{6.3}$$

Bounding R_2. We now bound the regret after switching. From Lemmas 6.2.5 and 6.2.6, we know that b_1 is never deleted, $b_1 \in \mathcal{A}^*$, and $|\mathcal{A}^*| \leq rank(b_{i^*_S})$. For any $\mathcal{A}^*$, applying Theorem 5.5.1 we get,

$$R_2 \leq 3|\mathcal{A}^*|T^{1/B}\log(6T|\mathcal{A}^*|^2 B) \sum_{j \in \mathcal{A}^* : \Delta_j > 0} \frac{1}{\Delta_j} \leq 3|\mathcal{A}^*|T^{1/B}\log(6TK^2B) \sum_{j \in \mathcal{B} : \Delta_j > 0} \frac{1}{\Delta_j}$$

By Lemma 6.2.7, $\mathbb{E}[|\mathcal{A}^*|] \leq \sqrt{K}$, hence

$$\mathbb{E}[R_2] \leq 3\sqrt{K}T^{1/B}\log(6TK^2B) \sum_{j : \Delta_j > 0} \frac{1}{\Delta_j} \tag{6.4}$$

Combining (6.3) and (6.4), we get

$$\mathbb{E}[R(T)|G] = \sum_{j : \Delta_j > 0} O\left(\frac{T^{1/B}\sqrt{K}\log(6K^2TB)}{\Delta_j}\right),$$

and by (6.2), this concludes the proof. □

Proof of Theorem 6.2.2. We bound the expected regret conditioned on G. Let R_1 and R_2 denote the regret incurred in phase I and II respectively.

Bounding R_1. Fix a bandit b_j. Let r denote any round such that $b_j \in \mathcal{A}$ *and* switching does not occur (at the end of round r). As in the proof of Theorem 6.2.1, we first show that $c_r = O\left(\frac{\log(1/\delta)}{\Delta_j^2}\right)$. Recall that $b_{i^*_r}$ is the candidate in round r. As b_j is not eliminated by $b_{i^*_r}$, we have $\widehat{P}_{i^*_r,j} \leq \frac{1}{2} + 5\gamma_r$, which implies (by event G) $P_{i^*_S,j} \leq \frac{1}{2} + 6\gamma_r$. Moreover, as switching doesn't occur, we have $\widehat{P}_{1,i^*_r} \leq \frac{1}{2} + 5\gamma_r$ (by Lemma 6.2.5, b_1 is never deleted from $\mathcal{A}$). By event G, we get $P_{1,i^*_r} \leq \frac{1}{2} + 6\gamma_r$. It now follows that $\Delta_{i^*_r,j} \leq 6\gamma_r$ and $\Delta_{1,i^*_r} \leq 6\gamma_r$. Consider now two cases:

1. $b_1 \succeq b_{i^*_r} \succeq b_j$. Then, by STI, $\Delta_{1,j} \leq 12\gamma_r$, and
2. $b_1 \succeq b_j \succeq b_{i^*_S}$. Then, by SST $\Delta_{1,j} \leq \Delta_{i^*_r,j} \leq 6\gamma_r$.

In either case, we have $\Delta_j = \Delta_{1,j} \leq 12\gamma_r$, which implies $c_r \leq \frac{\log(1/\delta)}{2\gamma_r^2} = O\left(\frac{\log(1/\delta)}{\Delta_j^2}\right)$.

We further divide R_1 into two kinds of regret: $R_1^{(c)}$ and $R_1^{(n)}$ where $R_1^{(c)}$ refers to the regret incurred by candidate arms and $R_1^{(n)}$ is the regret incurred by non-candidate arms.

Bounding $R_1^{(n)}$. For any bandit b_j, let T_j be a random variable denoting the number of comparisons of b_j (in phase I) when b_j is not a candidate. Also, let r be the last round such that $b_j \in \mathcal{A}$ and switching doesn't occur. So, b_j will participate in at most one round after r, and

$$T_j \leq \begin{cases} \sum_{\tau=1}^{r+1} c_\tau & \text{if } b_j \notin \mathcal{S} \\ |\mathcal{S}| \cdot \sum_{\tau=1}^{r+1} c_\tau & \text{if } b_j \in \mathcal{S} \end{cases}$$

Taking expectation over $\mathcal{S}$, we get

$$\begin{aligned} \mathbb{E}\left[T_j\right] &\leq \mathbb{E}\left[|\mathcal{S}| \sum_{\tau=1}^{r+1} c_\tau \mid b_j \in \mathcal{S}\right] \cdot \mathbf{P}(b_j \in \mathcal{S}) + \mathbb{E}\left[\sum_{\tau=1}^{r+1} c_\tau \mid b_j \notin \mathcal{S}\right] \cdot \mathbf{P}(b_j \notin \mathcal{S}) \\ &\leq \sum_{\tau=1}^{r+1} c_\tau \cdot \left(\frac{1}{\sqrt{K}} \cdot \mathbb{E}[|\mathcal{S}| \mid b_j \in \mathcal{S}] + 1\right) \;\leq\; (2 + \frac{1}{\sqrt{K}}) \cdot \sum_{\tau=1}^{r+1} c_\tau, \end{aligned}$$

where the third inequality uses $\mathbb{E}[|\mathcal{S}| \mid b_j \in \mathcal{S}] \leq 1 + \sqrt{K}$.

Moreover, using $c_r = O\left(\frac{\log(1/\delta)}{\Delta_j^2}\right)$, we have $\sum_{\tau=1}^{r+1} c_\tau = O\left(\frac{T^{1/B}\log(1/\delta)}{\Delta_j^2}\right)$. Thus,

$$\mathbb{E}[R_1^{(n)}] = \sum_j \mathbb{E}\left[T_j\right] \cdot \Delta_j \leq \sum_{j:\Delta_j>0} O\left(\frac{T^{1/B}\log\left(\frac{1}{\delta}\right)}{\Delta_{1,j}}\right) \leq O\left(\frac{T^{1/B} K \log\left(\frac{1}{\delta}\right)}{\Delta_{\min}}\right) \tag{6.5}$$

Bounding $R_1^{(c)}$. Observe that if b_j is a candidate in round r, then the regret incurred by b_j in round r is at most $Kc_r \cdot \Delta_{1,j}$. Also, $c_{r-1} \leq O\left(\frac{\log\left(\frac{1}{\delta}\right)}{\Delta_j^2}\right)$ because $b_j \in \mathcal{A}$ and switching hasn't occurred at end of round $r-1$. Thus, we have $c_r = T^{1/B} c_{r-1} \leq O\left(\frac{T^{1/B}\log\left(\frac{1}{\delta}\right)}{\Delta_j^2}\right)$. We can thus write

$$R_1^{(c)} = \sum_{r=1}^{B} \sum_j K c_r \cdot \Delta_j \cdot \mathbb{I}\left[i_r^* = j\right],$$

where $\mathbb{I}\left[i_r^* = j\right]$ is an indicator random variable denoting whether b_j was the candidate bandit in round r. Observe that there is exactly one candidate bandit, $b_{i_r^*}$, in each round. So,

$$\begin{aligned} R_1^{(c)} &= K\sum_{r=1}^{B} c_r \Delta_{i_r^*} \le K\sum_{r=1}^{B} O\left(\frac{T^{1/B}\log\left(\frac{1}{\delta}\right)}{\Delta_{i_r^*}^2}\right)\cdot \Delta_{i_r^*} \\ &= K\sum_{r=1}^{B} O\left(\frac{T^{1/B}\log\left(\frac{1}{\delta}\right)}{\Delta_{i_r^*}}\right) \le O\left(\frac{T^{1/B}KB\log\left(\frac{1}{\delta}\right)}{\Delta_{\min}}\right) \end{aligned} \tag{6.6}$$

Combining (6.5) and (6.6), we get

$$\mathbb{E}[R_1] \le O\left(\frac{T^{1/B}KB\log\left(\frac{1}{\delta}\right)}{\Delta_{\min}}\right) \tag{6.7}$$

Bounding R_2. Finally, we bound the regret in phase II where we only have bandits $\mathcal{A}^*$. From Lemmas 6.2.5 and 6.2.6, we know that $b_1 \in \mathcal{A}^*$, and $|\mathcal{A}^*| \le rank(b_{i_S^*})$. For any $\mathcal{A}^*$, applying Theorem 5.5.1 we get,

$$R_2 \le 3|\mathcal{A}^*|T^{1/B}\log(6T|\mathcal{A}^*|^2B)\sum_{j\in\mathcal{A}^*:\Delta_j>0}\frac{1}{\Delta_j} \le 3|\mathcal{A}^*|^2\cdot T^{1/B}\log(6TK^2B)\cdot\frac{1}{\Delta_{\min}}$$

By Lemma 6.2.7, $\mathbb{E}[|\mathcal{A}^*|^2] \le 2K$, and so:

$$\mathbb{E}[R_2] \le \frac{6T^{1/B}K\log(6TK^2B)}{\Delta_{\min}} \tag{6.8}$$

Finally, combining (6.7) and (6.8) completes the proof. □

6.3 A Recursive Algorithm for Batched Dueling Bandits

In this section, we describe a recursive algorithm, termed `R-SCOMP` for batched dueling bandits. The prior algorithms (`SCOMP` and `SCOMP2`) rely both on the seed set eliminating sub-optimal arms and on the fact that if the seed set is found to be sub-optimal, we can substantially *shrink the active set* and *switch* to the pairwise comparisons policy. we generalize SCOMP to R-SCOMP by requiring an input $m \geq 1$ and $\eta \in (0,1)$. At a high level, R-SCOMP recurses m times before switching to PCOMP. We maintain the property that each time it recurses, the active set shrinks by a factor of K^{η}. Note that when $m = 1$ and $\eta = 1/2$, we recover SCOMP.

The algorithm takes as input the set of bandits $\mathcal{B}$, time-horizon T, comparison parameters q and τ, integers B and m, and an accuracy parameter $\eta \in (0,1)$. Initially, we set $q = T^{1/B}$, $\tau = 1$ and $\delta = \frac{1}{2TK^2(B+m)}$. If $m = 0$, the algorithm executes PCOMP; else, the algorithm works in two phases:

- In the first phase, the algorithm samples a *seed set* $\mathcal{S}$ by including each bandit from $\mathcal{B}$ *independently* with probability $1/K^{1-\eta}$. This seed set is used to eliminate bandits from the active set $\mathcal{A}$ (like in SCOMP).
- Under a certain switching criteria, the algorithm recurses on the active set $\mathcal{A}$ with $m = m - 1$.

If $m \geq 1$, R-SCOMP performs all pairwise comparisons between $\mathcal{S}$ (seed set) and $\mathcal{A}$ (active bandits). Specifically, in round r, every active bandit is compared with every bandit in $\mathcal{S}$ for c_r times. If, for some bandit b_j, there exists bandit b_i such that $\widehat{P}_{i,j} > \frac{1}{2} + 3\gamma_r$, then bandit b_j is eliminated (from $\mathcal{A}$ as well as $\mathcal{S}$). If, in some round r, there exists bandit b_j such that b_j eliminates *all* bandits $b_i \in \mathcal{S}$, then the algorithm constructs a set $\mathcal{A}^* = \{b_j \in \mathcal{A} \mid \widehat{P}_{j,i} > \frac{1}{2} + \gamma_r \text{ for all } b_i \in \mathcal{S}\}$, and recurses R-SCOMP on bandits $\mathcal{A}^*$ with parameter $\tau = r$, and q and δ as set before. Additionally, the number of recursive calls is set to $m - 1$. Observe that the elimination and the switching criteria are the same as in SCOMP. Note that comparisons at round r are repeated when a recursive call is invoked, and since there are at most m recursive calls, R-SCOMP uses at most $B + m$ adaptive rounds. We describe the algorithm formally in Algorithm 13.

The following theorem, which formalizes Theorem 5.3.4, is the main result of this section.

Theorem 6.3.1. *Given any set $\mathcal{B}$ of K bandits, time-horizon T, integers B and m, parameters $q = T^{1/B}$, $r = 1$ and $\delta = \frac{1}{2TK^2(B+m)}$, and an accuracy parameter $\eta > 0$,* R-SCOMP *uses*

Algorithm 13 R-SCOMP(Recursive Seeded Comparisons Algorithm)

```
1:  Input: Bandits $\mathcal{B}$, time-horizon $T$, comparison parameters $q$, $\tau$, and $\delta$, #recursive calls
    $m$, accuracy $\eta$
2:  active bandits $\mathcal{A} \leftarrow \mathcal{B}$, $c_r \leftarrow \lfloor q^r \rfloor$, $\gamma_r \leftarrow \sqrt{\log(1/\delta)/2c_r}$, $r \leftarrow \tau$
3:  if $m = 0$ then                                                    ▷ base case
4:      run PCOMP($\mathcal{B}, T, q, \tau$)
5:  $\mathcal{S} \leftarrow$ add elements from $\mathcal{B}$ into $\mathcal{S}$ w.p. $1/K^{1-\eta}$
6:  while number of comparisons $\leq T$ do
7:      for all $(b_i, b_j) \in \mathcal{S} \times \mathcal{A}$, compare $b_i$ and $b_j$ for $c_r$ times and compute $\widehat{P}_{i,j}$
8:      if $\exists b_i \in \mathcal{S}$, $b_j \in \mathcal{A}$, $\widehat{P}_{i,j} > \frac{1}{2} + 3\gamma_r$ then                ▷ elimination
9:          $\mathcal{A} \leftarrow \mathcal{A} \setminus \{b_j\}$, $\mathcal{S} \leftarrow \mathcal{S} \setminus \{b_j\}$
10:     if $\exists b_j$ such that $\widehat{P}_{j,i} > \frac{1}{2} + 3\gamma_r$ for all $b_i \in \mathcal{S}$ then        ▷ switching
11:         construct set $\mathcal{A}^* = \{b_j \in \mathcal{A} \mid \widehat{P}_{j,i} > \frac{1}{2} + \gamma_r$ for all $b_i \in \mathcal{S}\}$
12:         $r^* \leftarrow r$, $T^* \leftarrow$ # comparisons until round $r^*$, break
13:     $r \leftarrow r + 1$
14: run R-SCOMP($\mathcal{A}^*, T - T^*, q, r^*, \delta, m - 1, \eta$)                    ▷ recursive call
```

at most $B + m$ batches, and has expected regret bounded by

$$\mathbb{E}[R(T)] = \sum_{j:\Delta_j > 0} O\left(\left(m \cdot K^{\eta} + K^{(1-\eta)^m}\right) \cdot \frac{T^{1/B}\log(6K^2TB)}{\Delta_j}\right)$$

under strong stochastic transitivity and stochastic triangle inequality.

6.3.1 The Analysis

We now provide the regret analysis of R-SCOMP, and prove Theorem 6.3.1. To keep the exposition in this section self-contained, we restate Lemma 6.2.3 which will be used to define a good event for R-SCOMP.

Lemma 6.3.2. *For any $r \in [B]$, and for any pair b_i, b_j that are compared c_r times, we have*

$$\mathbf{P}\left(|P_{i,j} - \widehat{P}_{i,j}| > \gamma_r\right) \leq 2\delta,$$

where $\gamma_r = \sqrt{\log(\frac{1}{\delta})/2c_r}$.

As before, we analyze the regret of R-SCOMP under event a *good* event, G. We show that G occurs with high probability; in the event that G does not occur (denoted $\overline{G}$), we incur a regret of T. Towards defining G, we say that an estimate $\widehat{P}_{i,j}$ at the end of batch r is *correct* if $|\widehat{P}_{i,j} - P_{i,j}| \leq \gamma_r$. We say that G occurs if every estimate in every batch is correct.

Lemma 6.3.3. *The probability that every estimate in the execution of* R-SCOMP *is correct is at least* $1 - 1/T$.

Proof. Applying Lemma 6.3.2 and taking a union bound over all pairs and batches (note R-SCOMP has at most $B + m$ batches), we get that the probability that some estimate is incorrect is at most $K^2 \times (B + m) \times 2\delta = \frac{1}{T}$ where $\delta = \frac{1}{2K^2T(B+m)}$. Thus, $\mathbf{P}(\overline{G}) \leq \frac{1}{T}$. $\square$

Using Lemma 6.3.3, the expected regret R-SCOMP can be written as follows:

$$\begin{aligned}\mathbb{E}[R(T)] &= \mathbb{E}[R(T) \mid G] \cdot \mathbf{P}(G) + \mathbb{E}[R(T) \mid \overline{G}] \cdot \mathbf{P}(\overline{G}) \\ &\leq \mathbb{E}[R(T) \mid G] + T \cdot \frac{1}{T} = \mathbb{E}[R(T) \mid G] + 1 \qquad (6.9)\end{aligned}$$

By Lemma 6.3.3 and (6.9), we only need to bound the expected regret under G; that is, we need to bound $\mathbb{E}[R(T) \mid G]$. Henceforth, we assume the SST and STI properties. Recall that for a bandit b_j, we define $E_j = \{b_i \in \mathcal{B} : \Delta_{i,j} > 0\}$; that is, the set of bandits superior to bandit b_j, and $rank(b_j) = |E_j|$. Conditioned on event G, the following Lemmas 6.3.4,6.3.5 and 6.3.6 hold for R-SCOMP.

Lemma 6.3.4. *The best bandit* b_1 *is never deleted.*

Proof. In R-SCOMP, b_i deletes b_j in batch r if $\widehat{P}_{i,j} > \frac{1}{2} + 3\gamma_r$. If b_1 is deleted due to some bandit b_j, then by applying Lemma 6.3.2, we get $P_{j,1} > \frac{1}{2} + 2\gamma_r$, a contradiction. $\square$

Lemma 6.3.5. *When the algorithm invokes a recursive call on* $\mathcal{A}^*$*, we have* $b_1 \in \mathcal{A}^*$ *and* $|\mathcal{A}^*| \leq rank(b_{i^*_{\mathcal{S}}})$ *where* $b_{i^*_{\mathcal{S}}}$ *is the best bandit in* $\mathcal{S}$.

Proof. Let $\mathcal{A}$ denote the set of active bandits in the some execution of R-SCOMP. Note that a recursive call is invoked when, in some batch r, there exists $b_{j^*} \in \mathcal{A}$ such that $\widehat{P}_{j^*,i} > \frac{1}{2} + 3\gamma_r$ for all $b_i \in \mathcal{S}$, Moreover, $\mathcal{A}^* = \{b_j \in \mathcal{A} \mid \widehat{P}_{j,i} > \frac{1}{2} + \gamma_r \text{ for all } b_i \in \mathcal{S}\}$. Consider any $b_i \in \mathcal{S}$. Given G, $\widehat{P}_{j^*,i} > \frac{1}{2} + 3\gamma_r$ implies that $P_{j^*,i} > \frac{1}{2} + 2\gamma_r$. By SST, $P_{1,i} \geq P_{j^*,i}$, and again using event G, $\widehat{P}_{1,i} > \frac{1}{2} + \gamma_r$. Thus, $b_1 \in \mathcal{A}^*$. We now bound $|\mathcal{A}^*|$. Let $b_{i^*_{\mathcal{S}}}$ be the best bandit in $\mathcal{S}$, i.e., the bandit of smallest rank. Consider any bandit $b_j \in \mathcal{A}^*$. We have $\widehat{P}_{j,i^*_{\mathcal{S}}} > \frac{1}{2} + \gamma_r$, which implies (by event G) that $P_{j,i^*_{\mathcal{S}}} > \frac{1}{2}$. So, we must have $b_j \succ b_{i^*_{\mathcal{S}}}$. Consequently, $\mathcal{A}^* \subseteq \{b_j \in \mathcal{B} : b_j \succ b_{i^*_{\mathcal{S}}}\}$, which implies $|\mathcal{A}^*| \leq rank(b_{i^*_{\mathcal{S}}})$. $\square$

Lemma 6.3.6. *We have* $\mathbb{E}[rank(b_{i^*_{\mathcal{S}}})] \leq K^{(1-\eta)}$.

Proof. The R be a random variable denoting $rank(b_{i^*_{\mathcal{S}}})$. Note that $R = k$ if, and only if, the first $k-1$ bandits are not sampled into $\mathcal{S}$, and the k^{th} bandit is sampled into $\mathcal{S}$. Thus, R is a geometric random variable with success probability $p := \frac{1}{K^{1-\eta}}$. Thus, $\mathbb{E}[R] = \frac{1}{p} = K^{(1-\eta)}$. $\square$

Using Lemmas 6.3.4, 6.3.5 and 6.3.6, we complete the proof of Theorem 6.3.1.

Proof of Theorem 6.3.1. The proof proceeds by induction on m. When $m = 0$, `R-SCOMP` runs `PCOMP` and the result follows by Theorem 5.5.1 (proving the base case). Now, suppose that $m \geq 1$. We bound the expected regret of `R-SCOMP` conditioned on G. Let R_1 and R_2 denote the regret incurred before and after the first recursive call.

Bounding R_1. Fix a bandit b_j. Let r denote the last round such that $b_j \in \mathcal{A}$ *and* we do not recurse at the end of round r. Let $b_{i^*_{\mathcal{S}}}$ be the best bandit in $\mathcal{S}$. As b_j is not eliminated by $b_{i^*_{\mathcal{S}}}$, we have $\widehat{P}_{i^*_{\mathcal{S}},j} \leq \frac{1}{2} + 3\gamma_r$, which implies (by event G) $P_{i^*_{\mathcal{S}},j} \leq \frac{1}{2} + 4\gamma_r$. Moreover, as switching doesn't occur, we have $\min_{i\in\mathcal{S}} \widehat{P}_{1,i} \leq \frac{1}{2} + 3\gamma_r$ (by Lemma 6.3.4, b_1 is never deleted from $\mathcal{A}$). By SST, we conclude that $P_{1,i^*_{\mathcal{S}}} \leq \frac{1}{2} + 4\gamma_r$. It now follows that $\Delta_{i^*_{\mathcal{S}},j} \leq 4\gamma_r$ and $\Delta_{1,i^*_{\mathcal{S}}} \leq 4\gamma_r$. Consider now two cases:

1. $b_1 \succeq b_{i^*_{\mathcal{S}}} \succeq b_j$. Then, by STI, $\Delta_{1,j} \leq 8\gamma_r$, and

2. $b_1 \succeq b_j \succeq b_{i^*_{\mathcal{S}}}$. Then, by SST $\Delta_{1,j} \leq \Delta_{i^*_{\mathcal{S}},j} \leq 4\gamma_r$.

In either case, we have $\Delta_j = \Delta_{1,j} \leq 8\gamma_r$, which implies $c_r \leq \frac{\log(1/\delta)}{2\gamma_r^2} \leq \frac{32\log(1/\delta)}{\Delta_j^2}$.

Now, let T_j be a random variable denoting the number of comparisons of b_j with other bandits before the recursive call. By definition of round r, bandit b_j will participate in at most one round after r (before recursing). So, we have

$$T_j \leq \begin{cases} |\mathcal{S}| \cdot \sum_{\tau=1}^{r+1} c_\tau & \text{if } b_j \notin \mathcal{S} \\ K \cdot \sum_{\tau=1}^{r+1} c_\tau & \text{if } b_j \in \mathcal{S} \end{cases}$$

Taking expectation over $\mathcal{S}$, we get

$$\mathbb{E}\left[T_j\right] \leq \mathbb{E}\left[K\sum_{\tau=1}^{r+1} c_\tau \,|\, b_j \in \mathcal{S}\right] \cdot \mathbf{P}(b_j \in \mathcal{S}) + \mathbb{E}\left[|\mathcal{S}|\sum_{\tau=1}^{r+1} c_\tau \,|\, b_j \notin \mathcal{S}\right] \cdot \mathbf{P}(b_j \notin \mathcal{S})$$
$$\leq \left(K\sum_{\tau=1}^{r+1} c_\tau\right) \cdot \frac{1}{K^{1-\eta}} + \mathbb{E}[|\mathcal{S}|\,|\, b_j \notin \mathcal{S}] \cdot \sum_{\tau=1}^{r+1} c_\tau \;\leq\; 2K^\eta \sum_{\tau=1}^{r+1} c_\tau,$$

where the third inequality uses $\mathbb{E}[|\mathcal{S}|\,|\, b_j \notin \mathcal{S}] \leq K^\eta$. Moreover,

$$\sum_{\tau=1}^{r+1} c_\tau \leq 2T^{1/B} \cdot c_r = O\left(\frac{T^{1/B}\log(1/\delta)}{\Delta_j^2}\right).$$

Thus,

$$\mathbb{E}[R_1] = \sum_j \mathbb{E}\left[T_j\right] \cdot \Delta_j = \sum_{j:\Delta_j>0} O\left(\frac{K^\eta T^{1/B}\log(6K^2TB)}{\Delta_j}\right) \tag{6.10}$$

Bounding R_2. We now bound the regret after a recursive call is invoked. From Lemmas 6.3.4 and 6.3.5, we know that b_1 is never deleted, $b_1 \in \mathcal{A}^*$, and $|\mathcal{A}^*| \leq rank(b_{i^*_{\mathcal{S}}})$. For any $\mathcal{A}^*$, on applying the inductive hypothesis with $m-1$ recursive calls, we get

$$\mathbb{E}[R_2 \mid \mathcal{A}^*] \leq \sum_{j:\Delta_j>0} O\left(\left((m-1)\cdot|\mathcal{A}^*|^\eta + |\mathcal{A}^*|^{(1-\eta)^{m-1}}\right) \cdot \frac{T^{1/B}\log(6|\mathcal{A}^*|^2TB)}{\Delta_j}\right)$$
$$\leq \sum_{j:\Delta_j>0} O\left(\left((m-1)\cdot K^\eta + |\mathcal{A}^*|^{(1-\eta)^{m-1}}\right) \cdot \frac{T^{1/B}\log(6K^2TB)}{\Delta_j}\right)$$

since $|\mathcal{A}^*| \leq K$. Taking an expectation over $\mathcal{A}^*$, we obtain

$$\mathbb{E}[R_2] \leq \sum_{j:\Delta_j>0} O\left(\left((m-1)\cdot K^\eta + \mathbb{E}\left[|\mathcal{A}^*|^{(1-\eta)^{m-1}}\right]\right) \cdot \frac{T^{1/B}\log(6K^2TB)}{\Delta_j}\right)$$

Finally, observe that by Jensen's inequality, we have $\mathbb{E}\left[|\mathcal{A}^*|^{(1-\eta)^{m-1}}\right] \leq \mathbb{E}[|\mathcal{A}^*|]^{(1-\eta)^{m-1}}$, and by Lemma 6.3.6 $\mathbb{E}[|\mathcal{A}^*|] \leq K^{1-\eta}$. Combining these observations, we get $\mathbb{E}\left[|\mathcal{A}^*|^{(1-\eta)^{m-1}}\right] \leq K^{(1-\eta)^m}$. Thus, we obtain

$$\mathbb{E}[R_2] \leq \sum_{j:\Delta_j>0} O\left(\left((m-1)\cdot K^\eta + K^{(1-\eta)^m}\right) \cdot \frac{T^{1/B}\log(6K^2TB)}{\Delta_j}\right) \tag{6.11}$$

Finally, combining (6.10) and (6.11) completes the induction. □

6.4 Experimental Results

We provide a summary of computational results of our algorithms for the batched dueling bandits problem. We conducted our computations using C++ and Python 2.7 with a 2.3 Ghz Intel Core *i*5 processor and 16 GB 2133 MHz LPDDR3 memory.

Experimental Setup. We compare all our algorithms, namely `PCOMP`, `SCOMP`, and `SCOMP2` to a representative set of sequential algorithms for dueling bandits. Specifically, we use the dueling bandit library due to [76], and compare our algorithms to RUCB [117], RMED1 [76], and Beat-the-Mean [114]. Henceforth, we refer to Beat-the-Mean as BTM. We plot the cumulative regret $R(t)$ incurred by the algorithms against time t. Furthermore, to illustrate the dependence on B, we run another set of experiments on `SCOMP2` and plot the cumulative regret $R(t)$ incurred by `SCOMP2` against time t for varying values of B.[3] We perform these experiments using both real-world and synthetic data. We use the following datasets:

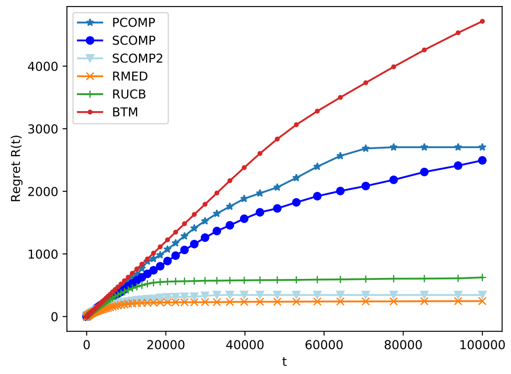
Six rankers. This real-world dataset is based on the 6 retrieval functions used in the engine of ArXiv.org.

Sushi. The Sushi dataset is based on the Sushi preference dataset [72] that contains the preference data regarding 100 types of Sushi. A preference dataset using the top-16 most popular types of sushi is obtained.

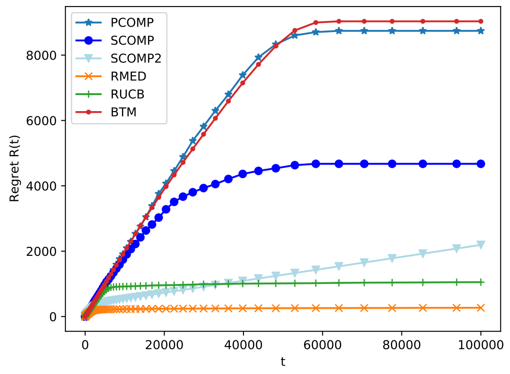
BTL-Uniform. We generate synthetic data using the Bradley-Terry-Luce (BTL) model. Under this model, each arm $b_i \in \mathcal{B}$ is associated with a weight $w_i > 0$ (sampled uniformly in the interval $(0, 1]$), and we set $P_{i,j} = w_i/(w_i + w_j)$. We set the number of arms $K = 100$. Note that the data generated in this way satisfies SST and STI [112]. We refer to this data as `SYN-BTL`.

Hard-Instance. The last dataset is a synthetic dataset inspired by the hard instances that we construct for proving our lower bound (see Theorem 5.3.5). Again, we set $K = 100$, and pick $\ell \in [K]$ uniformly at random as the Condorcet winner. We select Δ uniformly in

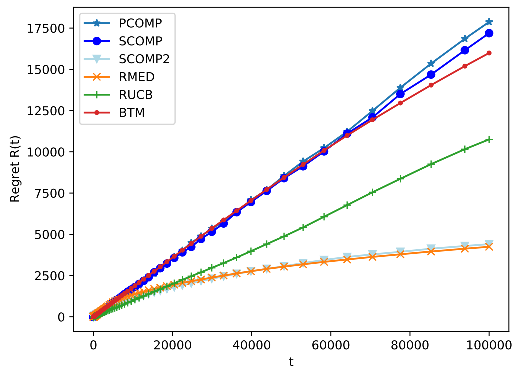
[3]We also conducted these experiment for `PCOMP` and `SCOMP` and the conclusions were similar.



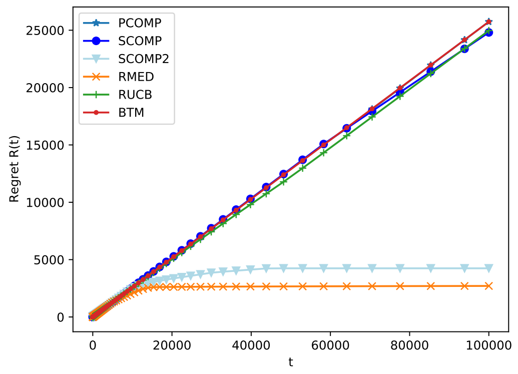
(a) Six rankers



(b) Sushi



(c) SYN-BTL



(d) SYN-CD

Figure 6.1: Regret v/s t plots of algorithms

$(0, 0.5)$, and set $P_{\ell,i} = \frac{1}{2} + \Delta$ for $i \neq \ell$. Furthermore, for all $i, j \neq \ell$, we set $P_{i,j} = 1/2$. We refer to this data as SYN-CD.

Note that there exists a Condorcet winner in all datasets. Moreover, the SYN-BTL dataset satisfies SST and STI. We repeat each experiment 10 times and report the average regret. In our algorithms, we use the KL-divergence based confidence bound (as in RMED1) for elimination as it performs much better empirically (and our theoretical bounds continue to hold). In particular, we replace lines 5, 6 and 8 in PCOMP, SCOMP and SCOMP2, respectively, with KL-divergence based elimination criterion that eliminates an arm i if there exists another arm j if $\hat{P}_{ij} < \frac{1}{2}$ and $N_{ij} \cdot D_{\mathrm{KL}}(\hat{P}_{ij}, \frac{1}{2}) > \log(T\delta)$ where N_{ij} is the number of times arm i and j are played together. We report the average cumulative regret at each time step.

Comparison with sequential dueling bandit algorithms. As mentioned earlier, we compare our algorithms against a representative set of sequential dueling bandits algorithms (RUCB [117], RMED1 [76], and BTM [114]). Note that the purpose of these experiments is to perform a sanity check to ensure that our batched algorithms, using a small number of batches, perform well when compared with sequential algorithms. We set $\alpha = 0.51$ for RUCB, and $f(K) = 0.3K^{1.01}$ for RMED1, and $\gamma = 1.3$ for BTM. We chose these parameters as they are known to perform well both theoretically and empirically [76]. We set $T = 10^5$, $\delta = 1/TK^2$ and $B = \lfloor \log(T) \rfloor = 16$. We plot the results in Figure 6.1. We observe that `SCOMP2` performs comparably to RMED1 in all datasets, even outperforms RUCB in 3 out of the 4 datasets, and always beats BTM. Notice that both `PCOMP` and `SCOMP` considerably outperform BTM on the six rankers and sushi data; however their performance degrades on the synthetic data demonstrating the dependence on K.

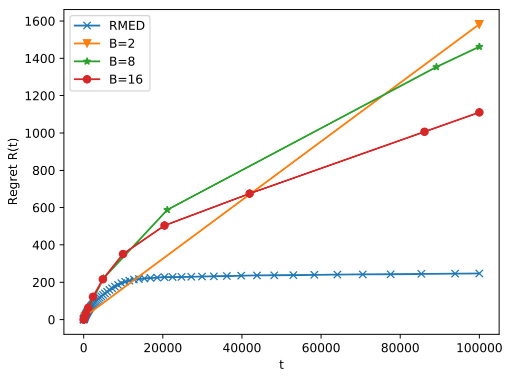
Trade-off with number of batches B. We study the trade-off of cumulative regret against the number of batches using `SCOMP2`. We set $T = 10^5$, and vary $B \in \{2, 8, 16\}$. We also plot the regret incurred by RMED1 as it performs the best amongst all sequential algorithms (and thus serves as a good benchmark). We plot the results in Figure 6.2. We observe that as we increase the number of batches, the (expected) cumulative regret decreases. Furthermore, we observe that on the synthetic datasets (where $K = 100$), the regret of `SCOMP2` approaches that of RMED1; in fact, the regret incurred is almost identical for the `SYN-BTL` dataset.

6.5 Lower Bound

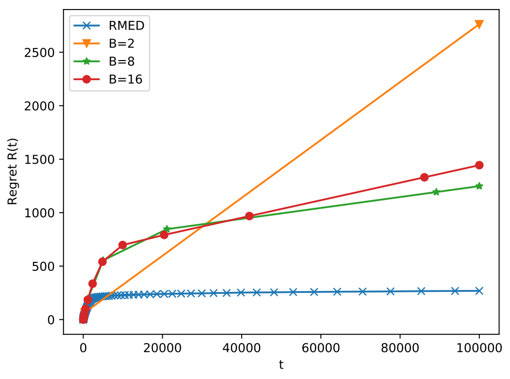
In this section, we present a lower bound for the batched dueling bandits problem under the SST and STI setting. Note that this lower bound also applies to the more general Condorcet winner setting. The main result of this section is the following:

Theorem 6.5.1. *Given an integer $B > 1$, and any algorithm that uses at most B batches, there exists an instance of the K-armed batched dueling bandit problem that satisfies the SST and STI conditions such that the expected regret*

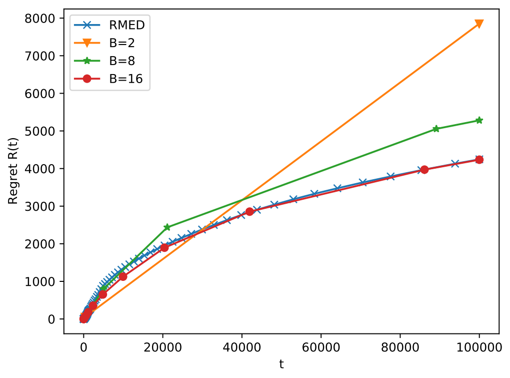
$$\mathbb{E}[R_T] = \Omega\left(\frac{KT^{1/B}}{B^2\Delta_{\min}}\right),$$



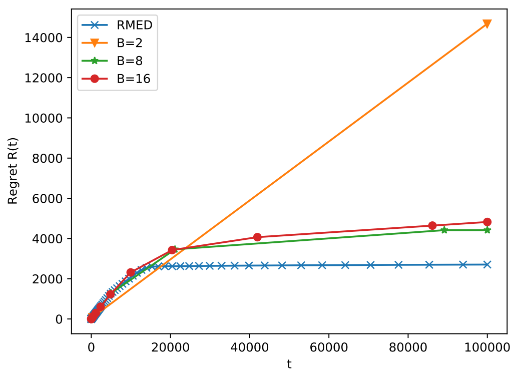
(a) Six rankers



(b) Sushi



(c) `SYN-BTL`



(d) `SYN-CD`

Figure 6.2: Regret v/s B for `SCOMP2`.

where $\Delta_{\min}$ is defined with respect to the particular instance.

In order to prove this theorem, we will construct a family of instances such that any algorithm for batched dueling bandits cannot simultaneously beat the above regret lower bound over all instances in the family. We exploit the fact that the algorithm is unaware of the particular instance chosen from the family at run-time, and hence, is unaware of the gap $\Delta_{\min}$ under that instance.

Family of Instances $\mathcal{I}$:

- Let F be an instance where $P_{i,j} = \frac{1}{2}$ for all $i, j \in \mathcal{B}$.
- For $j \in [B]$, let $\Delta_j = \frac{\sqrt{K}}{24B} \cdot T^{(j-1)/2B}$. For $j \in [B]$ and $k \in [K]$, let $E_{j,k}$ be an instance where bandit b_k is the Condorcet winner such that $P_{k,l} = \frac{1}{2} + \Delta_j$ for all $l \in [K] \setminus \{k\}$ and $P_{l,m} = \frac{1}{2}$ for all $l, m \in [K] \setminus \{k\}$.
- The family of instances $\mathcal{I} := \{E_{j,k}\}_{j\in[B],k\in[K]} \cup \{F\}$.

6.5.1 Proof of Theorem 6.5.1

Let us fix an algorithm $\mathcal{A}$ for this problem. Let $T_j = T^{j/B}$ for $j \in [B]$. Let t_j be the total (random) number of comparisons until the end of batch j during the execution of $\mathcal{A}$. We will overload notation and denote by I^t the distribution of observations seen by the algorithm when the underlying instance is I. We will sometimes use $P_{i,j}(I)$ for the probability of i beating j under an instance I to emphasize the dependence on I. We will also write $\Delta_{\min}(I)$ to emphasize the dependence on the underlying instance I.

We define event A_j as follows:

$$A_j = \{t_{j'} < T_{j'}, \forall j' < j \text{ and } t_j \geq T_j\},$$

and denote by $E_{j,k}(A_j)$ the event that A_j occurs given that the instance selected is $E_{j,k}$. Similarly, $F(A_j)$ denotes the event that A_j occurs when the instance selected is F. Now, define

$$p_j = \frac{1}{K}\sum_{l=1}^{K} \mathbf{P}(E_{j,l}(A_j)).$$

Observe that p_j is the average probability of event A_j conditional on the instance having gap Δ_j.

Lemma 6.5.2. $\sum_{j=1}^{B} p_j \geq \frac{1}{2}$.

Proof. Note that the event A_j is determined by observations until T_{j-1}. This is because

$t_{j-1} < T_{j-1}$, and once the observations until t_{j-1} are seen: the next batch j determines whether or not A_j occurs. Hence, in order to bound the probability of A_j under two different instances F and $E_{j,l}$ we use the Pinsker's inequality as

$$|\mathbf{P}(F(A_j)) - \mathbf{P}(E_{j,l}(A_j))| \leq \sqrt{\frac{1}{2} D_{\mathrm{KL}}(F^{T_{j-1}} || E_{j,l}^{T_{j-1}})}$$

for $l \in [K]$. Let τ_l be the random variable for the number of times arm l is played until T_{j-1}. We first bound $D_{\mathrm{KL}}(F^{T_{j-1}} || E_{j,l}^{T_{j-1}})$ as

$$\begin{aligned} D_{\mathrm{KL}}(F^{T_{j-1}} || E_{j,l}^{T_{j-1}}) &\overset{(a)}{=} \sum_{t=1}^{T_{j-1}} D_{\mathrm{KL}}\big(P_{t_1,t_2}(F) \ || \ P_{t_1,t_2}(E_{j,l})\big) \\ &\overset{(b)}{\leq} \sum_{t=1}^{T_{j-1}} \Pr_F(\text{arm } l \text{ is played in trial } t) \cdot D_{\mathrm{KL}}\left(\frac{1}{2} \ || \ \frac{1}{2} + \Delta_j\right) \\ &\overset{(c)}{\leq} \mathbb{E}_F[\tau_l] \cdot 4\Delta_j^2 \, , \end{aligned} \tag{6.12}$$

where (a) follows from the fact that, given F, the outcome of comparisons are independent across trials, (b) follows from the fact that the KL-divergence between $P_{t_1,t_2}(F)$ and $P_{t_1,t_2}(E_{j,k})$ is non-zero only when arm l is played in trial t, and (c) follows from the fact that $D_{\mathrm{KL}}(p||q) \leq \frac{(p-q)^2}{q\cdot(1-q)}$. Using the above bounds, we have that

$$\begin{aligned} \frac{1}{K}\sum_{l=1}^{K} |\mathbf{P}(F(A_j)) - \mathbf{P}(E_{j,l}(A_j))| &\leq \frac{1}{K}\sum_{l=1}^{K} \sqrt{\frac{1}{2} D_{\mathrm{KL}}(F^{T_{j-1}} || E_{j,l}^{T_{j-1}})} \\ &\leq \frac{1}{K}\sum_{l=1}^{K} \sqrt{\frac{1}{2} \cdot 4\Delta_j^2 \mathbb{E}_F[\tau_l]} = \frac{1}{K}\sum_{l=1}^{K} \sqrt{2\Delta_j^2 \mathbb{E}_F[\tau_l]} \\ &\overset{(a)}{\leq} \sqrt{\frac{2\Delta_j^2 \mathbb{E}_F[\sum_{l=1}^{K} \tau_l]}{K}} \\ &\overset{(b)}{\leq} \sqrt{\frac{2\Delta_j^2 \cdot 2T_{j-1}}{K}} = \frac{1}{2B} \, , \end{aligned}$$

where (a) follows from the concavity of $\sqrt{x}$ and Jensen's inequality, and (b) follows from the fact that $\sum_{l=1}^{K} \tau_l \leq T_{j-1}$. We thus have

$$|\mathbf{P}(F(A_j)) - p_j| = |\mathbf{P}(F(A_j)) - \frac{1}{K}\sum_{l=1}^{K} \mathbf{P}(E_{j,l}(A_j))|$$
$$\leq \frac{1}{K}\sum_{l=1}^{K} |\mathbf{P}(F(A_j)) - \mathbf{P}(E_{j,l}(A_j))| \leq \frac{1}{2B}\,.$$

Finally, we can write

$$\sum_{j=1}^{B} p_j \geq \sum_{j=1}^{B} (\mathbf{P}(F(A_j)) - \frac{1}{2B}) \geq \sum_{j=1}^{B} \mathbf{P}(F(A_j)) - \frac{1}{2} \geq \frac{1}{2}\,.$$

□

As a consequence of this lemma, we can conclude that there exists some $j \in [B]$ such that $p_j \geq \frac{1}{2B}$. We focus on the event where gap is Δ_j, and prove that when $p_j \geq \frac{1}{2B}$, $\mathcal{A}$ must suffer a high regret leading to a contradiction. The next lemma formalizes this.

Lemma 6.5.3. *If, for some j, $p_j \geq \frac{1}{2B}$, then*

$$\sup_{I:\Delta_{\min}(I)=\Delta_j} \mathbb{E}[R_T(I)] \geq \Omega\left(\frac{KT^{1/B}}{B^2\Delta_j}\right)$$

Proof. Fix $k \in [K]$. We will construct a family of instances $\{Q_{j,k,l}\}_{l \neq k}$ where $Q_{j,k,l}$ is defined as:

Instance $Q_{j,k,l}$: Arm l is the Condorcet winner and the pairwise preferences are defined as:

$$P_{lm} = \frac{1}{2} + 2\Delta_j, \forall m \in [K] \setminus \{l\}; \qquad P_{km} = \frac{1}{2} + \Delta_j, \forall m \in [K] \setminus \{l, k\};$$

and $P_{mm'} = \frac{1}{2}$ for remaining pairs (m, m').

We also let $Q_{j,k,k} := E_{j,k}$. Note that the regret is $\geq \Delta_j$ if the underlying instance is $Q_{j,k,l}$

and the pair played is not (b_l, b_l). We have that

$$\sup_{I:\Delta_{\min}(I)=\Delta_j} \mathbb{E}[R_T(I)] \geq \Delta_j \sum_{t=1}^{T} \frac{1}{K} \sum_{l \neq k} Q^t_{j,k,l}\left((b_{t_1}, b_{t_2}) \neq (b_l, b_l)\right) ,$$

where $Q^t_{j,k,l}$ denotes the distribution of observations available at time t under instance $Q_{j,k,l}$ and $Q^t_{j,k,l}\left((b_{t_1}, b_{t_2}) \neq (b_l, b_l)\right)$ is the probability that the algorithm does not play arm (b_l, b_l) at time t under $Q^t_{j,k,l}$. In order to bound the above quantity we will need the following lemma from [48].

Lemma 6.5.4 (Lemma 3 of [48]). *Let $Q_1, \cdots Q_K$ be probability measures on some common probability space $(\Omega, \mathcal{F})$, and $\psi : \Omega \to [K]$ be any measurable function (i.e., test). Then, for any tree $\mathcal{T} = ([K], E)$ with vertex set $[K]$ and edge set E,*

$$\frac{1}{K} \sum_{i=1}^{K} Q_i(\psi \neq i) \geq \frac{1}{K} \sum_{(l,l') \in E} \int \min\{dQ_l, dQ_{l'}\} .$$

Using the above lemma for the star graph centered at k, we have that

$$\begin{aligned}
\sup_{I:\Delta_{\min}(I)=\Delta_j} \mathbb{E}[R_T(I)] &\geq \Delta_j \sum_{t=1}^{T} \frac{1}{K} \sum_{l \neq k} \int \min\{dQ^t_{j,k,k}, dQ^t_{j,k,l}\} \\
&\overset{(a)}{\geq} \Delta_j \sum_{t=1}^{T_j} \frac{1}{K} \sum_{l \neq k} \int \min\{dQ^t_{j,k,k}, dQ^t_{j,k,l}\} \\
&\overset{(b)}{\geq} \Delta_j \sum_{t=1}^{T_j} \frac{1}{K} \sum_{l \neq k} \int \min\{dQ^{T_j}_{j,k,k}, dQ^{T_j}_{j,k,l}\} \\
&\geq \Delta_j \sum_{t=1}^{T_j} \frac{1}{K} \sum_{l \neq k} \int_{A_j} \min\{dQ^{T_j}_{j,k,k}, dQ^{T_j}_{j,k,l}\} \\
&\overset{(c)}{\geq} \Delta_j \sum_{t=1}^{T_j} \frac{1}{K} \sum_{l \neq k} \int_{A_j} \min\{dQ^{T_{j-1}}_{j,k,k}, dQ^{T_{j-1}}_{j,k,l}\} ,
\end{aligned} \tag{6.13}$$

where (a) follows because $T_j \leq T$, (b) follows due to the fact that $\int \min\{dP, dQ\} = 1 - D_{\mathrm{TV}}(P, Q)$ and the fact that $D_{\mathrm{TV}}(Q^{T_j}_{j,k,k}, Q^{T_j}_{j,k,l})$ is at least $D_{\mathrm{TV}}(Q^t_{j,k,k}, Q^t_{j,k,l})$ as the sigma algebra $\mathcal{F}_{Q^t_{j,k,k}}$ of $Q^t_{j,k,k}$ is a subset of the sigma algebra $\mathcal{F}_{Q^{T_j}_{j,k,k}}$ of $Q^{T_j}_{j,k,k}$, (c) follow from the fact that the event A_j is determined by observations until T_{j-1} as explained in the proof of Lemma 6.5.2. We then have that

$$\begin{aligned}
\int_{A_j} \min\{dQ_{j,k,k}^{T_{j-1}}, dQ_{j,k,l}^{T_{j-1}}\} &= \int_{A_j} \frac{dQ_{j,k,k}^{T_{j-1}} + dQ_{j,k,l}^{T_{j-1}} - |dQ_{j,k,k}^{T_{j-1}} - dQ_{j,k,l}^{T_{j-1}}|}{2} \\
&= \frac{Q_{j,k,k}^{T_{j-1}}(A_j) + Q_{j,k,l}^{T_{j-1}}(A_j)}{2} - \int_{A_j} \frac{|dQ_{j,k,k}^{T_{j-1}} - dQ_{j,k,l}^{T_{j-1}}|}{2} \\
&\overset{(a)}{\geq} Q_{j,k,k}^{T_{j-1}}(A_j) - \frac{1}{2} D_{\text{TV}}\left(Q_{j,k,k}^{T_{j-1}}, Q_{j,k,l}^{T_{j-1}}\right) - D_{\text{TV}}\left(Q_{j,k,k}^{T_{j-1}}, Q_{j,k,l}^{T_{j-1}}\right) \\
&= Q_{j,k,k}^{T_{j-1}}(A_j) - \frac{3}{2} D_{\text{TV}}\left(Q_{j,k,k}^{T_{j-1}}, Q_{j,k,l}^{T_{j-1}}\right), \qquad (6.14)
\end{aligned}$$

where (a) follows from the fact that $D_{\text{TV}}(P,Q) = \sup_A |P(A) - Q(A)|$. Let us define τ_l to be the random variable for the number of times arm l is played until T_{j-1} We also have that

$$\begin{aligned}
\frac{1}{K}\sum_{l\neq k} D_{\text{TV}}\left(Q_{j,k,k}^{T_{j-1}}, Q_{j,k,l}^{T_{j-1}}\right) &\leq \frac{1}{K}\sum_{l\neq k}\sqrt{\frac{1}{2}D_{\text{KL}}(Q_{j,k,k}^{T_{j-1}}||Q_{j,k,l}^{T_{j-1}})} \\
&\overset{(a)}{\leq} \frac{1}{K}\sum_{l\neq k}\sqrt{\frac{1}{2}\cdot 16\Delta_j^2\mathbb{E}_{E_{j,k}}[\tau_l]} = \frac{1}{K}\sum_{l\neq k}\sqrt{8\Delta_j^2\mathbb{E}_{E_{j,k}}[\tau_l]} \\
&\overset{(b)}{\leq} \sqrt{\frac{8\Delta_j^2\mathbb{E}_{E_{j,k}}[\sum_{l\neq k}\tau_l]}{K}} \\
&\overset{(c)}{\leq} \sqrt{\frac{8\Delta_j^2}{K}2T_{j-1}} = \frac{1}{6B}, \qquad (6.15)
\end{aligned}$$

where (a) follows from a similar calculation as eq. (6.12) in the proof of Lemma 6.5.2, (b) follows from the concavity of $\sqrt{x}$ and Jensen's inequality, and (c) follows from the fact that $\sum_{l=1}^{K}\tau_l \leq T_{j-1}$.

Combining eqs. (6.13) to (6.15) we have that

$$\sup_{I:\Delta_{\min}(I)=\Delta_j} \mathbb{E}[R_T(I)] \geq \Delta_j T_j\left(\mathbf{P}(E_{j,k}(A_j)) - \frac{1}{4B}\right).$$

Since the above inequality holds for all $k \in [K]$, by averaging we get

$$\begin{aligned}
\sup_{I:\Delta_{\min}(I)=\Delta_j} \mathbb{E}[R_T(I)] &\geq \Delta_j T_j \left(\frac{1}{K} \sum_{k=1}^{K} \mathbf{P}(E_{j,k}(A_j)) - \frac{1}{4B} \right) \\
&= \Delta_j T_j \left(p_j - \frac{1}{4B} \right) \\
&\geq \Delta_j T_j \frac{1}{4B} \,.
\end{aligned}$$

Substituting the value of $\Delta_j T_j$ we get

$$\begin{aligned}
\sup_{I:\Delta_{\min}(I)=\Delta_j} \mathbb{E}[R_T(I)] &\geq \Delta_j T_j \frac{1}{4B} = \frac{\sqrt{K}}{24B} T^{-(j-1)/2B} T^{j/B} \frac{1}{4B} \\
&= \frac{\sqrt{K}}{24B} T^{(j-1)/2B} T^{1/B} \frac{1}{4B} = \Omega\left(\frac{KT^{1/B}}{B^2 \Delta_j} \right) \,.
\end{aligned}$$

□

Finally, $\sum_{j=1}^{B} p_j \geq \frac{1}{2}$ implies that there exists $j \in [B]$ with $p_j \geq 1/2B$. Combining the two lemmas above, we get that there exists $j \in [B]$ with $p_j \geq 1/2B$ such that the algorithm incurs a regret of $\Omega\left(\frac{KT^{1/B}}{B^2\Delta_j}\right)$. In this case, there must exist an instance $E_{j,k}$ with gap $\Delta_{\min}(E_{j,k}) = \Delta_j$ such that the regret of the algorithm under $E_{j,k}$ is $\Omega\left(\frac{KT^{1/B}}{B^2\Delta_j}\right)$. This completes the proof of our lower bound.

Chapter 7

An Improved Algorithm for Batched Dueling Bandits under Condorcet Condition

7.1 Overview

In this chapter we provide detailed proofs, and formalize Theorems 5.3.6 and 5.3.7. The main contributions presented in this chapter are as follows.

7.1.1 Contributions

- We design an algorithm, denoted `C2B`, for the batched dueling bandit problem, and analyze its regret under the Condorcet condition. This algorithm achieves a smooth trade-off between the expected regret and the number of batches, B.
- Crucially, when $B = \log(T)$, our regret bounds nearly match the best regret bounds [76, 117] known in the fully sequential setting. Hence, our results show that $O(\log T)$ rounds are sufficient to achieve asymptotically optimal regret as a function of T.
- Our results rely on new ideas for showing that the Condorcet winner arm can be 'trapped' using few adaptive rounds with high (constant) probability while incurring a

reasonable amount of regret. We can then integrate over this space of probabilities to obtain a bound on the expected regret (in the same vein as [117]). Once the Condorcet arm is 'trapped', we can quickly eliminate all other 'sub-optimal' arms and minimize regret in the process.

- Finally, we run computational experiments to validate our theoretical results. We show that `C2B`, using $O(\log T)$ batches, achieves almost the same performance as fully sequential algorithms (which effectively use T batches) over a variety of real datasets.

7.1.2 Results and Techniques

We provide both high-probability and expected regret bounds as mentioned in Chapter 5. We restate the theorems for completeness.

Theorem 5.3.6. *For any integer $B \geq 1$, there is an algorithm for the K-armed dueling bandit problem that uses at most B rounds with the following guarantee. For any $\delta > 0$, with probability at least $1 - \delta - \frac{1}{T}$, its regret under the Condorcet condition is at most*

$$R(T) \leq O\left(T^{1/B} \cdot \frac{K^2 \log(K)}{\Delta_{\min}^2} \cdot \log\left(\frac{\log K}{\Delta_{\min}}\right)\right) + O\left(T^{2/B} \cdot K^2 \cdot \sqrt{\frac{1}{\delta}}\right) + \sum_{j \neq a^*} O\left(\frac{T^{1/B} \cdot \log(KT)}{\Delta_j}\right).$$

Theorem 5.3.7. *For any integer $B \geq 1$, there is an algorithm for the K-armed dueling bandit problem that uses at most B rounds, with expected regret under the Condorcet condition at most*

$$\mathbb{E}[R(T)] = O\left(T^{1/B} \cdot \frac{K^2 \log(K)}{\Delta_{\min}^2} \cdot \log\left(\frac{\log K}{\Delta_{\min}}\right)\right) + O\left(T^{2/B} \cdot K^2\right) + \sum_{j \neq a^*} O\left(\frac{T^{1/B} \cdot \log(KT)}{\Delta_j}\right).$$

When the number of rounds $B = \log(T)$, we obtain a batched algorithm that achieves the asymptotic optimality (in terms of T), even for sequential algorithms. We formalize this observation in Corollary 5.3.8.

Technical Challenges. The only other approach for batched dueling bandits (under the Condorcet condition) is the algorithm PCOMP (see Chapter 5), which performs all-pairs comparisons among arms in an active set. Such an approach cannot achieve regret better than $O(K^2 \log T)$ because the active set may remain large throughout. In order to achieve better regret bounds, we focused on the stronger SST+STI condition (see Chapter 6). In this setting, our main idea is to first sample a *seed set*, and use this seed set to eliminate sub-optimal arms. Their algorithm proceeds by performing all pairwise comparisons between the seed set and the set of active arms. However, the analysis of these 'seeded comparison' algorithms crucially rely on the total-ordering imposed by the SST and STI assumptions. Unfortunately, there is no such structure to exploit in the Condorcet setting: if the seed set does not contain the Condorcet winner, we immediately incur high regret.

The existing fully sequential algorithms such as RUCB [117] and RMED [76] are *highly adaptive* in nature. For instance, RUCB plays each *candidate* arm against an optimistic *competitor* arm using upper confidence bounds (UCB) on pairwise probabilities. This allows RUCB to quickly filter out candidates and uncover the Condorcet arm. Similarly, RMED plays each arm against a carefully selected *competitor* arm that is likely to beat this arm. However, such *competitors* can change frequently over trials in both RUCB and RMED. Since the batched setting requires comparisons to be predetermined, we do not have the flexibility to adapt to such changes in *competitors*. Hence, these existing fully sequential algorithms cannot be easily implemented in our setting.

Furthermore, we might also be tempted to consider an *explore-then-exploit* strategy where we first *explore* to find the Condorcet arm and *exploit* by playing this arm for remaining trials. However, this strategy is likely to fail because identifying the Condorcet arm with high probability might involve performing many comparisons, directly leading to high $(\Omega(K^2 \log T))$ regret; on the other hand, if the Condorcet winner is not identified with high probability, the exploit phase becomes expensive. This motivated us to consider algorithms that allow some form of *recourse*; that is, unless an arm is found to be sub-optimal, it must be given the opportunity to participate in the comparisons (as it could be the Condorcet winner).

The idea behind our new algorithm is to identify the Condorcet winner i^* in a small *expected* number of rounds, after which it uses this arm as an "anchor" to eliminate sub-optimal arms while incurring low regret. To identify the best arm, in each round we define a candidate arm and compare it against arms that it "defeats". Arms that are not defeated by the candidate arm are compared to *all* active arms: this step ensures that the Condorcet winner is eventually discovered. We show that i^* becomes the candidate, and *defeats all other arms* within a small number of rounds (though the algorithm may not know if this has occurred). Additionally, once this condition is established, it remains invariant in future rounds. This allows us to eliminate sub-optimal arms and achieve low regret.

Comparison to RUCB. Initially, RUCB puts all arms in a pool of potential champions, and "optimistically" (using a upper confidence bound) performs all pairwise comparisons. Using these, it constructs a set of candidates C. If $|C| = 1$, then that arm is the hypothesised Condorcet winner and placed in a set B. Then, a randomized strategy is employed to choose a champion arm a_c (from sets C and B) which is compared to arm a_d which is most likely to beat it. The pair (a_c, a_d) is compared, the probabilities are updated and the algorithm continues. Although our algorithm also seeks to identify the best arm, we do not employ the UCB approach nor do we use any randomness. In their analysis, [117] show that the best arm eventually enters the set B, and remains in B: we also show a similar property for our algorithm in the analysis. Finally, similar to the analysis of [117], we first give a high-probability regret bound for our algorithm which we then convert to a bound on the expected regret.

7.2 The Batched Algorithm

In this section, we describe a B-round algorithm for the K-armed dueling bandit problem under the Condorcet condition. Recall that given a set of K arms, $\mathcal{B} = \{1, \ldots, K\}$, and a positive integer $B \leq \log(T)$, we wish to find a sequence of B batches of noisy comparisons with low regret. Given arms i and j, recall that $p_{i,j} = \frac{1}{2} + \Delta_{i,j}$ denotes the probability of i winning over j where $\Delta_{i,j} \in (-1/2, 1/2)$. We use i^* to denote the Condorcet winner; recall that i^* is a Condorcet winner if $p_{i^*,j} \geq 1/2$ for all $j \in \mathcal{B}$. To simplify notation, we use $\Delta_j = \Delta_{i^*,j}$. Before describing our algorithm, we first define some notation. We use $\mathcal{A}$ to denote the current set of *active* arms; i.e., the arms that have not been eliminated. We will use index r for rounds or batches. If pair (i, j) is compared in round r, it is compared

$q_r = \lfloor q^r \rfloor$ times where $q = T^{1/B}$. We define the following quantities at the *end* of each round r:

- $N_{i,j}(r)$ is the total number of times the pair (i, j) has been compared.
- $\widehat{p}_{i,j}(r)$ is the frequentist estimate of $p_{i,j}$, i.e.,

$$\widehat{p}_{i,j}(r) = \frac{\#\ i \text{ wins against } j \text{ until end of round } r}{N_{i,j}(r)}\,. \tag{7.1}$$

- Two confidence-interval radii for each (i, j) pair:

$$c_{i,j}(r) = \sqrt{\frac{2\log(2K^2 q_r)}{N_{i,j}(r)}} \qquad \text{and} \qquad \gamma_{i,j}(r) = \sqrt{\frac{\log(K^2BT)}{2N_{i,j}(r)}} \tag{7.2}$$

We now describe our B-round algorithm, called **Catching the Condorcet winner in Batches** (or, `C2B`). At a high-level, the algorithm identifies the best arm i^* in a small expected number of rounds, after which it uses this arm as an "anchor" to eliminate sub-optimal arms while incurring low regret. In every round r, we do the following:

1. We define a *defeated set* $D_r(i)$ for every active arm i; this set comprises arms that are defeated *with confidence* by i. Specifically, $j \in D_r(i)$ if $\widehat{p}_{i,j}(r-1) > 1/2 + c_{i,j}(r-1)$.
2. Then, we define a *candidate* i_r as the arm that defeats the most number of arms; that is, $i_r = \arg\max_{i\in\mathcal{A}} |D_r(i)|$.
3. For every arm $i \neq i_r$:
    - If $i \in D_r(i_r)$, then we compare i to i_r for q_r times. The idea here is to use i_r as an anchor against i. We will show that i^* becomes the candidate i_r in a small number of rounds. Then, this step ensures that we eliminate arms efficiently using i^* as an anchor.
    - If $i \notin D_r(i_r)$, then i is compared to all arms in $\mathcal{A}$ for q_r times. This step crucially protects the algorithm against cases where a sub-optimal arm becomes the candidate (and continues to become the candidate). For example, suppose $K = [5]$

and the arms are linearly ordered as $1 \succ 2 \succ \cdots \succ 5$. Furthermore suppose that in some round r, we have that (a) 2 defeats $3, 4, 5$ and (b) 1 (best arm) defeats 2 but not the others. So, 2 is the candidate in round r; if 1 is not compared to $3, 4, 5$, then 2 would continue to be the candidate (leading to high regret).

4. If, for any arm j, there is arm i such that $\widehat{p}_{i,j}(r) > \frac{1}{2} + \gamma_{i,j}(r)$, then j is eliminated from $\mathcal{A}$.

This continues until T total comparisons are performed. See Algorithm 14 for a formal description. The main result of this section is to show that `C2B` achieves the guarantees stated in Theorems 5.3.6 and 5.3.7.

Algorithm 14 `C2B` (Catching the Condorcet winner in Batches)

```
1: **Input:** Arms $\mathcal{B}$, time-horizon $T$, integer $B \geq 1$
2: active arms $\mathcal{A} \leftarrow \mathcal{B}$, $r \leftarrow 1$, emprical probabilities $\widehat{p}_{i,j}(0) = \frac{1}{2}$ for all $i, j \in \mathcal{B}^2$
3: **while** number of comparisons $\leq T$ **do**
4:     **if** $\mathcal{A} = \{i\}$ for some $i$ **then** play $(i, i)$ for remaining trials
5:     $D_r(i) \leftarrow \{j \in \mathcal{A} : \widehat{p}_{i,j}(r-1) > \frac{1}{2} + c_{i,j}(r-1)\}$
6:     $i_r \leftarrow \arg\max_{i \in \mathcal{A}} |D_r(i)|$
7:     **for** $i \in \mathcal{A} \setminus \{i_r\}$ **do**
8:         **if** $i \in D_r(i_r)$ **then**
9:             compare $(i_r, i)$ for $q_r$ times
10:        **else**
11:            for each $j \in \mathcal{A}$, compare $(i, j)$ for $q_r$ times
12:    compute $\widehat{p}_{i,j}(r)$ values
13:    **if** $\exists i, j : \widehat{p}_{i,j}(r) > \frac{1}{2} + \gamma_{i,j}(r)$ **then**
14:        $\mathcal{A} \leftarrow \mathcal{A} \setminus \{j\}$
15:    $r \leftarrow r + 1$
```

Overview of the Analysis. We provide a brief outline of the proofs of our main results. Let $\delta > 0$ be any value. Towards proving Theorem 5.3.6, we first define two events:

- The first event, denoted G, ensures that i^* is not eliminated during the execution of `C2B`. We show that $\mathbf{P}(G) \geq 1 - 1/T$.

- The second event, denoted $E(\delta)$, says that there exists a round $C(\delta)$ (defined later) such that for all $r > C(\delta)$, the estimate $\widehat{p}_{i,j}(r-1)$ satisfies the confidence interval of $c_{i,j}(r-1)$. Moreover, $\mathbf{P}(E(\delta)) \geq 1 - \delta$.

By union bound, $\mathbf{P}(G \cap E(\delta)) \geq 1 - \delta - 1/T$. Together, we use G and $E(\delta)$ to argue that:

- the best arm, i^*, is *not defeated* by any arm i in any round $r > C(\delta)$,
- and that there exists a round $r(\delta) \geq C(\delta)$ such that for every round after $r(\delta)$, arm i^* defeats *every other arm*.

Under the event $G \cap E(\delta)$, we analyze the regret in two parts: (i) regret incurred up to round $r(\delta)$, which is upper bounded by $K^2 \sum_{r \leq r(\delta)} q^r$ and (ii) regret after $r(\delta)$, which is the regret incurred in eliminating sub-optimal arms using i^* as an anchor. Finally, we can use the high-probability bound to also obtain a bound on the expected regret, proving Theorem 5.3.7.

7.2.1 The Analysis

In this section, we prove high-probability and expected regret bounds for `C2B`. Recall that $q = T^{1/B}$, and that $q \geq 2$. We first prove the following lemma which will be used to prove that i^* is never eliminated.

Lemma 7.2.1. *For any batch $r \in [B]$, and for any pair (i, j), we have*

$$\mathbf{P}\left(|\widehat{p}_{i,j}(r) - p_{i,j}| > \gamma_{i,j}(r)\right) \leq 2\eta,$$

where $\eta = 1/K^2BT$.

Proof. Note that $\mathbb{E}[\widehat{p}_{i,j}(r)] = p_{i,j}$, and applying Hoeffding's inequality gives

$$\mathbf{P}\left(|\widehat{p}_{i,j}(r) - p_{i,j}| > \gamma_{i,j}(r)\right) \leq 2\exp\left(-2N_{i,j}(r) \cdot \gamma_{i,j}(r)^2\right) \leq 2\eta.$$

□

We first define the *good* event G as follows.

Definition 7.2.1 (Event G). *An estimate $\widehat{p}_{i,j}(r)$ at the end of batch r is* ***strongly-correct*** *if $|\widehat{p}_{i,j}(r) - p_{i,j}| \leq \gamma_{i,j}(r)$. We say that event G occurs if every estimate in every batch $r \in [B]$ is strongly-correct.*

The following two lemmas show that G occurs with high probability and that i^* is not eliminated under G.

Lemma 7.2.2. *The probability that every estimate in every batch of* `C2B` *is strongly-correct is at least* $1 - 1/T$.

Proof. Applying Lemma 7.2.1 and taking a union bound over all pairs and batches, we get that the probability that some estimate is not strongly-correct is at most $\binom{K}{2} \times B \times 2\eta \le \frac{1}{T}$ where $\eta = 1/K^2BT$. Thus, $\mathbf{P}(\overline{G}) \le \frac{1}{T}$. □

We now show that, under event G, the best arm i^* is never eliminated.

Lemma 7.2.3. *Conditioned on G, the best arm i^* is never eliminated from $\mathcal{A}$ in the elimination step of* `C2B`.

Proof. In `C2B`, an arm j is deleted in batch r iff there is an arm $i \in \mathcal{A}$ with $\widehat{p}_{i,j}(r) > \frac{1}{2} + \gamma_{i,j}(r)$. If i^* is eliminated due to some arm j, then by definition of event G, we get $p_{j,i^*} \ge \widehat{p}_{i,j}(r) - \gamma_{i,j}(r) > \frac{1}{2}$, a contradiction. □

High-probability Regret Bound

We now prove Theorem 5.3.6. Fix any $\delta > 0$. We first define another good event as follows.

Definition 7.2.2 (Event $E(\delta)$)**.** *An estimate $\widehat{p}_{i,j}(r)$ in batch r is **weakly-correct** if $|\widehat{p}_{i,j}(r) - p_{i,j}| \le c_{i,j}(r)$. Let $C(\delta) := \lceil \frac{1}{2}\log_q(1/\delta) \rceil$. We say that event $E(\delta)$ occurs if for each batch $r \ge C(\delta)$, every estimate is weakly-correct.*

The next lemma shows that $E(\delta)$ occurs with probability at least $1 - \delta$.

Lemma 7.2.4. *For all $\delta > 0$, we have*

$$\mathbf{P}(\neg E(\delta)) \;=\; \mathbf{P}\left(\exists r \ge C(\delta), i, j : |\widehat{p}_{i,j}(r) - p_{i,j}| > c_{i,j}(r)\right) \;\le\; \delta.$$

Proof. For any pair i, j of arms and round r, let $B_{i,j}(r)$ denote the event that $|\widehat{p}_{i,j}(r) - p_{i,j}| > c_{i,j}(r)$. Note that $N_{ij}(r) \le \sum_{s=1}^{r} q_s \le 2q_r$. For any integer n, let $s_{ij}(n)$ denote the sample average of n independent Bernoulli r.v.s with probability p_{ij}. By Hoeffding's bound,

$$\mathbf{P}[|s_{ij}(n) - p_{ij}| > c] \le 2e^{-2nc^2}, \qquad \text{for any } c \in [0,1].$$

We now bound

$$\begin{aligned}
\mathbf{P}[B_{ij}(r)] &\le \sum_{n=0}^{2q_r} \mathbf{P}[B_{ij}(r) \wedge N_{ij}(r) = n] \\
&\le \sum_{n=0}^{2q_r} \mathbf{P}\left[|s_{ij}(n) - p_{ij}| > \sqrt{\frac{2\log(2K^2 q_r)}{n}}\right] \le \sum_{n=0}^{2q_r} 2\exp\left(-2n \cdot \frac{2\log(2K^2 q_r)}{n}\right) \\
&\le 4q_r \cdot \frac{1}{(2K^2 q_r)^4} \le \frac{1}{4K^2 \cdot q_r^2}
\end{aligned}$$

The second inequality uses the definition of $c_{ij}(r)$ when $N_{ij}(r) = n$. The last inequality uses $K \ge 2$. Next, by a union bound over arms and rounds, we can write the desired probability as

$$\begin{aligned}
\mathbf{P}(\exists r \ge C(\delta), i, j : B_{i,j}(r)) &\le \sum_{r \ge C(\delta)} \sum_{i<j} \mathbf{P}(B_{i,j}(r)) \\
&\le \sum_{r \ge C(\delta)} \binom{K}{2} \cdot \frac{1}{4K^2 \cdot q_r^2} \le \sum_{r \ge C(\delta)} \frac{1}{8q_r^2} \\
&\le \sum_{r \ge C(\delta)} \frac{1}{2q^{2r}} = \frac{1}{2q^{2C(\delta)}} \cdot \left(1 + \frac{1}{q^2} + \frac{1}{q^4} + \cdots\right) \le \frac{1}{q^{2C(\delta)}} \le \delta
\end{aligned} \tag{7.3}$$

The second inequality above uses the bound on $\mathbf{P}[B_{ij}(r)]$. The first inequality in (7.3) uses $q_r = \lfloor q^r \rfloor \ge q^r - 1 \ge \frac{q^r}{2}$ as $q \ge 2$. The last inequality in (7.3) uses the definition of $C(\delta)$.

The lemma now follows by the definition of event $\neg E(\delta)$ as $\exists r \ge C(\delta), i, j : B_{i,j}(r)$. $\square$

We will analyze our algorithm under both events G and $E(\delta)$. *Conditioned on these*, we next show:

- The best arm, i^*, is *not defeated* by any arm i in any round $r > C(\delta)$ (Lemma 7.2.5).
- Furthermore, there exists a round $r(\delta) \ge C(\delta)$ such that arm i^* defeats *every other arm*, in every round after $r(\delta)$ (Lemma 7.2.7).

Intuitively, these observations imply that our algorithm identifies the best arm after $r(\delta)$ rounds. Thus, beyond round $r(\delta)$, we only perform pairwise comparisons of the form (i^*, i) for $i \neq i^*$: thus, i^* is used as an anchor to eliminate sub-optimal arms. Note that event G is required to ensure that i^* is not eliminated (especially in rounds before $C(\delta)$ where the Lemma 7.2.4 does not apply). We now prove the aforementioned observations.

Lemma 7.2.5. *Conditioned on G and $E(\delta)$, for any round $r > C(\delta)$, arm i^* is not defeated by any other arm, i.e., $i^* \notin \cup_{i \neq i^*} D_r(i)$.*

Proof. Fix any round $r \geq C(\delta) + 1$. Suppose that $i^* \in D_r(i)$ for some other arm i. This implies that $\widehat{p}_{i,i^*}(r-1) > \frac{1}{2} + c_{i,i^*}(r-1)$. But under event $E(\delta)$, we have $|\widehat{p}_{i,i^*}(r-1) - p_{i,i^*}| \leq c_{i,i^*}(r-1)$ because $r - 1 \geq C(\delta)$. Combined, these two observations imply $p_{i,i^*} > \frac{1}{2}$, a contradiction. □

To proceed, we need the following definitions.

Definition 7.2.3. *The candidate i_r of round r is called the **champion** if $|D_r(i_r)| = |\mathcal{A}| - 1$; that is, if i_r defeats every other active arm.*

Definition 7.2.4. *Let $r(\delta) \geq C(\delta) + 1$ be the smallest integer such that*

$$q^{r(\delta)} \geq 2A \log A, \qquad \text{where } A := \frac{32}{\Delta^2_{\min}} \cdot \log(2K^2).$$

We use the following inequality based on this choice of $r(\delta)$.

Lemma 7.2.6. *The above choice of $r(\delta)$ satisfies*

$$q^r > \frac{8}{\Delta^2_{\min}} \cdot \log\left(2K^2 q_r\right), \qquad \forall r \geq r(\delta).$$

Proof. Using the fact that $q_r \leq q^r$, it suffices to show $q^r \geq \frac{8}{\Delta^2_{\min}} \cdot (\log(2K^2) + \log q^r)$. Moreover,

$$\log(2K^2) + \log q^r \leq \left(1 + \log(2K^2)\right) \cdot (1 + \log q^r) \leq 4 \cdot \log(2K^2) \cdot \log q^r,$$

where the last inequality uses $K \geq 2$, $r \geq r(\delta) \geq 1$ and $q \geq 2$. So, it suffices to show:

$$q^r > A \cdot \log(q^r), \quad \forall r \geq r(\delta), \qquad \text{where } A = \frac{32}{\Delta_{\min}^2} \cdot \log(2K^2) \tag{7.4}$$

Below, let $x = q^r$, $R := 2A \log A$ and function $f(x) := x - A \log x$. We will show that $f(x) > 0$ for all $x \geq R$, which would imply (7.4) because $q^{r(\delta)} \geq R$. As $R \geq A$, and f is increasing for $x \geq A$, it suffices to show that $f(R) \geq 0$. Indeed,

$$\frac{f(R)}{A} = 2\log A - \log(2A\log A) = \log A - \log(2\log A) > 0,$$

where the inequality uses $A \geq 8$. $\square$

Then, we have the following.

Lemma 7.2.7. *Conditioned on G and $E(\delta)$, the best arm i^* is the champion in every round $r > r(\delta)$.*

Proof. We first argue that i^* is compared to all active arms in each round $r \geq r(\delta)$. By Lemma 7.2.3, we know $i^* \in \mathcal{A}$. By Lemma 7.2.5, we have $i^* \notin D_r(j)$ for all $j \neq i^*$ because $r \geq r(\delta) \geq 1 + C(\delta)$. If candidate $i_r \neq i^*$, then i^* will be compared to all $j \in \mathcal{A}$ (since $i^* \notin D_r(i_r)$). On the other hand, if $i_r = i^*$, then (1) for any $j \in D_r(i^*)$, arm j is only compared to i^*, and (2) for any $j \in \mathcal{A} \setminus D_r(i^*)$, arm j is compared to all active arms including i^*.

Next, we show that for any round $r \geq r(\delta) + 1$, arm i^* defeats all other arms, i.e., $|D_r(i^*)| = |\mathcal{A}| - 1$. This would imply that $i_r = i^*$ and i^* is the champion. Consider any arm $j \in \mathcal{A} \setminus i^*$. Since i^* is compared to all active arms in round $r - 1 \geq r(\delta)$, we have

$$N_{i^*,j}(r-1) \geq q^{r-1} > \frac{8}{\Delta_{\min}^2} \cdot \log\left(2K^2 q_{r-1}\right),$$

where the second inequality is by Lemma 7.2.6 with $r - 1 \geq r(\delta)$. Now, by definition, we have

$$c_{i^*,j}(r-1) = \sqrt{\frac{2\log\left(2K^2 q_{r-1}\right)}{N_{i^*,j}(r-1)}} < \sqrt{\frac{2\log\left(2K^2 q_{r-1}\right)}{\frac{8}{\Delta_{\min}^2}\log\left(2K^2 q_{r-1}\right)}} = \frac{\Delta_{\min}}{2}.$$

Given this, it is easy to show that i^* defeats arm j in round r. Conditioned on $E(\delta)$, we know that $|\widehat{p}_{i^*,j}(r-1) - p_{i^*,j}| \le c_{i^*,j}(r-1) < \frac{\Delta_{\min}}{2}$. Then, we have

$$\widehat{p}_{i^*,j}(r-1) > p_{i^*,j} - \frac{\Delta_{\min}}{2} = \frac{1}{2} + \Delta_j - \frac{\Delta_{\min}}{2} \ge \frac{1}{2} + \frac{\Delta_{\min}}{2} > \frac{1}{2} + c_{i^*,j}(r-1).$$

Therefore, $j \in D_r(i^*)$. It now follows that for any round $r \ge r(\delta)+1$, arm i^* is the champion. □

We are now ready to prove Theorem 5.3.6.

Proof of Theorem 5.3.6. First, recall that in round r of C2B, any pair is compared $q_r = \lfloor q^r \rfloor$ times where $q = T^{1/B}$. Since $q^B = T$, C2B uses at most B rounds.

For the rest of proof, we fix $\delta > 0$. We now analyze the regret incurred by C2B, conditioned on events G and $E(\delta)$. Recall that $\mathbf{P}(G) \ge 1 - 1/T$ (Lemma 7.2.2), and $\mathbf{P}(E(\delta)) \ge 1-\delta$ (Lemma 7.2.4). Thus, $\mathbf{P}(G \cap E(\delta)) \ge 1 - \delta - 1/T$. Let R_1 and R_2 denote the regret incurred before and after round $r(\delta)$ (see Definition 7.2.4) respectively.

Bounding R_1. This is the regret incurred up to (and including) round $r(\delta)$. We upper bound the regret by considering all pairwise comparisons every round $r \le r(\delta)$.

$$\begin{aligned} R_1 \;&\le\; K^2 \cdot \sum_{r \le r(\delta)} q_r \le \; K^2 \cdot \sum_{r\le r(\delta)} q^r \;\le\; 2K^2 \cdot q^{r(\delta)} \\ &\le 2K^2 \cdot \max\left\{q \cdot 2A \log A \,,\, q^{C(\delta)+1}\right\}, \end{aligned}$$

where the last inequality uses Definition 7.2.4; recall $A = \frac{32}{\Delta_{\min}^2} \cdot \log(2K^2)$. Plugging in the value of $C(\delta) \le 1 + \frac{1}{2}\log_q(1/\delta)$, we obtain

$$R_1 \le O(K^2) \cdot \max\left\{ q \cdot \frac{\log K}{\Delta_{\min}^2} \cdot \log\left(\frac{\log K}{\Delta_{\min}}\right) \,,\, q^2\sqrt{\frac{1}{\delta}} \right\}. \tag{7.5}$$

Bounding R_2. This is the regret in rounds $r \ge r(\delta)+1$. By Lemma 7.2.7, arm i^* is the champion in all these rounds. So, the only comparisons in these rounds are of the form (i^*, j) for $j \in \mathcal{A}$.

Consider any arm $j \neq i^*$. Let T_j be the total number of comparisons that j participates in after round $r(\delta)$. Let r be the penultimate round that j is played in. We can assume that $r \geq r(\delta)$ (otherwise arm j will never participate in rounds after $r(\delta)$, i.e., $T_j = 0$). As arm j is *not* eliminated after round r,

$$\widehat{p}_{i^*,j}(r) \leq \frac{1}{2} + \gamma_{i^*,j}(r).$$

Moreover, by $E(\delta)$, we have $\widehat{p}_{i^*,j}(r) \geq p_{i^*,j} - c_{i^*,j}(r)$ because $r \geq r(\delta) \geq C(\delta)$. So,

$$\frac{1}{2} + \Delta_j = p_{i^*,j} \leq \widehat{p}_{i^*,j}(r) + c_{i^*,j}(r) \leq \frac{1}{2} + \gamma_{i^*,j}(r) + c_{i^*,j}(r).$$

It follows that

$$\Delta_j \leq \gamma_{i^*,j}(r) + c_{i^*,j}(r) \leq \frac{3}{\sqrt{2}}\sqrt{\frac{\log(2K^2BT)}{N_{i^*,j}(r)}}$$

where the final inequality follows by definition of c and γ. On re-arranging, we get $N_{i^*,j}(r) \leq \frac{9\log(2K^2BT)}{2\Delta_j^2}$. As $r+1$ is the last round that j is played in, and j is only compared to i^* in each round after $r(\delta)$,

$$T_j \;\leq\; N_{i^*,j}(r+1) \;\leq\; N_{i^*,j}(r) + 2q \cdot N_{i^*,j}(r) \;\leq\; \frac{15q \cdot \log(2K^2BT)}{\Delta_j^2}.$$

The second inequality follows since j is compared to i^* in rounds r and $r+1$, and the number of comparisons in round $r+1$ is $\lfloor q^{r+1} \rfloor \leq q \cdot (2q_r) \leq 2q \cdot N_{i^*,j}(r)$. Adding over all arms j, the total regret accumulated beyond round $r(\delta)$ is

$$R_2 = \sum_{j \neq i^*} T_j \Delta_j \leq \sum_{j \neq i^*} O\left(\frac{q \cdot \log(KT)}{\Delta_j}\right). \tag{7.6}$$

Combining (7.5) and (7.6), and using $q = T^{1/B}$, we obtain

$$R(T) \leq O\left(T^{1/B} \cdot \frac{K^2 \log(K)}{\Delta_{\min}^2} \cdot \log\left(\frac{\log K}{\Delta_{\min}}\right)\right) + O\left(T^{2/B} \cdot K^2 \cdot \sqrt{\frac{1}{\delta}}\right) + \sum_{j \neq i^*} O\left(\frac{T^{1/B} \cdot \log(KT)}{\Delta_j}\right).$$

This completes the proof Theorem 5.3.6. $\square$

Expected Regret Bound

In this section, we present the proof of Theorem 5.3.7. We first state the definitions needed in the proof. Let $F_X(\cdot)$ denote the cumulative density function (CDF) of a random variable X; that is, $F_X(x) = \mathbf{P}(X \leq x)$. The inverse CDF of X, denoted F_X^{-1}, is defined as $F_X^{-1}(z) = \inf\{x : \mathbf{P}(X \leq x) \geq z\}$ where $z \in [0,1]$. We will use the identity $\mathbb{E}[X] = \int_0^1 F_X^{-1}(z)dz$.

Proof of Theorem 5.3.7. First, note that in round r of `C2B`, any pair is compared $q_r = \lfloor q^r \rfloor$ times where $q = T^{1/B}$. Since $q^B = T$, `C2B` uses at most B rounds.

Let $R(T)$ be the random variable denoting the regret incurred by `C2B`. By Theorem 5.3.6, we know that with probability at least $1 - \delta - 1/T$,

$$R(T) \leq O\left(T^{1/B} \cdot \frac{K^2 \log(K)}{\Delta_{\min}^2} \cdot \log\left(\frac{\log K}{\Delta_{\min}}\right)\right) + O\left(T^{2/B} \cdot K^2 \cdot \sqrt{\frac{1}{\delta}}\right) + \sum_{j \neq i^*} O\left(\frac{T^{1/B} \cdot \log(KT)}{\Delta_j}\right).$$

Thus, $F_{R(T)}^{-1}(1 - \delta - 1/T) \leq G(\delta)$ where

$$G(\delta) := A + O\left(T^{2/B} \cdot K^2 \cdot \sqrt{\frac{1}{\delta}}\right) + B$$

where to simplify notation we set $A = O\left(T^{1/B} \cdot \frac{K^2 \log(K)}{\Delta_{\min}^2} \cdot \log\left(\frac{\log K}{\Delta_{\min}}\right)\right)$ and $B = \sum_{j \neq i^*} O\left(\frac{T^{1/B} \cdot \log(KT)}{\Delta_j}\right)$. Using the identity for expectation of a random variable, we get

$$\begin{aligned}
\mathbb{E}[R(T)] &= \int_0^1 F_{R(T)}^{-1}(z)dz \\
&= \int_0^{1-\frac{1}{T}} F_{R(T)}^{-1}(z)dz + \underbrace{\int_{1-\frac{1}{T}}^{T} F_{R(T)}^{-1}(z)dz}_{\leq T \cdot \frac{1}{T} = 1} \\
&\leq \int_0^{1-\frac{1}{T}} F_{R(T)}^{-1}(z)dz \quad + \quad 1 \\
&= \int_{1-\frac{1}{T}}^{0} F_{R(T)}^{-1}\left(1 - \delta - \frac{1}{T}\right)(-d\delta) \quad + \quad 1 \\
&\leq \int_0^{1-\frac{1}{T}} G(\delta)d\delta \quad + \quad 1 \\
&\leq A + O\left(T^{2/B} \cdot K^2\right) + B + 1
\end{aligned}$$

where the fourth equality follows by setting $1 - q - 1/T = \delta$ and the final inequality follows since $\int_0^1 \left(\frac{1}{\delta}\right)^{1/2} \leq 2$. Thus,

$$\mathbb{E}[R(T)] \leq O\left(T^{1/B} \cdot \frac{K^2 \log(K)}{\Delta_{\min}^2} \cdot \log\left(\frac{\log K}{\Delta_{\min}}\right)\right) + O\left(T^{2/B} \cdot K^2\right) + \sum_{j \neq i^*} O\left(\frac{T^{1/B} \cdot \log(KT)}{\Delta_j}\right).$$

This completes the proof of Theorem 5.3.7. □

7.3 Computational Results

In this section, we provide details of our computational experiments. The goal of our experiments is to answer the following questions: (i) How does the regret of C2B using $B = \lfloor \log(T) \rfloor$ batches compare to that of existing fully sequential as well as batched algorithms? and (ii) Can the regret of C2B match the regret of the best known sequential algorithms; if yes, then how many rounds suffice to achieve this? Towards answering (i), we compare C2B to a representative set of sequential algorithms for dueling bandits using the library due to [76]. We compare C2B to the sequential algorithms RUCB [117], RMED [76], and Beat-the-Mean (BTM) [114]. We allow these algorithms to work as prescribed; that is, they work in $B = T$ batches. The reason that we chose these sequential algorithms is that our batched algorithm (C2B) is based on a similar paradigm, and such a comparison demonstrates the power of adaptivity in this context. We also compare C2B to the batched algorithm SCOMP2 [4]. We plot the cumulative regret $R(t)$ incurred by the algorithms against time t. We set $B = \lfloor \log(T) \rfloor$ for C2B and SCOMP2 in this experiment. For (ii), we increased B by a small amount; we found that the performance of C2B improves noticeably when given a constant number of additional rounds (we use $B = \lfloor \log(T) \rfloor + 6$ in this experiment). We perform these experiments using the following real-world datasets.

Six rankers. This dataset is based on the 6 retrieval functions used in the engine of ArXiv.org.

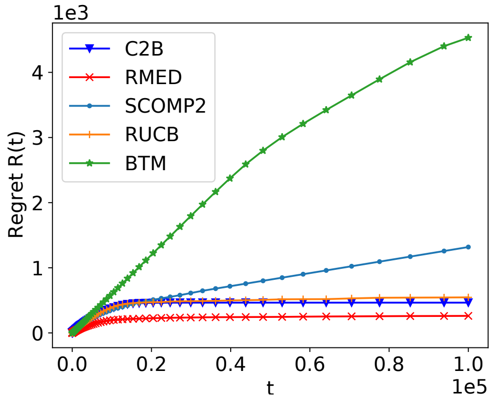
Sushi. The Sushi dataset is based on the Sushi preference dataset [72] that contains the preference data regarding 100 types of Sushi. A preference dataset using the top-16 most popular types of sushi is obtained.

Irish election data. The Irish election data for Dublin and Meath is available at *preflib.org*. It contains partial preference orders over candidates. As in [5], these are transformed into preference matrices by selecting a subset of candiates to ensure that a Condorcet winner exists. There are 12 candidates in the **Irish-Meath** dataset, and 8 in the **Irish-Dublin** dataset.

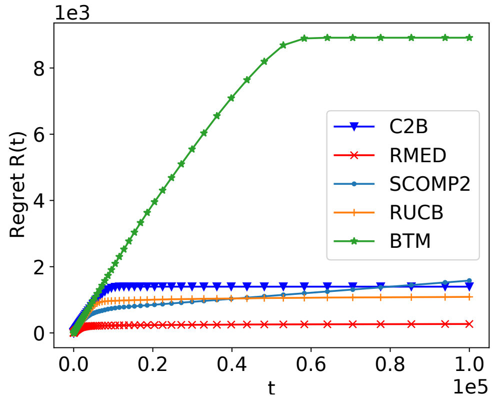
MSLR and Yahoo! data. We also run experiments on two web search ranking datasets: the Microsoft Learning to Rank (MSLR) dataset [94] and the Yahoo! Learning to Rank Challenge Set 1 [28]. These datasets have been used in prior work on online ranker evaluation [116, 81]. We use preference matrices generated using the "navigational" configuration (see [81] for details). The MSLR dataset has 136 rankers and the Yahoo! dataset has 700 rankers. We sample 30 rankers from each dataset while ensuring the existence of a Condorcet winner. In this way, we obtain two datasets, denoted **MSLR30** and **Yahoo30**.

Note that there exists a Condorcet winner in all datasets. We repeat each experiment 20 times and report the average regret. In our algorithm, we use the *KL-divergence based confidence bound* due to [76] for elimination as it performs much better empirically, and our theoretical bounds continue to hold (see §7.4). This KL-divergence based elimination criterion eliminates an arm i in round r if $I_i(r) - I^*(r) > \log(T) + f(K)$ where $I_i(r) = \sum_{j:\widehat{p}_{i,j}(r)<\frac{1}{2}} N_{i,j}(r) \cdot D_{\mathrm{KL}}(\widehat{p}_{i,j}(r), \frac{1}{2})$ and $I^*(r) = \min_{j\in[K]} I_i(r)$.

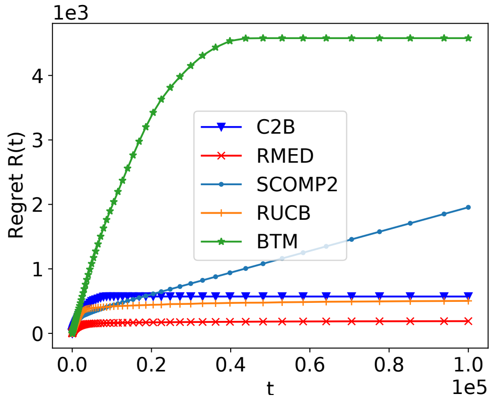
Computational Results. As mentioned earlier, we compare our algorithms against a representative set of sequential dueling bandits algorithms (RUCB, RMED, and BTM). We set $\alpha = 0.51$ for RUCB, and $f(K) = 0.3K^{1.01}$ for RMED and `C2B`, and $\gamma = 1.3$ for BTM: these parameters are known to perform well both theoretically and empirically [76]. We set $T = 10^6$ for MSLR30 and Yahoo30 datasets (as they have larger number of arms), and $T = 10^5$ for the remaining four. For the first set of experiments, we set $B = \lfloor \log(T) \rfloor$. We observe that `C2B` always outperforms BTM and beats SCOMP2 on most of the datasets. We observe that even when SCOMP2 beats `C2B` it has a slightly linear curve (implying that its regret would keep increasing as T increases) while the regret curve of `C2B` is mostly flat. Furthermore, `C2B` performs comparably to RUCB in all datasets except Yahoo30. We plot the results in Figure 7.1. In the second set of experiments, we set $B = \lfloor \log(T) \rfloor + 6$. We observe that `C2B` always outperforms RUCB and, in fact, performs comparably to RMED



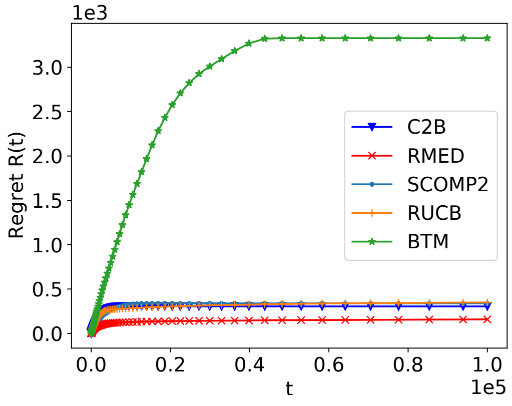
(a) Six rankers



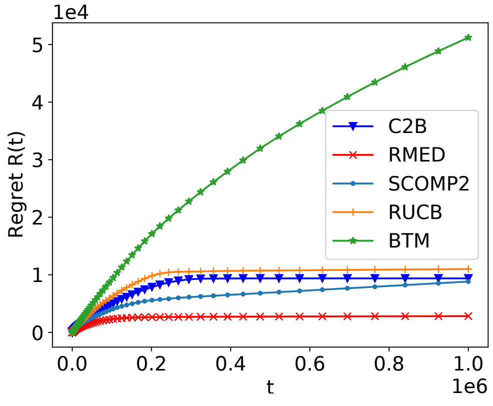
(b) Sushi



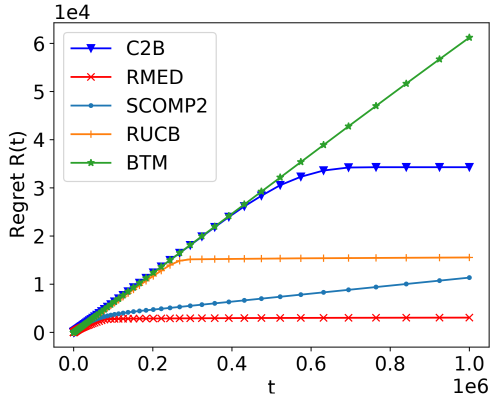
(c) Irish-Meath



(d) Irish-Dublin

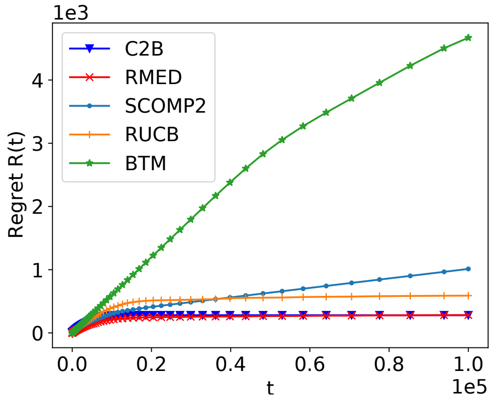


(e) MSLR30

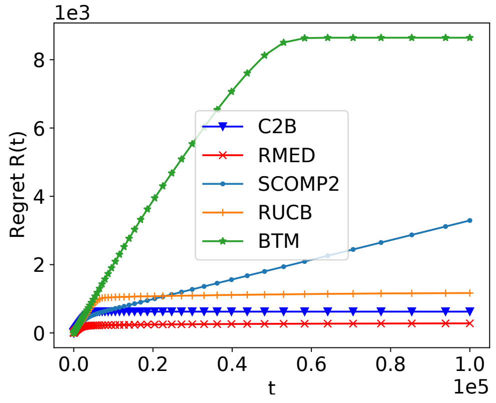


(f) Yahoo30

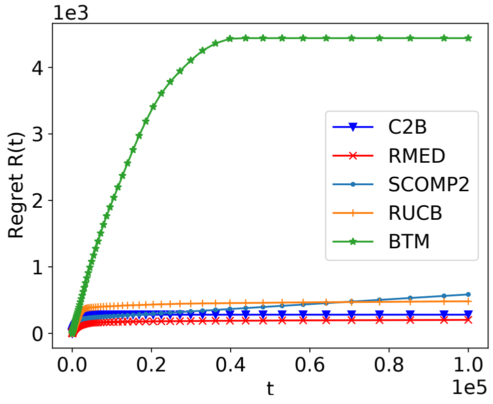
Figure 7.1: Regret v/s t plots of algorithms when $B = \lfloor \log(T) \rfloor$



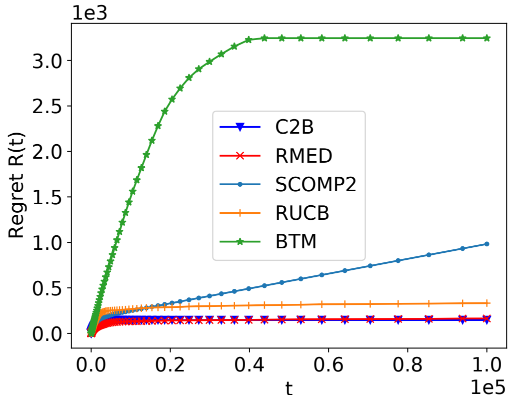
(a) Six rankers



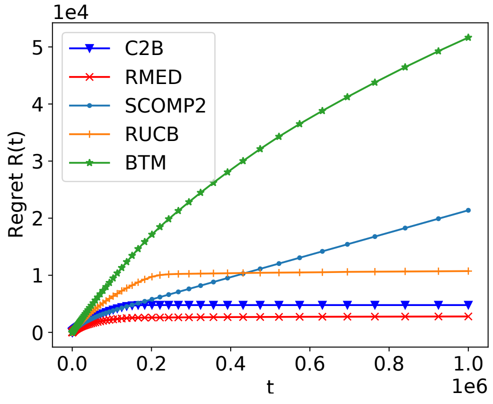
(b) Sushi



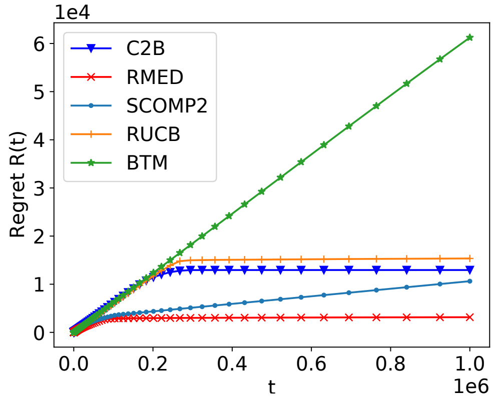
(c) Irish-Meath



(d) Irish -Dublin



(e) MSLR30



(f) Yahoo30

Figure 7.2: Regret v/s t plots of algorithms when $B = \lfloor \log(T) \rfloor + 6$

on all datasets except Yahoo30. We plot the results in Figure 7.2. Finally, we note that SCOMP2 exhibits varying performance across runs (even on the same dataset) and we think that this is due to the randomness involved in selecting the "seed set".

7.4 The Batched Algorithm with KL-based Elimination Criterion

In this section, we modify `C2B` to use a Kullback-Leibler divergence based elimination criterion. We provide a complete description of the algorithm, denoted `C2B-KL`, in Algorithm 15. In what follows, we highlight the main differences of `C2B-KL` from `C2B`. Recall the following notation. We use $\mathcal{A}$ to denote the current set of *active* arms; i.e., the arms that have not been eliminated. We use index r for rounds or batches. If pair (i, j) is compared in round r, it is compared $q_r = \lfloor q^r \rfloor$ times where $q = T^{1/B}$. We define the following quantities at the *end* of each round r:

- $N_{i,j}(r)$ is the total number of times the pair (i, j) has been compared.
- $\widehat{p}_{i,j}(r)$ is the frequentist estimate of $p_{i,j}$, i.e.,

$$\widehat{p}_{i,j}(r) = \frac{\#\ i \text{ wins against } j \text{ until end of round } r}{N_{i,j}(r)}. \tag{7.7}$$

- A confidence-interval radius for each (i, j) pair:

$$c_{i,j}(r) = \sqrt{\frac{2\log(2K^2 q_r)}{N_{i,j}(r)}}$$

- We define a term $I_j(r)$ which, at a high-level, measures how unlikely it is for j to be the Condorcet winner at the end of batch r:

$$I_j(r) = \sum_{i:\widehat{p}_{i,j}(r)\geq\frac{1}{2}} D_{\mathrm{KL}}\left(\widehat{p}_{i,j}(r), \frac{1}{2}\right) \cdot N_{i,j}(r),$$

where $D_{\mathrm{KL}}(p,q)$ denotes the Kullback–Leibler divergence between two Bernoulli distributions: $B(p)$ and $B(q)$. We define $I^*(r) = \min_{j\in\mathcal{A}} I_j(r)$.

The B-round algorithm, `C2B-KL`, proceeds exactly as `C2B`. The only change is in the *elimination criterion*, which we describe next.

Elimination Criterion. In round r, if, for any arm j, we have $I_j(r) - I^*(r) > \log(T) + f(K)$, then j is eliminated from $\mathcal{A}$. Here $f(K)$ is a non-negative function of K, independent of r.

The main result of this section is to show that `C2B-KL` achieves the following guarantee.

Theorem 7.4.1. *For any integer $B \geq 1$, there is an algorithm for the K-armed dueling bandit problem that uses at most B rounds. Furthermore, for any $\delta > 0$, with probability at least $1 - \delta - \frac{1}{T}\cdot e^{K\log(C)-f(K)}$, where C is some constant (see Lemma 7.4.2), its regret under the Condorcet condition is at most*

$$R(T) \leq O\left(T^{1/B}\cdot\frac{K^2\log(K)}{\Delta_{\min}^2}\cdot\log\left(\frac{\log K}{\Delta_{\min}}\right)\right) + O\left(T^{2/B}\cdot K^2\cdot\sqrt{\frac{1}{\delta}}\right) + \sum_{j\neq i^*} O\left(\frac{T^{1/B}\cdot\log(T)}{\Delta_j}\right) + \sum_{j\neq i^*} O\left(\frac{T^{1/B}\cdot f(K)}{\Delta_j}\right)$$

Remark. Setting $f(K) > K\log(C)$, we get the same asymptotic expected regret bound as in Theorem 5.3.7. Following [76], we set $f(K) = 0.3K^{1.01}$ in our experiments.

We require the following result in the proof of Theorem 7.4.1.

Fact 7.4.1. *For any μ and μ_2 satisfying $0 < \mu_2 < \mu < 1$. Let $C_1(\mu,\mu_2) = (\mu-\mu_2)^2/(2\mu(1-\mu_2))$. Then, for any $\mu_3 \leq \mu_2$,*

$$D_{KL}(\mu_3,\mu) - D_{KL}(\mu_3,\mu_2) \geq C_1(\mu,\mu_2) > 0.$$

The high-level outline of the analysis is exactly the same as that of `C2B`. For completeness, we provide the analysis in the following section; however, we skip the proofs of lemmas that follow from the analysis of `C2B`.

Algorithm 15 C2B-KL

```
1: Input: Arms $\mathcal{B}$, time-horizon $T$, integer $B \geq 1$
2: active arms $\mathcal{A} \leftarrow \mathcal{B}$, $r \leftarrow 1$, emprical probabilities $\widehat{p}_{i,j}(0) = \frac{1}{2}$ for all $i, j \in \mathcal{B}^2$
3: while number of comparisons $\leq T$ do
4:     if $\mathcal{A} = \{i\}$ for some $i$ then play $(i, i)$ for remaining trials
5:     $D_r(i) \leftarrow \{j \in \mathcal{A} : \widehat{p}_{i,j}(r-1) > \frac{1}{2} + c_{i,j}(r-1)\}$
6:     $i_r \leftarrow \arg\max_{i \in \mathcal{A}} |D_r(i)|$
7:     for $i \in \mathcal{A} \setminus \{i_r\}$ do
8:         if $i \in D_r(i_r)$ then
9:             compare $(i_r, i)$ for $q_r$ times
10:        else
11:            for each $j \in \mathcal{A}$, compare $(i, j)$ for $q_r$ times
12:    compute $\widehat{p}_{i,j}(r)$ values
13:    if $\exists j : I_j(r) - I^*(r) > \log(T) + f(K)$ then
14:        $\mathcal{A} \leftarrow \mathcal{A} \setminus \{j\}$
15:        $r \leftarrow r + 1$
```

7.4.1 The Analysis

In this section, we prove the high-probability regret bound for C2B-KL. Recall that $q = T^{1/B}$, and that $q \geq 2$. We first show that, with high probability, i^* is not eliminated during the execution of the algorithm. The following lemma formalizes this.

Lemma 7.4.2. *Let G denote the event that the best arm i^* is not eliminated during the execution of* C2B-KL*. We can bound the probability of $\overline{G}$ as follows.*

$$\mathbf{P}(\overline{G}) \leq \frac{1}{T} \cdot e^{K \log(C) - f(K)},$$

where $C = \max_j C(j) + 1$, is a constant, with $C(j) = \left(\frac{1}{e^{D_{KL}\left(p_{j,i^*}, 1/2\right)} - 1} + \frac{e^{C_1\left(p_{i^*,j}, 1/2\right)}}{\left(e^{C_1\left(p_{i^*,j}, 1/2\right)} - 1\right)^2} \right)$.

Proof. Let n_j denote the number of times i^* and j are compared. Let $\widehat{p}_{i^*,j}(n_j)$ denote the frequentist estimate of $p_{i^*,j}$ when i^* and j are compared n_j times (we will abuse notation and use $\widehat{p}_{i^*,j}$ when n_j is clear from context). Let $S \in 2^{[K] \setminus \{i^*\}} \setminus \emptyset$, and consider vector $\{n_j \in \mathbb{N} : j \in S\}$. We define $A = \sum_{j \in S} D_{\text{KL}}\left(\widehat{p}_{j,i^*}, 1/2\right) \cdot n_j$. Let $D(S; \{n_j : j \in S\})$ denote the event that i^* and j are compared n_j times and $\widehat{p}_{i^*,j} \leq 1/2$ for all $j \in S$, and that $A > \log(T) + f(K)$. The probability of this event upper bounds the probability that i^*

is eliminated (as per our elimination criterion) when i^* and j are compared n_j times, and $\widehat{p}_{i^*,j} \leq 1/2$ for all $j \in S$. We will show that

$$\mathbf{P}(D(S; \{n_j : j \in S\})) \leq \frac{e^{-f(K)}}{T} \prod_{j \in S} \left(e^{-n_j D_{\mathrm{KL}}\left(p_{j,i^*}, 1/2\right)} + n_j e^{C_1\left(p_{j,i^*}, 1/2\right)} \right) \tag{7.8}$$

where $C_1(\mu_1, \mu_2) = (\mu_1 - \mu_2)^2/(2\mu_1(1-\mu_2))$. Using the above, we first show that by taking a union bound over all $S \in 2^{[K]\setminus\{i^*\}} \setminus \emptyset$ and $\{n_j : j \in S\}$, we obtain the final result. We have

$$\begin{aligned}
\mathbf{P}(\overline{G}) &\leq \sum_{S \in 2^{[K]\setminus\{i^*\}} \setminus \emptyset} \sum_{n_j \in \mathbb{N}^{|S|}} \mathbf{P}(D(S; \{n_j : j \in S\})) \\
&\leq \sum_{S \in 2^{[K]\setminus\{i^*\}} \setminus \emptyset} \sum_{n_j \in \mathbb{N}^{|S|}} \frac{e^{-f(K)}}{T} \prod_{j \in S} \left(e^{-n_j D_{\mathrm{KL}}\left(p_{j,i^*}, 1/2\right)} + n_j e^{C_1\left(p_{j,i^*}, 1/2\right)} \right) \\
&= \frac{e^{-f(K)}}{T} \sum_{S \in 2^{[K]\setminus\{i^*\}} \setminus \emptyset} \prod_{j \in S} \sum_{n_j \in \mathbb{N}} \left(e^{-n_j D_{\mathrm{KL}}\left(p_{j,i^*}, 1/2\right)} + n_j e^{C_1\left(p_{j,i^*}, 1/2\right)} \right) && (7.9) \\
&= \frac{e^{-f(K)}}{T} \sum_{S \in 2^{[K]\setminus\{i^*\}} \setminus \emptyset} \prod_{j \in S} \left(\frac{1}{e^{D_{\mathrm{KL}}\left(p_{j,i^*}, 1/2\right)} - 1} + \frac{e^{C_1\left(p_{j,i^*}, 1/2\right)}}{\left(e^{C_1\left(p_{j,i^*}, 1/2\right)} - 1\right)^2} \right) && (7.10) \\
&\leq \frac{e^{-f(K)}}{T} \sum_{S \in 2^{[K]\setminus\{i^*\}} \setminus \emptyset} (C-1)^{|S|} \leq \frac{e^{-f(K)}}{T} \cdot C^K && (7.11) \\
&= \frac{1}{T} \cdot e^{K \log(C) - f(K)}
\end{aligned}$$

where (7.9) follows by swapping the order of summation and multiplication, (7.10) uses $\sum_{n=1}^{\infty} e^{-nx} = 1/(e^x - 1)$ and $\sum_{n=1}^{\infty} n e^{-nx} = e^x/(e^x-1)^2$, and (7.11) follows by letting $C(j) = \left(\frac{1}{e^{D_{\mathrm{KL}}\left(p_{j,i^*}, 1/2\right)} - 1} + \frac{e^{C_1\left(p_{j,i^*}, 1/2\right)}}{\left(e^{C_1\left(p_{j,i^*}, 1/2\right)} - 1\right)^2} \right)$, $C = \max_j C(j) + 1$ and the binomial theorem. To complete the proof, we need to prove (7.8).

For the remainder of this proof, we fix $S \in 2^{[K]\setminus\{i^*\}} \setminus \emptyset$, and vector $\{n_j \in \mathbb{N} : j \in S\}$. Observe that

$$\mathbf{P}(D(S; \{n_j : j \in S\})) = \mathbf{P}\left(A > \log(T) + f(K)\right) = \mathbf{P}\left(T < e^{-f(K)} \cdot e^A\right)$$

where we defined $A = \sum_{j \in S} D_{\mathrm{KL}}\left(\widehat{p}_{j,i^*}, 1/2\right) \cdot n_j$. By Markov's inequality, we have

$$\mathbf{P}\left(e^{-f(K)} \cdot e^A > T\right) \leq \frac{\mathbb{E}[e^{-f(K)} \cdot e^A]}{T} = \frac{e^{-f(K)}}{T} \cdot \mathbb{E}[e^A] \tag{7.12}$$

where the last equality follows since $f(K)$ is constant (with respect to $\{n_j\}$ values). So, it suffices to bound $\mathbb{E}[e^A]$. Towards this end, we define the following term:

$$P_j(x_j) = \mathbf{P}\left(\widehat{p}_{j,i^*} \geq \frac{1}{2} \text{ and } D_{\mathrm{KL}}\left(\widehat{p}_{j,i^*}, \frac{1}{2}\right) \geq x_j\right).$$

Then, we have

$$\begin{aligned}
\mathbb{E}[e^A] &= \int_{\{x_j\} \in [0,\log(2)]^{|S|}} \exp\left(\sum_{j \in S} n_j x_j\right) \prod_{j \in S} d(-P_j(x_j)) \\
&= \prod_{j \in S} \int_{x_j \in [0,\log 2]} e^{n_j x_j} d(-P_j(x_j)) && (7.13) \\
&= \prod_{j \in S} \left([-e^{n_j x_j} P_j(x_j)]_0^{\log(2)} + \int_{x_j \in [0,\log(2)]} n_j e^{n_j x_j} P_j(x_j) dx_j \right) && (7.14) \\
&= \prod_{j \in S} \left(P_j(0) + \int_{x_j \in [0,\log(2)]} n_j e^{n_j x_j} P_j(x_j) dx_j \right) \\
&\leq \prod_{j \in S} \left(e^{-n_j D_{\mathrm{KL}}\left(p_{j,i^*}, 1/2\right)} + \int_{x_j \in [0,\log(2)]} n_j e^{n_j x_j} e^{-n_j\left(x_j + C_1\left(p_{j,i^*}, 1/2\right)\right)} dx_j \right) && (7.15) \\
&= \prod_{j \in S} \left(e^{-n_j D_{\mathrm{KL}}\left(p_{j,i^*}, 1/2\right)} + \int_{x_j \in [0,\log(2)]} n_j e^{C_1\left(p_{j,i^*}, 1/2\right)} dx_j \right) \\
&\leq \prod_{j \in S} \left(e^{-n_j D_{\mathrm{KL}}\left(p_{j,i^*}, 1/2\right)} + n_j e^{C_1\left(p_{j,i^*}, 1/2\right)} \right)
\end{aligned}$$

where (7.13) follows from the independence of the comparisons. We obtain (7.14) by applying integration by parts, (7.15) follows from the Chernoff bound and Fact 7.4.1; here $C_1(\mu_1, \mu_2) = (\mu_1 - \mu_2)^2/(2\mu_1(1-\mu_2))$, and the final inequality follows by observing that $\int_{x_j \in [0,\log(2)]} n_j e^{C_1\left(p_{j,i^*}, 1/2\right)} dx_j = n_j e^{C_1\left(p_{j,i^*}, 1/2\right)} \cdot \int_{x_j \in [0,\log(2)]} dx_j = n_j e^{C_1\left(p_{j,i^*}, 1/2\right)} \log(2)$. Note that log refers to the natural logarithm, so we have $\log(2) \leq 1$. Combined with (7.12), this completes the proof of (7.8). □

High-probability Regret Bound

We now prove Theorem 7.4.1. Fix any $\delta > 0$. We first define event $E(\delta)$ as before.

Definition 7.4.1 (Event $E(\delta)$)**.** *An estimate $\widehat{p}_{i,j}(r)$ in batch r is **weakly-correct** if $|\widehat{p}_{i,j}(r) - p_{i,j}| \leq c_{i,j}(r)$. Let $C(\delta) := \lceil \frac{1}{2} \log_q(1/\delta) \rceil$. We say that event $E(\delta)$ occurs if for each batch $r \geq C(\delta)$, every estimate is weakly-correct.*

The next lemma shows that $E(\delta)$ occurs with probability at least $1 - \delta$. Since $E(\delta)$ does not depend on the elimination criterion, its proof follows from the analysis of `C2B`.

Lemma 7.4.3. *For all $\delta > 0$, we have*

$$\mathbf{P}(\neg E(\delta)) \;=\; \mathbf{P}\left(\exists r \geq C(\delta), i, j : |\widehat{p}_{i,j}(r) - p_{i,j}| > c_{i,j}(r)\right) \;\leq\; \delta.$$

As before, we analyze our algorithm under both events G and $E(\delta)$. Recall that, under event G, the best arm i^* is not eliminated. *Conditioned on these*, we next show:

- The best arm, i^*, is *not defeated* by any arm i in any round $r > C(\delta)$ (Lemma 7.4.4).
- Furthermore, there exists a round $r(\delta) \geq C(\delta)$ such that arm i^* defeats *every other arm*, in every round after $r(\delta)$ (Lemma 7.4.6).

We re-state the formal lemmas next.

Lemma 7.4.4. *Conditioned on G and $E(\delta)$, for any round $r > C(\delta)$, arm i^* is not defeated by any other arm, i.e., $i^* \notin \cup_{i \neq i^*} D_r(i)$.*

To proceed, we need the following definitions.

Definition 7.4.2. *The candidate i_r of round r is called the **champion** if $|D_r(i_r)| = |\mathcal{A}| - 1$; that is, if i_r defeats every other active arm.*

Definition 7.4.3. *Let $r(\delta) \geq C(\delta) + 1$ be the smallest integer such that*

$$q^{r(\delta)} \geq 2A \log A, \qquad \textit{where } A := \frac{32}{\Delta_{\min}^2} \cdot \log(2K^2).$$

We use the following inequality based on this choice of $r(\delta)$.

Lemma 7.4.5. *The above choice of $r(\delta)$ satisfies*

$$q^r > \frac{8}{\Delta_{\min}^2} \cdot \log\left(2K^2 q_r\right), \qquad \forall r \geq r(\delta).$$

Then, we have the following.

Lemma 7.4.6. *Conditioned on G and $E(\delta)$, the best arm i^* is the champion in every round $r > r(\delta)$.*

We are now ready to prove Theorem 7.4.1.

Proof of Theorem 7.4.1. First, recall that in round r of C2B, any pair is compared $q_r = \lfloor q^r \rfloor$ times where $q = T^{1/B}$. Since $q^B = T$, C2B uses at most B rounds.

For the rest of proof, we fix $\delta > 0$. We now analyze the regret incurred by C2B, conditioned on events G and $E(\delta)$. Recall that $\mathbf{P}(G) \geq 1 - \frac{1}{T} \cdot e^{K\log(C) - f(K)}$ (Lemma 7.4.2), and $\mathbf{P}(E(\delta)) \geq 1 - \delta$ (Lemma 7.4.3). Thus, $\mathbf{P}(G \cap E(\delta)) \geq 1 - \delta - \frac{1}{T} \cdot e^{K\log(C) - f(K)}$. Let R_1 and R_2 denote the regret incurred before and after round $r(\delta)$ (see Definition 7.4.3) respectively.

Bounding R_1. We can bound R_1 as in the proof of Theorem 5.3.6; so, we get

$$R_1 \leq O(K^2) \cdot \max\left\{ q \cdot \frac{\log K}{\Delta_{\min}^2} \cdot \log\left(\frac{\log K}{\Delta_{\min}}\right), \ q^2\sqrt{\frac{1}{\delta}} \right\}. \tag{7.16}$$

Bounding R_2. This is the regret in rounds $r \geq r(\delta) + 1$. By Lemma 7.4.6, arm i^* is the champion in all these rounds. So, the only comparisons in these rounds are of the form (i^*, j) for $j \in \mathcal{A}$.

Consider any arm $j \neq i^*$. Let T_j be the total number of comparisons that j participates in after round $r(\delta)$. Let r be the penultimate round that j is played in. We can assume that $r \geq r(\delta)$ (otherwise arm j will never participate in rounds after $r(\delta)$, i.e., $T_j = 0$). As arm j is *not* eliminated after round r,

$$I_j(r) - I^*(r) \leq \log(T) + f(K).$$

By Lemma 7.4.6, $I^*(r) = 0$ (since i^* is the *champion*, the summation is empty). So, we have $I_j(r) \leq \log(T) + f(K)$. Observe that

$$I_j(r) \geq D_{\mathrm{KL}}\left(\widehat{p}_{i^*,j}(r), \frac{1}{2}\right) N_{i^*,j}(r) \tag{7.17}$$

We can lower bound $D_{\mathrm{KL}}\left(\widehat{p}_{i^*,j}(r), \frac{1}{2}\right)$ as follows.

$$D_{\mathrm{KL}}\left(\widehat{p}_{i^*,j}(r), \frac{1}{2}\right) \geq \left(\widehat{p}_{i^*,j}(r) - \frac{1}{2}\right)^2 \geq \left(p_{i^*,j} - c_{i^*,j}(r) - \frac{1}{2}\right)^2 \geq \left(\frac{\Delta_j}{2}\right)^2$$

where the first inequality follows from Pinsker's inequality, the second inequality uses Lemma 7.4.3 and the final inequality uses the fact that $c_{i^*,j}(r) \leq \frac{\Delta_{\min}}{2}$, which follows by the choice of $r(\delta)$. Plugging this into (7.17), we get

$$\frac{\Delta_j^2}{4} \cdot N_{i^*,j}(r) \leq \log(T) + f(K)$$

which on re-arranging gives

$$N_{i^*,j}(r) \leq \frac{4(\log(T) + f(K))}{\Delta_j^2}.$$

As $r+1$ is the last round that j is played in, and j is only compared to i^* in each round after $r(\delta)$,

$$T_j \;\leq\; N_{i^*,j}(r+1) \;\leq\; N_{i^*,j}(r) + 2q \cdot N_{i^*,j}(r) \;\leq\; \frac{12q \cdot (\log(T) + f(K))}{\Delta_j^2}.$$

The second inequality follows since j is compared to i^* in rounds r and $r+1$, and the number of comparisons in round $r+1$ is $\lfloor q^{r+1} \rfloor \leq q \cdot (2q_r) \leq 2q \cdot N_{i^*,j}(r)$. Adding over all arms j, the total regret accumulated beyond round $r(\delta)$ is

$$R_2 = \sum_{j \neq i^*} T_j \Delta_j \leq \sum_{j \neq i^*} O\left(\frac{q \cdot (\log(T) + f(K))}{\Delta_j}\right). \tag{7.18}$$

Combining (7.16) and (7.18), and using $q = T^{1/B}$, we obtain

$$R(T) \leq O\left(T^{1/B} \cdot \frac{K^2 \log(K)}{\Delta_{\min}^2} \cdot \log\left(\frac{\log K}{\Delta_{\min}}\right)\right) \quad + O\left(T^{2/B} \cdot K^2 \cdot \sqrt{\frac{1}{\delta}}\right) + \sum_{j \neq i^*} O\left(\frac{T^{1/B} \cdot \log(T)}{\Delta_j}\right) + \sum_{j \neq i^*} O\left(\frac{T^{1/B} \cdot f(K)}{\Delta_j}\right)$$

This completes the proof Theorem 7.4.1. $\square$

Chapter 8

Directions for Future Work

We conclude this thesis with a couple of directions for future work.

Batched Algorithms for Other Stochastic Optimization Problems. A generalization of `SSC` and `ScnSC` asks for covering a submodular function when the costs of the stochastic items are given by a metric; for example, items may correspond to physical locations and 'selecting' an item corresponds to moving to the selected location using a shortest path. We can extend the approach from Chapter 2 to handle such costs and provide algorithms that obtain good approximation guarantees under limited adaptivity. The deterministic problem is well understood [67] but the stochastic variant has not yet been studied. The stochastic variant can be used to model the *Informative Path Planning* (IPP) problem that has applications in UAV Search-and-Rescue and Equivalence Class Determination [82] and warrants investigation. Furthermore, it would be interesting to investigate other models for the UAV Search-and-Rescue problem and design practical algorithms with limited adaptivity for it; for example, what if we have k UAVs that can communicate in some limited way? *Can we design algorithms that achieve good theoretical guarantees while being easy to implement in practice?*

The concept of *adaptive submodularity* generalizes `SSC` and captures certain problems that are not captured by `SSC`, for example, the influence maximization problem. [56] show that influence maximization under the *independent cascade* model and the *full adoption* model are special cases of adaptive submodularity. Hence, guarantees for adaptive submodularity (under full/limited adaptivity) imply guarantees for the influence maximization problem under these models. However, as demonstrated by the results in Chapter 4, we can obtain better guarantees for specific problems by exploiting the inherent structure in the problem; for example, we obtain a constant factor non-adaptive solution for `SSClass` whereas taking `SSClass` as a special case of `SSC` does not give us such strong guarantees. The influence maximization problem captures diffusion dynamics (depending on the model) and has important applications in capturing spread of epidemics and viral marketing. A direction for future research is to *design algorithms with limited adaptivity* for the influence maximization problem.

Stochastic Optimization with Bandit Feedback. In many stochastic optimization problems involving uncertainty in the input (for example, `SSC`, `ScnSC` and `SSClass`), a common assumption is that the distribution over input instances is known. While this assumption holds in settings with readily available data (we can use historical data to approximate the distribution), it may not be applicable in general. For example, consider a sensor deployment application where the sensors are newly manufactured, and information about their failure rate is unavailable. Such settings pose a dual problem- we need to *simultaneously learn the input distributions while designing good algorithms for the problem.* This poses two crucial challenges: (i) what sort of feedback should we assume? and (ii) how do we measure the performance of our learning algorithms? Consider the following special case of diagnosing complex systems: we have a complex system comprising many components, each of which fails with some *known* probability. We say that the system is working if there are no failures, else system maintenance needs to be undertaken. In this case, the simple non-adaptive policy that tests components in decreasing order of failure probabilities is optimal (assuming uniform testing costs). Now, suppose that the failure probabilities are unknown. It is unclear how to approach such a problem: a crucial challenge arises from the fact that *any suggested policy is followed only until a failure is observed.* A "*desirable*" solution for this problem should have *low-regret* with respect to the optimal non-adaptive policy when failure probabilities are known. Towards this end, an open direction is to develop a framework to incorporate learning using bandit feedback and design a general strategy to obtain low-regret algorithms for such stochastic optimization problems. Some recent work in this area is [46, 50] but much more remains open.

Bibliography

[1] M. Adler and B. Heeringa. Approximating optimal binary decision trees. *Algorithmica*, 62(3-4):1112–1121, 2012.

[2] A. Agarwal, S. Assadi, and S. Khanna. Stochastic submodular cover with limited adaptivity. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*, page 323–342, 2019.

[3] A. Agarwal, R. Ghuge, and V. Nagarajan. An asymptotically optimal batched algorithm for the dueling bandit problem. In *Proceedings of the 36th Annual Conference on Neural Information Processing Systems*, 2022.

[4] A. Agarwal, R. Ghuge, and V. Nagarajan. Batched dueling bandits. In *Proceedings of the 39th International Conference on Machine Learning*, pages 89–110, 2022.

[5] A. Agarwal, N. Johnson, and S. Agarwal. Choice bandits. In *NeurIPS*, 2020.

[6] N. Ailon, Z. Karnin, and T. Joachims. Reducing Dueling Bandits to Cardinal Bandits. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

[7] S. Allen, L. Hellerstein, D. Kletenik, and T. Ünlüyurt. Evaluation of monotone dnf formulas. *Algorithmica*, 77, 11 2015.

[8] A. Asadpour and H. Nazerzadeh. Maximizing stochastic monotone submodular functions. *Manag. Sci.*, 62(8):2374–2391, 2016.

[9] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 05 2002.

[10] A. T. Azar and S. M. El-Metwally. Decision tree classifiers for automated medical diagnosis. *Neural Computing and Applications*, 23:2387–2403, 2013.

[11] E. Balkanski, A. Breuer, and Y. Singer. Non-monotone submodular maximization in exponentially fewer iterations. In *Advances in Neural Information Processing Systems*, pages 2359–2370, 2018.

[12] E. Balkanski, A. Rubinstein, and Y. Singer. An exponential speedup in parallel running time for submodular maximization without loss in approximation. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 283–302, 2019.

[13] E. Balkanski and Y. Singer. The adaptive complexity of maximizing a submodular function. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1138–1151, 2018.

[14] E. Balkanski and Y. Singer. Approximation guarantees for adaptive sampling. In *Proceedings of the 35th International Conference on Machine Learning*, pages 393–402, 2018.

[15] N. Bansal, A. Gupta, J. Li, J. Mestre, V. Nagarajan, and A. Rudra. When LP is the cure for your matching woes: Improved bounds for stochastic matchings. *Algorithmica*, 63(4):733–762, 2012.

[16] N. Bansal and V. Nagarajan. On the adaptivity gap of stochastic orienteering. *Mathematical Programming*, 154(1-2):145–172, 2015.

[17] O. Barinova, V. Lempitsky, and P. Kholi. On detection of multiple object instances using hough transforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1773–1784, 2012.

[18] M. Bateni, H. Esfandiari, and V. S. Mirrokni. Optimal distributed submodular optimization via sketching. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1138–1147, 2018.

[19] S. Behnezhad, M. Derakhshan, and M. Hajiaghayi. Stochastic matching with few queries: (1-ϵ) approximation. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1111–1124, 2020.

[20] G. Bellala, S. Bhavnani, and C. Scott. Active diagnosis under persistent noise with unknown noise distribution: A rank-based approach. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, volume 15, pages 155–163, 2011.

[21] G. Bellala, S. K. Bhavnani, and C. Scott. Group-based active query selection for rapid diagnosis in time-critical situations. *IEEE Transactions on Information Theory*, 58(1):459–478, 2012.

[22] V. Bengs, R. Busa-Fekete, A. E. Mesaoudi-Paul, and E. Hüllermeier. Preference-based online learning with dueling bandits: A survey. *Journal of Machine Learning Research*, 22(7):1–108, 2021.

[23] A. Bhalgat, A. Goel, and S. Khanna. Improved approximation results for stochastic knapsack problems. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1647–1665, 2011.

[24] S. K. Bhavnani, A. Abraham, C. Demeniuk, M. Gebrekristos, A. Gong, S. Nainwal, G. K. Vallabha, and R. J. Richardson. Network analysis of toxic chemicals and symptoms: implications for designing first-responder systems. *AMIA Annual Symposium Proceedings*, pages 51–55, 2007.

[25] G. Blanc, J. Lange, and L.-Y. Tan. Query strategies for priced information, revisited. In *Proceedings of the Thirty-second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2021)*, page 1638–1650, 2021.

[26] D. Bradac, S. Singla, and G. Zuzic. (Near) Optimal Adaptivity Gaps for Stochastic Multi-Value Probing. In *Approximation, Randomization, and Combinatorial Optimization*, volume 145, pages 49:1–49:21, 2019.

[27] V. T. Chakaravarthy, V. Pandit, S. Roy, P. Awasthi, and M. K. Mohania. Decision trees for entity identification: Approximation algorithms and hardness results. *ACM Transactions on Algorithms*, 7(2):15, 2011.

[28] O. Chapelle and Y. Chang. Yahoo! learning to rank challenge overview. In *Proceedings of the 2010 International Conference on Yahoo! Learning to Rank Challenge - Volume 14*, page 1–24. JMLR.org, 2010.

[29] M. Charikar, R. Fagin, V. Guruswami, J. Kleinberg, P. Raghavan, and A. Sahai. Query strategies for priced information (extended abstract). pages 582–591, 01 2000.

[30] C. Chekuri and K. Quanrud. Parallelizing greedy for submodular set function maximization in matroids and beyond. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 78–89, 2019.

[31] B. Chen and P. I. Frazier. Dueling Bandits with Weak Regret. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[32] Y. Chen, H. Shio, C. A. F. Montesinos, L. P. Koh, S. Wich, and A. Krause. Active detection via adaptive submodularity. In *Proceedings of the 31st International Conference on Machine Learning*, page I–55–I–63, 2014.

[33] F. Cicalese and E. S. Laber. A new strategy for querying priced information. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '05, page 674–683, 2005.

[34] F. Cicalese and E. S. Laber. On the competitive ratio of evaluating priced functions. *J. ACM*, 58(3), June 2011.

[35] F. Cicalese, E. S. Laber, and A. M. Saettler. Diagnosis determination: decision trees optimizing simultaneously worst and expected testing cost. In *Proceedings of the 31th International Conference on Machine Learning*, pages 414–422, 2014.

[36] T. M. Cover and J. A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing).* Wiley-Interscience, USA, 2006.

[37] S. Dasgupta. Analysis of a greedy active learning strategy. In *Advances in Neural Information Processing Systems*, pages 337–344, 2004.

[38] B. C. Dean, M. X. Goemans, and J. Vondrák. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Mathematics of Operations Research*, 33(4):945–964, 2008.

[39] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, page 588–599, 2004.

[40] A. Deshpande, L. Hellerstein, and D. Kletenik. Approximation algorithms for stochastic submodular set cover with applications to boolean function evaluation and min-knapsack. *ACM Trans. Algorithms*, 12(3), Apr. 2016.

[41] M. Dudik, K. Hofmann, R. E. Schapire, A. Slivkins, and M. Zoghi. Contextual Dueling Bandits. In *Proceedings of the 28th Conference on Learning Theory*, 2015.

[42] A. Ene, V. Nagarajan, and R. Saket. Approximation algorithms for stochastic k-tsp. In *37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 27:27–27:14, 2017.

[43] H. Esfandiari, A. Karbasi, A. Mehrabian, and V. Mirrokni. Regret bounds for batched bandits. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):7340–7348, 2021.

[44] H. Esfandiari, A. Karbasi, and V. Mirrokni. Adaptivity in adaptive submodularity. In *Proceedings of 34th Conference on Learning Theory*, volume 134, pages 1823–1846. PMLR, 2021.

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

[46] V. Gabillon, B. Kveton, Z. Wen, B. Eriksson, and S. Muthukrishnan. Adaptive submodular maximization in bandit setting. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[47] Z. Gao, Y. Han, Z. Ren, and Z. Zhou. Batched multi-armed bandits problem. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[48] Z. Gao, Y. Han, Z. Ren, and Z. Zhou. Batched multi-armed bandits problem. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 501–511, 2019.

[49] M. Garey and R. Graham. Performance bounds on the splitting algorithm for binary testing. *Acta Informatica*, 3:347–355, 1974.

[50] E. Gergatsouli and C. Tzamos. Online learning for min sum set cover and pandora's box. In *Proceedings of the 39th International Conference on Machine Learning*, pages 7382–7403, 2022.

[51] R. Ghuge, A. Gupta, and V. Nagarajan. The power of adaptivity for stochastic submodular cover. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3702–3712. PMLR, 6 2021.

[52] R. Ghuge, A. Gupta, and V. Nagarajan. Non-adaptive stochastic score classification and explainable halfspace evaluation. In K. Aardal and L. Sanità, editors, *Integer Programming and Combinatorial Optimization - 23rd International Conference*, pages 277–290. Springer, 2022.

[53] D. Gkenosis, N. Grammel, L. Hellerstein, and D. Kletenik. The Stochastic Score Classification Problem. In *26th Annual European Symposium on Algorithms*, volume 112, pages 36:1–36:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018.

[54] M. Goemans and J. Vondrák. Stochastic covering and adaptivity. In *LATIN 2006: Theoretical Informatics*, pages 532–543. Springer Berlin Heidelberg, 2006.

[55] D. Golovin and A. Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *J. Artif. Intell. Res. (JAIR)*, 42:427–486, 2011.

[56] D. Golovin and A. Krause. Adaptive submodularity: A new approach to active learning and stochastic optimization. *CoRR*, abs/1003.3967, 2017.

[57] H. González-Banos. A randomized art-gallery algorithm for sensor placement. In *Proceedings of the Seventeenth Annual Symposium on Computational Geometry*, page 232–240, 2001.

[58] N. Grammel, L. Hellerstein, D. Kletenik, and P. Lin. Scenario submodular cover. In *International Workshop on Approximation and Online Algorithms*, pages 116–128. Springer, 2016.

[59] S. Guha and K. Munagala. Multi-armed bandits with metric switching costs. In *Automata, Languages and Programming, 36th Internatilonal Colloquium (ICALP)*, pages 496–507, 2009.

[60] A. Guillory and J. A. Bilmes. Average-case active learning with costs. In *Algorithmic Learning Theory, 20th International Conference, ALT 2009, Porto, Portugal, October 3-5, 2009. Proceedings*, pages 141–155. Springer, 2009.

[61] A. Gupta, R. Krishnaswamy, V. Nagarajan, and R. Ravi. Running errands in time: Approximation algorithms for stochastic orienteering. *Math. Oper. Res.*, 40(1):56–79, 2015.

[62] A. Gupta and V. Nagarajan. A stochastic probing problem with applications. In *Integer Programming and Combinatorial Optimization - 16th International Conference*, pages 205–216, 2013.

[63] A. Gupta, V. Nagarajan, and R. Ravi. Approximation algorithms for optimal decision trees and adaptive TSP problems. *Math. Oper. Res.*, 42(3):876–896, 2017.

[64] A. Gupta, V. Nagarajan, and S. Singla. Adaptivity gaps for stochastic probing: Submodular and XOS functions. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1688–1702, 2017.

[65] K. Hofmann, S. Whiteson, and M. Rijke. Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval. *Inf. Retr.*, 16(1):63–90, 2 2013.

[66] L. Hyafil and R. L. Rivest. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5(1):15–17, 1976.

[67] S. Im, V. Nagarajan, and R. van der Zwaan. Minimum latency submodular cover. *ACM Trans. Algorithms*, 13(1):13:1–13:28, 2016.

[68] K. Jamieson, S. Katariya, A. Deshpande, and R. Nowak. Sparse Dueling Bandits. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, 2015.

[69] S. Javdani, Y. Chen, A. Karbasi, A. Krause, D. Bagnell, and S. S. Srinivasa. Near optimal bayesian active learning for decision making. In *AISTATS*, pages 430–438, 2014.

[70] H. Jiang, J. Li, D. Liu, and S. Singla. Algorithms and Adaptivity Gaps for Stochastic k-TSP. In *11th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 151, pages 45:1–45:25, 2020.

[71] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, page 133–142, New York, NY, USA, 2002. Association for Computing Machinery.

[72] T. Kamishima. Nantonac collaborative filtering: recommendation based on order responses. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*, pages 583–588, 2003.

[73] H. Kaplan, E. Kushilevitz, and Y. Mansour. Learning with attribute costs. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '05, page 356–365, 2005.

[74] A. Karbasi, V. S. Mirrokni, and M. Shadravan. Parallelizing thompson sampling. *CoRR*, abs/2106.01420, 2021.

[75] D. Kempe, J. M. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. *Theory of Computing*, 11:105–147, 2015.

[76] J. Komiyama, J. Honda, H. Kashima, and H. Nakagawa. Regret Lower Bound and Optimal Algorithm in Dueling Bandit Problem. In *Proceedings of the 28th Conference on Learning Theory*, 2015.

[77] J. Komiyama, J. Honda, and H. Nakagawa. Copeland Dueling Bandit Problem: Regret Lower Bound, Optimal Algorithm, and Computationally Efficient Algorithm. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

[78] S. R. Kosaraju, T. M. Przytycka, and R. S. Borgstrom. On an Optimal Split Tree Problem. In *Proceedings of the 6th International Workshop on Algorithms and Data Structures*, pages 157–168, 1999.

[79] A. Krause and C. Guestrin. Near-optimal nonmyopic value of information in graphical models. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, page 324–331, 2005.

[80] A. Krause and C. Guestrin. Near-optimal observation selection using submodular functions. In *Proceedings of the 22nd National Conference on Artificial Intelligence - Volume 2*, page 1650–1654. AAAI Press, 2007.

[81] C. Li, I. Markov, M. de Rijke, and M. Zoghi. Mergedts: A method for effective large-scale online ranker evaluation. *ACM Trans. Inf. Syst.*, 38(4):40:1–40:28, 2020.

[82] Z. W. Lim, D. Hsu, and W. S. Lee. Adaptive stochastic optimization: From sets to paths. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[83] H. Lin and J. Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, page 510–520, 2011.

[84] T.-Y. Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, 3 2009.

[85] Z. Liu, S. Parthasarathy, A. Ranganathan, and H. Yang. Near-optimal algorithms for shared filter evaluation in data stream systems. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 133–146, 2008.

[86] D. W. Loveland. Performance bounds for binary testing with arbitrary weights. *Acta Inform.*, 22(1):101–114, 1985.

[87] S. Mini, S. K. Udgata, and S. L. Sabat. Sensor deployment and scheduling for target coverage problem in wireless sensor networks. *IEEE Sensors Journal*, 14(3):636–644, 2014.

[88] B. Mirzasoleiman, A. Karbasi, A. Badanidiyuru, and A. Krause. Distributed submodular cover: Succinctly summarizing massive data. In *Advances in Neural Information Processing Systems*, pages 2881–2889, 2015.

[89] B. M. E. Moret. Decision trees and diagrams. *ACM Comput. Surv.*, 14(4):593—623, 1982.

[90] K. Munagala, U. Srivastava, and J. Widom. Optimization of continuous queries with shared expensive filters. In *27th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 215–224, 2007.

[91] S. K. Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, 2:345–389, 1997.

[92] F. Navidi, P. Kambadur, and V. Nagarajan. Adaptive submodular ranking and routing. *Oper. Res.*, 68(3):856–877, 2020.

[93] V. Perchet, P. Rigollet, S. Chassang, and E. Snowberg. Batched bandit problems. *The Annals of Statistics*, 44(2):660–681, 2016.

[94] T. Qin and T.-Y. Liu. Introducing letor 4.0 datasets. *ArXiv*, abs/1306.2597, 2013.

[95] G. Radanovic, A. Singla, A. Krause, and B. Faltings. Information gathering with peers: Submodular optimization with peer-prediction constraints. In *Proc. Conference on Artificial Intelligence (AAAI)*, 2 2018.

[96] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, page 43–52, New York, NY, USA, 2008. Association for Computing Machinery.

[97] S. Ramamohan, A. Rajkumar, and S. Agarwal. Dueling Bandits : Beyond Condorcet Winners to General Tournament Solutions. In *Advances in Neural Information Processing Systems 29*, 2016.

[98] R. A. Rossi and N. K. Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 2015.

[99] A. Saha and P. Gaillard. Versatile dueling bandits: Best-of-both world analyses for learning from relative preferences. In *International Conference on Machine Learning*, pages 19011–19026. PMLR, 2022.

[100] A. Saha and A. Gopalan. Combinatorial bandits with relative feedback. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 983–993, 2019.

[101] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[102] I. Simon, N. Snavely, and S. M. Seitz. Scene summarization for online image collections. In *11th International Conference on Computer Vision*, pages 1–8, 2007.

[103] R. Sipos, A. Swaminathan, P. Shivaswamy, and T. Joachims. Temporal corpus summarization using submodular word coverage. In *Proceedings of the 21st ACM Inter-*

national Conference on Information and Knowledge Management, CIKM '12, page 754–763, 2012.

[104] Y. Sui, V. Zhuang, J. W. Burdick, and Y. Yue. Multi-dueling Bandits with Dependent Arms. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence*, 2017.

[105] Y. Sui, M. Zoghi, K. Hofmann, and Y. Yue. Advancements in dueling bandits. In J. Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 5502–5510. ijcai.org, 2018.

[106] C. Sun, V. O. K. Li, J. C. K. Lam, and I. Leslie. Optimal citizen-centric sensor placement for air quality monitoring: A case study of city of cambridge, the united kingdom. *IEEE Access*, 7:47390–47400, 2019.

[107] T. Ünlüyurt. Sequential testing of complex systems: a review. *Discrete Applied Mathematics*, 142(1):189–205, 2004.

[108] T. Urvoy, F. Clerot, R. Feraud, and S. Naamane. Generic Exploration and K-armed Voting Bandits. In *Proceedings of the 30th International Conference on Machine Learning*, 2013.

[109] C. Wirth, R. Akrour, G. Neumann, and J. Fürnkranz. A survey of preference-based reinforcement learning methods. *J. Mach. Learn. Res.*, 18(1):4945–4990, 1 2017.

[110] L. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.

[111] H. Wu and X. Liu. Double thompson sampling for dueling bandits. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 649–657, 2016.

[112] Y. Yue, J. Broder, R. Kleinberg, and T. Joachims. The k-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538–1556, 2012. JCSS Special Issue: Cloud Computing 2011.

[113] Y. Yue and T. Joachims. Interactively optimizing information retrieval systems as a dueling bandits problem. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML ’09, page 1201–1208, New York, NY, USA, 2009. Association for Computing Machinery.

[114] Y. Yue and T. Joachims. Beat the mean bandit. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

[115] M. Zoghi, Z. Karnin, S. Whiteson, and M. de Rijke. Copeland Dueling Bandits. In *Advances in Neural Information Processing Systems 28*, 2015.

[116] M. Zoghi, S. Whiteson, and M. de Rijke. MergeRUCB: A method for large-scale online ranker evaluation. In *Proceedings of the 8th ACM International Conference on Web Search and Data Mining*, 2015.

[117] M. Zoghi, S. Whiteson, R. Munos, and M. de Rijke. Relative Upper Confidence Bound for the K-Armed Dueling Bandit Problem. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.