Adaptive Approximation Algorithms for Ranking, Routing and Classification

by

Fatemeh Navidi

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
2020

Doctoral Committee:

Assistant Professor Viswanath Nagarajan, Chair
Professor Marina Epelman
Professor Jon Lee
Professor Seth Pettie
Associate Professor Cong Shi

Fatemeh Navidi
navidi@umich.edu
ORCiD: 0000-0002-3157-9460

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor, Professor Viswanath Nagarajan, who guided me through this journey and was always ready to help whenever I had a problem in my research. He taught me to be more precise and organized and put so much time and dedication into advising me. I would also like to thank all of my collaborators for the amazing work experience we had together, and my committee members, without whose support this would not be possible.

I am very grateful for all the people I met at the University of Michigan, especially in Industrial and Operations Engineering Department. I was part of an amazing cohort, who soon were my friends more than classmates. I learnt many new things from every interaction I had with every faculty member. I want to specifically thank Professor Brian Denton, our amazing department chair, who is very supportive and has done so many brilliant things for the department. Also, I wish to thank Tina Picano Sroka and Rod Capps because of their endless kindness and always going the extra mile to help everyone. We are all very lucky to have such great staff members in the department.

I am definitely indebted to my family who supported me in every possible way and always prioritized what is best for me. It was not easy for them to deal with 5 years of not seeing each other, but they never made me feel bad about my decision. They tried their best to make me feel involved in their life, so I could be more comfortable about the distance between us and focus on my studies.

Last but not least, I would like to show my sincere appreciation towards my amazing

friends who were always there for me and made this a wonderful part of my life. I am so lucky to have the chance of knowing these fabulous people and I would not be the same person if it were not for their help and all the things they taught me. I wish I could talk about every single one of them here, but that would require much more space than my whole thesis. So I keep that in my heart, where it belongs.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

ABSTRACT

This dissertation aims to consider different problems in the area of stochastic optimization, where we are provided with more information about the instantiation of the stochastic parameters over time. With uncertainty being an inseparable part of every industry, several applications can be modeled as discussed. In this dissertation we focus on three main areas of applications: 1) ranking problems, which can be helpful for modeling product ranking, designing recommender systems, etc., 2) routing problems which can cover applications in delivery, transportation and networking, and 3) classification problems with possible applications in medical diagnosis and chemical identification. We consider three types of solutions for these problems based on how we want to deal with the observed information: static, adaptive and *a priori* solutions. In Chapter II, we study two general stochastic submodular optimization problems that we call Adaptive Submodular Ranking and Adaptive Submodular Routing. In the ranking version, we want to provide an adaptive sequence of weighted elements to cover a random submodular function with minimum expected cost. In the routing version, we want to provide an adaptive path of vertices to cover a random scenario with minimum expected length. We provide (poly)logarithmic approximation algorithms for these problems that (nearly) match or improve the best-known results for various special cases. We also implemented different variations of the ranking algorithm and observed that it outperforms other practical algorithms on real-world and synthetic data sets. In Chapter III, we consider the Optimal Decision Tree problem: an identification task that is widely used in active learning. We study this problem in presence of noise, where we want to perform

a sequence of tests with possible noisy outcomes to identify a random hypothesis. We give different static (non-adaptive) and adaptive algorithms for this task with almost logarithmic approximation ratios. We also implemented our algorithms on real-world and synthetic data sets and compared our results with an information theoretic lower bound to show that in practice, our algorithms' value is very close to this lower bound. In Chapter IV, we focus on a stochastic vehicle routing problem called *a priori* traveling repairman, where we are given a metric and probabilities of each vertices being active. We want to provide an *a priori* master tour originating from the root that is shortcut later over the observed active vertices. Our objective is to minimize the expected total wait time of active vertices, where the wait time of a vertex is defined as the length of the path from the root to this vertex. We consider two benchmarks to evaluate the performance of an algorithm for this problem: optimal *a priori* solution and the re-optimization solution. We provide two algorithms to compare with each of these benchmarks that have constant and logarithmic approximation ratios respectively.

CHAPTER I

Introduction

Traditional optimization models assume full information on the instances being solved, which is unrealistic in many situations. In order to remedy this limitation, there has been significant work in the area of optimization under uncertainty, which deals with various ways to model uncertain input. One popular way to handle uncertainty is by using stochastic models. In stochastic models we assume that the data is random with a specific distribution, based on which we can optimize an expected objective function. To design algorithms for such a problem, we need to specify our decisions under all possible outcomes of the data; therefore, computing an optimal stochastic policy is usually much harder than optimizing a deterministic problem. There are many examples of polynomial time deterministic algorithms that do not extend to stochastic settings. As the number of possible outcomes in a stochastic problem is often exponential in the input size, the time/space complexity of an algorithm that finds the optimal solution is often exponential as well. In many applications, we need much faster running times but we are willing to sacrifice optimality up to a point. This is the main idea behind *approximation algorithms*. An approximation algorithm provides a near optimal solution with time complexity that is a polynomial in the input size. For example, in case of the 0/1 maximization knapsack problem, there is a Fully Polynomial Time Approximation Scheme in the deterministic setting [61], and there is a constant factor

approximation algorithm in the stochastic setting [28]. Moreover, we want to make sure that the solution of an approximation algorithm is still close to the optimal solution. So, to evaluate its performance, we are interested in proving that the ratio of the algorithm's objective to the optimal value, which is called the approximation ratio of the algorithm, is not very large (small) for minimization (maximization) problems. For stochastic problems, we are interested in bounding the ratio of *expected* objectives. For instance, in [28], authors prove that the expected cost of their algorithm is at least a constant factor of the expected optimal cost for stochastic knapsack.

Typically, when dealing with uncertainty, the unknown data is revealed gradually over time. There are three main approaches for such problems. First, is to have a static (non-adaptive) algorithm that offers the same solution regardless of the additional information we receive. Second, we can design adaptive algorithms that can make decisions based on incremental information. Finally, we can offer an offline master solution which can be modified online (with very small overhead) in order to adjust to the additional data; this approach is known as *a priori* optimization.

In some applications, the run-time of the algorithm might be far more important than its solution value. In those cases, we might prefer to save time by generating the same sequence of actions regardless of the observed information, rather than modifying the solution. This corresponds to a static (non-adaptive) algorithm. For instance, in emergency rooms doctors usually perform a fixed sequence of tests rather than waiting to see each test result due to the importance of time in patients' status. Medical diagnosis is an application of our work in Chapter III and we present some non-adaptive algorithms there.

On the other hand, in some applications the algorithm's run-time is of secondary importance in comparison with the algorithm's solution value: here adaptive algorithms are preferred. For instance, in product ranking our focus is usually on maximizing the revenue

and we are less sensitive about the time. It turns out that in many cases, there is a huge gap between adaptive and non-adaptive algorithms. For example, in [37] authors show that there is polynomial "adaptivity gap" for stochastic set cover, which justifies why we prefer to have the extra overhead in real time. In Chapters II and III we focus on such adaptive policies.

A priori optimization ([15]) is especially beneficial when we need to solve instances of the same problem repeatedly. The small online time and space complexity in modifying the master solution for each instance is one of the main reasons that makes this method desirable. Another reason is that the master solution will give us some information about the modified solution for each instance that might be useful for management and planning. On the other hand, the objective of a master solution is usually costlier than a solution produced by a fully adaptive algorithm; therefore, deciding about the approach we want to use should come from a careful consideration of this trade-off. In Chapter IV we consider *a priori* solutions.

1.1 Summary of Contributions

In Chapter II, we discuss a general stochastic ranking problem where an algorithm needs to adaptively select a sequence of elements so as to "cover" a random scenario (drawn from a known distribution) at minimum expected cost. The coverage of each scenario is captured by an individual submodular function, where the scenario is said to be covered when its function value goes above a given threshold. We obtain a logarithmic factor approximation algorithm for this adaptive ranking problem, which is the best possible (unless $P = NP$). This problem unifies and generalizes many previously studied problems with applications in search ranking and active learning. The approximation ratio of our algorithm either matches or improves the best result known in each of these special cases. Furthermore, we extend our results to an

adaptive vehicle routing problem, where costs are determined by an underlying metric. This routing problem is a significant generalization of the previously-studied adaptive traveling salesman and traveling repairman problems. Our approximation ratio nearly matches the best bound known for these special cases. Finally, we present experimental results for some applications of adaptive ranking.

In Chapter III, we focus on a fundamental task in active learning, which involves performing a sequence of tests to identify an unknown hypothesis that is drawn from a known distribution. This problem, known as optimal decision tree induction, has been widely studied for decades and the asymptotically best-possible approximation algorithm has been devised for it. We study a generalization where certain test outcomes are noisy, even in the more general case when the noise is persistent, i.e., repeating a test gives the same noisy output, disallowing simple repetition as a way to gain confidence. We design new approximation algorithms for both the non-adaptive setting, where the test sequence must be fixed *a-priori*, and the adaptive setting where the test sequence depends on the outcomes of prior tests. Previous work in the area assumed at most a logarithmic number of noisy outcomes per hypothesis and provided approximation ratios that depended on parameters such as the minimum probability of a hypothesis. Our new approximation algorithms provide guarantees that are nearly best-possible and work for the general case of a large number of noisy outcomes per test or per hypothesis where the performance degrades smoothly with this number. Our results adapt and generalize methods used for submodular ranking and stochastic set cover. We evaluate the performance of our algorithms on two natural applications with noise: toxic chemical identification and active learning of linear classifiers. Despite our theoretical logarithmic approximation guarantees, our methods give solutions with cost very close to the information theoretic minimum, demonstrating the effectiveness of our methods.

In Chapter IV, we discuss the *a priori* traveling repairman problem, which is a stochastic

version of the classic traveling repairman problem (also called the traveling deliveryman or minimum latency problem). Given a metric (V, d) with a root $r \in V$, the traveling repairman problem (TRP) involves finding a tour originating from r that minimizes the sum of arrival-times at all vertices. In its *a priori* version, we are also given independent probabilities of each vertex being active. We want to find a master tour τ originating from r and visiting all vertices. The objective is to minimize the expected sum of arrival-times at all active vertices, when τ is shortcut over the inactive vertices. We obtain the first constant-factor approximation algorithm for *a priori* TRP under non-uniform probabilities. Previously, such a result was only known for uniform probabilities. We also present a $\mathcal{O}(\log n)$- approximation algorithm with respect to the re-optimization solution (which has access to full information), where n is the number of vertices. We believe that this is the first result for *a priori* TRP with respect to the re-optimization solution in any setting.

CHAPTER II

Adaptive Submodular Ranking and Routing

The results in this chapter appear in [60] and [71].

2.1 Introduction

Many stochastic optimization problems can be viewed as sequential decision processes of the following form. There is a known distribution $\mathcal{D}$ over a set of scenarios, and the goal is to cover the unknown realized scenario i^* drawn from $\mathcal{D}$. In each step, an algorithm chooses an element which partially covers i^* and receives some feedback from that element. This feedback is then used to update the distribution over scenarios (using conditional probabilities). So any solution in this setting is an adaptive sequence of elements. The objective is to minimize the expected cost incurred to cover the realized scenario i^*.

Furthermore, many different criteria to cover a scenario can be modeled as covering a suitable submodular function. Submodular functions are widely used in many domains, e.g. game theory, social networks, search ranking and document summarization; see [78, 62, 74, 65].

As an example of the class of problems that we address, consider a medical diagnosis application. There is a patient with an unknown disease and there are several possible tests that can be performed. Each test has a certain cost and its outcome (feedback) can be used to restrict the set of possible diseases. There are also *a priori* probabilities associated with

each disease. The task here is to obtain an adaptive sequence of tests so as to identify the disease at minimum expected cost.

As another example, consider a search engine application. On any query, different user-types are often interested in viewing different search results. Each user-type is associated with the set of results they are interested in and a threshold number of results they would like to see. There is also a probability distribution over user-types. After displaying each result (or a block of small number of results), the search engine receives feedback on which of those results were of interest to the realized user-type. The goal is to provide an adaptive sequence of results so as to minimize the expected number of results until the user-type is satisfied.

Yet another example arises in route planning for disaster management. After a major disaster such as an earthquake, normal communication networks are usually unavailable. So rescue operators would not know the precise locations of victims before actually visiting them. However, probabilistic information is often available based on geographical data etc. Then the task is to plan an adaptive route for a rescue vehicle that visits all the victims within minimum expected time.

In this chapter, we study an abstract stochastic optimization problem in the setting described above which unifies and generalizes many previously-studied problems such as *optimal decision trees* studied in [53], [63], [27], [20], [49] and [26], *equivalence class determination* (see [40] and [12]), *decision region determination* studied in [58] and *submodular ranking* studied in [6] and [55]. We obtain an algorithm with the best-possible approximation guarantee in all these special cases. We also obtain the first approximation algorithms for some other natural problems that are captured by our framework, such as stochastic versions of knapsack cover and matroid basis with correlated distributions. Moreover, our algorithm is very simple to state and implement. We also present experimental results on the optimal

decision tree problem and our algorithm performs very well.

We extend our framework to a vehicle-routing setting as well, where the elements are located in a metric and the cost corresponds to travel distance/time between these locations. As special cases, we recover the adaptive traveling salesman and repairman problems that were studied in [49]. Our approximation ratio almost matches the best result known for these special cases. Our approach has the advantage of being able to solve a more general problem while allowing for a simpler analysis. We note that submodular objectives are also commonly utilized in vehicle routing problems, see [23] and references therein for theoretical work and [80] for applications in information acquisition and robotics.

For some stochastic optimization problems, one can come up with approximately optimal solutions using static (non-adaptive) solutions that are insensitive to the feedback obtained, see e.g. stochastic (maximization) knapsack in [28] and stochastic matching in [10]. However, this is not the case for the adaptive submodular ranking problem. For all the special cases mentioned above, there are instances where the optimal adaptive value is much less than the optimal non-adaptive value. Thus, it is important to come up with an adaptive algorithm.

2.1.1 Adaptive Submodular Ranking

We start with some basics. A set function $f : 2^U \to \mathbb{R}_+$ on ground set U is said to be submodular if $f(A) + f(B) \geq f(A \cap B) + f(A \cup B)$ for all $A, B \subseteq U$. The function f is said to be monotone if $f(A) \leq f(B)$ for all $A \subseteq B \subseteq U$. We assume that set functions are given in the standard *value oracle* model, i.e. we can evaluate $f(S)$ for any $S \subseteq U$ in polynomial time.

In the adaptive submodular ranking problem (ASR) we have a ground set U of n *elements* with positive costs $\{c_e\}_{e \in U}$. We also have m *scenarios* with a probability distribution $\mathcal{D}$ given by probabilities $\{p_i\}_{i=1}^m$ totaling to one. Each scenario $i \in [m] := \{1, \cdots, m\}$ is specified by:

(i) a *monotone submodular* function $f_i : 2^U \to [0, 1]$ where $f_i(\emptyset) = 0$ and $f_i(U) = 1$ (any

monotone submodular function can be expressed in this form by scaling), and

(ii) a *feedback* function $r_i : U \to G$ where G is a set of possible feedback values.

We note that f_i and r_i need not be related in any way: this flexibility allows us to capture many different applications. Scenario $i \in [m]$ is said to be *covered* by any subset $S \subseteq U$ of elements such that $f_i(S) = 1$. The goal in ASR is to adaptively find a sequence of elements in U that minimizes the expected cost to cover a *random scenario* i^* drawn from $\mathcal{D}$. The identity of i^* is initially unknown to the algorithm. When the algorithm selects an element $e \in U$, it receives some feedback value $g = r_{i^*}(e) \in G$ which can be used to update the probability distribution of i^* using conditional probabilities. In particular, the probability of any scenario $i \in [m]$ with $r_i(e) \neq g$ would become zero. The sequence of selected elements is *adaptive* because it depends on the feedback received.

Example: Figure 2.1 demonstrates an example for ASR. In this example we have elements $U = \{e_1, e_2, e_3, e_4, e_5, e_6\}$ and 3 scenarios. Each element has cost 1 and there is a uniform probability distribution over scenarios. Each senario $i \in \{1, 2, 3\}$ is associated with a subset S_i with submodular function $f_i(S) = \frac{|S \cap S_i|}{|S_i|}$ and binary feedback function $r_i(e) = \mathbb{1}[e \in S_i]$. So the realized scenario i^* will be covered with subset $S \subseteq U$ if and only if $S_{i^*} \subseteq S$. And, the feedback from an element e is one if and only if $e \in S_{i^*}$. The decision tree in Figure 2.1 represents a feasible solution with expected cost $\frac{1}{3} \cdot 4 + \frac{1}{3} \cdot 3 + \frac{1}{3} \cdot 3 = \frac{10}{3}$.

A solution to ASR is represented by a decision tree $\mathcal{T}$, where each node is labeled by an element $e \in U$ and the branches out of such a node are labeled by the possible feedback we can receive after selecting e. Each node in $\mathcal{T}$ also corresponds to a *state* which is specified by the set E of previously selected elements and the feedback $\theta_e \in G$ of each $e \in E$. From this information, we can obtain a more abbreviated version of the state as (E, H) where H denotes the set of uncovered and compatible scenarios based on the observed feedback.

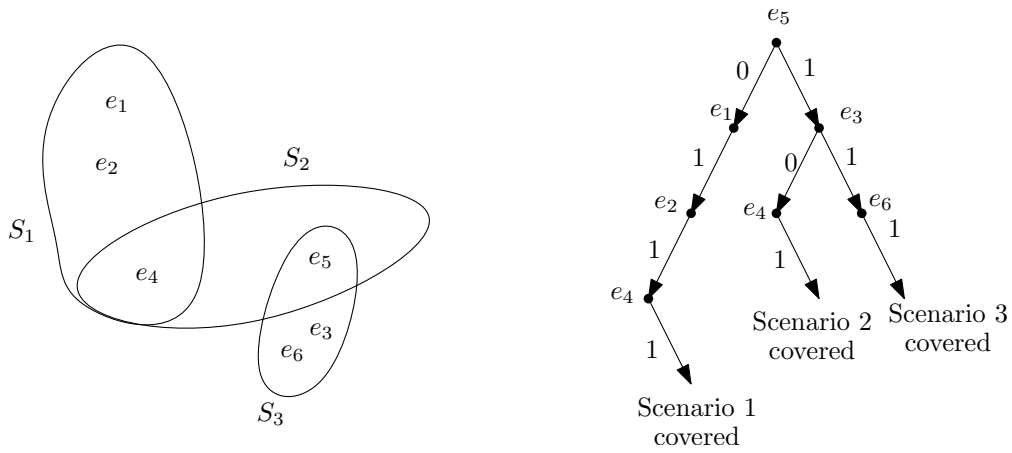

Figure 2.1: An example for ASR and a feasible solution

Formally,

$$H = \{i \in [m] \,:\, f_i(E) < 1,\ r_i(e) = \theta_e \text{ for all } e \in E\}$$

Every scenario $i \in [m]$ traces a root-leaf path in the decision tree $\mathcal{T}$ which at any node labeled by element $e \in U$, takes the branch labeled by feedback $r_i(e)$. Let T_i denote the sequence of elements on this path. In a feasible decision tree $\mathcal{T}$, each scenario $i \in [m]$ must be covered, i.e. $f_i(T_i) = 1$. The cost $C_{\mathcal{T}}(i)$ of $\mathcal{T}$ under scenario i is the total cost of the shortest prefix $\overline{T}_i$ of T_i such that $f_i(\overline{T}_i) = 1$. The objective in ASR is to minimize the expected cost $\sum_{i=1}^{m} p_i \cdot \left(\sum_{e \in \overline{T}_i} c_e\right)$. We emphasize that multiple scenarios may trace the same path in $\mathcal{T}$: in particular, it is *not* necessary to identify the realized scenario i^* in order to cover it.

We also note that cost is only incurred until the realized scenario i^* gets covered, even though the algorithm may not know this. In applications where scenarios correspond to users and the goal is to minimize cost incurred by the users, this is the natural definition. An example is the *multiple intent re-ranking* problem which models the search engine application (see Section 2.4.2). However, in some other applications (such as *optimal decision tree*), we are interested in algorithms that know exactly when to stop. For the applications that we consider, it turns out that this is still possible using the above definition: see Section 2.4 for details.

An important parameter in the analysis of our algorithm is the following:

$$\epsilon := \min_{\substack{e \in U : f_i(S \cup e) > f_i(S) \\ i \in [m],\, S \subseteq U}} f_i(S \cup e) - f_i(S). \tag{2.1}$$

It measures the minimum positive incremental value of any element. Such a parameter appears in all results on the submodular cover problem, eg. [87], [6].

2.1.2 Adaptive Submodular Routing

In the adaptive submodular routing problem (ASP) we have a ground set U of n *elements* which are located at vertices of a metric $(U \cup \{s\}, d)$, where s is a specified root vertex. Here $d : U \times U \to \mathbb{R}_+$ is a cost function that is symmetric (i.e. $d(x, y) = d(y, x)$ for all $x, y \in U$) and satisfies triangle inequality (i.e. $d(x, y) + d(y, z) \geq d(x, z)$ for all $x, y, z \in U$). We will use the terms element and vertex interchangeably. As before, we have m *scenarios* with a probability distribution $\mathcal{D}$ given by probabilities $\{p_i\}_{i=1}^m$ totaling to one, and each scenario $i \in [m]$ is associated with functions f_i and r_i. A feasible solution to ASP can again be represented by a decision tree $\mathcal{T}$, at the end of which each scenario is covered. Note that in the actual solution, we need to return to the root s after visiting the last vertex in $\mathcal{T}$. For any scenario i, let τ_i denote the root-leaf path traced in decision tree $\mathcal{T}$, and let π_i denote the shortest prefix of τ_i such that $f_i(\pi_i) = 1$. The cost $C_{\mathcal{T}}(i)$ of $\mathcal{T}$ under scenario i is the total cost of path π_i. Specifically, if $\pi_i = s, e_1, e_2, \cdots e_k$, the cost under scenario i would be $C_{\mathcal{T}}(i) = d(s, e_1) + \sum_{i=1}^{k-1} d(e_i, e_{i+1})$. The objective is to minimize the expected cost $\sum_{i=1}^m p_i \cdot C_{\mathcal{T}}(i)$. As with ASR, cost in ASP is only incurred until the realized scenario i^* is covered.

This problem differs from ASR only in the definition of the cost: here we want to minimize the expected metric-cost of the walk that covers i^*. Note also that ASP generalizes ASR (at the loss of a factor 2). To see this, for any ASR instance, consider the ASP instance on the metric $(U \cup \{s\}, d)$ induced by a star with center s and leaves U where $d(s, e) = c_e$ for all

$e \in U$.

2.1.3 Results

Our main result is an $O(\log \frac{1}{\epsilon} + \log m)$-approximation algorithm for adaptive submodular ranking (ASR) where $\epsilon > 0$ is as defined in (2.1) and m is the number of scenarios. Assuming $P \neq NP$, this result is asymptotically the best possible even when $m = 1$. This is because the set cover problem on k elements is a special case of ASR with $m = 1$ and parameter $\epsilon = 1/k$, and [30] showed that approximating set cover to within a $(1 - \epsilon) \ln k$ factor (for any $\epsilon > 0$) is NP-hard. Our algorithm is a simple adaptive greedy-style algorithm. At each step, we assign a score to each remaining element and select the element with maximum score. Such a simple algorithm was previously unknown even in the special case of optimal decision tree, despite a large number of papers on this topic, including [53], [63], [27], [1], [20], [44], [49], [38] and [26] .

For adaptive submodular routing (ASP) we provide an $O(\log^{2+\delta} n \cdot (\log \frac{1}{\epsilon} + \log m))$-approximation algorithm where $\delta > 0$ is any fixed constant and ϵ is as defined in (2.1). This algorithm utilizes some ideas from the algorithm for ASR, and involves combining a number of smaller tours into the final solution. We also make use of an algorithm for the (deterministic) submodular orienteering problem in a black-box fashion. Our result is nearly the best-possible because the group Steiner problem studied in [35] is a special case of ASP with $m = 1$ and parameter $\epsilon = 1/k$ where k denotes the number of groups. There is an $\Omega(\log^{2-\delta} n)$ factor hardness of approximation for group Steiner by [50] and the best approximation ratio known is $O(\log^2 n \cdot \log k)$ from [35].

We show that our framework is widely applicable by demonstrating a number of previously-studied stochastic optimization problems as special cases. In many cases, we match or improve upon prior approximation guarantees. We also obtain the first approximation algorithms for some other stochastic problems. More details on these applications can be found

in Section 2.4.

Finally, in Section 2.5 we provide computational results for the optimal decision tree problem (and its generalized version). We use a dataset arising in the identification of toxic chemicals based on binary symptoms. Our algorithm performs very well compared to some other natural algorithms.

Outline of key techniques. Our algorithm for ASR involves repeatedly selecting an element that maximizes a combination of (i) the expected increase in function value relative to the target of one, and (ii) a measure of gain in identifying the realized scenario. See Equation (2.2) for the formal selection criterion. Our analysis provides new ways of reasoning about adaptive decision trees. At a high level, our approach is similar to that for the minimum-latency TSP in [17] and [22]. We upper bound the probability that the algorithm incurs a certain power-of-two cost 2^k in terms of the probability that the optimal solution incurs cost $2^k/\alpha$, which is then used to establish an $O(\alpha)$ approximation ratio. Our main technical contribution is in relating these completion probabilities in the algorithm and the optimal solution (see Lemma II.3). In particular, a key step in our proof is a coupling of "bad" states in the algorithm (where the gain in terms of our selection criterion is small) with "bad" states in the optimum (where the cost incurred is high). This is reflected in the classification of the algorithm's states as good/ok/bad (Definition II.4) and the proof that the *expected* gain of the algorithm is large (Lemma II.5). Our algorithm and analysis for the adaptive routing problem (ASP) are along similar lines.

2.1.4 Related Works

The basic submodular cover problem (select a min-cost subset of elements that covers a given submodular function) was first considered by [87] who proved that the natural greedy algorithm is a $(1 + \ln \frac{1}{\epsilon})$-approximation algorithm. This result is tight because set cover is

a special case. The submodular cover problem corresponds to the special case of ASR with $m = 1$.

The deterministic submodular ranking problem was introduced by [6] who obtained an $O(\log \frac{1}{\epsilon})$-approximation algorithm when all costs are unit. This is a special case of ASR when there is no feedback (i.e. $G = \emptyset$) and costs are uniform; note that the optimal ASR solution in this case is just a fixed sequence of elements. The result in [6] was based on an interesting "reweighted" greedy algorithm: the second term in our selection criterion (2.2) is similar to this. A different proof of the submodular ranking result, using a min-latency type analysis, was obtained in [55] which also implied an $O(\log \frac{1}{\epsilon})$-approximation algorithm with non-uniform costs. We also use a min-latency type analysis for ASR.

The first $O(\log m)$-approximation algorithm for optimal decision tree was obtained in [49], which is known to be best-possible from [20]. This result was extended to the equivalence class determination problem in [26]. Previous results based on a simple greedy "splitting" algorithm, had a logarithmic dependence on either costs or probabilities which can be exponential in m; see [63], [27], [1], [20] and [44]. The algorithms in [49] and [26] were significantly more complex than what we obtain here as a special case of ASR. In particular these algorithms proceeded in $O(\log m)$ phases, each of which required solving an auxiliary subproblem that reduced the number of possible scenarios by a constant factor. Using such a "phase based" algorithm and analysis for the general ASR problem only leads to an $O(\log m \cdot \log \frac{1}{\epsilon})$-approximation ratio because the subproblem to be solved in each phase is submodular ranking which only has an $O(\log \frac{1}{\epsilon})$-approximation ratio. Our work is based on a much simpler greedy-style algorithm and a new analysis, which leads to the $\mathcal{O}(\log m + \log 1/\epsilon)$ approximation ratio.

A different stochastic version of submodular ranking was considered in [55] where (i) the feedback was independent across elements and (ii) all the submodular functions needed

to be covered. In contrast, ASR involves a correlated scenario-based distribution and only the submodular function of the "realized" scenario i^* needs to be covered. Due to these differences, both the algorithm and analysis for ASR are different from [55]: our selection criterion (2.2) involves an additional "information gain" term, and our analysis requires a lot more work in order to handle correlations. We note that unlike ASR, the stochastic submodular ranking problem in [55] does not capture the optimal decision tree problem and its variants (equivalence class, decision region determination, etc).

For some previous special cases of ASR studied in [40], [12] and [58], one could obtain approximation algorithms via the framework of "adaptive submodularity" introduced by [38]. However, this approach does not apply to the general ASR problem and the approximation ratio obtained is at least $\Omega(\log^2 1/p_{min})$ where $p_{min} = \min_{i=1}^{m} p_i$ can be exponentially small in m. We note that the original paper by [38] claimed an $O(\log 1/p_{min})$ bound which was found to be erroneous by [70]; an updated version in [39] addresses this error but only obtains an $O(\log^2 1/p_{min})$ bound. So even in the case of uniform probabilities, our result provides an improved $O(\log m)$ approximation ratio compared to the $O(\log^2 m)$ ratio from [39]. We also note that our analysis is based on a completely different approach, which might be of independent interest.

Recently, [43] considered the scenario submodular cover problem, which can also be seens as a special case of ASR. This involves a *single* integer-valued submodular function for all scenarios which is defined on an expanded groundset $U \times G$ (i.e. pairs of "element, feedback" values). For this problem, our algorithm matches (in fact, improves slightly) the approximation ratio in [43] with a much simpler algorithm and analysis. We note that ASR is strictly more general than scenario submodular cover. For example, deterministic submodular ranking studied in [6] is a special case of ASR but not of scenario submodular cover.

A special case of the adaptive routing problem (ASP), the *adaptive TSP* was studied in [49], where the goal is to visit vertices in a random demand set. [49] obtained an $O(\log^2 n \cdot \log m)$-approximation algorithm for adaptive TSP and showed that any improvement on this would translate to a similar improvement for the group Steiner problem, which is a long standing open question. While our approximation ratio for ASP is slightly worse, it is much more general and involves a simpler analysis. For example, using ASP we can directly obtain an approximation algorithm for the variant of adaptive TSP where only a target number of demand vertices need to be visited.

A problem formulation similar to ASP was also studied in [64] where approximation algorithms were obtained under certain technical assumptions on the underlying submodular functions and probability distribution. To the best of our knowledge, the approach in [64] is not applicable to the general ASP problem considered here.

2.2 Algorithm for Adaptive Submodular Ranking

Recall that the state of our algorithm (i.e. any node in its decision tree) can be represented by (E, H) where (i) $E \subseteq U$ is the set of previously selected elements and (ii) $H \subseteq [m]$ is the set of scenarios that are compatible with feedback (on E) received so far and are still uncovered.

Intuitive explanation of the algorithm: At each state (E, H), our algorithm selects an element that maximizes the value computed in Equation (2.2). This can be viewed as the cost-effectiveness of any element e: the terms inside the paranthesis measure the gain from element e and this gain is normalized by the element's cost c_e. The gain of any element e comes from two sources:

1. *Information gain:* this corresponds to the first term in (2.2). Note that the feedback from element e can be used to define a partition of the scenarios in H, where all

scenarios in a part have the same feedback from e. Then, subset $L_e(H)$ is defined to be the complement of the largest-cardinality part; note that each part within $L_e(H)$ has size at most $|H|/2$. If the realized scenario happens to be in $L_e(H)$ then we make good progress in identifying the scenario: this is because the number of compatible scenarios decreases by (at least) a factor of two. The term $\sum_{j \in L_e(H)} p_j$ in (2.2) is just the probability that the realized scenario is in $L_e(H)$.

2. *Function coverage:* this corresponds to the second term in (2.2) and is based on the algorithm for *deterministic* submodular ranking from [6]. An important point here is that we consider the relative gain of each function f_i (for $i \in H$) which is the ratio of the increase in function value (i.e. $f_i(e \cup E) - f_i(E)$) to the remaining target (i.e. $1 - f_i(E)$), rather than just the absolute increase.

   Algorithm 1 gives a formal description. Note that we may not incur the cost for all selected elements under scenario i^* as the cost is only considered up to the point when i^* is covered.

Algorithm 1 ASR algorithm

$E \leftarrow \emptyset$, $H \leftarrow [m]$
while $H \neq \emptyset$ **do**
For any element $e \in U$, let $B_e(H)$ denote the set with maximum cardinality amongst

$$\{i \in H : r_i(e) = t\}, \quad \text{for } t \in G.$$

Define $L_e(H) = H \setminus B_e(H)$
Select element $e \in U \setminus E$ that maximizes:

$$\frac{1}{c_e} \cdot \left(\sum_{j \in L_e(H)} p_j \; + \; \sum_{i \in H} p_i \cdot \frac{f_i(e \cup E) - f_i(E)}{1 - f_i(E)} \right). \tag{2.2}$$

$E \leftarrow E \cup \{e\}$
Remove incompatible and covered scenarios from H based on the feedback from e.
end while
Output E

Note that H only contains uncovered scenarios. So, for all $i \in H$ we have $f_i(E) < 1$ and

the denominator in equation (2.2) is always positive. We have the following theorem:

Theorem II.1. *Algorithm 1 is an $\mathcal{O}(\log 1/\epsilon + \log m)$-approximation algorithm for* ASR.

Now, we analyze the performance of this algorithm. For any subset $T \subseteq [m]$ of scenarios, we use $\Pr(T) = \sum_{i \in T} p_i$. Let OPT denote an optimal solution to the ASR instance and ALG be the solution found by the above algorithm. Set $L := 15(1 + \ln 1/\epsilon + \log_2 m)$ and its choice will be clear later. We refer to the total cost incurred at any point in a solution as the *time*. We assume (without loss of generality, by scaling) that all costs are at least 1. For any $k = 0, 1, \cdots$, we define the following quantities:

- A_k is the set of uncovered scenarios of ALG at time $L \cdot 2^k$, and $a_k = \Pr(A_k)$. More formally, we have $A_k = \{i \in [m] : C_{ALG}(i) \geq L \cdot 2^k\}$ where $C_{ALG}(i)$ is the cost of scenario i in ALG.
- X_k is the set of uncovered scenarios of OPT at time 2^{k-1}, and $x_k = \Pr(X_k)$. That is, we have $X_k = \{i \in [m] : C_{OPT}(i) \geq 2^{k-1}\}$ where $C_{ALG}(i)$ is the cost of scenario i in OPT. Note that $x_0 = 1$.

To keep things simple, we will assume that all costs are integers. However, the analysis extends directly to the case of non-integer costs by replacing summations (over time t) with integrals.

Lemma II.2. *The expected cost of ALG and OPT can be bounded as follows.*

$$C_{ALG} \leq L \sum_{k \geq 0} 2^k a_k + L \qquad \text{and} \qquad C_{OPT} \geq \frac{1}{2} \sum_{k \geq 0} 2^{k-1} x_k \tag{2.3}$$

Proof. In ALG, for all $k \geq 1$, the probability of scenarios for which the cover time is in

$[2^{k-1}L, 2^kL)$ is equal to $a_{k-1} - a_k$. So we have:

$$C_{ALG} = \sum_{i\in[m]} p_i \cdot C_{ALG}(i) = \sum_{k\geq 1} \sum_{i\in A_{k-1}\setminus A_k} p_i \cdot C_{ALG}(i) \quad \leq \quad \sum_{k\geq 1} \sum_{i\in A_{k-1}\setminus A_k} p_i \cdot 2^k L$$

$$\leq \sum_{k\geq 1} 2^k L(a_{k-1} - a_k) + L(1 - a_0) \quad = \quad \sum_{k\geq 1} 2^k L a_{k-1} - \sum_{k\geq 1} 2^k L a_k + L(1 - a_0)$$

$$= 2\sum_{k\geq 0} 2^k L a_k - (\sum_{k\geq 0} 2^k L a_k - L a_0) + L(1 - a_0) \quad = \quad \sum_{k\geq 0} 2^k L a_k + L$$

On the other hand, in OPT, for all $k \geq 1$, the probability of scenarios for which the cover time is in $[2^{k-2}, 2^{k-1})$ is equal to $x_{k-1} - x_k$. So we have:

$$C_{OPT} = \sum_{i\in[m]} p_i \cdot C_{OPT}(i) \quad = \quad \sum_{k\geq 1} \sum_{i\in X_{k-1}\setminus X_k} p_i \cdot C_{OPT}(i) \tag{2.4}$$

$$\geq \quad \sum_{k\geq 1} \sum_{i\in X_{k-1}\setminus X_k} p_i \cdot 2^{k-2} \geq \sum_{k\geq 1} 2^{k-2}(x_{k-1} - x_k)$$

$$= \quad \sum_{k\geq 1} 2^{k-2} x_{k-1} - \sum_{k\geq 1} 2^{k-2} x_k = \sum_{k\geq 0} 2^{k-1} x_k - \frac{1}{2}(\sum_{k\geq 0} 2^{k-1} x_k - \frac{1}{2})$$

$$\geq \quad \frac{1}{2}\sum_{k\geq 0} 2^{k-1} x_k$$

Above we use the fact that $x_0 = 1$. □

Thus, if we upper bound each a_k by some multiple of x_k, it would be easy to obtain the approximation factor. However, this is not the case and instead we will prove:

Lemma II.3. *For all $k \geq 1$ we have $a_k \leq 0.2a_{k-1} + 3x_k$.*

Using this lemma we can prove Theorem II.1:

Proof. Let $Q = \sum_{k\geq 0} L \cdot 2^k a_k + L$, which is the bound on C_{ALG} from (2.3). Using Lemma II.3:

$$Q \;\leq\; L \cdot \sum_{k\geq 1} 2^k (0.2a_{k-1} + 3x_k) \;+\; L(a_0+1) \tag{2.5}$$

$$\leq\; 0.4L \cdot \sum_{k\geq 0} 2^k a_k \;+\; 6L \cdot \sum_{k\geq 1} 2^{k-1} x_k \;+\; L(a_0+1)$$

$$\leq\; 0.4(Q-L) \;+\; 6L\left(\sum_{k\geq 0} 2^{k-1}x_k \;-\; \frac{x_0}{2}\right) \;+\; 2L \;\leq\; 0.4\cdot Q + 12L\cdot C_{OPT} \tag{2.6}$$

The first inequality in (2.6) is by definition of Q and $a_0 \leq 1$, and the second inequality uses the bound on C_{OPT} from (2.3). Finally, we have $Q \leq 20L \cdot C_{OPT}$. Since $L = 15(1 + \ln 1/\epsilon + \log m)$ and $C_{ALG} \leq Q$, we obtain the theorem. □

2.2.1 Proof of Lemma II.3

We now prove Lemma II.3 for a fixed $k \geq 1$. Consider any time t between $L \cdot 2^{k-1}$ and $L \cdot 2^k$. Note that ALG's decision tree induces a partition of all the uncovered scenarios at time t, where each part H consists of all scenarios that are at a particular state (E, H) at time t. Let $R(t)$ denote the set of parts in this partition. We also use $R(t)$ to denote the collection of states corresponding to these parts. Note that all scenarios in A_k appear in $R(t)$ as these scenarios are uncovered even at time $L \cdot 2^k \geq t$. Similarly, all scenarios in $R(t)$ are in A_{k-1}. See Figure 2.3.

We have the following proposition:

Proposition II.3.1. Consider any state (E, H) and element $e \in E$. Then (i) the feedback values $\{r_i(e) : i \in H\}$ are all identical, and (ii) $L_e(H) = \emptyset$.

Proof. Note that by definition, at state (E, H) all scenarios in H are compatible with the feedback we have received from elements in E. Thus, all of them should have the same

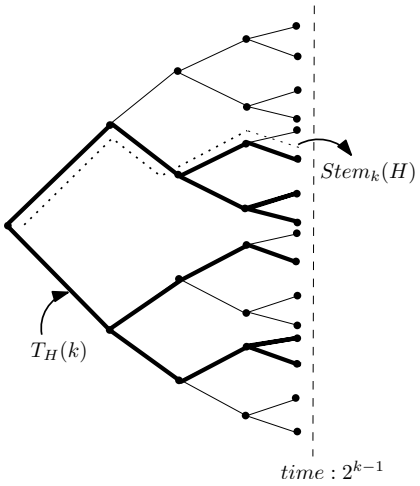

Figure 2.2: $\mathsf{Stem}_k(H)$ in OPT for $|G| = 2$

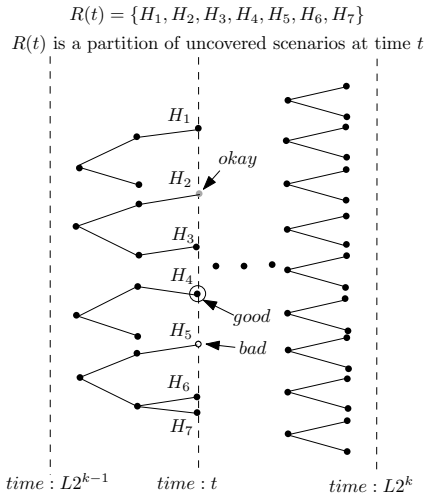

Figure 2.3: Bad, good and okay states in ALG

feedback for any element in E. Furthermore, for any $e \in E$, $L_e(H)$ is the complement of the largest part in the partition of H based on element e's feedback. According to the fact that all scenarios in H have the same feedback for element e, they are all in the same part, which is the largest part. So the complement of the largest part of the partition which is $L_e(H)$ is empty. □

For any $(E, H) \in R(t)$, note that E consists of all elements that have been completely selected before time t. The element that is being selected at state (E, H) is *not* included in E. Let $T_H(k)$ denote the subtree of OPT that corresponds to paths traced by scenarios in H up to time 2^{k-1}; this only includes elements that are completely selected by time 2^{k-1}. Note that each node (labeled by any element $e \in U$) in $T_H(k)$ has at most $|G|$ outgoing branches and one of them is labeled by the feedback corresponding to $B_e(H) = H \setminus L_e(H)$. We define $\mathsf{Stem}_k(H)$ to be the path in $T_H(k)$ that at each node (labeled e) follows the branch corresponding to $H \setminus L_e(H)$. See Figure 2.2 for an example. We also use $\mathsf{Stem}_k(H)$ to denote the set of elements that are completely selected on this path.

Definition II.4. Each state (E, H) in ALG is exactly one of the following types:

- **bad** if the probability of uncovered scenarios in H at the end of $\mathsf{Stem}_k(H)$ is at least $\frac{\Pr(H)}{3}$.
- **okay** if it is not bad and $\Pr(\cup_{e\in\mathsf{Stem}_k(H)} L_e(H))$ is at least $\frac{\Pr(H)}{3}$.
- **good** if it is neither bad nor okay and the probability of scenarios in H that get covered by $\mathsf{Stem}_k(H)$ is at least $\frac{\Pr(H)}{3}$.

See Figure 2.3. This is well defined, because by definition of $\mathsf{Stem}_k(H)$ each scenario in H is **(i)** uncovered at the end of $\mathsf{Stem}_k(H)$, or **(ii)** in $L_e(H)$ for some $e \in \mathsf{Stem}_k(H)$, or **(iii)** covered by some prefix of $\mathsf{Stem}_k(H)$, i.e. the function value reaches 1 on $\mathsf{Stem}_k(H)$. So the total probability of the scenarios in one of these 3 categories must be at least $\frac{\Pr(H)}{3}$. Therefore each state (E, H) is exactly one of these three types.

The following quantity turns out to be very useful in our proof of Lemma II.3.

$$Z := \sum_{t>L2^{k-1}}^{L2^k} \sum_{(E,H)\in R(t)} \max_{e\in U\setminus E} \frac{1}{c_e} \cdot \left(\Pr(L_e(H)) + \sum_{i\in H} p_i \cdot \frac{f_i(e\cup E) - f_i(E)}{1 - f_i(E)} \right) \tag{2.7}$$

$$= \sum_{t>L2^{k-1}}^{L2^k} \sum_{(E,H)\in R(t)} \max_{e\in U\setminus E} \frac{1}{c_e} \cdot \left(\sum_{i\in H} p_i \cdot \left(\mathbb{1}[i \in L_e] + \frac{f_i(e\cup E) - f_i(E)}{1 - f_i(E)} \right) \right) \tag{2.8}$$

Basically Z corresponds to the total "gain" according to our algorithm's selection criterion (2.2) accrued from time $L2^{k-1}$ to $L2^k$, over all the decision paths. We note that if costs are not integer, we can consider an integral over time $t \in (L2^{k-1}, L2^k]$ instead of the summation, and the rest of the analysis is essentially unchanged. Now, we obtain a lower and upper bound for Z and combine them to prove Lemma $II.3$. The lower bound views Z as a sum of terms over t, and uses the fact that the gain is "high" for good/ok states as well as the bound on probability of bad states (Proposition II.4.1). The upper bound views Z as a sum of terms over scenarios and uses the fact that if the total gain for a scenario is "high" then it must be already covered.

Proposition II.4.1. For any $t \in [(L2^{k-1}, L2^k]$, we have $\sum_{\substack{(E,H)\in R(t)\\(E,H):bad}} \Pr(H) \leq 3x_k$.

Proof. Note that $Stem_k(H) \subseteq T_H(k)$. Recall that $T_H(k)$ was the subtree of OPT up to time 2^{k-1} that only contains the scenarios in H. So, the probability of uncovered scenarios at the end of $Stem_k(H)$ is at most the probability of scenarios in H that are not covered in OPT by time 2^{k-1}. This probability is at least $\Pr(H)/3$ for the bad state (E, H) based on its definition. Now, since states in $R(t)$ induce a subpartition of scenarios, we have

$$x_k \;=\; \sum_{i\in X_k} p_i \;\geq\; \sum_{\substack{(E,H)\in R(t)\\(E,H):bad}} \;\sum_{i\in H\cap X_k} p_i \;\geq\; \sum_{\substack{(E,H)\in R(t)\\(E,H):bad}} \Pr(H)/3.$$

Rearranging, we obtain the desired inequality.

□

Lemma II.5. *We have* $Z \geq L \cdot (a_k - 3x_k)/3$.

Proof. Considering only the good/okay states in each $R(t)$ in the expression (2.7):

$$Z \geq \sum_{t>L2^{k-1}}^{L2^k} \left(\sum_{\substack{(E,H)\in R(t)\\(E,H):okay}} \max_{e\in U\setminus E} \frac{\Pr(L_e(H))}{c_e} \;+\; \sum_{\substack{(E,H)\in R(t)\\(E,H):good}} \max_{e\in U\setminus E} \sum_{i\in H} \frac{p_i}{c_e} \cdot \frac{f_i(e\cup E) - f_i(E)}{1 - f_i(E)} \right)$$

Fix any time t. For any state $(E, H) \in R(t)$ define $W(H) = \mathsf{Stem}_k(H) \setminus E$. The total cost of elements in $\mathsf{Stem}_k(H)$ is at most 2^{k-1}; so $c(W(H)) \leq 2^{k-1}$.

Case 1. (E, H) is an okay state. Since $W(H) \subseteq U \setminus E$ we can write:

$$\begin{aligned} \max_{e\in U\setminus E} \frac{\Pr(L_e(H))}{c_e} \quad &\geq \quad \max_{e\in W(H)} \frac{\Pr(L_e(H))}{c_e} \quad \geq \quad \frac{\sum_{e\in W(H)} \Pr(L_e(H))}{c(W(H))} \\ &\geq \quad \frac{\Pr(\cup_{e\in W(H)} L_e(H))}{2^{k-1}} \quad = \quad \frac{1}{2^{k-1}} \cdot \Pr(\cup_{e\in \mathsf{Stem}_k(H)} L_e(H)) \quad \geq \quad \frac{\Pr(H)}{3\cdot 2^{k-1}} \end{aligned} \tag{2.9}$$

The equality in (2.9) uses $\cup_{e\in E} L_e(H) = \emptyset$ (by Proposition II.3.1), and the last inequality is by definition of an okay state.

Case 2. (E, H) is a good state. Below, we use $F \subseteq H$ to denote the set of scenarios that get covered in $\mathsf{Stem}_k(H)$; by definition of a good state, we have $\Pr(F) \geq \Pr(H)/3$. Again using $W(H) \subseteq U \setminus E$, we have:

$$\begin{aligned}
&\max_{e\in U\setminus E} \frac{1}{c_e}\sum_{i\in H} p_i \cdot \frac{f_i(e\cup E) - f_i(E)}{1-f_i(E)} \geq \max_{e\in W(H)} \frac{1}{c_e}\sum_{i\in H} p_i \cdot \frac{f_i(e\cup E) - f_i(E)}{1-f_i(E)} \\
&\geq \frac{1}{c(W(H))}\sum_{e\in W(H)}\sum_{i\in H} p_i \cdot \frac{f_i(e\cup E) - f_i(E)}{1-f_i(E)} \\
&= \frac{1}{c(W(H))}\sum_{i\in H} p_i \sum_{e\in W(H)} \frac{f_i(e\cup E) - f_i(E)}{1-f_i(E)} \quad \geq \quad \frac{1}{2^{k-1}}\sum_{i\in H} p_i \cdot \frac{f_i(W(H)\cup E) - f_i(E)}{1-f_i(E)} \qquad (2.10) \\
&= \frac{1}{2^{k-1}}\sum_{i\in H} p_i \cdot \frac{f_i(\mathsf{Stem}_k(H)) - f_i(E)}{1-f_i(E)} \quad \geq \quad \sum_{i\in F}\frac{p_i}{2^{k-1}} \quad = \quad \frac{\Pr(F)}{2^{k-1}} \quad \geq \quad \frac{\Pr(H)}{3\cdot 2^{k-1}} \qquad (2.11)
\end{aligned}$$

The last inequality in (2.10) is by submodularity of the f_is, and the next equality is by definition of $W(H)$. The first inequality in (2.11) is based on this fact that $f_i(Stem_k(H)) = 1$ for any $i \in F$ and the last inequality is by definition of a good state. Now, we combine (2.9) and (2.11):

$$\begin{aligned}
Z \quad &\geq \quad \sum_{t>L2^{k-1}}^{L2^k} \sum_{\substack{(E,H)\in R(t)\\(E,H):okay}} \frac{\Pr(H)}{3\cdot 2^{k-1}} \quad + \quad \sum_{t>L2^{k-1}}^{L2^k} \sum_{\substack{(E,H)\in R(t)\\(E,H):good}} \frac{\Pr(H)}{3\cdot 2^{k-1}} \\
&= \quad \sum_{t>L2^{k-1}}^{L2^k} \frac{\Pr(R(t)) - \sum_{\substack{(E,H)\in R(t)\\(E,H):bad}} \Pr(H)}{3\cdot 2^{k-1}} \quad \geq \quad \sum_{t>L2^{k-1}}^{L2^k} \frac{a_k - 3x_k}{3\cdot 2^{k-1}} \quad = \quad \frac{L\cdot(a_k - 3x_k)}{3} \qquad (2.12)
\end{aligned}$$

The first equality uses the fact that the states $(E, H) \in R(t)$ are exactly one of the types bad/okay/good. The last inequality uses Proposition II.4.1 and that $\{H : (E, H) \in R(t)\}$ contains all scenarios in A_k. □

Lemma II.6. *We have* $Z \leq a_{k-1} \cdot (1 + \ln 1/\epsilon + \log m)$.

Proof. For any scenario $i \in A_{k-1}$ (i.e. uncovered in ALG by time $L2^{k-1}$) let $\widehat{\pi}_i$ be the path traced by i in ALG's decision tree. For each element e that appears in $\widehat{\pi}_i$, we say that e is selected during the interval $K_{e,i} = (D_e, D_e + c_e]$ where D_e is the total cost of elements in $\widehat{\pi}_i$ before e. Let π_i be the sub-path of $\widehat{\pi}_i$ consisting of elements selected between time $2^{k-1}L$ and 2^kL or when i gets covered (whatever happens earlier). Let $t_{e,i} \leq c_e$ denote the width of the interval $K_{e,i} \cap (L2^{k-1}, L2^k]$. Note that there can be at most two elements e in π_i with $t_{i,e} < c_e$: one that is being selected at time $L2^{k-1}$ and another at $L2^k$.

Recall that for any $L2^{k-1} < t \leq L2^k$, every scenario in $R(t)$ appears in A_{k-1}. So only scenarios in A_{k-1} can contribute to Z and we rewrite (2.8) by interchanging summations as follows:

$$\begin{aligned} Z &= \sum_{i \in A_{k-1}} p_i \cdot \sum_{e \in \pi_i} t_{e,i} \cdot \frac{1}{c_e} \left(\frac{f_i(e \cup E) - f_i(E)}{1 - f_i(E)} + \mathbb{1}[i \in L_e(H)] \right) \\ &\leq \sum_{i \in A_{k-1}} p_i \cdot \left(\sum_{e \in \pi_i} \frac{f_i(e \cup E) - f_i(E)}{1 - f_i(E)} + \sum_{e \in \pi_i} \mathbb{1}[i \in L_e(H)] \right) \end{aligned} \tag{2.13}$$

Above, for any $e \in \pi_i$ we use (E, H) to denote the state at which e is selected.

Fix any scenario $i \in A_{k-1}$. For the first term, we use Claim II.6.1 below and the definition of ϵ in Equation (2.1). This implies $\sum_{e \in \pi_i} \frac{f_i(e \cup E) - f_i(E)}{1 - f_i(E)} \leq 1 + \ln \frac{1}{\epsilon}$. To bound the second term, note that if scenario $i \in L_e(H)$ when ALG selects element e, then the number of compatible scenarios decreases by at least a factor of *two* in path π_i. So such an event can happen at most $\log_2 m$ times along the path π_i. Thus we can write $\sum_{e \in \pi_i} \mathbb{1}[i \in L_e(H)] \leq \log_2 m$. The lemma now follows from Equation (2.13). □

We now complete the proof of Lemma II.3.

Proof. By Lemma II.5 and Lemma II.6 we have:

$$L \cdot (a_k - 3x_k)/3 \quad \leq \quad Z \quad \leq \quad a_{k-1} \cdot (1 + \ln 1/\epsilon + \log m) = \quad a_{k-1} \cdot \frac{L}{15}$$

Rearranging, we obtain $a_k \leq 0.2 \cdot a_{k-1} + 3x_k$ as needed. □

Claim II.6.1 (Claim 2.1 in [6]). Let $f : 2^U \to [0, 1]$ be any monotone function with $f(\emptyset) = 0$ and $\epsilon = \min\{f(S \cup \{e\}) - f(S) \ : \ e \in U, S \subseteq U, f(S \cup \{e\}) - f(S) > 0\}$. Then, for any sequence $\emptyset = S_0 \subseteq S_1 \subseteq \cdots S_k \subseteq U$ of subsets, we have

$$\sum_{t=1}^{k} \frac{f(S_t) - f(S_{t-1})}{1 - f(S_{t-1})} \ \leq \ 1 + \ln \frac{1}{\epsilon}.$$

2.3 Algorithm for Adaptive Submodular Routing

Recall that the adaptive submodular routing problem (ASP) is a generalization of ASR to a vehicle-routing setting where costs correspond to a metric $(U \cup \{s\}, d)$. Here, U denotes the set of elements and s is a special root vertex. The rest of the input is exactly as in ASR: we are given m scenarios where each scenario $i \in [m]$ has some probability p_i, a submodular function f_i and a feedback function r_i. The goal is to compute an adaptive tour (that begins and ends at s) and covers a random scenario i^* at minimum expected cost, where the cost corresponds to the cost of the path of elements we need to take until we cover the realized scenario. For any walk P, when it is clear from context, we will also use P to refer to the vertices/elements on this walk.

An important subproblem in our algorithm for ASP is the *submodular orienteering problem* (SOP), defined as follows. There is a metric $(U \cup \{s\}, d)$ with root s, a monotone submodular function $f : 2^V \to \mathbb{R}_+$ and a bound B. The goal is to compute a tour P originating from s of cost at most B that maximizes $f(P)$. A (ρ, σ)-bicriteria approximation algorithm for SOP returns a tour P such that cost of P is at most $\sigma \cdot B$ and $f(P) \geq \mathsf{OPT}/\rho$, where OPT

is the maximum value of a tour of cost at most B. Our ASP algorithm will make use of a (ρ, σ)-bicriteria approximation algorithm denoted ALG-SOP.

Intuitive explanation of the algorithm: Our algorithm involves concatenating a sequence of smaller tours (each originating from s) where the tour costs increase geometrically. Each such tour is obtained as a solution to a suitably defined instance of SOP. The SOP instance encountered at state (E, H) involves the function $g_{(E,H)}$ defined in (2.14). Similar to the Equation (2.2), which is the definition of "gain" of an individual element in the ASR algorithm, function $g_{(E,H)}(T)$ measures the collective gain from any subset T of elements. This again comprises of two parts:

1. *Information gain:* this is the first term in (2.14). The definition of subsets $L_e(H)$ is the same as for ASR. If the realized scenario happens to be in $L_e(H)$ *for any* $e \in T$ then it is clear that we make good progress in identifying the scenario: the number of compatible scenarios decreases by (at least) a factor of two. The term $\Pr\left(\cup_{e \in T} L_e(H)\right)$ in (2.14) is just the probability that the realized scenario is in $L_e(H)$ for some $e \in T$.
2. *Function coverage:* this is the second term in (2.14) and is based on the algorithm for *deterministic* submodular routing from [55].

Crucially, both of these terms in $g_{(E,H)}$ are monotone submodular functions: so SOP can be used.

As with ASR, the algorithm for ASP may not incur the cost of the entire walk traced under scenario i^*: recall that the cost is only incurred until i^* gets covered.

We can always assume that $P_u \subseteq U \setminus E$ in Line 9: this is because $g_{(E,H)}(e) = 0$ for all $e \in E$ as in Proposition II.3.1. In the rest of this section, we will prove the following result.

Theorem II.7. *If ALG-SOP is any (ρ, σ)-bicriteria approximation algorithm for SOP, our algorithm for* ASP *is an $\mathcal{O}(\sigma\rho(\log 1/\epsilon + \log m))$-approximation algorithm.*

Algorithm 2 ASP algorithm

$E \leftarrow \emptyset, \pi \leftarrow \varnothing, H \leftarrow [m]$ and $D = 15\rho(1 + \ln \frac{1}{\epsilon} + \log m)$.
for phase $k = 0, 1, 2, ...$ **do**
  If $H = \emptyset$ then output π and end the algorithm.
  for iteration $u = 1, 2, ..., D$ **do**
    For any element $e \in U \setminus E$, let $B_e(H)$ denote the set with maximum cardinality amongst $\{i \in H : r_i(e) = t\}$ for $t \in G$; and define $L_e(H) := H \setminus B_e(H)$
    Define the submodular function

$$g_{(E,H)}(T) := \Pr\left(\cup_{e \in T} L_e(H)\right) + \sum_{i \in H} p_i \cdot \frac{f_i(E \cup T) - f_i(E)}{1 - f_i(E)}, \quad \forall T \subseteq U \tag{2.14}$$

    Use ALG-SOP to approximately solve the SOP instance on metric $(U \cup \{s\}, d)$ with root s, submodular function $g_{(E,H)}$ and cost bound 2^k to obtain tour P_u.
    $E \leftarrow E \cup P_u$ and concatenate P_u to π to form a new tour
    Remove incompatible and covered scenarios from H based on the feedback from P_u
  end for
end for

We can use the following known result on SOP.

Theorem II.8. *[18] For all constant $\delta > 0$, there is a polynomial time $(\mathcal{O}(1), \mathcal{O}(\log^{2+\delta} n))$-bicriteria approximation algorithm for the Submodular Orienteering problem.*

By combining Theorems II.8 and II.7 we obtain:

Corollary II.9. *For any constant $\delta > 0$, there is an $\mathcal{O}((\log 1/\epsilon + \log m) \cdot \log^{2+\delta} n)$-approximation algorithm for the adaptive submodular routing problem.*

Instead of Theorem II.8, we can also use the *quasi-polynomial* time $\mathcal{O}(\log n)$-approximation algorithm for SOP from [23], which implies:

Corollary II.10. *There is a quasipolynomial time $\mathcal{O}((\log 1/\epsilon + \log m) \cdot \log n)$-approximation algorithm for* ASP.

2.3.1 Analysis

We start by showing that the use of SOP is well-defined.

Proposition II.10.1. For any state (E, H) in Algorithm 2, the function $g_{(E,H)}$ is monotone and submodular.

Proof. First note that for any monotone submodular function f_i and $E \subseteq U$, we have $f_i(E \cup T) - f_i(E)$ is a monotone submodular function of T. Also $f(T) = \Pr(\bigcup_{e \in T} L_e(H))$ is a weighted coverage function, so it is monotone submodular. Now, since a weighted sum of submodular functions is submodular, the following function is submodular:

$$\sum_{i \in H} p_i \cdot \frac{f_i(E \cup T) - f_i(E)}{1 - f_i(E)} + \Pr(\bigcup_{e \in T} L_e(H))$$

which is equal to $g_{(E,H)}(T)$. □

In the following, we use cost and time interchangeably. We will refer to the outer-loop in Algorithm 2 by *phase* and the inner-loop by *iteration*. Define $\bar{L} := 2D \cdot \sigma$. Then we have the following proposition:

Proposition II.10.2. All vertices that are added to E in the j-th phase are visited in π by time $\bar{L} \cdot 2^j$.

Proof. In each phase k, we add D tours of cost at most $2^k\sigma$ each. So a vertex that is added in phase j is visited by time $\sum_{k=0}^{j} 2^k D \cdot \sigma \leq 2^{j+1} D \cdot \sigma = \bar{L} \cdot 2^j$. □

Let ALG be the solution produced by Algorithm 2 and OPT be the optimal solution. For any $k = 0, 1, \cdots$, we define the following quantities:

- A_k is the set of uncovered scenarios of ALG at the end of phase k, and $a_k = \Pr(A_k)$.
- X_k is the set of uncovered scenarios of OPT at time 2^{k-1}, and $x_k = \Pr(X_k)$. Note that $x_0 = 1$.

Lemma II.11. *The expected cost of ALG and OPT can be bounded as follows.*

$$C_{ALG} \leq \bar{L} \sum_{k \geq 0} 2^k a_k + \bar{L} \qquad \text{and} \qquad C_{OPT} \geq \frac{1}{2} \sum_{k \geq 0} 2^{k-1} x_k \tag{2.15}$$

Proof. By Proposition II.10.2, for all $k \geq 1$ every scenario in $A_{k-1} \setminus A_k$ in ALG is covered by time $\bar{L}2^k$. So we can write exactly the same inequalities as in the proof of Lemma II.2. □

As for ASR, in order to prove Theorem II.7, it suffices to prove:

Lemma II.12. *For any $k \geq 0$, we have $a_k \leq 0.2a_{k-1} + 3x_k$.*

2.3.2 Proof of Lemma II.12

Throughout this section we fix phase k to its value in Lemma II.12. Consider any iteration u in phase k of the algorithm. ALG's decision tree induces a partition of all the uncovered scenarios at iteration u, where each part H consists of all scenarios that are at a particular state (E, H) at the start of iteration u. Let $R_k(u)$ denote the set of parts in this partition. We also use $R_k(u)$ to denote the collection of states corresponding to these parts. Note that all scenarios in A_k appear in $R_k(u)$ as these scenarios are uncovered even at the end of phase k. Similarly, all scenarios in $R_k(u)$ are in A_{k-1}.

The analysis is similar to that for Lemma II.3. Analogous to the quantity Z in the proof of Lemma II.3, we will use:

$$\bar{Z} := \sum_{u=1}^{D} \sum_{(E,H) \in R_k(u)} \max_{P \in \mathcal{A}(E,H,k)} g_{(E,H)}(P) \tag{2.16}$$

Above, $\mathcal{A}(E, H, k)$ denotes the set of feasible tours to the SOP instance solved in iteration u of phase k, and (E, H) denotes the state at the beginning of this iteration. We prove Lemma II.12 by upper/lower bounding $\bar{Z}$.

For any $(E, H) \in R_k(u)$, note that E consists of all elements that have been selected before iteration u. The set of elements that are selected at iteration u are *not* included in

E. We also define $T_H(k)$ and $\mathsf{Stem}_k(H)$ as in Section 2.2. Recall, $T_H(k)$ is the subtree of OPT that corresponds to paths traced by scenarios in H up to time 2^{k-1}; this only includes elements that are completely selected by time 2^{k-1}. And $\mathsf{Stem}_k(H)$ is the path in $T_H(k)$ that at each node (labeled e) follows the branch corresponding to $H \setminus L_e(H)$. Again we also use $\mathsf{Stem}_k(H)$ to denote the set of elements that are on this path. We will also use the definition of "bad", "okay" and "good" states from Definition II.4. Then, exactly as in Proposition II.4.1 we have:

Proposition II.12.1. For any iteration u in phase k, $\sum_{\substack{(E,H)\in R_k(u)\\(E,H):bad}} \Pr(H) \le 3x_k$.

Lemma II.13. *We have* $\bar{Z} \ge D \cdot (a_k - 3x_k)/3$.

Proof. Considering only the good/okay states in each $R_k(u)$ in the expression (2.16):

$$\begin{aligned}\bar{Z} &= \sum_{u=1}^{D} \sum_{(E,H)\in R_k(u)} \max_{P\in\mathcal{A}(E,H,k)} \left(\sum_{i\in H} p_i \cdot \frac{f_i(E\cup P) - f_i(E)}{1-f_i(E)} + \Pr(\bigcup_{e\in P} L_e(H)) \right) \\ &\ge \sum_{u=1}^{D} \sum_{\substack{(E,H)\in R_k(u)\\(E,H):okay}} \max_{P\in\mathcal{A}(E,H,k)} \Pr\left(\bigcup_{e\in P} L_e(H)\right) \\ &\qquad + \sum_{u=1}^{D} \sum_{\substack{(E,H)\in R_k(u)\\(E,H):good}} \max_{P\in\mathcal{A}(E,H,k)} \sum_{i\in H} p_i \cdot \frac{f_i(E\cup P) - f_i(E)}{1-f_i(E)}\end{aligned}$$

Fix any iteration u. For any state $(E,H) \in R_k(u)$ define $W(H) = \mathsf{Stem}_k(H) \setminus E$. Note that the cost of $\mathsf{Stem}_k(H)$ is at most 2^{k-1}, so the tour obtained by doubling this path is in $\mathcal{A}(E,H,k)$: i.e. the tour originates from s and has cost at most 2^k. We call this tour $\overline{W}(H)$.

Case 1. (E,H) is an okay state. Since $\overline{W}(H) \in \mathcal{A}(E,H,k)$,

(2.17)

$$\max_{P\in\mathcal{A}(E,H,k)} \Pr\left(\bigcup_{e\in P} L_e(H)\right) \;\ge\; \Pr\left(\bigcup_{e\in W(H)} L_e(H)\right) \;=\; \Pr\left(\bigcup_{e\in \mathsf{Stem}_k(H)} L_e(H)\right) \;\ge\; \frac{\Pr(H)}{3}$$

The equality above uses $\cup_{e\in E} L_e(H) = \emptyset$ (by Proposition II.3.1), and the last inequality is by Definition II.4 of an okay state.

Case 2. (E, H) is a good state. Below, we use $F \subseteq H$ to denote the set of scenarios that get covered in $\mathsf{Stem}_k(H)$; by definition of a good state, we have $\Pr(F) \geq \Pr(H)/3$. Again using $\overline{W}(H) \in \mathcal{A}(E, H, k)$,

$$\begin{aligned} \max_{P\in\mathcal{A}(E,H,k)} \sum_{i\in H} p_i \cdot \frac{f_i(P\cup E) - f_i(E)}{1 - f_i(E)} \;&\geq\; \sum_{i\in H} p_i \cdot \frac{f_i(W(H)\cup E) - f_i(E)}{1 - f_i(E)} \\ = \sum_{i\in H} p_i \cdot \frac{f_i(\mathsf{Stem}_k(H)) - f_i(E)}{1 - f_i(E)} \geq \sum_{i\in F} p_i \;&=\; \Pr(F) \;\geq\; \frac{\Pr(H)}{3} \end{aligned} \tag{2.18}$$

The first equality of (2.18) is by definition of $W(H)$. The next inequality is based on the fact that $f_i(\mathsf{Stem}_k(H)) = 1$ for any $i \in F$ and the last inequality is by definition of a good state. Now, we combine (2.17) and (2.18) with the definition of $\bar{Z}$:

$$\begin{aligned} \bar{Z} &\geq \sum_{u=1}^{D} \sum_{\substack{(E,H)\in R_k(u)\\(E,H):okay}} \frac{\Pr(H)}{3} + \sum_{u=1}^{D} \sum_{\substack{(E,H)\in R_k(u)\\(E,H):good}} \frac{\Pr(H)}{3} \\ &= \sum_{u=1}^{D} \frac{\Pr(R_k(u)) - \sum_{\substack{(E,H)\in R_k(u)\\(E,H):bad}} \Pr(H)}{3} \\ &\geq \sum_{u=1}^{D} \frac{a_k - 3x_k}{3} \quad = \quad \frac{D\cdot(a_k - 3x_k)}{3} \end{aligned} \tag{2.19}$$

The first equality uses the fact that the states corresponding to each $(E, H) \in R_k(u)$ are exactly one of the types bad/okay/good. The last inequality uses Proposition II.12.1 and that $R_k(u)$ contains all scenarios in A_k. □

Lemma II.14. *We have* $\bar{Z} \leq a_{k-1} \cdot \rho(1 + \ln \frac{1}{\epsilon} + \log m)$.

Proof. For any scenario $i \in A_{k-1}$ (i.e. uncovered in ALG at the end of phase $k-1$) let π_i

be the path traced by i in ALG's decision tree, starting from the end of phase $k-1$ to the end of phase k or when i gets covered (whatever happens first). Formally, we represent π_i as a sequence of tuples (E_{iu}, H_{iu}, P_{iu}) for each iteration u in phase k, where (E_{iu}, H_{iu}) is the state at the start of iteration u and P_{iu} is the new tour chosen by ALG at this state.

Recall that for any iteration u, every scenario in $R_k(u)$ appears in A_{k-1}. So only scenarios in A_{k-1} can contribute to $\bar{Z}$, because every part H in $R_k(u)$ is a subset of A_{k-1}. Furthermore, since ALG-SOP is a (ρ, σ)-bicriteria approximation algorithm, it selects paths P_u such that $\rho \cdot g_{(E,H)}(P_u) \geq \max_{P \in \mathcal{A}(E,H,k)} g_{(E,H)}(P)$. So we can bound $\bar{Z}$ from above as follows:

$$\begin{aligned}
\bar{Z} = \sum_{u=1}^{D} \sum_{(E,H)\in R_k(u)} \max_{P\in\mathcal{A}(E,H,k)} g_{(E,H)}(P) \quad &\leq \quad \rho \cdot \sum_{u=1}^{D} \sum_{(E,H)\in R_k(u)} g_{(E,H)}(P_u) \\
&\leq \rho \cdot \sum_{u=1}^{D} \sum_{(E,H)\in R_k(u)} \left(\sum_{i\in H} \left(p_i \cdot \frac{f_i(E\cup P_u) - f_i(E)}{1-f_i(E)} \right) + \Pr\Big(\bigcup_{e\in P_u} L_e(H) \Big) \right) \\
&= \rho \cdot \sum_{u=1}^{D} \sum_{(E,H)\in R_k(u)} \sum_{i\in H} p_i \cdot \left(\frac{f_i(E\cup P_u) - f_i(E)}{1-f_i(E)} + \mathbb{1}\left[i \in \cup_{e\in P_u} L_e(H) \right] \right) \\
&\leq \rho \cdot \sum_{i\in A_{k-1}} p_i \cdot \left(\sum_{(E_{iu},H_{iu},P_{iu})\in\pi_i} \left(\frac{f_i(P_{iu}\cup E_{iu}) - f_i(E_{iu})}{1-f_i(E_{iu})} + \mathbb{1}[i \in \cup_{e\in P_{iu}} L_e(H_{iu})] \right) \right) \qquad (2.20) \\
&= \rho \cdot \sum_{i\in A_{k-1}} p_i \cdot \left(\sum_{(E_{iu},H_{iu},P_{iu})\in\pi_i} \frac{f_i(P_{iu}\cup E_{iu}) - f_i(E_{iu})}{1-f_i(E_{iu})} + \sum_{(E_{iu},H_{iu},P_{iu})\in\pi_i} \mathbb{1}[i \in \cup_{e\in P_{iu}} L_e(H_{iu})] \right) \qquad (2.21)
\end{aligned}$$

where the inequality (2.20) is due to an interchange of summation and the fact that each part H of $R_k(u)$ is a subset of A_{k-1}. Now, fix any scenario $i \in A_{k-1}$. For the first term in (2.21), we use Claim II.6.1 and the definition of ϵ in (2.1). This implies $\sum_{(E_{iu},H_{iu},P_{iu})\in\pi_i} \frac{f_i(P_{iu}\cup E_{iu}) - f_i(E_{iu})}{1-f_i(E_{iu})} \leq 1 + \ln \frac{1}{\epsilon}$. To bound the second term, note that if at some iteration u with state (E, H) the algorithm selects subset P_u, and if scenario $i \in \cup_{e\in P_u} L_e(H)$ then the number of possible scenarios decreases by at least a factor of *two* in path π_i. So such an event can happen at most $\log_2 m$ times along the path π_i. Thus we can write

$\sum_{(E_{iu},H_{iu},P_{iu})\in\pi_i} \mathbb{1}[i \in \bigcup_{e\in P_{iu}} L_e(H_{iu})] \leq \log_2 m$. The lemma follows from (2.21). □

Now we can complete the proof of Lemma II.12.

Proof. By Lemma II.13 and Lemma II.14 we have:

$$D \cdot (a_k - 3x_k)/3 \quad \leq \quad \bar{Z} \quad \leq \quad a_{k-1} \cdot \rho(1 + \ln 1/\epsilon + \log m) = \quad a_{k-1} \cdot \frac{D}{15}$$

Rearranging, we obtain $a_k \leq 0.2 \cdot a_{k-1} + 3x_k$ as needed. □

2.4 Applications

In this section we discuss various applications of ASR. For some of these applications, we obtain improvements over previously known results. For many others, we match (or nearly match) the previous best results using a simpler algorithm and analysis. Some of the applications discussed below are new, for which we provide the first approximation algorithms. Table 2.1 summarizes some of these applications. As defined, cost in ASR and ASP is only incurred until the realized scenario i^* gets covered and the algorithm may not know this (see Section 2.1.1). This definition is suitable for the applications discssed in Sections 2.4.1, 2.4.2, 2.4.3 and 2.4.10. However, for the other applications (Sections 2.4.4, 2.4.5, 2.4.6, 2.4.7, 2.4.8 and 2.4.9) the algorithm needs to know explicitly when to stop. For these applications, we also mention the stopping criteria used and show that it coincides with the (usual) criterion of just covering i^*. So Theorem II.1 or II.7 can be applied in all cases.

2.4.1 Deterministic Submodular Ranking

In this problem we are given a set of n elements and m monotone submodular functions $f_1, f_2, \ldots, f_m$ where each $f_i : 2^{[n]} \to [0, 1]$. We also have a non-negative weight w_i associated with each $i \in [m]$. The goal is to find a static linear ordering of the elements that minimizes

Problem	Previous best result	Our result
Adaptive Multiple Intent Re-ranking	-	$\mathcal{O}(\log K + \log m)$
Generalized Optimal Decision Tree	-	$\mathcal{O}(\log m)$
Decision Region Determination	$\mathcal{O}(r \log m)$ in exp time	$\mathcal{O}(r \log m)$ in poly time
Stochastic Knapsack Cover	-	$\mathcal{O}(\log m + \log W)$
Stochastic Matroid Basis	-	$\mathcal{O}(\log m + \log q)$
Adaptive Traveling Repairman Problem	$\mathcal{O}(\log^2 n \log m)$	$\mathcal{O}(\log^{2+\delta} n(\log m + \log n))$
Adaptive Traveling Salesman Problem	$\mathcal{O}(\log^2 n \log m)$	$\mathcal{O}(\log^{2+\delta} n(\log m + \log n))$

Table 2.1: Some applications of adaptive submodular ranking.

the weighted summation of functions' cover time, where the cover time of a function f_i is the first time that its value reaches one. This is a special case of ASR where there is no feedback. Formally, we consider the ASR instance with the same f_is, $G = \emptyset$, and probabilities $p_i = w_i/(\sum_{j=1}^{n} w_j)$. Theorem II.1 directly gives an $\mathcal{O}(\log m + \log \frac{1}{\epsilon})$-approximation algorithm. Moreover, by observing that in (2.2) for any state (E, H) we have $L_e(H) = \emptyset$, we can strengthen the upper bound in Lemma II.6 to $Z \leq a_{k-1} \cdot (1 + \ln 1/\epsilon)$. This implies that our algorithm is an $\mathcal{O}(\log \frac{1}{\epsilon})$-approximation, matching the best result in [6] and [55].

2.4.2 Adaptive Multiple Intents Re-ranking.

This is an adaptive version of the multiple intents re-ranking problem, introduced in [7] with applications to search ranking. There are n results to a particular search query, and m different users. Each user i is characterized by a subset S_i of the results that s/he is interested in and a threshold $K_i \leq |S_i|$: user i gets "covered" after seeing at least K_i results from the subset S_i. There is also a probability distribution $\{p_i\}_{i=1}^{m}$ on the m users, from which the realized user i^* is chosen. An algorithm displays results one by one and receives feedback on $e \in S_{i^*}$, i.e. whether result e is relevant to user i^*. The goal is to find an adaptive ordering of the results that minimizes the expected number of results to cover user i^*. We note that the algorithm need not know when this occurs, i.e. when to stop.

This can be modeled as ASR with results corresponding to elements U and users corresponding to the m scenarios. The feedback values are $G = \{0, 1\}$ and the feedback functions are given by $r_i(e) = \mathbb{1}(e \in S_i)$ for all $i \in [m]$ and $e \in U$. For each scenario $i \in [m]$, the submodular function $f_i(S) = \min(|S \cap S_i|, K_i)/K_i$. Letting $K = \max_{i \in [m]} K_i$, we can see that the parameter ϵ is equal to $1/K$. So Theorem II.1 implies an $\mathcal{O}(\log K + \log m)$-approximation algorithm. We note however that in the deterministic setting, there are better $\mathcal{O}(1)$-approximation algorithms in [9], [83] and [56]. These results are based on a different linear-program-based approach: extending such an approach to the stochastic case is still an interesting open question.

2.4.3 Minimum Cost Matroid Basis

Consider the following stochastic network design problem. We are given an undirected graph (V, E) with edge costs. However, only a random subset $E^* \subseteq E$ of the edges are active. We assume an explicit scenario-based joint distribution for E^*: there are m scenarios where each scenario $i \in [m]$ occurs with probability p_i and corresponds to active edges $E^* = E_i$. An algorithm learns whether/not an edge e is active only upon *testing* e which incurs time c_e. An algorithm needs to adaptively test a subset $S \subseteq E$ of edges so that $S \cap E^*$ achieves the maximum possible connectivity in the active graph (V, E^*), i.e. $S \cap E^*$ must contain a maximal spanning forest of graph (V, E^*). The objective is to minimize the expected time before the tested edges achieve maximal connectivity in the active graph. The algorithm need not know when this occurs, i.e. when to stop.

We can model this as an ASR instance with edges E as elements and scenarios as described above. The feedback values are $G = \{0, 1\}$ and $r_i(e) = \mathbb{1}(e \in E_i)$ for all $i \in [m]$ and $e \in E$. The submodular functions are $f_i(S) = \frac{\mathsf{rank}_i(S \cap E_i)}{\mathsf{rank}_i(E_i)}$ where rank_i is the rank function of the graphic matroid on (V, E_i). The f_is are monotone and submodular due to the submodularity of matroid rank functions. Moreover, the parameter ϵ is at least $\frac{1}{q}$ where $q = |V|$. So

Theorem II.1 implies an $\mathcal{O}(\log m + \log q)$-approximation algorithm. We note that the same result also holds for a general matroid: where a random (correlated) subset of elements is active and the goal is to find a basis over the active elements at minimum expected cost.

2.4.4 Optimal Decision Tree (ODT)

This problem captures many applications in active learning, medical diagnosis and databases; see e.g. [20] and [27]. There are m possible hypotheses with a probability distribution $\{p_i\}_{i=1}^m$, from which an unknown hypothesis i^* is drawn. There are also a number of binary tests; each test e costs c_e and returns a positive outcome if i^* lies in some subset Y_e of hypotheses and a negative outcome if $i^* \in [m] \setminus Y_e$. It is assumed that i^* can be uniquely identified by performing all tests. The goal is to perform an adaptive sequence of tests so as to identify hypothesis i^* at the minimum expected cost.

This can be cast as an ASR instance as follows. We associate elements with tests U and scenarios with hypotheses $[m]$. The feedback values are $G = \{0, 1\}$ and the feedback functions are given by $r_i(e) = \mathbb{1}(i \in Y_e)$ which denotes the outcome of test e on hypothesis i. In order to define the submodular functions, let

$$T_e(i) = \begin{cases} [m] \setminus Y_e & \text{if } i \in Y_e \\ Y_e & \text{if } i \notin Y_e \end{cases}, \qquad \forall e \in U \text{ and } i \in [m].$$

Then, for each scenario $i \in [m]$, define the submodular function $f_i(S) = |\cup_{e \in S} T_e(i)| \cdot \frac{1}{m-1}$. Note that at any point in the algorithm where we have performed a set S of tests, the set $\bigcup_{e \in S} T_e(i^*)$ consists of all hypothesis that have a different outcome from i^* in at least one of the tests in S. So i^* is uniquely identified after performing tests S if and only if $f_{i^*}(S) = 1$. The algorithm's stopping criterion is the first point when the number of compatible hypotheses/scenarios reaches one: this coincides with the point where f_{i^*} gets covered. Note that the parameter ϵ is equal to $\frac{1}{m}$; so by Theorem II.1 we obtain an $\mathcal{O}(\log m)$-approximation algorithm which is known to be best-possible (unless P=NP), as shown by

[20]. Although this problem has been extensively studied, previously such a result was known only via a complex algorithm in [49] and [26]. We also note that our result extends in a straightforward manner to provide an $O(\log m)$ approximation in the case of *multiway* tests (corresponding to more than two outcomes) as studied in [20].

Generalized Optimal Decision Tree Our algorithm also extends to the setting when we do not have to uniquely identify the realized hypothesis i^*. Here we are given a threshold t such that it suffices to output a subset H^* of at most t hypotheses with $i^* \in H^*$. This can be handled easily by setting:

$$f_i(S) = \min\left\{ |\cup_{e\in S} T_e(i)| \cdot \frac{1}{m-t}, 1 \right\}, \quad \text{for all } S \subseteq U \text{ and } i \in [m].$$

Note that this time we will have $f_i(S) = 1$ if and only if at least $m - t$ hypotheses differ from i on at least one test in S; so this corresponds to having at most t possible hypotheses. The algorithm's stopping criterion here is the first point when the number of compatible hypotheses is at most t: again, this coincides with the point where f_{i^*} gets covered. And Theorem II.1 implies an $\mathcal{O}(\log m)$-approximation algorithm. To the best of our knowledge, this is the first approximation algorithm in this setting.

2.4.5 Equivalence Class Determination

This is an extension of ODT that was introduced to model noise in Bayesian active learning by [40]. As in ODT, there are m hypotheses with a probability distribution $\{p_i\}_{i=1}^m$ and binary tests where each test e has a positive outcome for hypotheses in Y_e. We are additionally given a partition Q of $[m]$. For each $i \in [m]$, let $Q(i)$ be the subset in the partition that contains i. The goal now is to minimize the expected cost of tests until we recognize the *part* of Q containing the realized hypothesis i^*.

We can model this as an ASR instance with tests as elements and hypotheses as scenarios.

The feedback functions are the same as in ODT. The submodular functions are:

$$f_i(S) = \frac{|\cup_{e\in S}(T_e(i)\cap Q(i)^c)|}{|Q(i)^c|}, \quad \text{for all } S\subseteq U \text{ and } i\in[m].$$

Above, $T_e(i)$ are as defined above for ODT and A^c denotes the complement of any set $A\subseteq[m]$. Note that f_is are monotone submodular with values between 0 and 1. Furthermore, $f_i(S)=1$ means that $Q(i)^c\subseteq\cup_{e\in S}T_e(i)$, which means that the set of compatible hypotheses based on the tests S is a subset of $Q(i)$. The algorithm's stopping criterion here is the first point when the set of compatible hypotheses is a subset of *any* $Q(i)$, which coincides with the point where f_{i^*} gets covered. Again, Theorem II.1 implies an $\mathcal{O}(\log m)$-approximation algorithm. This matches the best previous result of [26], and again our algorithm is much simpler.

2.4.6 Decision Region Determination

This is an extension of ODT that was introduced in order to allow for *decision making* in Bayesian active learning. As elaborated in [58], this problem has applications in robotics, medical diagnosis and comparison-based learning. Again, there are m hypotheses with a probability distribution $\{p_i\}_{i=1}^m$ and binary tests where each test e has a positive outcome for hypotheses in Y_e. In addition, there are a number of overlapping decision regions $D_j\subseteq[m]$ for $j\in[t]$. Each region D_j corresponds to the subset of hypotheses under which a particular decision $j\in[t]$ is applicable. The goal is to minimize the expected cost of tests so as to find some decision region D_j containing the realized hypothesis i^*. Following prior work, two additional parameters are useful for this problem: r is the maximum number of decision regions that contain a hypothesis and d is the maximum size of any decision region. Our main result here is:

Theorem II.15. *There is an* $\mathcal{O}(\log m+\min(d, r\log d))$*-approximation algorithm for decision region determination.*

This improves upon a number of previous papers on decision region determination (DRD). [58] obtained an $\mathcal{O}(\min(r,d)\cdot\log^2\frac{1}{\min_i p_i})$-approximation algorithm running in time exponential in $\min(r,d)$. Then, [25] obtained an $\mathcal{O}(r\cdot\log^2\frac{1}{\min_i p_i})$-approximation algorithm for this problem in polynomial time. The approximation ratio was later improved by [43] to $\mathcal{O}(\min(r,d)\cdot\log m)$ which however required time exponential in $\min(r,d)$. In contrast, our algorithm runs in polynomial time.

Before proving Theorem II.15, we provide two different algorithms for DRD.

Approach 1: an $O(r\log m)$-approximation algorithm for DRD. Here we model DRD as ASR with tests as elements and hypotheses as scenarios. The feedback functions are the same as in ODT. For each $i\in[m]$ and $j\in[t]$ such that $i\in D_j$ define $f_{i,j}(S)=\frac{|\bigcup_{e\in S}(T_e(i)\cap {D_j}^c)|}{|{D_j}^c|}$. Clearly $f_{i,j}$s are monotone submodular with values between 0 and 1. Also, $f_{i,j}(S)=1$ means that ${D_j}^c\subseteq\bigcup_{e\in S}T_e(i)$, which means that the set of compatible hypotheses based on the tests S is a subset of decision region D_j. However, we may stop when it is determined that the realized hypothesis is in *any one* of the decision regions. This criterion (for hypothesis i) corresponds to at least one $f_{i,j}(S)=1$ among $\{j: i\in D_j\}$. Using an idea from [45], we can express this criterion as a submodular-cover requirement. Define:

$$f_i(S)=1-\prod_{j:i\in D_j}(1-f_{i,j}(S)),\qquad \text{for all } S\subseteq U \text{ and } i\in[m].$$

One can verify that $f_i(S)=1$ if and only if $\exists j: i\in D_j$ and $f_{i,j}(S)=1$. The algorithm's stopping criterion is the first point when the set of compatible hypotheses is a subset of *any* decision region D_j, which coincides with the point where f_{i^*} gets covered. We can also see that f_i is monotone and submodular. Note that here the parameter ϵ is equal to $\min_i\prod_{j:i\in D_j}\frac{1}{|{D_j}^c|}$, which is much smaller than in previous applications. Still, we have $\epsilon=\Omega(m^{-r})$. So in this case, Theorem II.1 implies an $\mathcal{O}(r\log m)$-approximation algorithm where r is the maximum number of decision regions that contain a hypothesis.

Approach 2: an m-approximation algorithm for DRD. Here we use a simple greedy

splitting algorithm. At any state with compatible scenarios $H \subseteq [m]$ the algorithm selects the minimum cost element that splits H. Formally, it selects:

$$\arg\min\{c_e : e \in U \text{ with } H \cap Y_e \neq \emptyset \text{ and } H \cap Y_e^c \neq \emptyset\}.$$

The algorithm terminates when the compatible scenarios H is contained in any decision region.

As the number of compatible scenarios reduces by at least one after each chosen element, the depth of the algorithm's decision tree is at most m. Consider any depth $k \in \{1, \cdots m\}$ in this decision tree. Note that the states occurring at depth k induce a partition of all scenarios $I \subseteq [m]$ that are yet uncovered (at depth k). For each scenario $i \in I$, let $R_i \subseteq I$ denote all scenarios that are compatible with i at depth k, and let C_i denote the minimum cost of an element that splits R_i. Note that all scenarios i occurring at the same state at depth k will have the same R_i and C_i. Moreover, the k^{th} element chosen by the algorithm under any scenario $i \in I$ costs exactly C_i. So the algorithm's expected cost at depth k is exactly $\sum_{i \in I} p_i \cdot C_i$. The next claim shows that $OPT \geq \sum_{i \in I} p_i \cdot C_i$, which implies that the total expected cost of the algorithm is at most $m \cdot OPT$.

Claim II.15.1. The optimal cost of the DRD instance $OPT \geq \sum_{i \in I} p_i \cdot C_i$.

Proof. Consider any $i \in I$. Note that $R_i \subseteq I \subseteq [m]$ does not contain any decision region (otherwise i would have been covered before depth k which would contradict $i \in I$). So the optimal solution *must* select some element that splits R_i in its decision path for scenario i. As C_i is the minimum cost element that splits R_i, it follows that the optimal cost under scenario i is at least C_i. The claim now follows by taking expectations. □

Proof of Theorem II.15. This algorithm involves two phases. The first phase runs the $O(\log m)$-

approximation algorithm for *generalized ODT* (Section 2.4.4) on the given set of scenarios and elements with threshold d (this step ignores the decision regions). Crucially, the optimal value of this generalized ODT instance is at most that of the DRD instance. This follows simply from the fact that every decision region has size at most d: so the number of compatible scenarios at the end of any feasible DRD solution is always at most d. So the expected cost in the first phase is $O(\log m) \cdot OPT$. At the end of this phase, we will be left with a set M of at most d candidate scenarios and we still need to identify a valid decision region within that set. Let $\{M_1, \cdots M_s\}$ denote the partition of the m scenarios corresponding to the states at the end of the generalized ODT algorithm. So we have $|M_k| \leq d$ for all $k \in [s]$.

Next, in the second phase, we run one of the above mentioned algorithms on the DRD instance conditioned on scenarios M. For any $k \in [s]$ let $\mathcal{I}_k$ denote the DRD instance restricted to scenarios M_k where probabilities are normalized so as to sum to one. Crucially,

$$\sum_{k=1}^{s} \left(\sum_{i \in M_k} p_i \right) OPT(\mathcal{I}_k) \leq OPT, \tag{2.22}$$

where OPT is the optimal value of the original DRD instance. (2.22) follows directly by using the optimal tree for the original DRD instance as a feasible solution for each instance $\mathcal{I}_1, \cdots \mathcal{I}_s$.

Note that the DRD instance in the second phase always has at most d scenarios as $\max_{k=1}^{s} |M_k| \leq d$. So the two algorithms above have approximation ratios of $O(r \log d)$ and d respectively on this instance. Combined with (2.22) it follows that the expected cost in the second phase is $O(\min\{r \log d, d\}) \cdot OPT$. Adding the cost over both phases proves the theorem. □

2.4.7 Stochastic Knapsack Cover

In the knapsack cover problem, there are n elements, each with a cost and reward. We are also given a target W and our goal is to choose a subset of elements with minimum total cost such that the total reward is at least W. [54] gave a fully polynomial time approximation scheme for this problem. Here we consider a stochastic version of this problem where rewards are random and correlated across elements. Previously, [29] considered the case of independent rewards, and obtained a 3-approximation algorithm. We assume an explicit scenario-based distribution for the rewards. Formally, there are m scenarios where each scenario $i \in [m]$ occurs with probability p_i and corresponds to element rewards $\{r_i(e)\}_{e=1}^{n}$. We also assume that all rewards are integers between 0 and W. An algorithm knows the precise reward of an element $e \in [n]$ only upon selecting e. The goal is to adaptively select a sequence of elements so as to achieve total reward at least W, at minimum expected cost.

To model this problem as an instance of ASR, elements and scenarios are as described above. The feedback values are $G = \{0, 1, ..., W\}$ and the feedback functions are the rewards $r_i(\cdot)$ under each scenario $i \in [m]$. The submodular functions are $f_i(E) = \min(1, \frac{1}{W} \cdot \sum_{e \in E} r_i(e))$, where $r_i(e)$ is the reward of element e under scenario i. Note that $f_i(E) = 1$ if and only if the total reward of elements in E is at least W, which is also used as the stopping criterion for the algorithm. The parameter ϵ would be equal to $w/W \geq 1/W$, where w is the minimum positive reward. Using Theorem II.1, we obtain an $\mathcal{O}(\log m + \log \frac{W}{w})$-approximation algorithm.

We note that in the more general black-box distribution model (where we can only access the reward distribution through samples), there are hardness results that rule out any sub-polynomial approximation ratio by polynomial-time algorithms.

2.4.8 Scenario Submodular Cover

This was studied recently by [43] as a way to model correlated distributions in stochastic submodular cover.

We have a set U of elements with costs $\{c_e\}_{e \in U}$. Each element when selected, provides a random feedback from a set G: the feedback is correlated across elements. We are given a scenario-based distribution of elements' feedback values. There are m scenarios with probabilities $\{p_i\}_{i=1}^{m}$, from which the realized scenario i^* is drawn. Each scenario $i \in [m]$ specifies the feedback $r_i(e) \in G$ for each element $e \in U$. Let $*$ denote an unknown feedback value. There is also a "state based" utility function $f : (G \cup \{*\})^U \to \mathbb{Z}_{\geq 0}$ and an integer target Q. The function f is said to be covered if its value is at least Q. The goal is to (adaptively) select a sequence of elements so as to cover f at the minimum expected cost.

It is assumed f is monotone and submodular: as f is not a usual set function, one needs to extend the notions of monotonicity and submodularity to this setting. For any $g, g' \in (G \cup \{*\})^U$, we say g' is an *extension* of g and write $g' \succcurlyeq g$ if ${g'}_e = g_e$ for all $e \in U$ with $g_e \neq *$. For any $g \in (G \cup \{*\})^U$, $e \in U$ and $r \in G$, define $g_{e \leftarrow r}$ to be the vector which is equal to g on all coordinates $U \setminus \{e\}$ and has value r in coordinate e. Now, we say f is:

- *monotone* if receiving a feedback does not decrease its value, i.e. $f(g') \geq f(g)$ for all $g' \succcurlyeq g$.
- *submodular* if $f(g') - f({g'}_{e \leftarrow r}) \leq f(g) - f(g_{e \leftarrow r})$ for all $g' \succcurlyeq g$, $r \in G$ and $e \in U$ with ${g'}_e = *$.

For any subset $S \subseteq U$ and scenario $i \in [m]$, define $x(S, i) \in (G \cup \{*\})^U$ as:

$$x(S,i)_e = \begin{cases} r_i(e) & \text{if } e \in S \\ * & \text{if } e \in U \setminus S \end{cases}.$$

Note that function f is covered by subset $S \subseteq U$ if and only if $f(x(S, i^*)) \geq Q$.

We can model scenario submodular cover as an ASR instance with elements, scenarios and feedback as above. The submodular functions are $f_i(S) = \frac{1}{Q} \cdot \min\{f(x(S,i)), Q\}$ for all $S \subseteq U$ and $i \in [m]$. It can be seen that each f_i is monotone submodular (in the usual set function definition). Moreover, the parameter $\epsilon \geq 1/Q$ because function f is assumed to be integer-valued. The algorithm's stopping criterion is as follows. If S denotes the set of selected elements and $\theta_e \in G$ the feedback from each $e \in S$ then we stop when $f(\theta) \geq Q$ where $\theta_e = *$ for all $e \in U \setminus S$. Clearly, this is the same point when f_{i^*} reaches one.

So Theorem II.1 implies an algorithm with approximation ratio of $\mathcal{O}(\log m + \log \frac{1}{\epsilon})$, which is at least as good as the $\mathcal{O}(\log m + \log Q)$ bound in [43]. We might have $\frac{1}{\epsilon} \ll Q$ for some functions f, in which case our approximation ratio is slightly better than the previous one.

2.4.9 Adaptive Traveling Salesman Problem

This is a stochastic version of the basic TSP that was studied in [49]. We are given a metric $(U \cup \{s\}, d)$ where s is a root vertex, and there is demand at some random subset $S^* \subseteq U$ of vertices. The demand distribution is scenario-based: each scenario $i \in [m]$ occurs with probability p_i and has demand subset $S^* = S_i$. We get to know whether $u \in S^*$ or not upon visiting vertex $u \in U$. The goal is to build an adaptive tour originating from s that visits all the demands S^* at minimum expected distance.

As described in [49] it suffices to solve the related "isolation problem" where one wants to *identify* the realized scenario i^* at minimum expected distance and then use an approximate TSP to visit S_{i^*}. The isolation problem, which can be viewed as the metric version of ODT, can be modeled as adaptive submodular routing (ASP) by considering vertices as elements and scenarios as above. The feedback values are $G = \{0, 1\}$, and the feedback function is $r_i(e) = \mathbb{1}(e \in S_i)$ for all $e \in U$ and $i \in [m]$. The submodular functions are exactly the same as for the ODT problem (§2.4.4) where tests correspond to vertices: for each test $e \in U$, we use $Y_e = \{i \in [m] : e \in S_i\}$. Recall that parameter ϵ is equal to $1/m$. So Corollary II.9

implies an $\mathcal{O}(\log m \cdot \log^{2+\delta} n)$-approximation algorithm. This almost matches the best result known which is an $\mathcal{O}(\log^2 n \log m)$-approximation algorithm by [49].

Adaptive k-Traveling Salesman Problem. The input here is the same as adaptive TSP with an additional number k, and the goal is to minimize the expected distance taken to cover any k vertices of the demand subset S^*. As for adaptive TSP, we can model this problem as an instance of ASP. The only difference is in the definition of the submodular functions, which are now $f_i(T) = \frac{\min(|T \cap S_i|, k)}{k}$ for $T \subseteq U$ and $i \in [m]$. The algorithm stops at the first point when it has visited k demand vertices, which is the same as f_{i^*} getting covered. Here, parameter $\epsilon = 1/k$ and Corollary II.9 implies an $\mathcal{O}((\log m + \log k) \cdot \log^{2+\delta} n)$-approximation algorithm. To the best of our knowledge, this is the first approximation algorithm for this problem.

2.4.10 Adaptive Traveling Repairman Problem

This is a stochastic version of the traveling repairman problem (TRP) which was also studied in [49]. The setting is the same as adaptive TSP, but the objective here is to minimize the expected sum of distances to reach the demand vertices S^*.

We now show that this can also be viewed as a special case of ASP. Let $\mathcal{J}$ be a given instance of adaptive TRP with metric $(U \cup \{s\}, d)$, root s and demand scenarios $\{S_i \subseteq U\}_{i=1}^m$ with probabilities $\{p_i\}_{i=1}^m$. Let $q = \sum_{i=1}^m p_i |S_i|$. We create an instance $\mathcal{I}$ of ASP with elements U, $\sum_{i=1}^m |S_i|$ scenarios and feedback values $G = \{0, 1\}$. For each $i \in [m]$ and $e \in S_i$ we define scenario $h_{e,i}$ as follows:

- $h_{e,i}$ has probability of occurrence p_i/q.
- the submodular function $f_{e,i}(T) = |\{e\} \cap T|$ for $T \subseteq U$.
- $r_{e,i}(e') = \mathbb{1}(e' \in S_i)$ for $e' \in U$.

Note that the total probability of these $\sum_{i=1}^{m} |S_i|$ scenarios is one. The idea is that covering scenario $h_{e,i}$ in $\mathcal{I}$ corresponds to visiting vertex e when the realized scenario in $\mathcal{J}$ is i. Note that for any $i \in [m]$, the feedback functions for all the scenarios $\{h_{e,i} : e \in S_i\}$ are identical.

Claim II.15.2. $OPT(\mathcal{I}) = \frac{1}{q} \cdot OPT(\mathcal{J})$.

Proof. Consider an optimal solution R to the adaptive TRP instance $\mathcal{J}$. For each scenario $i \in [m]$, let τ_i denote the tour (originating from s) traced by R; note that τ_i visits every vertex in S_i, and let $C_{e,i}$ denote the distance to vertex $e \in S_i$ along τ_i. So $OPT(\mathcal{J}) = \sum_{i=1}^{m} p_i \sum_{e \in S_i} C_{e,i}$. We can also view R as a potential solution for the ASP instance $\mathcal{I}$. To see that this is a feasible solution, note that the tour traced by R under scenario $h_{e,i}$ (for any $i \in [m]$ and $e \in S_i$) is precisely the prefix of τ_i until vertex e, at which point the tour returns to s. So every scenario in $\mathcal{I}$ is covered. Moreover, the expected cost of R for $\mathcal{I}$ is exactly $\sum_{i=1}^{m} \sum_{e \in S_i} \frac{p_i}{q} C_{e,i} = \frac{1}{q} \cdot OPT(\mathcal{J})$. This shows that $OPT(\mathcal{I}) \leq \frac{1}{q} \cdot OPT(\mathcal{J})$.

Now, consider an optimal solution R' to the ASP instance $\mathcal{I}$. For each scenario $h_{e,i}$ (with $i \in [m]$ and $e \in S_i$), let $\sigma_{e,i}$ denote the tour (originating from s) traced by R' and let $\tau_{e,i}$ denote the shortest prefix of $\sigma_{e,i}$ that covers $f_{e,i}$. Let $C'_{e,i}$ denote the cost of the walk $\tau_{e,i}$, which is the cost under scenario $h_{e,i}$. So $OPT(\mathcal{I}) = \sum_{i=1}^{m} \sum_{e \in S_i} \frac{p_i}{q} C'_{e,i}$ Note that for each $i \in [m]$, the tours $\{\sigma_{e,i} : e \in S_i\}$ are identical (call it σ_i) because the feedback obtained under scenarios $\{h_{e,i} : e \in S_i\}$ are identical. So the walks $\{\tau_{e,i} : e \in S_i\}$ must be nested. We now view R' as a potential solution for the adaptive TRP instance $\mathcal{J}$. To see that this is feasible, note that the tour traced under scenario $i \in [m]$ is precisely σ_i which visits all vertices in S_i. Moreover, due to the nested structure of the walks $\{\tau_{e,i} : e \in S_i\}$, the distance

to any vertex $e \in S_i$ under scenario i is exactly $C'_{e,i}$. So the expected cost of R' for $\mathcal{J}$ is $\sum_{i=1}^{m} p_i \sum_{e \in S_i} C'_{e,i} = q \cdot OPT(\mathcal{I})$. This shows that $OPT(\mathcal{J}) \leq q \cdot OPT(\mathcal{I})$.

Combining the above two bounds, we obtain $OPT(\mathcal{I}) = \frac{1}{q} \cdot OPT(\mathcal{J})$ as desired.

□

Moreover, $\epsilon = 1$ for this ASP instance. Hence, Corollary II.9 implies an $\mathcal{O}(\log m \cdot \log^{2+\delta} n)$-approximation algorithm for adaptive TRP. Again, this almost matches the best result known for this problem which is an $\mathcal{O}(\log^2 n \log m)$-approximation algorithm by [49]. While our approximation ratios for adaptive TSP and TRP are slightly worse than those in [49], we obtain these results as direct applications of more general framework (ASP) with very little problem-specific work.

2.5 Experiments

We present experimental results for the Adaptive Multiple Intents Re-ranking (Adaptive MIR), Optimal Decision Tree (ODT) and Generalized ODT problems. We use expected number of elements as the objective, i.e. all costs are unit. The main difference between ODT and Generalized ODT is in the stopping criteria, which makes their coverage functions (f_is) different. Recall that in ODT, our goal is to uniquely identify the realized scenario. As discussed in Section 2.4.4:

$$f_i(S) = |\cup_{e \in S} T_e(i)| \cdot \frac{1}{m-1}, \tag{2.23}$$

where $T_e(i)$ is the set of all scenarios which have a different outcome from scenario i on test e. On the other hand, for Generalized ODT, we satisfy the scenario as soon as the number of compatible scenarios is at most t, for some input parameter t. Here we have:

$$f_i(S) = \min\left\{|\cup_{e \in S} T_e(i)| \cdot \frac{1}{m-t}, 1\right\} \tag{2.24}$$

Furthermore, recall that as stated in Section 2.4.2, in Adaptive MIR the submodular function for scenario $i \in [m]$ over set S is defined as:

$$f_i(S) = \min(|S \cap S_i|, K_i)/K_i \tag{2.25}$$

Where S_i is the interest set of user i and K_i is the minimum number of relevant results that this user wants to see to get covered.

2.5.1 Datasets

Real-world Dataset: For our experiments we used two real-world datasets. For Adaptive MIR, we use a data set called ML-100, which is the 100K example from the MovieLens [52] repository. It contains 100,000 ratings on a scale of [1,5] from 943 users (scenarios) on 1682 movies (elements) where each user has rated ≥ 20 movies. We binarized this dataset by setting all ratings < 3 to 0, which left us with $82,520$ ratings, where the average user had 87.5 ratings with a standard deviation of 81.2, which suggests a highly-skewed distribution. Now, we can consider each user as a scenario and each movie as an element. Furthermore, the subset associated with a scenario is equivalent to the subset of movies that the corresponding user has rated more than 2.

We use another data set for ODT and Generalized ODT, which is called WISER [1]. It contains information related to 79 binary symptoms (corresponding to elements in ODT) for 415 chemicals (equivalent to scenarios in ODT) which is used in the problem of toxic chemical identification of someone who has been exposed to these chemicals. This dataset has been used for testing algorithms for similar problems in other papers, eg. [11], [12] and [16]. For each symptom-chemical pair the data specifies whether/not that symptom is seen for that chemical. However the WISER data has 'unknown' entries for some pairs. In order to obtain instances for ODT from this, we generated 10 different datasets by assigning

[1] http://wiser.nlm.nih.gov/

random binary values to the 'unknown' entries. Then we removed all identical scenarios: otherwise ODT would not be feasible.

As probability distributions, we used permutations of the power-law distribution ($Pr[X = x] = Kx^{\alpha}$) for $\alpha = 0, -1$ and -2. For ODT and Generalized ODT, we use the power-law distribution with $\alpha = -1/2$ as well. Also, to be able to compare results meaningfully in these two problems, the same permutation was used for each α across all 10 datasets.

Synthetic Dataset: For ODT and Generalized ODT, we also used a synthetic dataset — SYN-K — that is parameterized by k; this is based on a hard instance for the greedy algorithm [63]. Given k, this instance has $m = 2k + 1$ scenarios and $n = k + 2$ elements as follows:

$$\begin{cases} \text{Scenario } i \in [1, k] \text{ has positive feedback on element } i \text{ and } k+1 \text{ and negative on the others.} \\ \text{Scenario } i \in [k+1, 2k] \text{ has positive feedback on element } i-k \text{ and } k+2 \text{ and negative on the others.} \\ \text{Scenario } 2k+1 \text{ has negative feedback on all elements.} \end{cases}$$

Also, the probabilities for the scenarios are as follows:

$$p_i = p_{i+k} = 2^{-i-2} \text{ for } i \in [1, k-1], \quad p_k = p_{2k} = 2^{-k-1} \quad \text{and} \quad p_{2k+1} = 2^{-1}.$$

2.5.2 Algorithms

In our experiments, we compare and contrast the results of four different algorithms for each problem, three of which are common for all the problems:

- **ASR:** Our algorithm that uses the objective described in (2.2) with corresponding f_is described in equations (2.25), (2.23) and (2.24), for Adaptive MIR, ODT and Generalized ODT respectively.
- **Static:** This is the algorithm from [6]. This algorithm is not feedback dependent and uses a measure which is similar to the second term in our measure (2.2). More specifically, this algorithm at each iteration chooses an element e that maximizes:

$$\sum_{i \in H} p_i \cdot \frac{f_i(e \cup E) - f_i(E)}{1 - f_i(E)}$$

with corresponding f_is for each problem, described in equations (2.25), (2.23) and (2.24).

- **AdStatic** This is a modified version of the aforementioned Static algorithm. It uses the observed feedback to skip redundant elements that have the same outcome on all the uncovered compatible scenarios.

For Adaptive MIR, we also perform the following algorithm:

- **Clustering:** This algorithm uses k-Means [5] to *a priori* partition U into 10 clusters. Each cluster $c_j, j \in [1, 10]$ is initially given a weight $w_{c_j} = 1$. To choose the next element $e \in U \setminus E$, a cluster $j \in [1, 10]$ is first chosen by sampling non-uniformly according to w_{c_j}. Next, an element $e \in c_j$ is chosen uniformly at random. If $e \in S_{i^*}$, the c_j is rewarded by setting $w_{c_j} = 2w_{c_j}$, else c_j is penalized by setting $w_{c_j} = 0.5w_{c_j}$.

For ODT and Generalized ODT, use the following algorithm as our fourth algorithm:

- **Greedy:** This is a classic greedy algorithm described in [63], [27], [1], [20], [44]. At each iteration, it chooses the element which keeps the decision tree as balanced as possible. More formally at each state (E, H) we choose an element $e \in U \setminus E$ that minimizes:

$$|\Pr(i \in H : r_i(e) = 1) - \Pr(i \in H : r_i(e) = 0)|$$

  While the rule is the same for ODT and Generalized ODT, the set of uncovered compatible scenarios may be different, which affects the sequence of chosen elements.

2.5.3 Results

The performance of these algorithms are reported in the tables below. We show normalized costs which is the actual cost divided by the minimum cost over all algorithms to show the differences better. The best algorithm is marked bold. For ODT and Adaptive MIR, we also report (as "Best cost") the actual minimum cost over the four algorithms.

Alg \ K_i	$\lvert S_i\rvert$	$[\lvert S_i\rvert/2, \lvert S_i\rvert)$	$[\lvert S_i\rvert/4, \lvert S_i\rvert)$	$[1, \lvert S_i\rvert/2)$	$[1, \lvert S_i\rvert/4)$
ASR	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Clustering	1.100	1.132	1.143	1.359	1.584
Static	10.080	5.497	5.000	3.216	2.666
AdStatic	1.013	1.017	1.020	1.042	1.073
Bestcost	92.50	70.38	60.98	26.21	14.48

Table 2.2: Costs for Adaptive MIR on ML-100 dataset for uniform distribution and some threshold K_i, which is randomly chosen from the specified interval.

Alg \ Perm	1	2	3
ASR	**1.000**	**1.000**	**1.000**
Clustering	1.116	1.095	1.103
Static	10.500	10.079	10.161
AdStatic	1.018	1.018	1.019
Bestcost	86.20	92.72	91.23

Table 2.3: Costs for Adaptive MIR on ML-100 dataset for power law distribution with $\alpha = -2$, and $K_i = |S_i|$.

Algorithm \ Dataset	1	2	3	4	5	6	7	8	9	10
ASR	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Greedy	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Static	1.179	1.189	1.180	1.211	1.190	1.191	1.218	1.166	1.193	1.203
AdStatic	1.035	1.038	1.033	1.036	1.033	1.033	1.043	1.035	1.032	1.036
Best cost	8.704	8.719	8.717	8.706	8.713	8.742	8.717	8.697	8.723	8.736

Table 2.4: Normalized costs for ODT with uniform distribution

Adaptive MIR: The results for this part is summarized in Table 2.2 and 2.3. As we can see, ASR consistently performs better than all other algorithms in this setting, with AdStatic is the second best. The performance of Static algorithm, which is significantly worse than its counterparts demonstrates the importance of *adaptive* algorithms. Table 2.3 shows the performance when $K_i = |S_i|$ and the scenarios are drawn from power-law distributions with $\alpha = -2$. For $\alpha = -2$, three random permutations are used to test the stability of the expected cost of each algorithm. The instability of the expected cost across permutations of the scenario distributions is indicative of the inherent skew in the dataset. Still, ASR consistently outperforms the other three algorithms.

Algorithm \ Dataset	1	2	3	4	5	6	7	8	9	10
ASR	1.001	1.002	1.001	1.003	1.002	1.003	1.001	1.003	1.001	1.003
Greedy	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Static	1.203	1.207	1.193	1.231	1.214	1.191	1.233	1.222	1.262	1.222
AdStatic	1.069	1.063	1.065	1.059	1.066	1.058	1.067	1.063	1.071	1.069
Best cost	8.415	8.427	8.429	8.400	8.422	8.449	8.419	8.403	8.431	8.449

Table 2.5: Normalized costs for ODT with power-law distribution $\alpha = -1/2$

Algorithm \ Dataset	1	2	3	4	5	6	7	8	9	10
ASR	1.038	1.051	1.010	**1.000**	1.005	**1.000**	1.024	1.027	1.041	1.006
Greedy	**1.000**	**1.000**	**1.000**	1.008	**1.000**	1.005	**1.000**	**1.000**	**1.000**	**1.000**
Static	1.308	1.361	1.320	1.320	1.284	1.336	1.335	1.345	1.339	1.383
AdStatic	1.199	1.250	1.193	1.209	1.149	1.195	1.198	1.237	1.187	1.237
Best cost	7.097	7.075	7.214	7.082	7.302	7.398	7.048	7.099	7.156	7.122

Table 2.6: Normalized costs for ODT with power-law distribution $\alpha = -1$

Algorithm \ Dataset	1	2	3	4	5	6	7	8	9	10
ASR	1.118	1.153	1.011	1.116	**1.000**	**1.000**	**1.000**	1.112	1.124	**1.000**
Greedy	**1.000**	**1.000**	**1.000**	**1.000**	1.050	1.193	1.096	**1.000**	**1.000**	1.011
Static	1.684	1.271	1.435	1.397	1.136	1.336	1.867	1.328	1.548	1.531
AdStatic	1.624	1.235	1.414	1.366	1.112	1.293	1.604	1.269	1.468	1.364
Best cost	3.721	4.085	4.753	4.149	5.884	4.195	4.267	4.373	4.224	4.952

Table 2.7: Normalized costs for ODT with power-law distribution $\alpha = -2$

ODT: Table 2.4 shows the expected costs of these algorithms for the ODT problem with uniform distribution. It turns out ASR and Greedy algorithms have the same cost for all datasets, while they both outperform Static and AdStatic. Table 2.5 shows the results when we have power-law distribution with $\alpha = -1/2$. Greedy does slightly better than ASR on all instances; both Greedy and ASR are much better than Static and AdStatic. Table 2.6 has the results for power-law distribution with $\alpha = -1$. Both Greedy and ASR still outperform Static and AdStatic on all instances. ASR achieves the best solution on 2 out of 10 instances, whereas Greedy is the best on the others. Table 2.7 is for power-law distribution with $\alpha = -2$. Here, ASR is the best on 4 out of 10 instances, and again both greedy and ASR outperform Static and AdStatic.

Alg \ Th	1	2	3	4	5
ASR	**1.000**	**1.000**	**1.000**	**1.000**	**1.001**
Greedy	**1.000**	**1.000**	**1.000**	**1.000**	**1.001**
Static	1.192	1.088	1.111	1.061	1.008
AdStatic	1.035	1.040	1.088	1.050	1.003

Table 2.8: Average cost for Generalized ODT with uniform distribution

Alg \ Th	1	2	3	4	5
ASR	1.003	**1.000**	**1.000**	**1.000**	1.004
Greedy	**1.000**	1.005	1.010	1.007	**1.002**
Static	1.218	1.126	1.084	1.084	1.054
AdStatic	1.065	1.075	1.060	1.068	1.050

Table 2.9: Average cost for Generalized ODT with power-law distribution $\alpha = -1/2$

Alg \ Th	1	2	3	4	5
ASR	1.020	1.010	**1.004**	1.085	1.064
Greedy	**1.001**	**1.004**	1.010	**1.000**	**1.000**
Static	1.333	1.213	1.177	1.120	1.111
AdStatic	1.205	1.176	1.163	1.113	1.108

Table 2.10: Average cost for Generalized ODT with power-law distribution $\alpha = -1$

Alg \ Th	1	2	3	4	5
ASR	1.063	1.048	1.074	**1.041**	**1.043**
Greedy	**1.035**	**1.038**	**1.045**	1.058	1.059
Static	1.453	1.356	1.324	1.285	1.258
AdStatic	1.375	1.342	1.315	1.282	1.256

Table 2.11: Average cost for Generalized ODT with power-law distribution $\alpha = -2$

Generalized ODT: For these tests, we report the average (normalized) costs for each distribution and threshold. Each entry is an average over the 10 datasets. Table 2.8 is for the uniform distribution, Table 2.9 is for power-law $\alpha = -1/2$, Table 2.10 is for power-law $\alpha = -1$ and Table 2.11 is for power-law $\alpha = -2$. ASR performs the best in about half the settings, and Greedy is the best in the others. Note that the best average-number is more than 1 in some cases: this shows that the corresponding algorithm was not the best on all 10 datasets. As for ODT, we see that both ASR and Greedy are better than Static and AdStatic in all cases.

Results on synthetic data: Table 2.12 shows the results on the synthetic instances. ASR and AdStatic have the best result simultaneously, and Greedy's performance is much worse. It is somewhat surprising that even Static performs much better than Greedy.

Dataset	SYN-50			SYN-100			SYN-150			SYN-200		
Alg \ Th	1	3	5	1	3	5	1	3	5	1	3	5
ASR	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Static	1.09	1.09	1.09	1.09	1.09	1.09	1.09	1.09	1.09	1.09	1.09	1.09
AdStatic	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Greedy	9.64	9.46	9.27	18.73	18.55	18.36	27.82	27.64	27.46	36.91	36.73	36.55

Table 2.12: Normalized cost for ODT (Threshold = 1) and Generalized ODT on SYN-K.

Summary: For Adaptive MIR, ASR consistently does better than all the other algorithms, and AdStatic is the second best. For ODT and Generalized ODT, both ASR and

Greedy perform well on the real dataset and the difference in their objectives is typically small. The largest gaps were for ODT with power-law distribution $\alpha = -2$ (Table 2.7) where Greedy is 19% worse than ASR on data 6 and ASR is 15% worse than Greedy on data 2. Combined with the fact that Greedy performs poorly on worst-case instances (Table 2.12), ASR can be a good alternative for Greedy in practice.

We also observe that it is important to use adaptive algorithms, as Static consistently performs the worst in all cases. For ODT, static is on average 30% worse than the best algorithm, and for Generalized ODT it is on average 18% worse. In Adaptive MIR, Static performance is highly dependent on the threshold and it can change from being 206% to more than 10 times worse (1050%) than ASR.

CHAPTER III

Optimal Decision Tree with Noisy Outcomes

The results in this chapter appear in [59].

3.1 Introduction

The classic optimal decision tree (ODT) problem involves identifying an initially unknown hypothesis $\bar{x}$ that is drawn from a known probability distribution over a set of m possible hypotheses. We can perform tests in order to distinguish between these hypotheses. Each test produces a binary outcome (positive or negative) and the precise outcome of each hypothesis-test pair is known beforehand.[1] So an instance of ODT can be viewed as a ± 1 matrix with the hypotheses as rows and tests as columns. The goal is to identify hypothesis $\bar{x}$ using the minimum number of tests in expectation.

As a motivating application, consider the following task in medical diagnosis [67]. A doctor needs to diagnose a patient's disease by performing tests. Given an *a priori* probability distribution over possible diseases, what sequence of tests should the doctor perform? Another application is in active learning [27]. Given a set of data points, one wants to learn a classifier that labels the points correctly as positive and negative. There is a set of m possible classifiers which is assumed to contain the true classifier. In the Bayesian setting, which we consider, the true classifier is drawn from some known probability distribution. The goal is

[1] We consider binary test outcomes only for simplicity: our results also hold for finitely many outcomes.

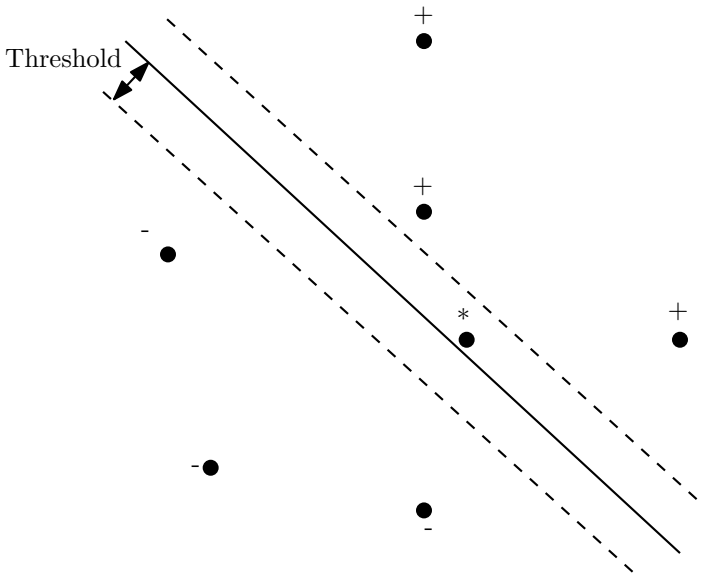

Figure 3.1: A binary classifier with a specified threshold. The label * can be either + or -.

to identify the true classifier by querying labels at the minimum number of points (in expectation). Other applications include entity identification in databases [20] and experimental design to choose the most accurate theory among competing candidates [40].

An important issue that is not considered in the classic ODT model is that of unknown or noisy outcomes. In fact, our research was motivated by a dataset involving toxic chemical identification where the outcomes of many hypothesis-test pairs are stated as unknown (one of our experimental results is also on this dataset). We might need to consider noise in Bayesian active learning to have a more robust model. For example, when we have linear classifiers the label of the data points that fall very close to the boundaries of the classifiers can change with a small perturbation. To avoid this, we can consider a threshold for the minimum distance of the data points from a classifier to have an accurate label. So if a data point's distance to the true classifier is less than that threshold, the label can be unknown (See Figure 3.1). Prior work incorporating noise in ODT [40] is restricted to settings with very sparse noise. In this chapter, we design approximation algorithms for the noisy optimal decision tree problem in full generality.

We consider a standard model for persistent noise. Some outcomes (i.e., entries in the hypothesis-test matrix) are random with a known distribution: for simplicity we treat each noisy outcome as an unbiased ± 1 random variable. Our results extend directly to the case when each noisy outcome has a different probability of being ± 1. Persistent noise means that repeating the same test always produces the same ± 1 outcome. We assume that the instance is identifiable, i.e., a unique hypothesis can always be identified irrespective of the noisy outcomes. (This assumption can be relaxed: see §3.5.)

We consider both non-adaptive policies (where the test sequence is fixed upfront) and adaptive policies (where the test sequence is built incrementally and depends on observed test outcomes). Clearly, adaptive policies perform at least as well as non-adaptive ones.[2] However, non-adaptive policies are very simple to implement (requiring minimal incremental computation) and may be preferred in time-sensitive applications. Our main contributions are:

- an $O(\log m)$-approximation algorithm for non-adaptive ODT with noise.
- an $O(\min(h, r) + \log m)$-approximation algorithm for adaptive ODT with noise, where h (resp. r) is the maximum number of noisy outcomes for any hypothesis (resp. test).
- an $O(\log m)$-approximation algorithm for adaptive ODT with noise when every test has at least $m - O(\sqrt{m})$ noisy outcomes.
- experimental results on applications to toxic chemical identification and active learning.

We note that both non-adaptive and adaptive versions (even for usual ODT) generalize the set cover problem: so an $\Omega(\log m)$ approximation ratio is the best possible (unless P=NP).

Related Work The optimal decision tree problem (without noise) has been extensively studied for several decades [34, 53, 67, 63, 1, 3, 21, 49]. The state-of-the-art result [49] is an

[2]There are also instances where the relative gap between the best adaptive and non-adaptive policies is $\tilde{\Omega}(m)$.

$O(\log m)$-approximation, for instances with arbitrary probability distribution and costs. It is also known that ODT cannot be approximated to a factor better than $\Omega(\log m)$, unless P=NP [20].

The application of ODT to Bayesian active learning was formalized in [27]. There are also several results on the *statistical complexity* of active learning. e.g. [8, 51, 72], where the focus is on proving bounds for structured hypothesis classes. In contrast, we consider arbitrary hypothesis classes and obtain *computationally efficient* policies with provable approximation bounds relative to the optimal (instance specific) policy. This approach is similar to that in [27, 44, 38, 40, 26, 58].

The noisy ODT problem was studied previously in [40]. Using a connection to adaptive-submodularity [38], they obtained an $O(\log^2 \frac{1}{p_{min}})$-approximation algorithm for noisy ODT in the presence of very few noisy outcomes; here $p_{min} \leq \frac{1}{m}$ is the minimum probability of any hypothesis.[3] In particular, the running time of the algorithm in [40] is exponential in the number of noisy outcomes per hypothesis, which is polynomial only if this number is at most logarithmic in the number of hypotheses/tests. Our result provides the following improvements (i) the running time is polynomial irrespective of the number of noisy outcomes and (ii) the approximation ratio is better by at least one logarithmic factor. We note that a better $O(\log m)$ approximation ratio (still only for very sparse noise) follows from subsequent work on the "equivalence class determination" problem by [26]. For this setting, our result is also an $O(\log m)$ approximation, but the algorithm is simpler. More importantly, ours is the first result that can handle *any* number of noisy outcomes.

Other variants of noisy ODT have also been considered, e.g. [69, 11, 24], where the goal is to identify the correct hypothesis with at least some target probability. The theoretical results in [24] provide "bicriteria" approximation bounds where the algorithm has a larger

[3]The paper [40] states the approximation ratio as $O(\log 1/p_{min})$ because it relied on an erroneous claim in [38]. The correct approximation ratio, based on [70, 39], is $O(\log^2 1/p_{min})$.

error probability than the optimal policy. Our setting is different because we require *zero* probability of error.

Many algorithms for ODT (including ours) rely on some underlying submodularity properties. We briefly survey some background results. The basic submodular cover problem was first considered by [87], who proved that the natural greedy algorithm is a $(1 + \ln \frac{1}{\epsilon})$-approximation algorithm, where ϵ is the minimal positive marginal increment of the function. [6] obtained an $O(\log \frac{1}{\epsilon})$-approximation algorithm for the submodular ranking problem, that involves simultaneously covering multiple submodular functions; [55] extended this result to also handle costs. [71] studied an adaptive version of the submodular ranking problem, which we explain in Chapter II. We utilize results/techniques from these papers.

Finally, we note that there is also work on minimizing the *worst-case* (instead of average case) cost in ODT and active learning [68, 75, 46, 45]. These results are incomparable to ours because we are interested in the average case, i.e. minimizing expected cost.

3.2 Problem Definition

We start with defining the optimal decision tree with noise (ODTN) formally. There is a set of m possible hypotheses with a probability distribution $\{\pi_x\}_{x=1}^m$, from which an unknown hypothesis $\bar{x}$ is drawn. There is also a set $U = [n]$ of binary tests. Each test $e \in U$ is associated with a 3-way partition $T^+(e), T^-(e), T^*(e)$ of the hypotheses, where the outcome of test e is (a) positive if $\bar{x}$ lies in $T^+(e)$, (b) negative if $\bar{x} \in T^-(e)$, and (c) positive or negative with probability $\frac{1}{2}$ each if $\bar{x} \in T^*(e)$ (these are noisy outcomes). We assume that conditioned on $\bar{x}$, each noisy outcome is independent. We also use $r_x(e)$ to denote the part

of test e that hypothesis x lies in, i.e.

$$r_x(e) = \begin{cases} -1 & \text{if } x \in T^-(e) \\ +1 & \text{if } x \in T^+(e) \\ * & \text{if } x \in T^*(e) \end{cases}$$

While we know the 3-way partition $T^+(e), T^-(e), T^*(e)$ for each test $e \in U$ upfront, we are not aware of the actual outcomes for the noisy hypothesis-test pairs. It is assumed that the realized hypothesis $\bar{x}$ can be uniquely identified by performing all tests, regardless of the outcomes of $*$-tests. This means that for every pair $x, y \in [m]$ of hypotheses, there is some test $e \in U$ with $x \in T^+(e)$ and $y \in T^-(e)$ or vice-versa. The goal is to perform an adaptive (or non-adaptive) sequence of tests to identify hypothesis $\bar{x}$ using the minimum *expected* number of tests. Note that expectation is taken over both the prior distribution of $\bar{x}$ and the random outcomes of noisy tests for $\bar{x}$.

In our algorithms and analysis, it will be convenient to work with an **expanded set** of hypotheses M. For a binary vector $b \in \{\pm 1\}^U$ and hypothesis $x \in [m]$, we say b is consistent with x and denote $b \sim x$, if $b_e = r_x(e)$ for each $e \in U$ with $r_x(e) \neq *$. Let $M = \{(b, x) \in \{\pm 1\}^U \times [m] : b \sim x\}$, and $M_x \subseteq M$ be all copies associated with a particular hypothesis $x \in [m]$; note that $\{M_x\}_{x=1}^m$ is a partition of M. Each "expanded" hypothesis $(b, x) \in M$ corresponds to the case where the true hypothesis $\bar{x} = x$ and the test-outcomes are given by b. We assign the probability $q_{b,x} = \pi_x / 2^{h_x}$ to each $(b, x) \in M$, where h_x is the number of *-tests for x. Note that conditioned on $\bar{x} = x$, the probability of observing outcomes b is exactly 2^{-h_x}; so $\Pr[\bar{x} = x \text{ and test outcomes are } b] = q_{b,x}$. For any $(b, x) \in M$ and $e \in U$, define $r_{b,x}(e) = b(e)$ to be the observed outcome of test e if $\bar{x} = x$ and test-outcomes are b. For every expanded hypothesis $(b, x) \in M$ and test $e \in U$, define

$$T_{b,x}(e) = \begin{cases} T^+(e) & \text{if } r_{b,x}(e) = -1 \\ T^-(e) & \text{if } r_{b,x}(e) = +1 \end{cases}, \tag{3.1}$$

which is the subset of (original) hypotheses that can *definitely* be ruled-out based on test e if $\bar{x} = x$ and the test-outcomes are given by b. Note that hypotheses in $T^*(e)$ are never part of $T_{b,x}(e)$ as their outcome on test e can be positive/negative (so they cannot be ruled-out). For every hypothesis $(b, x) \in M$, define a monotone submodular function $f_{b,x} : 2^U \to [0, 1]$:

$$f_{b,x}(S) = |\bigcup_{e \in S} T_{b,x}(e)| \cdot \frac{1}{m-1}, \qquad \forall S \subseteq U, \tag{3.2}$$

which equals the fraction of the $m - 1$ hypotheses (excluding x) that have been ruled-out based on the tests in S if $\bar{x} = x$ and test-outcomes are given by b. Assuming $\bar{x} = x$ and test-outcomes are given by b, hypothesis x is uniquely identified after tests S if and only if $f_{b,x}(S) = 1$.

A **non-adaptive** policy is specified by just a permutation of tests. The policy performs tests in this sequence and eliminates incompatible hypotheses until there is a unique compatible hypothesis (which is $\bar{x}$). Note that the number of tests performed under such a policy is still random (depends on $\bar{x}$ and outcomes of noisy tests). An **adaptive** policy chooses tests incrementally, depending on prior test outcomes. The *state* of a policy is a tuple (E, d) where $E \subseteq U$ is a subset of tests and $d \in \{\pm 1\}^E$ denotes the observed outcomes on tests in E. An adaptive policy is specified by a mapping $\Phi : 2^U \times \{\pm 1\}^U \to U$ from states to tests, where $\Phi(E, d)$ is the next test to perform at state (E, d). Equivalently, we can view a policy as decision tree with nodes corresponding to states, labels at nodes representing the test performed at that state and branches corresponding to the ± 1 outcome at the current test. As the number of states can be exponential, we cannot hope to specify arbitrary adaptive policies. Instead, we want implicit policies Φ, where given *any* state (E, d), the test $\Phi(E, d)$ can be computed *efficiently*. This would imply that the total time taken under any outcome is polynomial. We note that an optimal policy Φ^* can be very complex and the map $\Phi^*(E, d)$ may not be efficiently computable. We will still compare the performance of our (efficient) policy to Φ^*.

In this chapter, we consider the **persistent noise** model. That is, repeating a test e with $\bar{x} \in T^*(e)$ always produces the same outcome. An alternative model is non-persistent noise, where each run of test e with $\bar{x} \in T^*(e)$ produces an independent random outcome. The persistent noise model is more appropriate to handle missing data. It also contains the non-persistent noise model as a special case (by introducing multiple tests with identical partitions). One can easily obtain an adaptive $O(\log^2 m)$-approximation algorithm for the non-persistent model that succeeds with high probability using existing algorithms for noiseless ODT [21] and repeating each test $O(\log m)$ times. The persistent-noise model that we consider is much harder.

3.3 Non-Adaptive Algorithm

Our algorithm is based on a reduction to the submodular ranking problem [6], defined below.

Submodular Function Ranking (SFR) An instance of SFR consists of a ground set U of elements and a collection of monotone submodular functions $\{f_1, ..., f_m\}$, $f_x : 2^U \to [0, 1]$, with $f_x(\emptyset) = 0$ and $f_x(U) = 1$ for all $x \in [m]$. Additionally, there is a weight $w_x \geq 0$ for each $x \in [m]$. A solution is a permutation of the elements U. Given any permutation σ of U, the **cover time** of function f is $C(f, \sigma) := \min\{t \mid f(\cup_{i\in[t]}\sigma(i)) = 1\}$ where $\sigma(i)$ is the i^{th} element in σ. In words, it is the earliest time when the value of f reaches the unit threshold. The goal is to find a permutation σ of $[n]$ with minimal weighted cover time $\sum_{x\in[m]} w(x) \cdot C(f_x, \sigma)$. We will use the following result:

Theorem III.1 ([6])**.** *There is an $O(\log \frac{1}{\epsilon})$-approximation for SFR where ϵ is minimum positive marginal increment of any function.*

The non-adaptive ODTN problem can be expressed as an instance of SFR as follows. The

elements are the tests U. For each hypothesis-copy $(b, x) \in M$ there is a function $f_{b,x}$ (see (3.2)) with weight $q_{b,x}$. Based on the definition of these functions, the parameter $\epsilon = \frac{1}{m-1}$. To see the equivalence, note that a solution to non-adaptive ODTN is also a permutation σ of U and hypothesis x is uniquely identified under outcome (b, x) exactly when function $f_{b,x}$ has value one. Moreover, the objective of the ODTN problem is the expected number of tests in σ to identify the realized hypothesis $\bar{x}$, which equals

$$\sum_{x=1}^{m} \pi_x \sum_{b \sim x} 2^{-h_x} \cdot C_{b,x}(\sigma) = \sum_{(b,x) \in M} q_{b,x} \cdot C_{b,x}(\sigma),$$

where $C_{b,x}(\sigma)$ is the cover-time of function $f_{b,x}$. It now follows that this SFR instance is equivalent to the non-adaptive ODTN instance. However, we cannot apply Theorem III.1 directly to obtain an $O(\log m)$ approximation. This is because we have an exponential number of functions (note that $|M|$ can be exponential in m), which means that a direct implementation of the algorithm from [6] requires exponential time. Nevertheless, we will show that a variant of the SFR algorithm can be used to obtain:

Theorem III.2. *There is an $O(\log m)$-approximation for non-adaptive ODTN.*

The SFR algorithm [6] is a greedy-style algorithm that at any point, having already chosen tests E, assigns a score to each test $e \in U \setminus E$ of

$$G_E(e) := \sum_{(b,x) \in M : f_{b,x}(E) < 1} q_{b,x} \frac{f_{b,x}(\{e\} \cup E) - f_{b,x}(E)}{1 - f_{b,x}(E)} = \sum_{(b,x) \in M} q_{b,x} \cdot \Delta_E(b, x, e), \tag{3.3}$$

$$\Delta_E(b, x, e) = \begin{cases} \frac{f_{b,x}(\{e\} \cup E) - f_{b,x}(E)}{1 - f_{b,x}(E)}, & \text{if } f_{b,x}(E) < 1; \\ 0, & \text{otherwise.} \end{cases} \tag{3.4}$$

where $\Delta_E(b, x, e)$ is the "gain" of test e for the hypothesis-copy (b, x). At each step, the algorithm chooses the test with the maximum score. However, we do not know how to

compute the score (3.3) in polynomial time. Instead, using the fact that $G_E(e)$ is the expectation of $\Delta_E(b, x, e)$ over the hypothesis-copies $(b, x) \in M$, we will show that we can obtain a constant-approximate maximizer by sampling. Moreover, we use the following extension of Theorem III.1, which follows directly from the analysis in [55].

Theorem III.3 ([6, 55])**.** *Consider the SFR algorithm that selects at each step, an element e with $G_E(e) \geq \Omega(1) \cdot \max_{e' \in U} G_E(e')$. This is an $O(\log \frac{1}{\epsilon})$ approximation algorithm.*

So, if we always find an approximate maximizer for $G_E(e)$ by sampling then Theorem III.2 would follow from Theorem III.3. However, this sampling approach is not sufficient because it can fail when the value $G_E(e)$ is very small. In order to deal with this, a key observation is that when the score $G_E(e)$ is small *for all* tests e then it must be that (with high probability) the already-performed tests E uniquely identify hypothesis $\bar{x}$. So any future tests would not affect the expected cover time by much. This is formalized below.

As the realized hypothesis $\bar{x}$ can always be identified uniquely, for any pair $x, y \in [m]$ of hypotheses, there is a test where x and y have opposite outcomes (i.e. one is $+$ and the other $-$). So there is a set $\mathcal{L}$ of at most $\binom{m}{2}$ tests where hypothesis $\bar{x}$ will be uniquely identified by performing all the tests in $\mathcal{L}$. The non-adaptive ODTN algorithm (Algorithm 3 below) involves two phases. In the first phase, we run the SFR algorithm using sampling to get estimates $\overline{G_E}(e)$ of the scores. If at some step, the maximum sampled score is less than m^{-5} then we go to the second phase where we perform all the tests in $\mathcal{L}$ and stop. The number of samples used to obtain each estimate is polynomial in m; so the overall runtime is polynomial.

We use the following sampling lemma, which follows from Chernoff bounds.

Lemma III.4. *Let X be a $[0, 1]$ bounded random variable with $\mathbb{E}X \geq \Omega(m^{-5})$. Let $\bar{X}$ denote the average of m^6 many independent samples of X. Then $\Pr\left[\bar{X} \notin [\frac{1}{2}\mathbb{E}X, 2\mathbb{E}X]\right] \leq e^{-\Omega(m)}$.*

Algorithm 3 Non-adaptive ODTN algorithm.

initially $E \leftarrow \emptyset$ and sequence $\sigma = \emptyset$.
while $E \neq U$ **do** (*Phase 1 begins*)
  for each $e \in U$, compute an estimate $\overline{G_E}(e)$ of the score $G_E(e)$ by sampling from M independently $N = m^6$ times.
  let e^* denote the test $e \in U \setminus E$ that maximizes $\overline{G_E}(e)$.
  if $\overline{G_E}(e) \geq \frac{1}{4}m^{-5}$ **then**
    update $E \leftarrow E \cup \{e^*\}$ and append e^* to sequence σ.
  else
    exit the while loop. (*Phase 1 ends*)
  end if
end while
append the tests in $\mathcal{L} \setminus E$ to sequence σ. (*Phase 2*)
output non-adaptive sequence σ.

Proof. Let $X_1, ..., X_N$ be i.i.d. samples of random variable X where $N = m^6$ is the number of samples. Letting $Y = \sum_{i \in [N]} X_i$, the usual Chernoff bound implies for any $\delta \in (0, 1)$,

$$\Pr\left(Y \notin [(1-\delta)\mathbb{E}Y, (1+\delta)\mathbb{E}Y]\right) \leq \exp(-\frac{\delta^2}{2} \cdot \mathbb{E}Y).$$

Setting $\delta = \frac{1}{2}$ and using the assumption $\mathbb{E}Y = N \cdot \mathbb{E}X = \Omega(m)$, the lemma follows. □

The next lemma shows that sampling does find an approximate maximizer unless the score is very small, and also bounds the "failure" probability.

Lemma III.5. *Consider any step in the algorithm with* $S = \max_{e \in U} G_E(e)$ *and* $\bar{S} = \max_{e \in U} \overline{G_E}(e)$ *with* $\overline{G_E}(e^*) = \bar{S}$. *Call this step a failure if (i)* $\bar{S} < \frac{1}{4}m^{-5}$ *and* $S \geq \frac{1}{2}m^{-5}$, *or (ii)* $\bar{S} \geq \frac{1}{4}m^{-5}$ *and* $G_E(e^*) < \frac{S}{4}$. *Then the probability of failure is at most* $e^{-\Omega(m)}$.

Proof. We will consider the two types of failures separately. For the first type, suppose $S \geq \frac{1}{2}m^{-5}$. Using Lemma III.4 on the test $e \in U$ with $G_E(e) = S$, we obtain

$$\Pr[\bar{S} < \frac{1}{4}m^{-5}] \leq \Pr[\overline{G_E}(e) < \frac{1}{4}m^{-5}] \leq e^{-\Omega(m)}.$$

So the probability of the first type of failure is at most $e^{-\Omega(m)}$.

For the second type of failure, we consider two further cases:

- $S < \frac{1}{8}m^{-5}$. For any $e \in U$ we have $G_E(e) \leq S < \frac{1}{8}m^{-5}$. Note that $\overline{G_E}(e)$ is the the average of m^6 many independent samples each with mean $G_E(e)$. We now upper bound the probability of the event $\mathcal{B}_e$ that $\overline{G_E}(e) \geq \frac{1}{4}m^{-5}$. We first artificially increase each sample mean to $\frac{1}{8}m^{-5}$: note that this only increases the probability of event $\mathcal{B}_e$. Now, using Lemma III.4 we obtain $\Pr[\mathcal{B}_e] \leq e^{-\Omega(m)}$. By a union bound, it follows that $\Pr[\bar{S} \geq \frac{1}{4}m^{-5}] \leq \sum_{e \in U} \Pr[\mathcal{B}_e] \leq e^{-\Omega(m)}$.
- $S \geq \frac{1}{8}m^{-5}$. Consider now any $e \in U$ with $G_E(e) < S/4$. By Lemma III.4 (artificially increasing $G_E(e)$ to $S/4$ if needed), it follows that $\Pr[\overline{G_E}(e) > S/2] \leq e^{-\Omega(m)}$. Now consider the test e' with $G_E(e') = S$. Again, by Lemma III.4, it follows that $\Pr[\overline{G_E}(e') \leq S/2] \leq e^{-\Omega(m)}$. This means that test e^* has $\overline{G_E}(e^*) \geq \overline{G_E}(e') > S/2$ and $G_E(e^*) \geq S/4$ with probability $1 - e^{-\Omega(m)}$. In other words, assuming $S \geq \frac{1}{8}m^{-5}$, the probability that $G_E(e^*) < S/4$ is at most $e^{-\Omega(m)}$.

Adding the probabilities over all possibilities for failures, the lemma follows. □

Based on Lemma III.5, in the remaining analysis we condition on our algorithm encountering *no failures*: this occurs with probability $1 - e^{-\Omega(m)}$.

Lemma III.6. *Assume that there are no failures. Consider the end of phase 1 in our algorithm, i.e. the first step with $\overline{G_E}(e^*) < \frac{1}{4}m^{-5}$. The probability that the realized hypothesis $\bar{x}$ is not uniquely identified at this point is at most m^{-2}.*

Proof. Let E denote the set of all previous tests and z the probability of *not* identifying the realized hypothesis $\bar{x}$ after performing tests E. For a contradiction, suppose that $z > m^{-2}$.

Let $p_x(y) = \Pr_{b\sim x}[y \text{ not ruled out by } E|\bar{x} = x]$ denote the probability that when $\bar{x} = x$, hypothesis y is not ruled out after performing tests E. Note that

$$\begin{aligned} z &= \sum_{x=1}^{m} \pi_x \cdot \Pr_{b\sim x}[E \text{ doesn't rule out } [m] \setminus x] \\ &\leq \sum_{x=1}^{m} \pi_x \sum_{y\in[m]\setminus x} p_x(y) \\ &\leq m \sum_{x=1}^{m} \left(\pi_x \cdot \max_{y\in[m]\setminus x} p_x(y) \right) \end{aligned}$$

It follows that there is some $x \in [m]$ with $\pi_x \cdot \max_{y\in[m]\setminus x} p_x(y) \geq \frac{z}{m^2}$. So there is some $x, y \in [m]$ with $\pi_x \cdot p_x(y) \geq \frac{z}{m^2} > m^{-4}$, where we used the assumption that $z > m^{-2}$. Recall that there is some test e' that separates x and y deterministically. Let $B' = \{b \sim x : y \notin \cup_{e\in E} T_{b,x}(e)\}$. Note that $\sum_{b\in B'} q_{b,x} = \pi_x \cdot p_x(y)$. For any $b \in B'$ we have (i) $y \notin \cup_{e\in E} T_{b,x}(e)$ which implies $f_{b,x}(E)$ and (ii) $y \in T_{b,x}(e')$ (this is true for *all* $b \sim x$). Therefore $\Delta_E(b, x, e') \geq \frac{1}{m-1}$ for all $b \in B'$, which implies:

$$G_E(e') \geq \sum_{b\sim x} q_{b,x} \Delta_E(b, x, e^*) \geq \sum_{b\in B'} q_{b,x} \frac{1}{m-1} = \frac{\pi_x \cdot p_x(y)}{m-1} > m^{-5}.$$

So we obtain $S = \max_{e\in U} G_E(e) \geq m^{-5}$ and $\overline{G_E}(e^*) < \frac{1}{4} m^{-5}$ (as this is the end of phase 1). This means that our algorithm has encountered a failure (see case (i) in Lemma III.5), which is a contradiction to the "no failure" assumption. □

Proof of Theorem III.2. Assume that there are no failures. We bound the expected costs (number of tests) from phase 1 and 2 separately. By Lemma III.5, the test chosen in each step of phase 1 is a 4-approximate maximizer (see case (ii) failure) of the score used in the ASR algorithm. So, by Theorem III.3, the expected cost in phase 1 is at most $O(\log m)$ times the optimum. By Lemma III.6, the probability of performing tests from phase 2 is at

most m^{-2}. As there are $|\mathcal{L}| \leq m^2$ tests in phase 2, the expected cost is $O(1)$. So we obtain an $O(\log m)$-approximation algorithm.

3.4 Adaptive Algorithms

Our adaptive algorithm chooses between two algorithms ($ODTN_r$ and $ODTN_h$) based on the noise sparsity parameters h (maximum number of noisy outcome per hypothesis), and r (maximum number of noisy outcome per test). These two algorithms maintain the posterior probability of each hypothesis based on the previous test outcomes, and use these probabilities to calculate a "score" for each test. The score of a test has two components (i) a term that prioritizes splitting the candidate hypotheses in a balanced way and (ii) terms that correspond to the expected number of hypotheses eliminated. We maintain the following information at each point in the algorithm: already performed tests $E \subseteq U$, compatible hypotheses $H \subseteq [m]$ and the posterior probability for each $x \in H$. The main difference between the two algorithms we have, is in the defining the metric for component (i).

The high-level idea in both algorithms is to view any ODT instance $\mathcal{I}$ as a suitable instance $\mathcal{J}$ of adaptive submodular ranking (ASR). Then we will use and modify an existing framework of analysis of ASR from Chapter II.

An equivalent ASR instance $\mathcal{J}$. This involves the expanded hypothesis set $M = \cup_{x=1}^{m} M_x$ where M_x are all copies (b, x) of hypothesis $x \in [m]$. Each hypothesis $(b, x) \in M$ occurs with probability $q_{b,x} = \pi_x / 2^{h_x}$. To reduce notation, we use $q(S) = \sum_{(b,x)\in S} q_{b,x}$ for any subset $S \subseteq M$. Each hypothesis (b, x) is also associated with: (i) submodular function $f_{b,x} : 2^U \to [0, 1]$ and (ii) feedback function $r_{b,x} : U \to \{+, -\}$ where $r_{b,x}(e)$ is the outcome of test e under hypothesis (b, x). The goal in the ASR instance is to adaptively select a subset $S \subseteq U$ such that the value $f_{b,x}(S) = 1$ for the realized hypothesis (b, x). The objective is to

minimize the expected cost $\mathbb{E}[|S|]$. Note that hypothesis (b, x) is covered when $f_{b,x}(E) = 1$, which is equivalent to identifying hypothesis $x \in [m]$ when the test outcomes are given by b.

Lemma III.7. *The ASR instance $\mathcal{J}$ is equivalent to ODT instance $\mathcal{I}$.*

Proof. We will show that any feasible decision tree for instance $\mathcal{J}$ (resp. $\mathcal{I}$) is also feasible for instance $\mathcal{I}$ (resp. $\mathcal{I}$) with the same objective. In one direction, let $\mathcal{T}$ be a decision tree for ASR instance $\mathcal{J}$. For any hypothesis $(b, x) \in M$ let $P_{b,x}$ denote the unique path traced in $\mathcal{T}$ and let $S_{b,x}$ denote the tests performed. Then we have $f_{b,x}(S_{b,x}) = 1$ which means $\bigcup_{e \in S_{b,x}} T_{b,x}(e) = [m] \setminus x$. Now consider $\mathcal{T}$ as a decision tree for the ODTN instance $\mathcal{I}$. *Condition* on hypothesis $x \in [m]$ and outcomes b on the $*$-tests for x, which occurs with probability $q_{b,x} = \pi_x / 2^{h_x}$. Then, it is clear that the feedback from any test e is $r_{b,x}(e)$ and so the path traced in $\mathcal{T}$ is just $P_{b,x}$. Moreover, the set of incompatible hypotheses based on test e is $T_{b,x}(e)$. So the set of incompatible hypotheses at the end of $P_{b,x}$ is $\bigcup_{e \in S_{b,x}} T_{b,x}(e) = [m] \setminus x$, which means x is identified. Taking an expectation over all x and b, it follows that $\mathcal{T}$ is a feasible decision tree for $\mathcal{I}$ with cost at most that for instance $\mathcal{J}$.

In the other direction, let $\mathcal{T}'$ be any decision tree for instance ODTN $\mathcal{I}$. Again *condition* on hypothesis $x \in [m]$ and outcomes b on the $*$-tests for x (with probability $q_{b,x}$). Then a unique path $P'_{b,x}$ is traced in $\mathcal{T}'$, and let $S'_{b,x}$ denote the tests on this path. As before, the set of incompatible hypotheses at the end of $P'_{b,x}$ is $\bigcup_{e \in S'_{b,x}} T_{b,x}(e) = [m] \setminus x$ because x is identified. Now consider $\mathcal{T}'$ as a decition tree for ASR instance $\mathcal{J}$. Under hypothesis b, x, it is clear that path $P'_{b,x}$ is traced and so tests $S'_{b,x}$ are selected. It follows that $f_{b,x}(S'_{b,x}) = 1$ which means that hypothesis b, x is covered at the end of $P'_{b,x}$. So $\mathcal{T}'$ is a feasible decision

tree for $\mathcal{J}$. Taking expectations, the cost for $\mathcal{J}$ is at most that for instance $\mathcal{I}$. □

3.4.1 $\mathcal{O}(h + \log m)$-Approximation Algorithm

This algorithm is based on directly applying the ASR algorithm from Chapter II on instance $\mathcal{J}$. It is described in Algorithm 4 below. This involves maintaining the set $H' \subseteq M$ of all compatible and uncovered hypotheses-copies, and iteratively selecting the test e with maximum score (3.6).

Algorithm 4 Algorithm for ASR instance $\mathcal{J}$ from Chapter II.

initially $E \leftarrow \emptyset, H' \leftarrow M$.
while $H' \neq \emptyset$ **do**
    for any test $e \in U$, let $L_e(H')$ be the smaller cardinality set among:

$$\{(b,x) \in H' : r_{b,x}(e) = +1\} \text{ and } \{(b,x) \in H' : r_{b,x}(e) = -1\}. \tag{3.5}$$

    select test $e \in U \setminus E$ that maximizes:

$$q\left(L_e(H')\right) + \sum_{(b,x)\in H'} q_{b,x} \cdot \frac{f_{b,x}(e \cup E) - f_{b,x}(E)}{1 - f_{b,x}(E)}. \tag{3.6}$$

    remove incompatible and covered hypotheses from H' based on the feedback from e.
    $E \leftarrow E \cup \{e\}$
end while

One issue that we need to handle is that the size of the expanded hypothesis set M is exponentially large; recall that $|M| \leq 2^h \cdot m$. So direct use of this algorithm requires exponential time in maintaining H' and computing the score (3.6). However, we can use the structure of instance $\mathcal{J}$ to implicitly perform these calculations in polynomial time.

Maintaining and using H'. We just maintain the *number* $n_x = |M_x \cap H'|$ of copies of each original hypothesis $x \in [m]$. In each iteration, after performing test e, this number n_x (for $x \in [m]$) can be easily updated as follows:

If $r_x(e) = *$ then n_x reduces by factor 2; otherwise n_x remains the same.

For any test e, we can also implicitly describe set $L_e(H')$ in (3.5) as follows. Note that

$$|\{(b,x) \in H' : r_{b,x}(e) = +1\}| = \sum_{x \in [m]: r_x(e) = +1} n_x \; + \; \frac{1}{2} \sum_{x \in [m]: r_x(e) = *} n_x.$$

Similarly,

$$|\{(b,x) \in H' : r_{b,x}(e) = -1\}| = \sum_{x \in [m]: r_x(e) = -1} n_x \; + \; \frac{1}{2} \sum_{x \in [m]: r_x(e) = *} n_x.$$

So the smaller of the two sets in (3.5) can be described by the smaller of the above two expressions that involve n_xs. Moreover, if $L_e(H')$ corresponds to the $+1$ outcome, we can calculate

$$q(L_e(H') = \sum_{x \in [m]: r_x(e) = +1} \frac{\pi_x}{2^{h_x}} \cdot n_x \; + \; \frac{1}{2} \sum_{x \in [m]: r_x(e) = *} \frac{\pi_x}{2^{h_x}} \cdot n_x.$$

If $L_e(H')$ corresponds to the -1 outcome then there is a similar expression.

Computing the score (3.6)**.** We already discussed how to calculate $q(L_e(H'))$. So it only remains to calculate the second term in (3.6). Let $H = \{x \in [m] : M_x \cap H' \neq \emptyset\} = \{x \in [m] : n_x \geq 1\}$ denote the compatible (original) hypotheses, i.e. those with at least one copy in H'. For each hypothesis $x \in H$ let $p_x = n_x \cdot \frac{\pi_x}{2^{h_x}}$. To reduce notation we use the shorthand $p(S) = \sum_{x \in S} p_x$ for any subset $S \subseteq [m]$. Note that $p_x/p(H)$ equals the posterior probability of hypothesis $x \in H$, based on all previously observed test outcomes. We will show that the second term in (3.6) is equal to:

(3.7)

$$\frac{|T^-(e) \cap H|}{|H| - 1} \cdot p\left(T^+(e) \cap H\right) \; + \; \frac{|T^+(e) \cap H|}{|H| - 1} \cdot p\left(T^-(e) \cap H\right) + \frac{1}{2} \frac{|H \setminus T^*(e)|}{|H| - 1} \cdot p\left(T^*(e) \cap H\right).$$

We first prove the following lemma that relies on the definitions of $f_{b,x}$, $T_{b,x}$ etc.

Lemma III.8. *Consider any state (E, H') of Algorithm 4 and $(b,x) \in H'$. The following are true:*

1. $\bigcup_{e\in E} T_{b,x}(e) = [m] \setminus H$. *So* $f_{b,x}(E) = \frac{m-|H|}{m-1}$.

2. *For any* $e \in U$, $f_{b,x}(E \cup e) - f_{b,x}(E) = \frac{|H\cap T_{b,x}(e)|}{m-1}$.

3. *If* $x \in H \cap T^+(e)$ *then* $f_{b,x}(E \cup e) - f_{b,x}(E) = \frac{|H\cap T^-(e)|}{m-1}$.

4. *If* $x \in H \cap T^-(e)$ *then* $f_{b,x}(E \cup e) - f_{b,x}(E) = \frac{|H\cap T^+(e)|}{m-1}$.

5. *If* $x \in H \cap T^*(e)$ *then*

$$\sum_{(b,x)\in H'_x} q_{b,x}\left(f_{b,x}(E \cup e) - f_{b,x}(E)\right) = \frac{1}{2}\sum_{(b,x)\in H'_x} q_{b,x}\left(\frac{|H \cap T^-(e)|}{m-1} + \frac{|H \cap T^+(e)|}{m-1}\right),$$

where $H'_x = M_x \cap H'$.

Proof. (1) As $(b,x) \in H'$, the outcome of each test $e \in E$ must have been $r_{b,x}(e)$. By definition, $T_{b,x}(e)$ consists of all hypotheses $y \in [m]$ with $r_y(e) \neq *$ and $r_y(e) \neq r_{b,x}(e)$. So $\bigcup_{e\in E} T_{b,x}(e) \subseteq [m]$ is precisely the set of incompatible hypotheses at this state. In other words, $\bigcup_{e\in E} T_{b,x}(e) = [m] \setminus H$ and $f_{b,x}(E) = \frac{|\bigcup_{e\in E} T_{b,x}(e)|}{m-1} = \frac{m-|H|}{m-1}$.

(2) The set of hypotheses in H that are compatible with (b,x) after test e are $H \setminus T_{b,x}(e)$. So based on (1) we can write:

$$f_{b,x}(E \cup e) - f_{b,x}(E) = \frac{m - |H \setminus T_{b,x}(e)|}{m-1} - \frac{m-|H|}{m-1} = \frac{|H \setminus T_{b,x}(e)| - |H|}{m-1} = \frac{|H \cap T_{b,x}(e)|}{m-1}.$$

(3-4) These follow directly from (2) using the definition of $T_{b,x}(e)$.

(5) It is easy to see (as observed before) that half the hypothesis-copies $(b,x) \in H' \cap M_x = H'_x$ have $r_{b,x}(e) = +1$ (which implies $T_{b,x}(e) = T^-(e)$) and the rest have $r_{b,x}(e) = -1$ (which implies $T_{b,x}(e) = T^+(e)$). So using (2),

$$\sum_{(b,x)\in H'_x} q_{b,x}\left(f_{b,x}(E \cup e) - f_{b,x}(E)\right) = \sum_{\substack{(b,x)\in H'_x \\ b(e)=-1}} q_{b,x}\left(\frac{|H \cap T^+(e)|}{m-1}\right) + \sum_{\substack{(b,x)\in H'_x \\ b(e)=+1}} q_{b,x}\left(\frac{|H \cap T^-(e)|}{m-1}\right),$$

which proves the statement as exactly half the copies in H'_x have $b(e) = +1$. □

Lemma III.9. *The second term in* (3.6) *equals* (3.7) *for any state* (E, H') *of Algorithm 4 and test* e*.*

Proof. Consider any $x \in H$. By its definition, we have $p_x = \sum_{(b,x)\in H'\cap M_x} q_{b,x}$. Now using Lemma III.8, we have the following three cases:

- If $x \in H \cap T^+(e)$ then

$$\sum_{(b,x)\in H'\cap M_x} q_{b,x} \left(\frac{f_{b,x}(E \cup e) - f_{b,x}(E)}{1 - f_{b,x}(E)} \right) = p_x \cdot \frac{|H \cap T^-(e)|}{|H| - 1}.$$

- If $x \in H \cap T^-(e)$ then

$$\sum_{(b,x)\in H'\cap M_x} q_{b,x} \left(\frac{f_{b,x}(E \cup e) - f_{b,x}(E)}{1 - f_{b,x}(E)} \right) = p_x \cdot \frac{|H \cap T^+(e)|}{|H| - 1}.$$

- If $x \in H \cap T^*(e)$ then

$$\sum_{(b,x)\in H'\cap M_x} q_{b,x} \left(\frac{f_{b,x}(E \cup e) - f_{b,x}(E)}{1 - f_{b,x}(E)} \right) = \frac{p_x}{2} \cdot \frac{|H \cap T^-(e)| + |H \cap T^+(e)|}{|H| - 1}.$$

Summing over all $x \in H$, it follows that the second term in (3.6) equals (3.7). □

Therefore, we have proved:

Theorem III.10. *There is an* $\mathcal{O}(h + \log m)$*-approximation algorithm for adaptive ODTN, where* h *is the maximum number of noisy outcomes for a hypothesis.*

Proof. Consider the ASR instance $\mathcal{J}$ and Algorithm 4. As discussed above, this algorithm can be implemented in polynomial time. Using the result from Chapter II it follows that this algorithm has an $O(\log |M| + \log 1/\epsilon) = O(h + \log m)$ approximation ratio, because $|M| \leq 2^h \cdot m$ and $\epsilon \geq 1/m$. □

3.4.2 $\mathcal{O}(r + \log m)$-Approximation Algorithm

This algorithm is based on a different definition of the "score" in Algorithm 4. In particular, we change (3.5) and (3.6) as described in Algorithm 5 below.

Algorithm 5 Modified algorithm for ASR instance $\mathcal{J}$.

initially $E \leftarrow \emptyset, H' \leftarrow M$.
while $H' \neq \emptyset$ **do**
  $H \leftarrow \{x \in [m] : M_x \cap H' \neq \emptyset\}$.
  for each test $e \in U$, let $L_e(H)$ be the smaller cardinality set among $T^+(e) \cap H$ and $T^-(e) \cap H$.
  for each test $e \in U$, define

$$L'_e(H') = \{(b,x) \in H' : x \in L_e(H)\} = H' \cap \left(\cup_{x \in L_e(H)} M_x\right). \tag{3.8}$$

  select test $e \in U \setminus E$ that maximizes:

$$q\left(L'_e(H')\right) + \sum_{(b,x) \in M_x \cap H'} q_{b,x} \cdot \frac{f_{b,x}(e \cup E) - f_{b,x}(E)}{1 - f_{b,x}(E)}. \tag{3.9}$$

  remove incompatible and covered hypotheses from H' based on the feedback from e.
  $E \leftarrow E \cup \{e\}$
end while

As for Algorithm 4, this algorithm can also be implemented in polynomial time. Recall that we maintain and use H' implicitly as follows. For each original hypothesis $x \in [m]$, we store $n_x = |M_x \cap H'|$. So $H = \{x \in [m] : n_x \geq 1\}$ and for any test $e \in U$, $L_e(H)$ is defined as the smaller of the sets $\{x \in [m] : r_x(e) = +1, n_x \geq 1\}$ and $\{x \in [m] : r_x(e) = -1, n_x \geq 1\}$. This suffices to describe $L'_e(H')$ implicitly. Moreover, for each $x \in [m]$ we store $p_x = n_x \cdot \frac{\pi_x}{2^{h_x}}$; note that $p_x/p(H)$ is the current posterior probability of any hypothesis $x \in H$. Note that the "score" (3.9) in this algorithm differs from (3.6) only in the first term: we now use $q(L'_e(H'))$ instead of $q(L_e(H'))$. Therefore, as shown in §3.4.1, the second term in (3.9) (which is also the second term in (3.6)) equals (3.7). For easier reference, the polynomial time implementation is described explicitly in Algorithm 6. Note that we keep track of H and $\{p_x\}_{x=1}^m$ directly without using the numbers $\{n_x\}_{x=1}^m$. Also, the stopping criterion remains the same because $H' = \emptyset$ if and only if $|H| = 1$. To see this, note that hypothesis-

copy (b, x) is covered exactly when $H = \{x\}$ and H' consists of all compatible uncovered hypothesis-copies.

Algorithm 6 Polynomial time implementation.

initially $E \leftarrow \emptyset$, $H \leftarrow [m]$ and $p_x \leftarrow \pi_x$ for all $x \in [m]$.
while $|H| > 1$ **do**
  for any test $e \in U$, let $L_e(H)$ be the smaller cardinality set among $T^+(e) \cap H$ and $T^-(e) \cap H$
  select test $e \in U \setminus E$ that maximizes:

$$p\left(L_e(H)\right) + \frac{|T^-(e) \cap H|}{|H| - 1} \cdot p\left(T^+(e) \cap H\right) + \frac{|T^+(e) \cap H|}{|H| - 1} \cdot p\left(T^-(e) \cap H\right) + \frac{1}{2} \frac{|H \setminus T^*(e)|}{|H| - 1} \cdot p\left(T^*(e) \cap H\right). \tag{3.10}$$

  if outcome of test e is $+$ **then**
    update $H \leftarrow H \setminus T^-(e)$
  else
    update $H \leftarrow H \setminus T^+(e)$
  end if
  $E \leftarrow E \cup \{e\}$
  for $x \in H$ **do**
    if $r_x(e) = *$ **then**
      $p_x \leftarrow p_x/2$
    end if
  end for
end while

We will show the following result:

Theorem III.11. *Algorithm 6 is a polynomial-time $\mathcal{O}(r + \log m)$-approximation algorithm for adaptive ODTN, where r is the maximum number of noisy outcomes per test.*

Using the equivalence of Algorithms 6 and 5, it suffices to prove the approximation ratio for Algorithm 5 applied to the ASR instance $\mathcal{J}$. The proof is very similar to the analysis in Chapter II. So we only provide an outline of the overall proof, while emphasizing the differences. Let ALG denote the policy described in Algorithm 5 and OPT the optimal adaptive policy. For $k = 0, 1, \cdots$, define the following quantities:

- $A_k \subseteq M$ is the set of uncovered hypotheses in ALG at time $L \cdot 2^k$, and $a_k = q(A_k)$ is the total probability of these hypotheses.
- Y_k is the set of uncovered hypotheses in OPT at time 2^{k-1}, and $y_k = q(Y_k)$ is the total

probability of these hypotheses.

Here $L = O(r + \log m)$. The key step is to show:

$$a_k \leq 0.2a_{k-1} + 3y_k, \qquad \text{for all } k \geq 1. \tag{3.11}$$

As shown in Chapter II, this suffices to show that Algorithm 5 is an $O(L)$-approximation algorithm, which implies Theorem III.11. In order to prove (3.11) we use the quantity:

$$Z := \sum_{t > L2^{k-1}}^{L2^k} \sum_{(E,H') \in R(t)} \max_{e \in U \setminus E} \left(\sum_{(b,x) \in L'_e(H')} q_{b,x} + \sum_{(b,x) \in H'} q_{b,x} \cdot \frac{f_{b,x}(e \cup E) - f_{b,x}(E)}{1 - f_{b,x}(E)} \right) \tag{3.12}$$

Above, $R(t)$ denotes the set of states (E, H') that occur at time t in ALG. (3.11) will be proved by separately lower and upper bounding Z. Recall the following Lemma from Chapter II:

Lemma II.5. *We have $Z \geq L \cdot (a_k - 3y_k)/3$.*

Although the definition of $L'_e(H')$ is different here, the proof of this lower bound is identical to what presented in Chapter II and we do not repeat it here. The proof of the upper bound (analogous to Lemma II.6 in Chapter II) requires new ideas, and is shown below.

Lemma III.12. *We have $Z \leq a_{k-1} \cdot (1 + \ln m + r + \log m)$.*

Proof. For any hypothesis $(b, x) \in A_{k-1}$ (i.e. uncovered in ALG by time $L2^{k-1}$) let $\sigma_{b,x}$ be the path traced by (b, x) in ALG's decision tree, starting from time $2^{k-1}L$ and ending at $2^k L$ or when (b, x) gets covered. Recall that for any $L2^{k-1} < t \leq L2^k$, any hypothesis in H' for any state in $R(t)$ appears in A_{k-1}. So only hypotheses in A_{k-1} can contribute to Z and we rewrite (3.12) as:

$$\begin{aligned} Z &= \sum_{(b,x) \in A_{k-1}} q_{b,x} \cdot \sum_{e \in \sigma_{b,x}} \left(\frac{f_{b,x}(e \cup E) - f_{b,x}(E)}{1 - f_{b,x}(E)} + \mathbf{1}[(b,x) \in L'_e(H')] \right) \\ &\leq \sum_{(b,x) \in A_{k-1}} q_{b,x} \cdot \left(\sum_{e \in \sigma_{b,x}} \frac{f_{b,x}(e \cup E) - f_{b,x}(E)}{1 - f_{b,x}(E)} + \sum_{e \in \sigma_{b,x}} \mathbf{1}[(b,x) \in L'_e(H')] \right) \end{aligned} \tag{3.13}$$

Above, for any $e \in \sigma_{b,x}$ we use (E, H') to denote the state at which e is selected.

Fix any hypothesis $(b, x) \in A_{k-1}$. For the first term, we use Lemma III.13 below and the definition of ϵ. This implies $\sum_{e \in \sigma_{b,x}} \frac{f_{b,x}(e \cup E) - f_{b,x}(E)}{1 - f_{b,x}(E)} \leq 1 + \ln \frac{1}{\epsilon} \leq 1 + \ln m$ as parameter $\epsilon \geq 1/m$ for $f_{b,x}$.

Now, we bound the second term by proving the inequality below:

$$\sum_{e \in \sigma_{b,x}} \mathbf{1}[(b, x) \in L'_e(H')] \leq r + \log m \tag{3.14}$$

To prove this inequality, consider hypotheses in H'. Now, if hypothesis $(b, x) \in L'_e(H')$ when ALG selects test e, then x would be in $L_e(H)$. Suppose $L_e(H) = T^+(e) \cap H$; the other case is identical. Let $D_e(H) = T^-(e) \cap H$ and $S_e(H) = T^*(e) \cap H$. As $x \in L_e(H)$, it must be that path $\sigma_{b,x}$ follows the + branch out of e. Also, the number of candidate hypotheses on this path after test e is

$$|L_e(H)| + |S_e(H)| \leq \frac{|L_e(H)|}{2} + \frac{|D_e(H)|}{2} + |S_e(H)| = \frac{|H|}{2} + \frac{|S_e(H)|}{2} \leq \frac{|H|}{2} + \frac{r}{2}.$$

The first inequality uses the definition of $L_e(H)$ and the last inequality uses the bound of r on the number of hypotheses with $*$ outcomes. Hence, each time that $(b, x) \in L'_e(H')$ along path $\sigma_{b,x}$, the number of candidate hypotheses changes as $|H_{new}| \leq \frac{1}{2}|H_{old}| + \frac{r}{2}$. This implies that after $\log_2 m$ such events, $|H| \leq r$. Let $\sigma'_{b,x}$ denote the portion of path $\sigma_{b,x}$ after $|H|$ drops below r. Note that each time $(b, x) \in L'_e(H')$ we have $L_e(H) \neq \emptyset$: so $|H|$ reduces by at least one after each such test e. Hence $\sum_{e \in \sigma'_{b,x}} \mathbf{1}[(b, x) \in L'_e(H')] \leq r$. As the portion of path $\sigma_{b,x}$ until $|H| \leq r$ contributes at most $\log_2 m$ to the left-hand-side in (3.14), the total is at most $r + \log_2 m$ as needed. □

Lemma III.13 ([6]). *Let $f : 2^U \to [0,1]$ be any monotone function with $f(\emptyset) = 0$ and $\epsilon = \min\{f(S \cup \{e\}) - f(S) : e \in U, S \subseteq U, f(S \cup \{e\}) - f(S) > 0\}$. Then, for any sequence $\emptyset = S_0 \subseteq S_1 \subseteq \cdots S_k \subseteq U$ of subsets, we have $\sum_{t=1}^{k} \frac{f(S_t)-f(S_{t-1})}{1-f(S_{t-1})} \leq 1 + \ln \frac{1}{\epsilon}$.*

Setting $L = 15(1 + \ln m + r + \log_2 m)$ and applying Lemmas II.5 and II.6 completes the proof of (3.11) and hence Theorem III.11.

Tight Example. Our analysis above is tight, as shown by the following instance with $r = m$ where the algorithm's cost is $\Omega(m)$ times the optimum. The instance has $m = 6k+1$ hypotheses, which are partitioned into $A = \{a_1, \cdots a_{3k}\}$, $B = \{b_1, \cdots, b_{3k-3}\}$, $C = \{c_1, c_2, c_3\}$ and $\{c^*\}$. The probability of hypothesis c^* is $1 - \epsilon$ and each of the other hypotheses has probability $\frac{\epsilon}{6k}$. We will use $\epsilon = k^{-3} \to 0$. We also have four types of tests (unspecified hypotheses have $*$ outcomes).

- α-tests: for each $j \in [k]$, test α_j is $+1$ on $C \cup \{c^*\}$ and -1 on $\{a_{3j-2}, a_{3j-1}, a_{3j}\}$.
- β-tests: for each $j \in [k-1]$, test β_j is $+1$ on $C \cup \{c^*\}$ and -1 on $\{b_{3j-2}, b_{3j-1}, b_{3j}\}$.
- γ-tests: for each pair $s, t \neq c^*$ of hypotheses there is a test that is $+1$ on s and -1 on t.
- Test δ is $+1$ on A and -1 on all other hypotheses. Test δ' is $+1$ on B and -1 on all others. Test δ'' is $+1$ on C and -1 on all others.

Bound on the Optimal Cost: We first perform tests δ, δ' and δ''. If all three outcomes are -1 then we identify hypothesis c^* uniquely; this happens with probability $1 - \epsilon$. Otherwise, we continue to perform all the γ-tests which suffices to identify the realized hypothesis (this happens with total probability ϵ). The expected cost is at most $3 + \epsilon \cdot m^2 = \mathcal{O}(1)$ using

$\epsilon = 1/k^3$.

Cost of our Algorithm: We will only describe the sequence of tests under realized hypothesis c^*, which will provide a lower bound on the algorithm's cost. We claim that the algorithm will select tests $\alpha_1, \beta_1, \alpha_2, \beta_2, \cdots \alpha_{k-1}, \beta_{k-1}$. We show this by induction. Consider the candidate hypotheses H at any point in this sequence. Note that the alternating choice of α and β tests implies that we will always have $\{c^*\} \cup C \in H$ and either $|H \cap A| = 3 + |H \cap B|$ or $|H \cap A| = |H \cap B|$. In either case, the "lighter" side of test δ is always $H \cap A$, the lighter side of test δ' is always $H \cap B$ and the lighter side of test δ'' is always C. In particular, for any test $e \in \{\delta, \delta', \delta''\}$ we have $c^* \notin L_e(H)$ and the score of e in Equation (3.10) is:

$$p(L_e(H)) \;+\; \frac{|T^+(e) \cap H|}{|H| - 1} \cdot p(T^-(e) \cap H) \;+\; \frac{|T^-(e) \cap H|}{|H| - 1} \cdot p(T^+(e) \cap H) \quad \leq \quad \epsilon + \frac{1}{2} + \epsilon.$$

Above we used the fact that hypothesis c^* does not lie in the lighter side and it has probability $1 - \epsilon$ (so the total remaining probability is at most ϵ). On the other hand, the score of all remaining α and β tests will be equal (by symmetry) and has value at least $1 - \epsilon$ as c^* lies in the lighter side of these tests. Finally, all γ-tests have a score of at most $\epsilon + \frac{1}{2}$. So the algorithm can choose any remaining α or β test at this point, proving the inductive step. Thus the expected cost of our algorithm is at least $(1 - \epsilon)(2k - 2) = \Omega(m)$.

Combined algorithm. By simply using the better of the two ODTN algorithms, we obtain:

Corollary III.14. *There is an $\mathcal{O}(\min(h, r) + \log m)$-approximation algorithm for adaptive ODTN, where h is the maximum number of noisy outcomes per hypothesis, and r is the maximum number of noisy outcomes per test.*

3.5 Handling Non-Identifiable Instances

We have assumed so far that the true hypothesis $\bar{x}$ can always be uniquely identified. In other words, for every pair x, y of hypotheses, there is some test e that distinguishes them deterministically, i.e. $x \in T^+(e)$ and $y \in T^-(e)$ or vice versa. In this section, we show that we can still obtain non-adaptive and adaptive algorithms (similar to Theorem III.2 and Corollary III.14) without this assumption. However, it is necessary to change the stopping criterion as we will have to stop with multiple compatible hypothesis on some decision paths.

Define a *similarity graph* G on m nodes (corresponding to hypotheses) with an edge (x, y) if there is *no* test separating x and y deterministically. Our algorithms' performance guarantees will now also depend on the maximum degree d of G; note that $d = 0$ in the perfectly identifiable case. For each hypothesis $i \in [m]$, let $D_i \subseteq [m]$ denote the set containing i and all its neighbors in G. We now define two stopping criteria as follows:

- The *neighborhood* stopping criterion involves stopping when the set H of compatible hypotheses is contained in *some* D_i, where i might or might not be the true hypothesis $\bar{x}$.
- The *clique* stopping criterion involves stopping when H is contained in some clique of G.

Note that clique stopping is clearly a stronger notion of identification than neighborhood stopping. That is, if the clique-stopping criterion is satisfied then so is the neighborhood-stopping criterion. We obtain a non-adaptive algorithm with approximation ratio $O((d +$

$1)\log m)$ for neighborhood-stopping and an adaptive algorithm with approximation ratio $O(d + \min(h, r) + \log m)$ for clique-stopping as well as neighborhood-stopping.

Non-adaptive algorithm. Here, we apply the approach using submodular ranking using different submodular functions. In particular, for each $(b, x) \in M$ and $i \in [m]$, we define a submodular function:

$$f^i_{b,x}(S) = |\bigcup_{e \in S} T_{b,x}(e) \cap \overline{D_i}| \cdot \frac{1}{m - |D_i|}, \qquad \forall S \subseteq U,$$

where $\overline{D_i} = [m] \setminus D_i$. Assuming $\bar{x} = x$ and test-outcomes are given by b, we know $x \in D_i$ after tests S if and only if $f^i_{b,x}(S) = 1$. We now define $f_{b,x}$ to be the "OR combination" of functions $f^i_{b,x}$ where $x \in D_i$, i.e.,

$$f_{b,x}(S) = 1 - \prod_{i: x \in D_i} (1 - f^i_{b,x}(S)), \qquad \forall S \subseteq U.$$

This function ensures that $f_{b,x}(S) = 1$ if and only if $f^i_{b,x}(S) = 1$ for some $i \in [m]$. Moreover, this function $f_{b,x}$ is monotone and submodular; see [45]. The minimum positive marginal increment $\epsilon = m^{-d}$ as $|\{i : x \in D_i| \leq d$ for all $x \in [m]$. The non-adaptive algorithm is then identical to that in § 3.3 and we obtain an $O(\log \frac{1}{\epsilon})$ approximation, i.e.,

Theorem III.15. *There is a non-adaptive $O(d \log m)$ approximation algorithm for ODTN with the neighborhood-stopping criterion.*

Adaptive algorithm Here, we run a two-phase algorithm. In the first phase, we identify some subset $N \subseteq [m]$ containing $\bar{x}$ with $|N| \leq d + 1$. This involves an ASR instance $\mathcal{J}'$ on the expanded hypothesis set M (as in § 3.4) but with the following submodular function for

each $(b, x) \in M$.

$$f_{b,x}(S) = |\bigcup_{e \in S} T_{b,x}(e)| \cdot \frac{1}{m-d-1}, \qquad \forall S \subseteq U.$$

Note that any adaptive policy for ODTN with either neighborhood-stopping or clique-stopping is also feasible for the ASR instance $\mathcal{J}'$. The proof is identical to that of Lemma III.7. Moreover, Algorithm 4 and 5 are also applicable to ASR instance $\mathcal{J}'$ (the only difference to instance $\mathcal{J}$ is the definition of the functions $f_{b,x}$). The parameter $\epsilon = \frac{1}{m-d-1} \geq \frac{1}{m}$. So, taking the better of Algorithms 4 and 5, we obtain an $O(\min(r, h) + \log m)$-approximation algorithm for $\mathcal{J}'$. Because the optimal value of instance $\mathcal{J}'$ is at most the optimal value OPT of the ODTN instance, the expected cost in this phase is $O(\min(r, h) + \log m) \cdot OPT$

Then, in the second phase, we run a simple splitting algorithm that iteratively selects any test e that splits the current set H of candidate hypotheses, i.e. $H \cap T_e^+ \neq \emptyset$ and $H \cap T_e^- \neq \emptyset$. The second phase continues until H is contained in (i) some clique (for clique-stopping) or (ii) some subset D_i (for neighborhood-stopping). Because, the number of compatible hypotheses $|H| \leq d + 1$ at the start of this phase, there are at most d iterations and the expected cost is at most $d \leq d \cdot OPT$. Combining both phases, we obtain:

Theorem III.16. *There is an adaptive $O(d + \min(h, r) + \log m)$-approximation algorithm for ODTN with the clique-stopping or neighborhood-stopping criterion.*

3.6 Adaptive Algorithm with Large Number of Noisy Outcomes

Our adaptive algorithms in the previous sections have a performance guarantee that depends on the noise sparsity $\min(r, h)$. This performance guarantee deteriorates when

there are a large number of noisy outcomes. In this section we consider ODTN instances with a large number of noisy outcomes, and obtain an approximation algorithm that relies on different ideas.

We define an α**-sparse** ($\alpha \leq 1/2$) ODTN instance as follows. There is a constant C such that $\max\{|T^+(e)|, |T^-(e)|\} \leq C \cdot m^\alpha$ for all tests $e \in U$. In words, the number of deterministic outcomes is at most $2Cm^\alpha$ for every test. Our main result here is:

Theorem III.17. *There is an adaptive $O(\log m)$-approximation algorithm for ODTN on any α-sparse instance, where $\alpha \leq \frac{1}{2}$.*

We first make some simple observations related to α-sparse instances.

Proposition III.18. *The optimal value $OPT \geq \Omega(m^{1-\alpha})$ for any α-sparse instance.*

Proof. By definition of α-sparse instances, the maximum number of candidate hypotheses that can be eliminated after performing a single test is m^α. As we need to eliminate $m-1$ hypotheses irrespective of the realized hypothesis $\bar{x}$, we need to perform at least $\frac{m-1}{m^\alpha} = \Omega(m^{1-\alpha})$ tests under every $\bar{x}$. □

Proposition III.19. *Consider any $W \subseteq U$ and $X \subseteq [m]$. For $y \in X$, let $\kappa(y) = |\{e \in W : r_y(e) \neq *\}|$ denote the number of tests in W for which y has ± 1 outcome. Then, the number of hypotheses in X with $\kappa(y) > |W|/2$ is at most $2Cm^\alpha$.*

Proof. Let $X' = \{y \in X : \kappa(y) > |W|/2\}$. Then:

$$|X'| \cdot \frac{|W|}{2} < \sum_{y \in X} \kappa(y) = \sum_{e \in W} |\{y \in X : r_y(e) \neq *\}| \leq |W| \cdot Cm^\alpha.$$

Rearraging, we get $|X'| \leq 2Cm^{\alpha}$ as needed. □

Main Idea. Consider the following naive algorithm: if H denotes the current set of candidate hypotheses, we choose a test $e \in U$ such that $\frac{1}{2}|T^{+}(e) \cap H| + \frac{1}{2}|T^{-}(e) \cap H|$ is maximized. Roughly speaking, this maximizes the expected number of hypotheses ruled out at each step. This algorithm can have a large cost. Nevertheless, it turns out that it makes good progress towards identifying hypotheses $x \in T^{*}(e)$; this is made formal in via the stochastic set cover instances in §3.6.1 and Lemma III.23. In particular, if at any point in this algorithm, a constant fraction of the tests so far have been $*$-tests for some hypothesis x then we can show that the algorithm has "low" cost conditioned on the true hypothesis $\bar{x} = x$. In order to handle hypotheses y with only a small fraction of $*$-tests, we modify the algorithm at all power-of-two steps: so the total number of modified step is $O(\log m)$. At each modified step, we collect $O(m^{\alpha})$ many hypotheses Z with the smallest number of $*$-tests performed so far, and check if $\bar{x} \in Z$. This is formalized in the algorithm Member(Z) in §3.6.2, where we also show that the additional cost incurred in each modified step is $O(m^{\alpha})$. Finally, using the lower bound in Proposition III.18 we can prove the $O(\log m)$ approximation ratio.

3.6.1 Relation to Stochastic Set Cover

We now establish a crucial relation to the stochastic set cover (SSC) problem [66, 55] and also state a useful result on SSC. This forms the basis of our algorithm.

An instance of SSC consists of a groundset V and k stochastic sets $S_1, \cdots S_k$ each of which is a subset of V. The distribution of each set S_i is known to the algorithm and the distributions of different sets are independent of each other. The instantiation of each set

is only known after it is selected. The goal is to find an adaptive policy that minimizes the expected number of sets to cover V. A natural adaptive greedy algorithm is known to be an $O(\log m)$-approximation where $m = |V|$ [66, 55]. At any point in this policy, if $A \subseteq V$ denotes the uncovered element s then we choose the set S_{i^*} that maximizes the expected number of new elements covered, i.e. $i^* = \arg\max_{i=1}^{k} \mathbb{E}[|S_i \cap A|]$. At any such step, we call a set β-greedy if it has expected coverage at least a $1/\beta$ fraction of the maximum. We need an extension of this result that involves picking β-greedy sets at just some constant fraction of the steps. Formally, call an SSC policy (β, ρ) greedy if for every $t \geq 1$, at least t/ρ steps among the first t steps involve picking a β-greedy set. The following result follows from a slight modification of the analysis in [55].

Theorem III.20 ([55])**.** *For any stochastic set cover instance, a (β, ρ) greedy policy costs at most $O(\beta\rho \log m)$ times the optimum.*

We now derive a lower bound on the optimal ODTN value in terms of certain SSC instances. For any $x \in [m]$, let $SSC(x)$ denote the stochastic set cover instance with groundset $V = [m] \setminus \{x\}$ (i.e. all hypotheses other than x) and sets U (i.e. all tests) where

$$S_e(x) = \begin{cases} T^+(e) \text{ with prob. } 1 & \text{if } x \in T^-(e) \\ T^-(e) \text{ with prob. } 1 & \text{if } x \in T^+(e) \\ T^-(e) \text{ or } T^+(e) \text{ with prob. } \frac{1}{2} \text{ each} & \text{if } x \in T^*(e) \end{cases}, \qquad \forall e \in U.$$

Lemma III.21. $OPT \geq \sum_{x \in [m]} \pi_x \cdot OPT_{SSC(x)}$.

Proof. Consider any feasible decision tree $\mathcal{T}$ for the ODTN instance and any hypothesis $x \in [m]$. If we *condition* on $\bar{x} = x$ then $\mathcal{T}$ corresponds to a feasible adaptive policy for $SSC(x)$. This is because (i) for any outcome-vector $b \sim x$, the tests performed in $\mathcal{T}$ must rule-out all the hypotheses $[m] \setminus x$ and (ii) the hypotheses ruled-out by any test e (conditioned on $\bar{x} = x$) is a random subset that has the same distribution as $S_e(x)$. Formally, let $P_{b,x}$ denote the path traced in $\mathcal{T}$ under test outcomes $b \sim x$, and $|P_{b,x}|$ the number of tests performed along this path. Then, this policy for $SSC(x)$ has cost $\sum_{b \sim x} 2^{-h_x} \cdot |P_{b,x}|$ as $\Pr[\text{observing outcomes } b | \bar{x} = x] = 2^{-h_x}$. So $OPT_{SSC(x)} \leq \sum_{b \sim x} 2^{-h_x} \cdot |P_{b,x}|$. Taking expectations over $x \in [m]$ the lemma follows. □

3.6.2 A Low-Cost Membership Oracle

One subroutine in our algorithm is an "oracle" called "Member", which takes a (small) subset $Z \subseteq [m]$ as input, and decides whether $\bar{x} \in Z$.

Note that steps 3, 9 and 18 are well-defined because the ODTN instance is assumed to be identifiable. If there is no new test in step 3 with $T^+(e) \cap Z' \neq \emptyset$ and $T^-(e) \cap Z' \neq \emptyset$, then we must have $|Z'| = 1$. If there is no new test in step 9 with $z \notin T^*(e)$ then we must have identified z uniquely, i.e. $Y = \emptyset$. Finally, in step 18, we use the fact that there are tests that deterministically separate every pair of hypotheses.

Lemma III.22. *If $\bar{x} \in Z$, then Member(Z) declares $\bar{x} = z$ with probability one; otherwise, Member(Z) declares $\bar{x} \notin Z$ with probability at least $1 - m^{-2}$. Moreover, the expected cost of Member(Z) is $O(|Z| + m^{\alpha})$.*

Algorithm 7 Member(Z) oracle that checks if $\bar{x} \in Z$.

initialize $Z' \leftarrow Z$.
while $|Z'| > 1$ **do**
    choose any new test $e \in U$ with both $T^+(e) \cap Z' \neq \emptyset$ and $T^-(e) \cap Z' \neq \emptyset$,
    let R be the set of hypotheses ruled out, let $Z' \leftarrow Z' \backslash R$.
end while
let $Z' = \{z\}$ at this step.
initialize $k \leftarrow 0$ and $Y \leftarrow \{x \in [m] \setminus \{z\} : x \text{ not ruled out so far}\}$.
while $Y \neq \emptyset$ and $k \leq 4 \log m$ **do**
    choose any new test $e \in U$ with $z \notin T^*(e)$.
    if outcome of test e is inconsistent with z **then**
        declare "$\bar{x} \notin Z$" and stop.
    else
        let R be the set of hypotheses ruled out, $Y \leftarrow Y \setminus R$ and $k \leftarrow k + 1$.
    end if
end while
if $Y = \emptyset$ then declare "$\bar{x} = z$" and stop.
let $W \subseteq U$ denote the tests performed in step 9 and

$$S = \{y \in [m] : r_y(e) = r_z(e) \text{ for at least } 2 \log m \text{ tests } e \in W\}. \tag{3.15}$$

for each $y \in S$, choose any test that deterministically separates y and z; let $W' \subseteq U$ denote the set of these tests.
if all tests in $W \cup W'$ had an outcome consistent with z **then**
    declare "$\bar{x} = z$".
else
    declare "$\bar{x} \notin Z$".
end if

Proof. If $\bar{x} \in Z$ then it is clear that $z = \bar{x}$ in step 6 and Member(Z) declares $\bar{x} = z$.

Now consider the case $\bar{x} \notin Z$. Recall that $z \in Z$ denotes the unique hypothesis that is still compatible in step 6. Note that Y denotes the set of compatible hypotheses among $[m] \setminus \{z\}$: so it always contains $\bar{x}$. Hence, $Y \neq \emptyset$ in step 16, which implies that $k = 4 \log m$. Recall the definition of set S from (3.15).

Case 1 If $\bar{x} \notin S$ then we have $\bar{x} \in T^*(e)$ for at least $2 \log m$ tests $e \in W$. As z has a deterministic outcome for each test in W, the probability that all outcomes in W are consistent with z is at most m^{-2}. So with probability at least $1 - m^{-2}$, some test in W must have an outcome (under $\bar{x}$) inconsistent with z, and based on step 19, we would declare $\bar{x} \notin Z$.

Case 2 If $\bar{x} \in S$ then we will identify correctly that $\bar{x} \neq z$ in step 19 as one of the tests in W' (step 18) separates $\bar{x}$ and z deterministically. So in this case we will always declare $\bar{x} \notin Z$.

In order to bound the cost, note that the number of tests performed are at most: $|Z|$ in step 3, $4 \log m$ in step 9 and $|S|$ in step 18. The key step is to bound $|S|$, for which we use Proposition III.19 with tests W and hypotheses $X = [m]$. Note that the tests in step 18 are only performed if $Y \neq \emptyset$ in step 16: so we must have $|W| = k = 4 \log m$. In the notation of Proposition III.19, we have $S = \{y \in X : \kappa(y) > |W|/2\}$ as $|W| = 4 \log m$. It now follows that $|S| \leq 2Cm^{\alpha}$. Hence the total number of tests is $|Z| + 4 \log m + |S| = O(|Z| + m^{\alpha})$. □

3.6.3 The Main Algorithm

We now describe the overall algorithm.

Note that the membership oracle is invoked $O(\log m)$ times as the total number t of tests

Algorithm 8 Overall algorithm for large number of noisy outcomes.

1: initialize compatible hypotheses $H \leftarrow [m]$, weights $w_x = 0$ for $x \in H$, number of tests $t \leftarrow 0$.
2: **while** $|H| > 1$ **do**
3: **if** t is a power of 2 **then**
4: let $Z \subseteq A$ be the subset of $2Cm^\alpha$ hypotheses with lowest w_x.
5: call Member(Z): if it identifies some Z-hypothesis as $\bar{x}$ then stop.
6: **end if**
7: perform test $e \in U$ maximizing $\frac{1}{2}(|T^+(e) \cap H| + |T^-(e) \cap H|)$.
8: update $w_x \leftarrow w_x + 1$ for each for each $x \in T^*(e)$.
9: remove incompatible hypotheses from H (based on the test e outcome).
10: $t \leftarrow t + 1$.
11: **end while**

used is always at most m. Using Lemma III.22, it is clear that $\bar{x}$ is identified correctly with probability at least $1 - \frac{1}{m}$. We now analyze the cost. The oracle Member is always invoked on $|Z| = O(m^\alpha)$ hypotheses. Using Lemma III.22, the expected number of tests due to step 4 is $O(m^\alpha \log m)$. In the rest of this section, we will bound the expected cost due to tests in step 7.

Truncated Decision Tree. Let $\mathcal{A}$ denote the decision tree corresponding to our algorithm. We only consider tests that correspond to step 7. For any expanded hypothesis $(b, x) \in M$ let $P_{b,x}$ denote the path traced in $\mathcal{A}$; so $|P_{b,x}|$ is the number of tests performed in step 7 under hypothesis (b, x). We will work with a truncated decision tree $\bar{\mathcal{A}}$, defined below.

Fix any $(b, x) \in M$. For any $t \geq 1$, let $\theta_{b,x}(t)$ denote the fraction of the first t tests in $P_{b,x}$ that are $*$-tests for hypothesis x. Recall that $P_{b,x}$ only contains tests from step 7. Let $\rho = 4$ and

$$\text{define } t_{b,x} = \max\left\{ t \in \{2^0, 2^1, \cdots, 2^{\log m}\} \ : \ \theta_{b,x}(t') \geq \frac{1}{\rho} \text{ for all } t' \leq t \right\}. \tag{3.16}$$

If $t_{b,x} > |P_{b,x}|$ then set $t_{b,x} = |P_{b,x}|$. The truncated decision tree $\bar{\mathcal{A}}$ is the subtree of $\mathcal{A}$

consisting of the first $t_{b,x}$ tests along path $P_{b,x}$, for each $(b, x) \in M$.

Relating costs of $\mathcal{A}$ and $\bar{\mathcal{A}}$. We show that the expected cost of $\mathcal{A}$ is at most twice that of $\bar{\mathcal{A}}$.

Lemma III.23. *For each $(b, x) \in M$, the number of tests performed $|P_{b,x}| \leq 2 \cdot t_{b,x}$. Hence, the expected cost of $\mathcal{A}$ is at most twice that of $\bar{\mathcal{A}}$.*

Proof. For the first statement, fix any $(b, x) \in M$. Recall that $P_{b,x}$ only contains tests from step 7. We only need to consider the case that $t_{b,x} < |P_{b,x}|/2$. Let $t'_{b,x} = 2 \cdot t_{b,x}$ which is a power-of-2. By (3.16) we know that there is some k with $t_{b,x} < k \leq t'_{b,x}$ and $\theta_{b,x}(k) < 1/\rho$. Hence $\theta_{b,x}(t'_{b,x}) < \frac{2}{\rho} < \frac{1}{2}$.

Consider the point in the algorithm after performing the first $t'_{b,x}$ tests (call them W) on $P_{b,x}$. Because $t'_e x$ is a power-of-two, the algorithm calls the member oracle in this iteration. Let X be the compatible hypotheses after the $t'_{b,x}$-th test on $P_{b,x}$. Because $\theta_{b,x}(t'_{b,x}) < 1/2$, at most $|W|/2$ tests in W are $*$-tests for hypothesis x: in other words the weight $w_x \leq |W|/2$ at this point in the algorithm. Let

$$X' = \left\{ y \in X : W \text{ has at most } \frac{|W|}{2} * \text{-tests for } y \right\} = \left\{ y \in X : w_y \leq \frac{|W|}{2} \right\}.$$

Using Proposition III.19 with W and X, it follows that $|X'| \leq 2Cm^\alpha$. Hence the number of hypotheses $y \in X$ with $w_y \leq |W|/2 \leq w_x$ is at most $2Cm^\alpha$, and so $x \in Z$ (recall that Z consists of $2Cm^\alpha$ hypotheses with the lowest weight). This means that after step 4, we would have correctly identified $\bar{x} = x$ and so $P_{b,x}$ ends. Hence $|P_{b,x}| \leq t'_{b,x}$. The first part of the lemma now follows from the fact that $t'_{b,x} \leq 2 \cdot t_{b,x}$.

The second statement in the lemma follows by taking expectation over all $(b, x) \in M$. $\square$

Bounding cost of $\bar{\mathcal{A}}$. Here we use the relation to stochastic set cover. Recall the instances $SSC(x)$ for $x \in [m]$. A key observation is:

Lemma III.24. *Consider step 7 in the main algorithm, where H denotes the set of compatible hypotheses and e is the chosen test. For any $x \in T^*(e)$,*

$$\frac{1}{2}\left(|T^+(e) \cap H| + |T^-(e) \cap H|\right) = \mathbb{E}[|S_e(x) \cap (H \setminus x)|] \geq \frac{1}{2} \cdot \max_{d \in U} \mathbb{E}[|S_d(x) \cap (H \setminus x)|].$$

Proof. Note that $\mathbb{E}[|S_e(x) \cap (H \setminus x)|] = \frac{1}{2}\left(|T^+(e) \cap H| + |T^-(e) \cap H|\right)$ because $x \in T^*(e)$. We consider two cases for test $d \in U$.

- If $x \in T^*(d)$ then

  $$\mathbb{E}[|S_d(x) \cap (H \setminus x)|] = \frac{1}{2}\left(|T^+(d) \cap H| + |T^-(d) \cap H|\right) \leq \frac{1}{2}\left(|T^+(e) \cap H| + |T^-(e) \cap H|\right),$$

  by the choice of e in step 7.

- If $x \in T^+(d) \cup T^-(d)$ then

  $$\mathbb{E}[|S_d(x) \cap (A \setminus x)|] \leq \max\{|T^+(d) \cap H|, |T^-(d) \cap H|\} \leq |T^+(d) \cap H| + |T^-(d) \cap H|,$$

  which is at most $|T^+(e) \cap H| + |T^-(e) \cap H|$ by the choice of e.

In either case, we obtain the lemma. $\square$

Fix any hypothesis $x \in [m]$ and consider decision tree $\bar{\mathcal{A}}_x$ obtained by *conditioning* $\bar{\mathcal{A}}$ on $\bar{x} = x$. The truncation (3.16) and Proposition III.24 together imply that $\bar{\mathcal{A}}_x$ is a $(2, \rho)$

greedy policy for $SSC(x)$. Now, using Theorem III.20, the expected cost of $\bar{\mathcal{A}}_x$ is $O(\log m) \cdot OPT_{SSC(x)}$.

Taking expectations over $x \in [m]$, the expected cost of $\bar{\mathcal{A}}$ is $O(\log m) \sum_{x=1}^{m} \pi_x \cdot OPT_{SSC(x)}$, which is $O(\log m) \cdot OPT$ by Lemma III.21. Combined with Lemma III.23, the expected cost of $\mathcal{A}$ is at most $O(\log m) \cdot OPT$. This bounds the cost dues to tests in step 7. Finally, adding in the contribution from step 4, we obtain an expected cost of

$$O(\log m) \cdot (m^{\alpha} + OPT) \leq_{(\text{as } \alpha < \frac{1}{2})} O(\log m) \cdot (m^{1-\alpha} + OPT) \leq_{(\text{Prop. } III.18)} O(\log m) \cdot OPT.$$

This completes the proof of Theorem III.17.

3.7 Extensions

Non-binary outcomes. We can also handle tests with an arbitrary set Σ of outcomes (instead of ± 1). This requires extending the outcomes b to be in Σ^U and applying this change to the definitions of sets $T_{b,x}$ (3.1) and submodular function $f_{b,x}$ (3.2).

Non-uniform noise distribution. Our results extend directly to the case where each noisy outcome has a different probability of being ± 1. Suppose that the probability of every noisy outcome is between δ and $1-\delta$. Then Theorems III.2 and III.14 continue to hold (irrespective of δ), and Theorem III.17 holds with a slightly worse $O(\frac{1}{\delta} \log m)$ approximation ratio.

3.8 Experiments

We implemented our algorithms, and performed experiments on real-world and synthetic data sets. We compared our algorithms' cost (expected number of tests) with an information

theoretic lower bound on the optimal cost and show that the difference is negligible. Thus, despite our logarithmic approximation ratios, the practical performance can be much better.

Chemicals with Unknown Test Outcomes One application of ODT is identifying chemical or biological materials. We considered a data set called WISER[4] (and we call it WISER-ORG here), which includes 400+ chemicals (hypothesis) and 78 binary tests. Every chemical has either positive, negative or unknown result on each test. We have performed our algorithms on both the original instance, in which some chemicals are not perfectly identifiable, and a modified version. In the modified version, to ensure every pair of chemicals can be distinguished, we removed the chemicals that are not identifiable from each other to obtain WISER-ID dataset with 255 chemicals (to do this, we used a greedy rule that iteratively drops the highest-degree hypothesis in the similarity graph).

Random Binary Classifiers with Margin Error We construct a dataset containing 100 two-dimensional points, by picking each of their attributes uniformly in $[-1000, 1000]$. We also choose 2000 random triples (a, b, c) to form linear classifiers $\frac{ax+by}{\sqrt{a^2+b^2}} + c \leq 0$, where $a, b \leftarrow N(0, 1)$ and $c \leftarrow U(-1000, 1000)$. The point labels are binary and we introduce noisy outcomes based on the distance of each point to a classifier. Specifically, for each threshold $d \in \{0, 5, 10, 20, 30\}$ we define dataset CL-d that has a noisy outcome for any classifier-point pair where the distance of the point to the boundary of the classifier is smaller than d. In order to ensure that the instances are perfectly identifiable, we remove "equivalent" classifiers

[4]https://wiser.nlm.nih.gov

and we are left with 234 classifiers.

Distributions For the distribution over the hypotheses we have considered permutations of power law distribution ($\Pr[X = x; \alpha] = \beta x^{-\alpha}$) for $\alpha = 0, 0.5$ and 1. Note that, $\alpha = 0$ corresponds to uniform distribution. To be able to compare the results across different classifiers' datasets meaningfully, we have considered the same permutation in each distribution.

Algorithms We implement the following algorithms: the adaptive $O(r+\log m)$-approximation (ODTN$_r$), the adaptive $O(h + \log m)$-approximation (ODTN$_h$), the non-adaptive $O(\log m)$-approximation (Non-Adap) and a slightly adaptive version of Non-Adap (Low-Adap). Algorithm Low-Adap considers the same sequence of tests as Non-Adap while (adaptively) skipping non-informative tests based on observed outcomes. The implementations of the adaptive and non-adaptive algorithms are available online.[5] We also consider three different stopping criteria: *unique* stopping for perfectly identifiable instances, *neighborhood* and *clique* stopping (defined in Section 3.5) for WISER-ORG dataset.

Algorithm \ Data	WISER-ID	Cl-0	Cl-5	Cl-10	Cl-20	Cl-30
Low-BND	**7.994**	**7.870**	**7.870**	**7.870**	**7.870**	**7.870**
ODTN$_r$	8.357	7.910	7.927	7.915	7.962	8.000
ODTN$_h$	9.707	7.910	7.979	8.211	8.671	8.729
Non-Adap	11.568	9.731	9.831	9.941	9.996	10.204
Low-Adap	9.152	8.619	8.517	8.777	8.692	8.803

Table 3.1: Cost of Different Algorithms for $\alpha = 0$ (Uniform Distribution).

[5] https://github.com/FatemehNavidi/ODTN ; https://github.com/sjia1/ODT-with-noisy-outcomes

Algorithm \ Data	WISER-ID	Cl-0	Cl-5	Cl-10	Cl-20	Cl-30
ODTN$_r$	0.008	0	0	0.002	0.003	0.006
ODTN$_h$	0.01	0	0	0	0.004	0.01
Non-Adap	1.463	0.937	1.047	1.092	1.056	1.158
Low-Adap	0.0317	0.0685	0.0541	0.0760	0.0206	0.0550

Table 3.2: Standard Deviation of Different Algorithms for $\alpha = 0$ (Uniform Distribution).

Algorithm \ Data	WISER-ID	Cl-0	Cl-5	Cl-10	Cl-20	Cl-30
Low-BND	7.702	7.582	7.582	7.582	7.582	7.582
ODTN$_r$	8.177	7.757	7.780	7.789	7.831	7.900
ODTN$_h$	9.306	7.757	7.829	8.076	8.497	8.452
Non-Adap	11.998	9.504	9.500	9.694	9.826	9.934
Low-Adap	8.096	7.837	7.565	7.674	8.072	8.310

Table 3.3: Cost of Different Algorithms for $\alpha = 0.5$.

Algorithm \ Data	WISER-ID	Cl-0	Cl-5	Cl-10	Cl-20	Cl-30
Low-BND	6.218	6.136	6.136	6.136	6.136	6.136
ODTN$_r$	7.367	6.998	7.121	7.150	7.299	7.357
ODTN$_h$	8.566	6.998	7.134	7.313	7.637	7.915
Non-Adap	11.976	9.598	9.672	9.824	10.159	10.277
Low-Adap	9.072	8.453	8.344	8.609	8.683	8.541

Table 3.4: Cost of Different Algorithms for $\alpha = 1$.

Parameters \ Data	WISER-ORG	WISER-ID	Cl-0	Cl-5	Cl-10	Cl-20	Cl-30
r	388	245	0	5	7	12	13
Avg-r	50.46	30.690	0	1.12	2.21	4.43	6.54
h	61	45	0	3	6	8	8
Avg-h	9.51	9.39	0	0.48	0.94	1.89	2.79

Table 3.5: Maximum and Average Number of Stars per Hypothesis and per Test in Different Datasets.

Results Tables 3.1, Tables 3.3 and Tables 3.4 show the expected costs of different algorithms on all uniquely identifiable datasets when the parameter α in the distribution over hypothesis is $0, 0.5$ and 1 correspondingly. These tables also report values of an information

Algorithm	Neighborhood Stopping	Clique Stopping
ODTN_r	11.163	11.817
ODTN_h	11.908	12.506
Non-Adap	16.995	21.281
Low-Adap	16.983	20.559

Table 3.6: Cost of Algorithms on WISER-ORG dataset with Neighborhood and Clique Stopping for Uniform Distribution.

theoretic lower bound (the entropy) on the optimal cost (Low-BND). We also report the sample standard deviation of all algorithms for uniform distribution in Tables 3.2. Since the approximation ratio of some of our algorithms are dependent on maximum number of unknown outcomes per hypothesis (h) and maximum number of unknown outcomes per test (r), we also have included these parameters as well as their average values. Table 3.6 summarizes the results on WISER-ORG with clique and neighborhood stopping criteria. We can see that ODTN_r consistently outperforms the other algorithms and is very close to the lower bound. Note that WISER-ORG dataset that is used to produce results in Table 3.6 has $m = 414$ hypotheses and $d = 54$ in its similarity graph, while WISER-ID in Table 3.1 is perfectly identifiable with $m = 255$ hypotheses.

CHAPTER IV

A Priori Traveling Repairman Problem

The results in this chapter appear in [42].

4.1 Introduction

A priori optimization ([15]) is an elegant model for stochastic combinatorial optimization, that is particularly useful when one needs to repeatedly solve instances of the same optimization problem. The basic idea here is to reduce the computational overhead of solving repeated problem instances by performing suitable pre-processing using distributional information. More specifically, in an *a priori* optimization problem, one is given a probability distribution Π over inputs and the goal is to find a "master solution" τ. Then, after observing the random input A (drawn from the distribution Π), the master solution τ is modified using a simple rule to obtain a solution τ_A for that particular input. The objective is to minimize the expected value of the master solution. For a problem with objective function ϕ, we are interested in:

$$\min_{\tau:\text{master solution}} \mathbb{E}_{A\leftarrow\Pi}\left[\phi(\tau_A)\right].$$

This chapter studies the *a priori* traveling repairman problem. The traveling repairman

problem (TRP) is a fundamental vehicle routing problem that involves computing a tour originating from a depot/root that minimizes the sum of latencies (i.e. the distance from the root on this tour) at all vertices. The TRP is also known as the traveling deliveryman or minimum latency problem, and has been studied extensively, see e.g. [73], [33], [36]. In the *a priori* TRP, the master solution τ is a tour visiting all vertices, and for any random input (i.e. subset A of vertices), the solution τ_A is simply obtained by visiting the vertices of A in the order given by τ.

An *a priori* solution is advantageous in settings when we repeatedly solve instances of the TRP that are drawn from a common distribution. For example, we may need to solve one TRP instance on each day of operations, where the distribution over instances is estimated from historical data. Using an *a priori* solution saves on computation time as we do not have to solve each instance from scratch. Moreover, for vehicle routing problems (VRPs) there are also practical advantages to have a pre-planned master tour, e.g. drivers have familiarity with the route followed each day. See [76], [19], and [31] for more discussion on the benefits of a pre-planned VRP solution.

4.1.1 Problem Definition.

The traveling repairman problem (TRP) is defined on a finite metric (V, d) where V is a vertex set and $d : V \times V \to \mathbb{R}_+$ is a distance function. We assume that the distances are symmetric and satisfy triangle inequality. There is also a designated root vertex $r \in V$. The goal is to find a tour τ originating from r that visits all vertices. The *latency* of any vertex v in tour τ is the length of the path from r to v along τ. The objective in TRP is to minimize

the sum of latencies of all vertices.

In the *a priori* TRP, in addition to the above input we are also given activation probabilities $\{p_v\}_{v\in V}$ at all vertices; we use Π to denote this distribution. In this chapter, as in most prior works on *a priori* optimization, we assume that the input distribution Π is independent accross vertices. So the active subset $A \subseteq V$ contains each vertex $v \in V$ independently with probability p_v. A solution to *a priori* TRP is a master tour τ originating from r and visiting all vertices. Given an active subset $A \subseteq V$, we restrict tour τ to vertices in A (by shortcutting over $V \setminus A$) to obtain tour τ_A, again originating from r. See Figure 4.1 for an example. For each $v \in A$, we use $\mathsf{LAT}_\tau^A(v)$ to denote the latency of v in tour τ_A. We also use $\mathsf{LAT}_\tau^A = \sum_{v\in A} \mathsf{LAT}_\tau^A(v)$ for the total latency under active subset $A \subseteq V$. The objective is to minimize

$$\mathsf{ELAT}_\tau = \mathbb{E}_{A\leftarrow\Pi}\left[\mathsf{LAT}_\tau^A\right] = \mathbb{E}_{A\leftarrow\Pi}\left[\sum_{v\in A} \mathsf{LAT}_\tau^A(v)\right] .$$

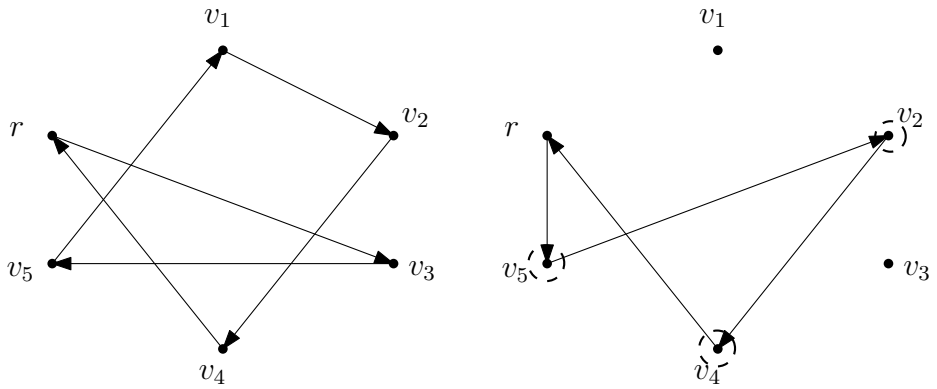

Figure 4.1: From left to right: the master tour τ and tour τ_A for active set $A = \{v_2, v_4, v_5\}$

There are two ways to evaluate an algorithm's approximation ratio for this problem. One option is to compare the expected cost of the a priori tour to the optimal offline solution,

which is the optimal *a priori* TRP solution:

$$\mathsf{OPT} = \min_{\tau} \mathsf{ELAT}_{\tau}$$

The second option is to compare the expected cost of the a priori tour with the *re-optimization* cost, which involves computing the optimal TRP solution for every possible active subset $A \subseteq V$. Formally, let OPT_A denote the optimal total latency of TRP instance with demands at subset $A \subseteq V$. Then, the re-optimization cost is:

$$\mathsf{ROPT} = E_A[\mathsf{OPT}_A] = \sum_{A \subseteq V} \left(\prod_{u \in A} p_u \prod_{w \in V \setminus A} (1 - p_w) \right) \cdot \mathsf{OPT}_A \ .$$

Note that the term in the parenthesis is the probability that A is the active subset. As OPT_A is the minimum total latency on set A, for any tour τ we have:

$$\mathsf{OPT}_A \leq \sum_{v \in A} \mathsf{LAT}_{\tau}^{A}(v)$$

In particular if τ is the optimal master tour, this results in:

$$\mathsf{ROPT} \leq \mathsf{OPT}$$

4.1.2 Results.

Our main results in this chapter are the first constant-factor approximation for the *a priori* TRP with respect to the optimal offline solution, as well as the first $\mathcal{O}(\log(n))$-approximation algorithm with respect to the re-optimization solution. In Section 4.2 we explain the algorithm and analysis for the first result. More precisely we prove the following theorem:

Theorem IV.1. *There is a constant-factor approximation algorithm for the* a priori *traveling repairman problem under independent probabilities.*

Previously, [84] obtained such a result under the restriction that all activation probabilities are *identical*, and posed the general case of non-uniform probabilities as an open question– which we resolve. Our result adds to the small list of *a priori* VRPs with provable worst-case guarantees: traveling salesman, capacitated vehicle routing and traveling repairman.

In fact, we obtain Theorem IV.1 by a generic reduction of *a priori* TRP from non-uniform to uniform probabilities, formalized below.

Theorem IV.2. *There is a* (6.27ρ)*-factor approximation algorithm for the* a priori *traveling repairman problem under independent probabilities, where* ρ *denotes the best approximation ratio for the problem under uniform probabilities.*

Clearly, Theorem IV.1 follows by combining Theorem IV.2 with the constant-factor approximation algorithm for *a priori* TRP under uniform probabilities by [84]. As the constant factor in [84] for uniform probabilities is quite large, there is the possibility of improving it using a different algorithm: Theorem IV.2 would be applicable to any such future improvement and yield a corresponding improved result for non-uniform probabilities.

In Section 4.3, we state an algorithm and compare its solution to the re-optimization solution. More formally, we prove the following theorem:

Theorem IV.3. *There is an* $\mathcal{O}(\log n)$*-approximation algorithm for the* a priori *traveling repairman problem under independent probabilities with respect to the re-optimization solution, where* n *denotes the number of vertices.*

To do that, first we provide a constant-factor approximation algorithm with respect to

the re-optimization solution if our metric is a relaxed 2-Hierarchically well Separated Tree (2-HST). More specifically, we show the following theorem:

Theorem IV.4. *There is a constant-factor approximation algorithm for the* a priori *traveling repairman problem with a relaxed 2-HST metric under independent probabilities with respect to the re-optimization solution.*

Then, we prove the following result:

Theorem IV.5. *If there is an $\mathcal{O}(\alpha)$-approximation algorithm for a priori TRP on a relaxed 2-HST w.r.t the re-optimization solution, then there is an $\mathcal{O}(\alpha \cdot \log n)$-approximation algorithm for a priori TRP on a general metric w.r.t the re-optimization solution.*

Finally, by combining the above Theorems, we can prove Theorem IV.3.

4.1.3 Related Work.

The *a priori* optimization model was introduced in [57] and [13], see also the survey by [15]. These papers considered the setting where the metric is itself random and carried out asymptotic analysis (as the number of vertices grows large). They obtained such results for the minimum spanning tree, traveling salesman, capacitated vehicle routing and traveling salesman facility location problems.

Approximation algorithms for *a priori* optimization are more recent: these can handle arbitrary problem instances. Such results are known for the traveling salesman problem (TSP), capacitated VRP and traveling repairman (TRP). We briefly discuss them below.

The *a priori* TSP has been extensively studied. In particular, there is a randomized 4-approximation algorithm for independent probabilities by [79]. The same paper also gave a deterministic 8-approximation algorithm; the constant was later improved to 6.5 in [85]. These algorithms were based on a random-sampling approach ([47, 86]) that was previously used in other network design problems. For arbitrary (black-box) distributions, [77] gave a randomized $O(\log n)$-approximation algorithm which actually does not even need any knowledge on the distribution. Later, [41] proved an $\Omega(\log n)$ lower bound on the approximation ratio of any deterministic algorithm for *a priori* TSP under arbitrary distributions.

The capacitated VRP with stochastic demands ([14]) is another well-studied *a priori* optimization problem. Here, we have a vehicle with limited capacity Q at the root that needs to satisfy demands at various vertices. The demand at each vertex is an independent random variable with a known distribution. A master solution to this problem is a tour τ that visits every vertex; after demands are observed, the vehicle visits vertices in the same order as τ while performing additional refill-trips to the root whenever it runs out of items. The objective is to minimize the total length of the tour. A 2.5-approximation algorithm for this problem in the case of *identical* demand distributions was given in [14]. Later, [48] obtained a randomized 2.5-approximation algorithm for this problem under non-identical distributions.

The *a priori* TRP was recently studied in [84], where a constant-factor approximation algorithm was obtained for the case of uniform independent probabilities. They left open the problem under non-uniform probabilities: Theorem IV.2 resolves this positively. The algo-

rithm in [84] was based on many ideas from the deterministic TRP, but it needed stochastic counterparts of various properties. As noted in [84], their proof relied heavily on the probabilities being uniform and it was unclear how to handle non-uniform probabilities.

We note that the deterministic traveling repairman problem (TRP) has been studied extensively, both in exact algorithms ([73, 33, 88]) and approximation algorithms ([17], [36], [4], [22]). It was shown to be NP-hard even on weighted trees by [81], and a polynomial time approximation scheme (PTAS) on such metrics was given by [82]. On general metrics, the best approximation ratio known is 3.59 due to [22]; it is also known that one cannot obtain a PTAS.

4.2 *A Priori* TRP with Respect to the Optimal Offline Solution

Consider an instance $\mathcal{I}$ of *a priori* TRP on metric (V, d) with probabilities $\{p_v\}_{v \in V}$. We show how to "reduce" this instance to one with uniform probabilities, which would prove Theorem IV.2. Our approach is natural: we replace each vertex $v \in V$ with a group S_v of co-located vertices, where each new vertex is active with a uniform probability p. Let $\mathcal{J}$ denote the new instance and $(\widehat{V}, d)$ the new metric. Intuitively, when p is chosen much smaller than the p_vs and $|S_v| \approx p_v/p$, the scaled uniform instance $\mathcal{J}$ should behave similar to $\mathcal{I}$. However, proving such a result formally requires significant technical work. For example, the master tour found by an algorithm for the scaled (uniform) instance might not visit all the co-located copies consecutively. We define a *consecutive* master tour for $\mathcal{J}$ as one that visits all co-located vertices consecutively. Then, we show an approximate equivalence

between (i) master tours in $\mathcal{I}$ and (ii) consecutive master tours in $\mathcal{J}$. This relies on the independence across vertices and the correspondence between the events "vertex v is active in $\mathcal{I}$" and "at least one vertex of S_v is active in $\mathcal{J}$". This is formalized in Section 4.2.2. Then, we show in Section 4.2.4 that any master tour for instance $\mathcal{J}$ can be modified to a "consecutive" master tour with the same or better overall expected latency. Finally, in order to maintain a polynomial-size instance $\mathcal{J}$ (this is reflected in the choice of p), we need to take care of vertices with very small probability separately. In Section 4.2.3 we show that the overall effect of the small-probability vertices is tiny if they are visited in non-decreasing order of distances at the end of our master tour.

Algorithm 9 Reducing non-uniform instance $\mathcal{I}$ to uniform instance $\mathcal{J}$

1: $Y \leftarrow \{v \in V \,|\, p_v < 1/n^2\}$ denotes the low probability vertices.
2: $X \leftarrow \{v \in V \,|\, p_v \geq 1/n^2\}$ denotes all other vertices.
3: $p \leftarrow \frac{1}{n} \min_{v \in X} p_v$
4: Construct instance $\mathcal{J}$ with vertex set $\widehat{V}$ that contains for each $v \in X$, a set S_v of $t_v = \lceil \frac{p_v}{p} \rceil$ copies of v. The distance between any two vertices of S_v is zero for all $v \in X$. The distance between any vertex of S_u and any vertex of S_v is $d(u, v)$. All vertices in $\widehat{V}$ have a uniform activation probability p.
5: Run any approximation algorithm for *uniform a priori* TRP on $\mathcal{J}$ to obtain master tour $\widehat{\pi}$.
6: Run procedure MakeConsecutive($\widehat{\pi}$) to ensure that $\widehat{\pi}$ visits each group S_v consecutively.
7: Obtain tour π by visiting vertices of X in the same order that S_vs are visited in $\widehat{\pi}$.
8: Extend π by visiting vertices $w \in Y$ in *non-decreasing* order of $d(r, w)$, to obtain tour $\bar{\pi}$.
9: **return** $\bar{\pi}$.

Algorithm 9 describes the reduction formally. In Step 6, Algorithm 9 relies on a procedure MakeConsecutive that modifies tour $\widehat{\pi}$ such that it visits all copies of the same node consecutively. We will prove Theorem IV.2 by analyzing this algorithm.

4.2.1 Overview of Analysis.

We first assume that the master tour $\widehat{\pi}$ on instance $\mathcal{J}$ already visits copies of each vertex consecutively: so there is no need for Step 6. We split this proof into two parts corresponding

to the X-vertices (normal probabilities) and Y-vertices (low probabilities). The analysis for X-vertices (Section 4.2.2) is the main part, where we show that the optimal values of $\mathcal{I}$ and $\mathcal{J}$ are within a constant factor of each other. In Lemma IV.8 we show that a constant-factor perturbation in probabilities of V will only change the cost of any solution (including the optimal) by a constant factor. Then we prove (in Lemma IV.9) that the optimal value of instance $\mathcal{J}$ is within a constant factor of the optimal value of $\mathcal{I}$: although $\mathcal{J}$ has many more vertices than $\mathcal{I}$, the proof exploits the fact that the expected number of active vertices is roughly the same as $\mathcal{I}$. Lemma IV.10 proves the other direction for the cost of our algorithm, i.e. the cost of Algorithm 9 for $\mathcal{I}$ is at most that of the consecutive master tour for $\mathcal{J}$. To handle the Y-vertices, we use a simple expected distance lower-bound to show (in Section 4.2.3) that visiting Y at the end of our tour only adds a small factor to the overall expected cost.

Note that we assumed above that the master tour $\widehat{\pi}$ visits copies of each vertex consecutively. It is possible that the algorithm for uniform *a priori* TRP in [84] already has this property, in which case the analysis outlined above suffices. However, by providing an explicit subroutine (MakeConsecutive) that ensures this consecutive property, our approach can be combined with *any* algorithm for uniform *a priori* TRP. The details of the MakeConsecutive procedure and its analysis appear in Section 4.2.4.

4.2.2 Analysis for Vertices in X.

Here we analyze the steps of the algorithm that deal with vertices in X, i.e. with probability at least $\frac{1}{n^2}$. In order to reduce notation, we will assume here that $X = V$ which is

the entire vertex set. Recall that $p = \frac{1}{n} \cdot \min_{v \in V} p_v$. Also define $\bar{p}_v = \min\left\{(1+\frac{1}{n})p_v, 1\right\}$, $t_v = \lceil p_v/p \rceil$ and $q_v = 1-(1-p)^{t_v}$ for each $v \in V$. We will refer to the instances on metric (V,d) with probabilities $\{p_v\}_{v\in V}$, $\{q_v\}_{v\in V}$ and $\{\bar{p}_v\}_{v\in V}$ as $\mathcal{I}_p$, $\mathcal{I}_q$ and $\mathcal{I}_{\bar{p}}$ respectively. Note that the original instance is $\mathcal{I} = \mathcal{I}_p$. For simplicity we use $\mathbf{p}, \mathbf{q}$ and $\bar{\mathbf{p}}$ to refer to the vector of probabilities for each corresponding distribution.

Lemma IV.6. *For any $v \in V$, we have $p_v(1-\frac{1}{e}) \le q_v \le \bar{p}_v \le p_v(1+\frac{1}{n})$.*

Proof. Note that for every real number x we have $1+x \le e^x$: using $x = -p$ and raising both sides to the power of t_v we obtain $(1-p)^{t_v} \le e^{-pt_v}$. Now we have:

$$q_v = 1-(1-p)^{t_v} \ge 1-e^{-pt_v} \ge 1-e^{-p\cdot\frac{p_v}{p}} = 1-e^{-p_v} \ge (1-\frac{1}{e})p_v \ .$$

The second inequality uses $t_v = \lceil p_v/p \rceil$ and the last one uses $1-e^{-x} \ge (1-1/e)x$ for any $x \in [0,1]$ with $x = p_v$. Now, to prove the other inequality we consider the bionomial expansion of $(1-p)^{t_v}$ and cut it off for the powers greater than 1. So we have:

$$q_v = 1-(1-p)^{t_v} \le 1-(1-pt_v) = pt_v \le p(\frac{p_v}{p}+1) \le p_v + \frac{p_v}{n} = p_v(1+\frac{1}{n}) \ .$$

Combined with the fact that $q_v \le 1$, we obtain $q_v \le \bar{p}_v$. □

Lemma IV.7. *Let π be any master tour on (V,d). Consider two probability distributions given by $\{q_v\}_{v\in V}$ and $\{\bar{p}_v\}_{v\in V}$ such that $0 \le q_v \le \bar{p}_v \le 1$ for each $v \in V$. Then the expected latency of π under $\{q_v\}_{v\in V}$ is at most that under $\{\bar{p}_v\}_{v\in V}$.*

Proof. Let function $f(p_1, \cdots p_n)$ denote the expected latency of π as a function of vertex probabilities $\{p_v\}$. We will show that all partial derivatives of f are non-negative. This

would imply the lemma.

We can express f as a multilinear polynomial

$$f(\mathbf{p}) = \sum_{A \subseteq V} \left(\prod_{u \in A} p_u \prod_{w \in V \setminus A} (1 - p_w) \right) \cdot \mathsf{LAT}_\pi^A \ .$$

Recall that LAT_π^A is the total latency of vertices in active set A in the shortcut tour π_A. So the v^{th} partial derivative is:

$$\frac{\partial f}{\partial p_v} = \sum_{A \subseteq V \setminus v} \left(\prod_{u \in A} p_u \prod_{w \in V \setminus A \setminus v} (1 - p_w) \right) \left(\mathsf{LAT}_\pi^{A \cup v} - \mathsf{LAT}_\pi^A \right) .$$

For any $A \subseteq V \setminus v$, it follows by triangle inequality that $\mathsf{LAT}_\pi^{A \cup v} \geq \mathsf{LAT}_\pi^A$. This shows that each term in the above summation is non-negative and so $\frac{\partial f}{\partial p_v} \geq 0$. □

Lemma IV.8. *Let π be any master tour on (V, d). Consider two probability distributions given by $\{q_v\}_{v \in V}$ and $\{\bar{p}_v\}_{v \in V}$ and some constant $\beta \leq 1$ such that $\beta \bar{p}_v \leq q_v \leq \bar{p}_v$ for each $v \in V$. Then the expected latency of π under $\{q_v\}_{v \in V}$ is at least β^3 times that under $\{\bar{p}_v\}_{v \in V}$.*

Proof. Let function $f(p_1, \cdots p_n)$ denote the expected latency of π under probabilities $\{p_v\}_{v \in V}$. For $\mathbf{q}$ and $\bar{\mathbf{p}}$ as in the lemma, we will show $f(\mathbf{q}) \geq \beta^3 \cdot f(\bar{\mathbf{p}})$. To this end, we now view f as the expected sum of terms corresponding to all possible edges used in the shortcut tour π_A (where A is the active set). Renumber the vertices as $1, 2, \cdots n$ in the order of appearance in π; so the root r is numbered 1. For any $i, j \in [n]$ let I_{ij} denote the indicator random variable for (ordered) edge (i, j) being used in the shortcut tour π_A. For any $j \in [n]$, let N_j denote

the number of active vertices among $\{j, j+1, \cdots n\}$. Then, the total latency of tour π_A is

$$\sum_{1 \le i < j \le n} d(i,j) \cdot I_{ij} \cdot N_j \ .$$

Under probabilities $\mathbf{q}$, for any $i < j$ we have $\mathbb{E}[I_{ij}] = q_i \cdot q_j \cdot \prod_{k=i+1}^{j-1}(1-q_k)$ which corresponds to the event that i and j are active but all vertices between i and j are inactive. Moreover, $\mathbb{E}[N_j | I_{ij} = 1] = 1 + \sum_{\ell=j+1}^{n} q_\ell$ using the independence across vertices. So we can write:

$$f(\mathbf{q}) = \sum_{1 \le i < j \le n} d(i,j) \cdot \mathbb{E}[I_{ij}] \cdot \mathbb{E}[N_j | I_{ij} = 1] = \sum_{1 \le i < j \le n} d(i,j) \cdot q_i \cdot q_j \cdot \prod_{k=i+1}^{j-1} (1-q_k) \left(1 + \sum_{\ell=j+1}^{n} q_\ell\right) \ .$$

Note that for any $i < j$, using the fact that $\beta \cdot \bar{\mathbf{p}} \le \mathbf{q} \le \bar{\mathbf{p}}$ we have:

$$q_i \cdot q_j \cdot \prod_{k=i+1}^{j-1} (1-q_k) \left(1 + \sum_{\ell=j+1}^{n} q_\ell\right) \ \ge \ \beta^3 \cdot \bar{p}_i \cdot \bar{p}_j \cdot \prod_{k=i+1}^{j-1} (1-\bar{p}_k) \left(1 + \sum_{\ell=j+1}^{n} \bar{p}_\ell\right).$$

This implies $f(\mathbf{q}) \ge \beta^3 \cdot f(\bar{\mathbf{p}})$ as desired. □

Lemma IV.9. *Instances $\mathcal{I}$ and $\mathcal{J}$ in Algorithm 9 satisfy*

$$\mathsf{OPT}(\mathcal{J}) \le \left(\frac{e}{e-1}\right)\left(1 + \frac{1}{n}\right)^4 \cdot \mathsf{OPT}(\mathcal{I}) \ .$$

Proof. Recall the three instances $\mathcal{I} = \mathcal{I}_p$, $\mathcal{I}_q$ and $\mathcal{I}_{\bar{p}}$ on the metric (V, d). Using $\mathbf{q} \le \bar{\mathbf{p}}$ (Lemma IV.6) and Lemma IV.7 we have $\mathsf{OPT}(\mathcal{I}_q) \le \mathsf{OPT}(\mathcal{I}_{\bar{p}})$. Further, using $\mathbf{p} \le \bar{\mathbf{p}} \le (1+1/n)\mathbf{p}$ and Lemma IV.8 we have $\mathsf{OPT}(\mathcal{I}_{\bar{p}}) \le (1+1/n)^3 \mathsf{OPT}(\mathcal{I}_p)$. So we obtain $\mathsf{OPT}(\mathcal{I}_q) \le (1+1/n)^3 \cdot \mathsf{OPT}(\mathcal{I})$.

For $\alpha = \frac{e}{e-1}(1+\frac{1}{n})$, we will show that $\mathsf{OPT}(\mathcal{J}) \le \alpha \cdot \mathsf{OPT}(\mathcal{I}_q)$ which would prove the lemma. Recall that instance $\mathcal{J}$ is defined on the "scaled" vertex set $\widehat{V} = \cup_{v \in V} S_v$. Let π be an optimal master tour for instance $\mathcal{I}_q$ and $\widehat{\pi}$ be its corresponding master tour for $\mathcal{J}$: i.e. $\widehat{\pi}$

visits each group S_v consecutively at the point when π visits v. It suffices to show that the expected latency $\mathsf{ELAT}_{\widehat{\pi}}$ of tour $\widehat{\pi}$ for $\mathcal{J}$ is at most $\alpha \cdot \mathsf{ELAT}_{\pi}$, where ELAT_{π} is the expected latency of tour π for $\mathcal{I}_q$.

Let $A \subseteq V$ and $\widehat{A} \subseteq \widehat{V}$ denote the random active subsets in the instances $\mathcal{I}_q$ and $\mathcal{J}$ respectively. For any $v \in V$, let $\mathcal{E}_v$ denote the event that $S_v \cap \widehat{A} \neq \emptyset$; note that these events are independent. Moreover, for any $v \in V$, $\Pr_{\widehat{A}}[\mathcal{E}_v] = \Pr_{\widehat{A}}[S_v \cap \widehat{A} \neq \emptyset] = q_v = \Pr_A[v \in A]$. Let $\mathsf{ELAT}_{\widehat{\pi}}(w) = \mathbb{E}_{A \leftarrow \Pi}\left[\sum_{v \in S_w} \mathsf{LAT}^A_{\widehat{\pi}}(v)\right]$ denote the total expected latency of vertices of S_w in tour $\widehat{\pi}$. Fix any vertex $w \in V$: we will show that $\mathsf{ELAT}_{\widehat{\pi}}(w)$ is at most $\alpha \cdot \mathsf{ELAT}_{\pi}(w)$, where $\mathsf{ELAT}_{\pi}(w)$ is the expected latency of vertex w in π. Summing over $w \in V$, this would imply $\mathsf{ELAT}_{\widehat{\pi}} \leq \alpha \cdot \mathsf{ELAT}_{\pi}$, and hence $\mathsf{OPT}(\mathcal{J}) \leq \alpha \cdot \mathsf{OPT}(\mathcal{I}_q)$.

Consider now a fixed $w \in V$. Note that the probability distribution of the vertices in $V \setminus \{w\}$ whose groups (in $\widehat{V}$) have at least one vertex in $\widehat{A}$ is *identical* to that of $A \setminus \{w\}$. In other words, the random subset $\{v \in V \setminus \{w\} : \mathcal{E}_v \text{ occurs for } \widehat{A} \subseteq \widehat{V} \setminus S_w\}$ has the same distribution as random subset $A \setminus \{w\}$. Below, we couple these two distributions: We *condition* on the events $\mathcal{E}_v$ for all $v \in V \setminus \{w\}$ (for tour $\widehat{\pi}$) which corresponds to conditioning on $A \setminus \{w\}$ being active (for tour π). Under this conditioning (denoted $\mathcal{E}$), the latency of any active S_w vertex in $\widehat{\pi}$ is deterministic and equal to the latency of w (if it is active) in π; let $L(\pi, w \,|\, \mathcal{E})$ denote this deterministic value. So the conditional expected latency of w is $L(\pi, w \,|\, \mathcal{E}) \cdot \Pr[w \in A] = L(\pi, w|\mathcal{E}) \cdot q_w$ where we used the independence of $A \setminus \{w\}$ and the event $w \in A$. Similarly, the total conditional expected latency of S_w in $\widehat{\pi}$ is

$$L(\pi, w|\mathcal{E}) \cdot \mathbb{E}[|\widehat{A} \cap S_w|] = L(\pi, w|\mathcal{E}) \cdot (pt_w) \leq L(\pi, w|\mathcal{E}) \cdot (p_w + p) \,.$$

The equality above uses the independence of $\{\mathcal{E}_v : v \in V \setminus \{w\}\}$ and $\widehat{A} \cap S_w$, and the inequality uses $t_w = \lceil p_w/p \rceil$. Thus, the total conditional expected latency of S_w in $\widehat{\pi}$ is at most $\frac{p_w+p}{q_w}$ times the conditional expected latency of w in π. Deconditioning, we obtain $\mathsf{ELAT}_{\widehat{\pi}}(w) \leq \frac{p_w+p}{q_w} \cdot \mathsf{ELAT}_{\pi}(w)$. Using Lemma IV.6, $\frac{p_w+p}{q_w} \leq \frac{e}{e-1}\,(1 + p/p_w) \leq \frac{e}{e-1}(1 + 1/n) = \alpha$. So $\mathsf{LAT}_{\widehat{\pi}}(w) \leq \alpha \cdot \mathsf{LAT}_{\pi}(w)$ as needed.

□

Lemma IV.10. *Consider any consecutive master tour $\widehat{\pi}$ on instance $\mathcal{J}$ with expected latency $\mathsf{ALG}(\mathcal{J})$. Then the expected latency of the resulting master tour π on instance $\mathcal{I}$ is*

$$\mathsf{ALG}(\mathcal{I}) \leq \left(\frac{e}{e-1}\right)^3 \left(1 + \frac{1}{n}\right)^3 \cdot \mathsf{ALG}(\mathcal{J})\,.$$

Proof. Let $\mathsf{ALG}(\mathcal{I}_p)$, $\mathsf{ALG}(\mathcal{I}_q)$ and $\mathsf{ALG}(\mathcal{I}_{\bar{p}})$ denote the expected latency of master tour π under probabilities $\mathbf{p}$, $\mathbf{q}$ and $\bar{\mathbf{p}}$ respectively. Below we use $\alpha = \frac{e}{e-1}(1 + \frac{1}{n})$. Using $\mathbf{p} \leq \bar{\mathbf{p}}$ and Lemma IV.7 we have $\mathsf{ALG}(\mathcal{I}_p) \leq \mathsf{ALG}(\mathcal{I}_{\bar{p}})$. Using $\frac{1}{\alpha} \cdot \bar{\mathbf{p}} \leq \mathbf{q} \leq \bar{\mathbf{p}}$ (Lemma IV.6) and Lemma IV.8, we have $\mathsf{ALG}(\mathcal{I}_{\bar{p}}) \leq \alpha^3 \cdot \mathsf{ALG}(\mathcal{I}_q)$. Combining these bounds, we have $\mathsf{ALG}(\mathcal{I}) \leq \alpha^3 \cdot \mathsf{ALG}(\mathcal{I}_q)$. Finally, it is easy to see that $\mathsf{ALG}(\mathcal{I}_q) \leq \mathsf{ALG}(\mathcal{J})$ as the probability of having at least one active vertex in group S_v (for any $v \in V$) in $\mathcal{J}$ is exactly equal the probability (q_v) of visiting v in $\mathcal{I}_q$. □

4.2.3 Overall Analysis Including Vertices in Y.

Now we have the tools to finish the proof of Theorem IV.2 assuming the tour $\widehat{\pi}$ in $\mathcal{J}$ is consecutive. Recall that π is the tour corresponding to $\widehat{\pi}$ on vertices X and $\bar{\pi}$ is the extended tour that also visits the vertices Y.

First, the analysis for the vertices X (Lemmas IV.9 and IV.10) yields:

Corollary IV.11. *The tour π on vertices X satisfies*

$$\mathbb{E}_A\left[\sum_{v\in A\cap X}\mathsf{LAT}^A_\pi(v)\right] \le (1+o(1))\left(\frac{e}{e-1}\right)^4 \rho\cdot\mathsf{OPT}_X\ ,$$

where ρ is the approximation ratio for uniform a priori *TRP and OPT_X is the optimal value of the instance restricted to vertices X.*

After extending tour π to $\bar\pi$, we can write the final expected latency as

$$\mathsf{ALG}(\mathcal{I}) = \mathbb{E}_A\left[\sum_{v\in A\cap X}\mathsf{LAT}^A_{\bar\pi}(v)+\sum_{v\in A\cap Y}\mathsf{LAT}^A_{\bar\pi}(v)\right] = \mathbb{E}_A\left[\sum_{v\in A\cap X}\mathsf{LAT}^A_{\pi}(v)\right]+\mathbb{E}_A\left[\sum_{v\in A\cap Y}\mathsf{LAT}^A_{\bar\pi}(v)\right] \tag{4.1}$$

where $A\subseteq V$ is the active subset. The last equality uses the fact that $\bar\pi$ visits all vertices of X (along π) before Y. The first term above can be bounded by Corollary IV.11. We now focus on the second term involving vertices Y.

Let L denote the length of tour $\bar\pi$ before visiting the first Y-vertex; note that this is a random variable. Clearly $\mathbb{E}[L]$ is at most the expected total latency of the X-vertices. Consider any $v\in Y$: by the ordering of the Y-vertices in master tour $\bar\pi$,

$$\mathsf{LAT}^A_{\bar\pi}(v) \le (L+(2N_v+1)\cdot d(r,v))\cdot\mathbf{1}_{v\in A}\ ,$$

where N_v is the number of active Y-vertices appearing before v. Taking expectations,

$$\begin{aligned}\mathbb{E}[\mathsf{LAT}^A_{\bar\pi}(v)] &\le p_v\cdot\mathbb{E}[L]+p_v\cdot d(r,v)\cdot(2\mathbb{E}[N_v]+1) \le p_v\cdot\mathbb{E}[L]+p_v\cdot d(r,v)\cdot(2n\cdot\frac{1}{n^2}+1)\\ &= p_v\cdot\mathbb{E}[L]+p_v\cdot d(r,v)\cdot(1+o(1)),\end{aligned}$$

The first inequality uses the fact that L, N_v and $\mathbf{1}_{v\in A}$ are independent. The second inequality uses that N_v is the sum of at most n Bernoulli random variables each with probability at most $\frac{1}{n^2}$.

Summing over all $v \in Y$, we obtain

$$\mathbb{E}_A\left[\sum_{v\in A\cap Y} \mathsf{LAT}^A_{\bar{\pi}}(v)\right] \le \left(\sum_{v\in Y} p_v\right)\cdot \mathbb{E}[L] + (1+o(1))\sum_{v\in Y} p_v \cdot d(r,v)$$
$$\le \frac{1}{n}\cdot \mathbb{E}[L] + (1+o(1))\sum_{v\in Y} p_v \cdot d(r,v)$$

where the last inequality uses $p_v \le 1/n^2$ for all $v \in Y$.

Let E_X denote the expected latency of the X-vertices: this is the first term in the right-hand-side of (4.1). Recall that $\mathbb{E}[L] \le E_X$. Using the above bound on the latency of Y-vertices,

$$\begin{aligned}
\mathsf{ALG}(\mathcal{I}) &\le E_X + \frac{1}{n}\cdot E_X + (1+o(1))\sum_{v\in Y} p_v\cdot d(r,v)\\
&= (1+o(1))\left(E_X + \sum_{v\in Y} p_v\cdot d(r,v)\right)\\
&\le (1+o(1))\left(\frac{e}{e-1}\right)^4 \rho\cdot\left(\mathsf{OPT}_X + \sum_{v\in Y} p_v\cdot d(r,v)\right) \qquad (4.2)\\
&\le (1+o(1))\left(\frac{e}{e-1}\right)^4 \rho\cdot \mathsf{OPT}. \qquad (4.3)
\end{aligned}$$

Above, inequality (4.2) uses Corollary IV.11. Inequality (4.3) uses the fact that the latency contribution of Y-vertices in *any* master tour is at least $\sum_{v\in Y} p_v\cdot d(r,v)$ and the latency of X-vertices is clearly at least OPT_X. This completes the proof of Theorem IV.2 assuming that $\widehat{\pi}$ visits each group S_v consecutively. The next section shows that this consecutive property can always be ensured.

4.2.4 Ensuring the Consecutive Property.

The main result here is:

Theorem IV.12. *Consider any instance $\mathcal{J}$ of uniform* a priori *TRP on vertices $\cup_{v\in X} S_v$ where the vertices in S_v are co-located for all $v \in X$. There is a polynomial time algorithm that given any master tour τ, modifies it into a consecutive tour having expected latency at most that of τ.*

While this result is intuitive, we note that it is not obvious to prove. This is because an optimal TRP solution can be fairly complicated even on simple metrics: for example, the optimum may cross itself several times on a line-metric ([2]) and the problem is NP-hard even on tree-metrics ([81]).

Algorithm 10 describes the procedure used to establish Theorem IV.12. We use Π to denote the distribution of active vertices, where each vertex has independent probability p.

It is obvious that each iteration of the while-loop decreases the number k of parts of S_z: so this procedure ends in polynomial time and produces a master tour that visits each S_v consecutively. The key part of the proof is in showing that the expected latency does not increase.

Algorithm 10 Algorithm to obtain a consecutive master tour.

Procedure MAKECONSECUTIVE(τ):

1: **for** $z \in V$ **do**
2: Let $C_z^1, C_z^2, ..., C_z^k$ be the minimal partition of S_z, where for every $i \in [k]$, the vertices in C_z^i appear consecutively in tour τ.
3: **while** there exists C_z^i and C_z^j with $i \neq j$ **do**
4: Construct tour τ_i from τ by relocating vertices C_z^j immediately after C_z^i
5: Construct tour τ_j from τ by relocating vertices C_z^i immediately before C_z^j
6: $\tau \leftarrow \text{argmin}_{\tau\in\{\tau_i,\tau_j\}} \mathbb{E}_{A\leftarrow\Pi}\left[\mathsf{LAT}_\tau^A\right]$
7: Update $k \leftarrow k-1$ and the new partition of S_z.
8: **end while**
9: **end for**

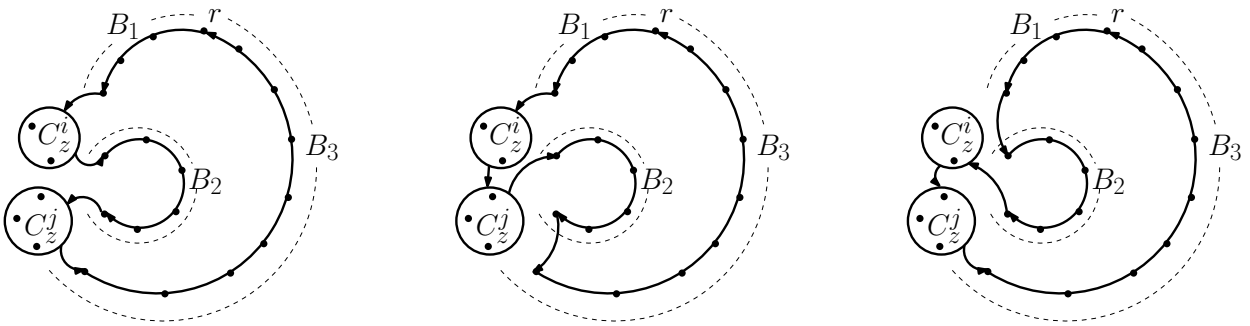

Figure 4.2: From left to right: tours τ, τ_i and τ_j

Lemma IV.13. *Let C_z^i and C_z^j be two parts of S_z with respect to the current tour τ in procedure* MakeConsecutive. *Then we have:*

$$\mathbb{E}_{A\leftarrow\Pi}\left[\mathsf{LAT}_{\tau}^{A}\right] \geq \min(\mathbb{E}_{A\leftarrow\Pi}\left[\mathsf{LAT}_{\tau_i}^{A}\right], \mathbb{E}_{A\leftarrow\Pi}\left[\mathsf{LAT}_{\tau_j}^{A}\right]).$$

Proof. Let $|C_z^i| = k_i$ and $|C_z^j| = k_j$. Without loss of generality we assume that τ visits C_z^i before C_z^j. To reduce notation we use V to denote the vertex set of instance $\mathcal{J}$ and let $U = C_z^i \cup C_z^j$. Recall that $\mathsf{LAT}_{\pi}^{A}(w)$ is the latency of vertex w in tour π when the subset A of vertices is active; also $\mathsf{LAT}_{\pi}^{A} = \sum_{w\in A} \mathsf{LAT}_{\pi}^{A}(w)$. For any $R \subseteq V$ and $S \subseteq R$ we use the notation $p(S,R) = p^{|S|} \cdot (1-p)^{|R\setminus S|}$ for the probability that S is the set of active vertices amongst R.

It suffices to show $\mathbb{E}_{A\leftarrow\Pi}\left[\mathsf{LAT}_{\tau}^{A}\right]$ is at least a convex combination of $\mathbb{E}_{A\leftarrow\Pi}\left[\mathsf{LAT}_{\tau_i}^{A}\right]$ and $\mathbb{E}_{A\leftarrow\Pi}\left[\mathsf{LAT}_{\tau_j}^{A}\right]$. More specifically we show that:

$$\mathbb{E}_{A\leftarrow\Pi}\left[\mathsf{LAT}_{\tau}^{A}\right] \geq \lambda \cdot \mathbb{E}_{A\leftarrow\Pi}\left[\mathsf{LAT}_{\tau_i}^{A}\right] + (1-\lambda) \cdot \mathbb{E}_{A\leftarrow\Pi}\left[\mathsf{LAT}_{\tau_j}^{A}\right] .$$

where $\lambda \in [0,1]$ is a value that will be set later. The above inequality is equivalent to proving

the following:

$$\sum_{A\subseteq V} p(A,V)\mathsf{LAT}_\tau^A \geq \sum_{A\subseteq V} p(A,V)\left(\lambda\cdot\mathsf{LAT}_{\tau_i}^A + (1-\lambda)\cdot\mathsf{LAT}_{\tau_j}^A\right) .$$

Let us define $B = A\setminus U$ and $C = A\cap U$. Basically C is the subset of active vertices among $U = C_z^i \cup C_z^j$, and B is the subset of active vertices among the rest of V. Then we can re-write the above inequality as follows:

$\sum_{B\subseteq V\setminus U} p(B, V\setminus U)\sum_{C\subseteq U} p(C,U)\mathsf{LAT}_\tau^{B\cup C}$

$$\geq \sum_{B\subseteq V\setminus U} p(B,V\setminus U)\sum_{C\subseteq U} p(C,U)\left(\lambda\cdot\mathsf{LAT}_{\tau_i}^{B\cup C} + (1-\lambda)\cdot\mathsf{LAT}_{\tau_j}^{B\cup C}\right) .$$

Therefore, it is enough to prove:

$$\sum_{C\subseteq U} p(C,U)\mathsf{LAT}_\tau^{B\cup C} \geq \sum_{C\subseteq U} p(C,U)\left(\lambda\cdot\mathsf{LAT}_{\tau_i}^{B\cup C} + (1-\lambda)\cdot\mathsf{LAT}_{\tau_j}^{B\cup C}\right), \forall B\subseteq V\setminus U. \tag{4.4}$$

In the rest of this proof we fix a subset $B\subseteq V\setminus U$. This can be viewed as conditioning on the event "B is the active set of vertices within $V\setminus U$"; we denote this event by $\mathcal{E}_B$. Let the order of visited vertices of $B\cup U$ in τ be $B_1, C_z^i, B_2, C_z^j, B_3$ where B_1, B_2, B_3 are ordered sets of vertices that form a partition of B. Therefore, together with C_z^i and C_z^j they form a partition of $B\cup U$. See Figure 4.2 for an example.

If $B_2 = \emptyset$ then all three tours τ, τ_i and τ_j become identical when restricted to $B\cup C$ for any $C\subseteq U$. So (4.4) is satisfied with an equality in this case. Below we assume $B_2\neq\emptyset$. We will prove the inequality (4.4) by considering the latency contributions of vertices in each of the 5 different parts $B_1, B_2, B_3, C_z^i, C_z^j$.

We define $l_w := \mathsf{LAT}_\tau^{B\cup\{w\}}(w)$ for all $w\in V$ and

$$T_i := \mathsf{LAT}_\tau^{B\cup C_z^i}(w) \quad \forall w\in C_z^i, \quad \text{and} \quad T_j := \mathsf{LAT}_\tau^{B\cup C_z^j}(w) \quad \forall w\in C_z^j. \tag{4.5}$$

Basically T_i (resp. T_j) is the length of the path in τ from the root to any vertex in C_z^i (resp. C_z^j) when the active vertices are $B \cup C_z^i$ (resp. $B \cup C_z^j$). Note that $T_j \geq T_i$ by triangle inequality. Also, let $L_\pi^B(w)$ be the expected latency of any vertex w for any tour $\pi \in \{\tau, \tau_i, \tau_j\}$ conditioned on the event $\mathcal{E}_B$. More formally:

$$L_\pi^B(w) = \sum_{C \subseteq U} p(C, U) \mathsf{LAT}_\pi^{B \cup C}(w), \qquad \forall w \in V \ .$$

Finally, defining the following terms will help us simplify our notation:

$$\Delta_i := \mathsf{LAT}_\tau^{B \cup C_z^i}(w) - \mathsf{LAT}_\tau^B(w) = \mathsf{LAT}_\tau^{B \cup C_z^i}(w) - l_w \qquad \forall w \in B_2 \cup B_3 \ . \tag{4.6}$$

$$\Delta_j := \mathsf{LAT}_\tau^{B \cup C_z^j}(w) - \mathsf{LAT}_\tau^B(w) = \mathsf{LAT}_\tau^{B \cup C_z^j}(w) - l_w \qquad \forall w \in B_3. \tag{4.7}$$

Note that Δ_i (resp. Δ_j) corresponds to the increase in latency (conditioned on $\mathcal{E}_B$) of any vertex appearing after C_z^i (resp. C_z^j) if some vertex in C_z^i (resp. C_z^j) is active. Note that the right hand side in (4.6) is the same for any w in the given set and as a result independent of w; the same observation is true for (4.7). Moreover, by triangle inequality, having a superset of active vertices can only increase the latency of any vertex: so Δ_i and Δ_j are non-negative.

Table 4.1 lists the expected latency of vertices in each of the five different parts, conditioned on $\mathcal{E}_B$. We use $\alpha_i = 1 - (1-p)^{k_i}$ and $\alpha_j = 1 - (1-p)^{k_j}$ as the probabilities of having at least one active vertex in parts C_z^i and C_z^j respectively.

We first prove the lemma assuming the entries stated in the table. Then we explain why each of these table entries is correct, which would complete the proof.

Tour π \ Type	B_1	B_2	B_3	C_z^i	C_z^j
τ	l_w	$l_w + \Delta_i\alpha_i$	$l_w + \Delta_i\alpha_i + \Delta_j\alpha_j$	T_ip	$T_jp + \Delta_i\alpha_ip$
τ_i	l_w	$l_w + \Delta_i(\alpha_i + \alpha_j - \alpha_i\alpha_j)$	$l_w + \Delta_i(\alpha_i + \alpha_j - \alpha_i\alpha_j)$	T_ip	T_ip
τ_j	l_w	l_w	$l_w + \Delta_j(\alpha_i + \alpha_j - \alpha_i\alpha_j)$	T_jp	T_jp

Table 4.1: The values of $L_\pi^B(w)$ for $w \in B_1 \cup B_2 \cup B_3 \cup C_z^i \cup C_z^j$, and $\pi \in \{\tau, \tau_i, \tau_j\}$

Completing proof of Lemma IV.13 using Table 4.1.

We now prove (4.4) for a suitable choice of $\lambda \in [0, 1]$. The value λ will not depend on the subset B: so (as discussed before) we can take an expectation over B to complete the proof of the lemma.

Choosing any λ such that $\lambda \leq \frac{\alpha_i}{\alpha_i+\alpha_j-\alpha_i\alpha_j}$ and $1-\lambda \leq \frac{\alpha_j}{\alpha_i+\alpha_j-\alpha_i\alpha_j}$, it follows from the first three columns of Table 4.1 (for B_1, B_2 and B_3) that:

$$L_\tau^B(w) \geq \lambda \cdot L_{\tau_i}^B(w) + (1-\lambda) \cdot L_{\tau_j}^B(w), \qquad \forall w \in B. \tag{4.8}$$

Next we show that the *total* latency contribution from U satisfies a similar inequality:

$$\sum_{w\in U} L_\tau^B(w) \geq \lambda \cdot \sum_{w\in U} L_{\tau_i}^B(w) + (1-\lambda) \cdot \sum_{w\in U} L_{\tau_j}^B(w). \tag{4.9}$$

To see this, note from the last two columns of the table that

$$\sum_{w\in U} L_\tau^B(w) \geq k_i \cdot T_ip + k_j \cdot T_jp, \quad \sum_{w\in U} L_{\tau_i}^B(w) = (k_i+k_j)T_ip, \quad \sum_{w\in U} L_{\tau_j}^B(w) = (k_i+k_j)T_jp\ .$$

So, to prove (4.9) it suffices to show $k_iT_ip + k_jT_jp \geq (k_i+k_j)(\lambda T_i + (1-\lambda)T_j)p$. Using the fact that $T_i \leq T_j$, it suffices to show $k_j \geq (k_i+k_j)(1-\lambda)$. In other words, choosing λ such that $1-\lambda \leq \frac{k_j}{k_i+k_j}$, we would obtain (4.9).

Finally, adding the inequalities (4.8) and (4.9) (which account for the latency contribution from all active vertices) we would obtain (4.4). We only need to ensure that there is some choice for λ satisfying the conditions we assumed, namely:

$$\lambda \leq \frac{\alpha_i}{\alpha_i + \alpha_j - \alpha_i\alpha_j}, \quad 1-\lambda \leq \frac{\alpha_j}{\alpha_i + \alpha_j - \alpha_i\alpha_j}, \quad \text{and} \quad 1-\lambda \leq \frac{k_j}{k_i + k_j} \ .$$

It can be verified directly that $\lambda = \frac{1-(1-p)^{k_i}}{1-(1-p)^{k_i+k_j}}$ satisfies these conditions. We provide more explanations about our choice of λ after completing the proof.

Obtaining the entries in Table 4.1.

Below we consider each vertex-type separately.

Vertices $w \in B_1$. By construction of τ_i and τ_j it is obvious that τ, τ_i and τ_j are identical until visiting any $w \in B_1$. So for any $C \subseteq U$ and $\pi \in \{\tau, \tau_i, \tau_j\}$ we have $\mathsf{LAT}_\pi^{B\cup C}(w) = \mathsf{LAT}_\tau^B(w) = \mathsf{LAT}_\tau^{B\cup\{w\}}(w) = l_w$. This means that $L_\pi^B(w) = l_w$ for all $\pi \in \{\tau, \tau_i, \tau_j\}$.

Vertices $w \in B_2$. Consider first tour τ. Note that if there is at least one active vertex in C_z^i (which happens with probability α_i) then the latency of any $w \in B_2$ will be $\mathsf{LAT}_\tau^{B\cup C_z^i}(w)$. However, if all vertices in C_z^i are inactive (which happens with probability $1-\alpha_i$) then the latency of w would be $\mathsf{LAT}_\tau^B(w)$. Now using (4.6) we have:

$$L_\tau^B(w) = \mathsf{LAT}_\tau^{B\cup C_z^i}(w) \cdot \alpha_i + l_w \cdot (1-\alpha_i) = (l_w + \Delta_i) \cdot \alpha_i + l_w \cdot (1-\alpha_i) = l_w + \Delta_i\alpha_i \ .$$

Now, we can use a similar logic for τ_i. Here, if there is any active vertex in $U = C_z^i \cup C_z^j$ (with probability $\alpha_i + \alpha_j - \alpha_i\alpha_j$) the latency of w is $\mathsf{LAT}_{\tau_i}^{B\cup U}(w)$, and if all of U is inactive the latency is l_w. Note that by definition of τ and τ_i and the fact that all vertices in C_z^i

appear consecutively on both tours, $\mathsf{LAT}_{\tau_i}^{B\cup U}(w) = \mathsf{LAT}_{\tau_i}^{B\cup C_z^i}(w) = \mathsf{LAT}_{\tau}^{B\cup C_z^i}(w)$. So we have $L_{\tau_i}^B = l_w + \Delta_i(\alpha_i + \alpha_j - \alpha_i\alpha_j)$.

Finally, by definition of τ_j we have $\mathsf{LAT}_{\tau_j}^{B\cup C}(w) = \mathsf{LAT}_{\tau}^{B}(w) = l_w$ for any $C \subseteq U$. So $L_{\tau_j}^B(w) = l_w$.

Vertices $w \in B_3$. Consider first tour τ. The latency of such a vertex w is:

- l_w if all of $U = C_z^i \cup C_z^j$ is inactive,
- $\mathsf{LAT}_{\tau}^{B\cup C_z^i}(w)$ if some vertex in C_z^i is active and all of C_z^j is inactive,
- $\mathsf{LAT}_{\tau}^{B\cup C_z^j}(w)$ if some vertex in C_z^j is active and all of C_z^i is inactive, and
- $\mathsf{LAT}_{\tau}^{B\cup C_z^i\cup C_z^j}(w)$ if some vertex in C_z^i and some vertex in C_z^j are active.

Therefore, we can write $L_{\tau}^B(w)$ as:

$$l_w(1-\alpha_i)(1-\alpha_j) + \mathsf{LAT}_{\tau}^{B\cup C_z^i}(w)\alpha_i(1-\alpha_j) + \mathsf{LAT}_{\tau}^{B\cup C_z^j}(w)\alpha_j(1-\alpha_i) + \mathsf{LAT}_{\tau}^{B\cup U}(w)\alpha_i\alpha_j \ .$$

From (4.6) and (4.7) we have $\mathsf{LAT}_{\tau}^{B\cup C_z^i} = l_w + \Delta_i$ and $\mathsf{LAT}_{\tau}^{B\cup C_z^j} = l_w + \Delta_j$. Also, since we assumed that $B_2 \neq \emptyset$, we have $\mathsf{LAT}_{\tau}^{B\cup U} = l_w + \Delta_j + \Delta_i$. Combined with the above equation,

$$L_{\tau}^B(w) = l_w + \Delta_i\alpha_i + \Delta_j\alpha_j \ .$$

Now for tour τ_i the latency would be equal to $\mathsf{LAT}_{\tau}^{B\cup C_z^i}(w) = l_w + \Delta_i$ if there is at least one active vertex among U which happens with probability $\alpha_i + \alpha_j - \alpha_i\alpha_j$. Otherwise it would be just l_w. So $L_{\tau_i}^B = l_w + \Delta_i(\alpha_i + \alpha_j - \alpha_i\alpha_j)$. Similarly, for tour τ_j we have $L_{\tau_j}^B = l_w + \Delta_j(\alpha_i + \alpha_j - \alpha_i\alpha_j)$.

Vertices $w \in C_z^i$. We start with tour τ. If $w \notin C$ then $\mathsf{LAT}_\tau^{B\cup C}(w) = 0$. Otherwise, w is active and using (4.5) we have $\mathsf{LAT}_\tau^{B\cup C}(w) = \mathsf{LAT}_\tau^{B\cup C_z^i}(w) = T_i$. So $L_\tau^B(w) = T_i p$.

As C_z^i appears in the same position in tours τ and τ_i, we also have $L_{\tau_i}^B(w) = T_i p$.

In tour τ_j, part C_z^i has moved to the position of part C_z^j in τ. Here, when $w \in C$ we have $\mathsf{LAT}_{\tau_j}^{B\cup C}(w) = T_j$. So $L_{\tau_j}^B(w) = T_j p$.

Vertices $w \in C_z^j$. As in the previous case, we have $L_{\tau_i}^B(w) = T_i p$ and $L_{\tau_j}^B(w) = T_j p$.

Now, consider tour τ. First note that if $w \notin C$, $\mathsf{LAT}_\tau^{B\cup C}(w) = 0$. Below we consider the cases that w is active, which happens with probability p. If there is at least one active vertex in C_z^i (which happens independently with probability α_i) we have $\mathsf{LAT}_\tau^{B\cup C}(w) = l_w + \Delta_i = T_j + \Delta_i$. And if there is no active vertex in C_z^i (with probability $1 - \alpha_i$), then we have $\mathsf{LAT}_\tau^{B\cup C}(w) = l_w = T_j$. So

$$L_\tau^B(w) = p\alpha_i \cdot (T_j + \Delta_i) + p(1 - \alpha_i) \cdot T_j = T_j p + \Delta_i \alpha_i p \; .$$

This completes the proof of all cases in Table 4.1, and hence Lemma IV.13.

□

4.2.5 Choice of λ in Proof of Lemma IV.13

Here we show that $\lambda = \frac{1-(1-p)^{k_i}}{1-(1-p)^{k_i+k_j}}$ (where $0 \le p \le 1$) satisfies the following inequalities:

$$\lambda \le \frac{\alpha_i}{\alpha_i + \alpha_j - \alpha_i \alpha_j} \tag{4.10}$$

$$1 - \lambda \le \frac{\alpha_j}{\alpha_i + \alpha_j - \alpha_i \alpha_j} \tag{4.11}$$

$$1 - \lambda \le \frac{k_j}{k_i + k_j} \tag{4.12}$$

where $\alpha_i = 1-(1-p)^{k_i}$ and $\alpha_j = 1-(1-p)^{k_j}$.

We define function $f(k) = 1-(1-p)^k$. Then we can write:

$$\lambda = \frac{f(k_i)}{f(k_i+k_j)}, \quad \alpha_i = f(k_i), \quad \alpha_j = f(k_j)$$

Clearly,

$$f(k_i+k_j) = f(k_i) + f(k_j) - f(k_i)f(k_j) \tag{4.13}$$

Now, we can re-write inequality (4.10) as:

$$\frac{f(k_i)}{f(k_i+k_j)} \leq \frac{f(k_i)}{f(k_i)+f(k_j)-f(k_i)f(k_j)}$$

which is true by equation ((4.13)).

For inequality (4.11), we rewrite it as:

$$1 - \frac{f(k_i)}{f(k_i+k_j)} \leq \frac{f(k_j)}{f(k_i)+f(k_j)-f(k_i)f(k_j)} = \frac{f(k_j)}{f(k_i+k_j)}$$

$$\Leftrightarrow \quad f(k_i+k_j) \leq f(k_i)+f(k_j)\ ,$$

which is true by (4.13) and the fact that $f(k) \geq 0$ for every k.

It remains to show the correctness of inequality (4.12) which can be written as:

$$\frac{f(k_i)}{f(k_i+k_j)} \geq \frac{k_i}{k_i+k_j} \quad \Leftrightarrow \quad \frac{f(k_i)}{k_i} \geq \frac{f(k_i+k_j)}{k_i+k_j}\ .$$

So it is enough to show that $g(k) = \frac{f(k)}{k}$ is decreasing, or equivalently $g'(k) \leq 0$. We can write:

$$g'(k) = \frac{kf'(k)-f(k)}{k^2} = \frac{(1-p)^k(1-k\log(1-p))-1}{k^2}$$
$$\leq \frac{(1-p)^k \cdot e^{-k\log(1-p)}-1}{k^2} = 0\ .$$

Above we used the inequality $1+x \leq e^x$ for all real x.

4.3 *A Priori* TRP with Respect to the Re-Optimization Solution

Similar to Section 4.2, we consider an instance $\mathcal{I}$ of *a priori* TRP on metric (V, d) with probabilities $\{p_v\}_{v \in V}$. In this section we provide an algorithm for *a priori* TRP and show it has $\mathcal{O}(\log n)$ approximation ratio with respect to the re-optimization solution, where n is the number of vertices. Recall that the re-optimization cost is:

$$\mathsf{ROPT} = \sum_{A \subseteq V} \left(\prod_{u \in A} p_u \prod_{w \in V \setminus A} (1 - p_w) \right) \cdot \mathsf{OPT}_A \ ,$$

where OPT_A denotes the optimal value of the TRP instance with demands at subset $A \subseteq V$.

As mentioned before, first we provide a constant-factor approximation algorithm with respect to the re-optimization solution if our metric is a relaxed 2-Hierarchically well Separated Tree (2-HST).

Definition IV.14. A 2-HST is a tree metric rooted at s such that:

1. for every vertex $v \in V$, all edges incident to v except for the one that lies on $s - v$ path has the same weight, and
2. cost of edges on any root to leaf path decrease by a factor of two in each step.

See Figure 4.3 for an illustration of a 2-HST.

Definition IV.15. A relaxed 2-HST is a tree metric rooted at s such that:

1. for every vertex $v \in V \setminus \{s\}$, all edges incident to v except for the one that lies on $s - v$ path has the same weight, and

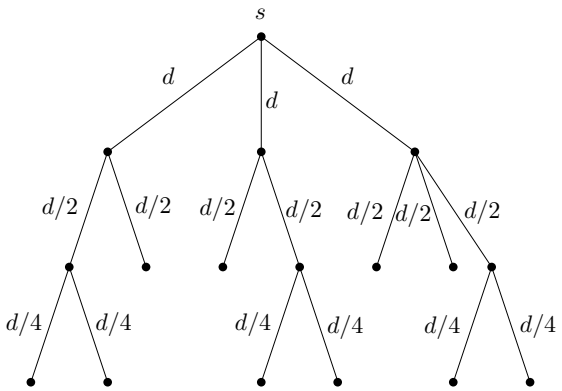

Figure 4.3: Illustration of a 2-HST metric

2. cost of edges on any root to leaf path decrease by a factor of two in each step.

The difference is that in a relaxed 2-HST, the edges incident to the root need not have equal weights.

After proving that the approximation ratio of our algorithm for a relaxed 2-HST with root s is constant with respect to the re-optimization solution, we show how we can change our metric to a relaxed 2-HST, such that we only lose a factor of $\mathcal{O}(\log n)$. In this part, we use the famous tree-embedding algorithm from [32]. Then we modify the resulting 2-HST to obtain a relaxed 2-HST rooted at r, and show how we can use their result to compare the expected *a priori* solution and the re-optimization solution of these metrics.

4.3.1 Algorithm for a Relaxed 2-HST

We now assume that the metric is induced by a relaxed 2-HST. This is given by an edge-weighted tree $\mathcal{T} = (V, E)$ of the following form. The tree $\mathcal{T}$ is rooted at $r \in V$ and the *depth* of any vertex u is the length of the $r - u$ path in $\mathcal{T}$. We note that r is also the root for the *a priori* TRP instance. For any $u \in V$ let $T_u \subseteq V$ denote all the vertices that appear at or below u in $\mathcal{T}$. The metric on $\mathcal{T}$ is $\ell : V \times V \to \mathbb{R}_+$, where $\ell(u, v)$ is the total weight of $u - v$

path in $\mathcal{T}$. Note that since $\mathcal{T}$ is a tree, if (u, v) is an edge, then $\ell(u, v)$ is the weight of this edge. For any vertex $u \neq r$ its *parent edge* is the edge incident to u that lies on the $u - r$ path; all other edges incident to u are its *child edges*. For every vertex $u \neq r$, all its child edges have the same weight and the weight of its parent edge is 2 times that of any child edge. Note that the edges incident to r may have different weights. Finally the distance $d(u, v)$ between any pair $u, v \in V$ of vertices is just the total weight on the $u - v$ path in $\mathcal{T}$.

The core of our algorithm is based on grouping vertices into clusters so that: (1) the latencies of the vertices in the same cluster are close to each other, and (2) total probability of each cluster is constant. Then we contract each cluster to form a single vertex and find a tour for the deterministic TRP on these new "clusters". In our original metric, this would correspond to a tour, where clusters are visited in the same order as the obtained deterministic tour, and vertices in each cluster are visited in an arbitrary order. Finally we take care of vertices that are not part of any cluster by appending them to this tour in an increasing order of their distance to the root. Here is a formal description of this algorithm:

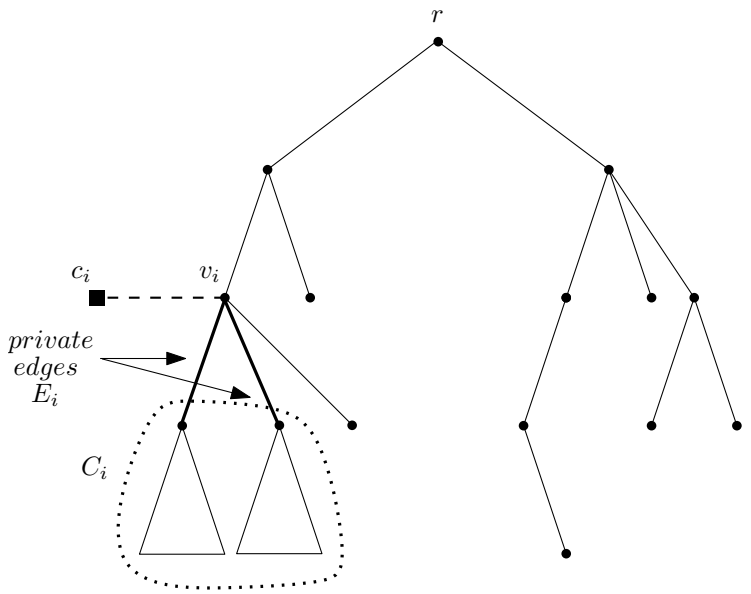

Figure 4.4: Forming cluster C_i and adding the corresponding vertex c_i

Algorithm 11 A priori solution on HSTs.

1: Set $M \leftarrow \emptyset$ and tree $\mathcal{T}' \leftarrow \mathcal{T}$
2: **for** $i = 1, 2, \cdots$ **do**
3: Find vertex $v_i \in V$ of maximum depth such that $\sum_{w \in T_{v_i} \setminus M} p_w \geq 1$.
4: If $v_i = r$ then exit the for-loop.
5: Choose $S \subseteq \{u \in V \ : \ u \text{ child of } v_i,\ T_u \setminus M \neq \emptyset\}$ such that:

$$1 \ \leq \ \sum_{u \in S} \ \sum_{w \in T_u \setminus M} p_w \ \leq \ 2$$

6: Set cluster $C_i = \bigcup_{u \in S} T_u \setminus M$, private edges $E_i = \{(v_i, u) : u \in S\}$ and $M \leftarrow M \cup C_i$.
7: Let τ_i denote any tour in $\mathcal{T}$ on vertices $\{v_i\} \cup C_i$ originating from v_i.
8: In tree $\mathcal{T}'$, add a new vertex c_i connected to vertex v_i; the weight of this new edge $\ell(v_i, c_i)$ is the same as other child edges of v_i. (See Figure 4.4)
9: **end for**
10: Let N denote the number of clusters formed above.
11: Find an approximately optimal tour δ for the deterministic TRP instance on tree metric $\mathcal{T}'$ with root r and vertices $\{c_i\}_{i=1}^N$.
12: Let π be the tour obtained by appending tours $\{\tau_i\}_{i=1}^N$ in the order that their corresponding cluster vertex is visited in δ.
13: Let U denote all the children of the root r, and cluster $B_u = T_u \setminus M$ for each $u \in U$.
14: Let σ be the tour in $\mathcal{T}$ visiting each cluster $\{B_u : u \in U\}$ in increasing order of $\ell(r, u)$.
15: Combine *a priori* tours π and σ into a single tour τ using Lemma IV.31.

The clusters $\{C_i\}_{i=1}^N$ are called heavy clusters (each such cluster has total probability

at least one). The clusters $\{B_u\}_{u\in U}$ are called light clusters (each such cluster has total probability less than one).

4.3.2 Analysis for Heavy Clusters

Here, we assume $\mathcal{I}$ only results in heavy clusters and we complete the analysis for all clusters in Section 4.3.3 and 4.3.4. For the analysis, we define a new instance $\mathcal{J}$ of *a priori* TRP on the tree metric $\mathcal{T}'$ where each vertex c_i has activation probability $q_i = 1 - \prod_{v\in C_i}(1-p_v)$. All the other vertices have 0 activation probability in instance $\mathcal{J}$. Note that q_i is exactly the probability that there is some active vertex in cluster C_i in the original instance $\mathcal{I}$. We first make the following simple observations.

Observation IV.16. *For each $i \in [N]$, all the private edges E_i have the same weight.*

Proof. Consider the iteration when cluster i is formed. Recall that $E_i = \{(v_i, u) : u \in S\}$ where $v_i \in V \setminus r$ and S is a subset of v_i's children. By Definition IV.15, all the child edges at v_i have the same weight. This proves the claim. □

Below, for any $i \in [N]$, w_i denotes the (common) weight of all edges in E_i.

Observation IV.17. *The private edges $\{E_i\}_{i=1}^N$ are pairwise disjoint.*

Proof. First, note that the cluster vertices $\{C_i\}_{i=1}^N$ are clearly pairwise disjoint. Now consider the iteration when cluster i is formed. Recall that S is some subset of v_i's children and cluster C_i contains *all* remaining vertices in $\bigcup_{u\in S} T_u$. Therefore, at the end of this iteration, we have $T_u \setminus M = \emptyset$ for all $u \in S$. So for any cluster j formed later, j's private edges cannot contain any E_i-edge. □

Note that the new tree $\mathcal{T}'$ contains the original tree $\mathcal{T}$. We will use ℓ to also denote the distance in the new tree-metric $\mathcal{T}'$.

Observation IV.18. *For any $i \in [N]$ and $z \in C_i$, the distance $\ell(c_i, z) \leq 3\ell(c_i, v_i)$ in tree-metric $\mathcal{T}'$.*

Proof. Consider the iteration when cluster i is formed. Recall that the new vertex c_i in tree $\mathcal{T}'$ is connected to the vertex v_i (chosen in step 3). By Definition IV.15, the weights on the path from v_i to z in $\mathcal{T}$ are geometrically decreasing by factor 2 in each step. So the distance $\ell(v_i, z) \leq 2\ell(c_i, v_i)$, and we have the distance $\ell(c_i, z) \leq \ell(c_i, v_i) + \ell(v_i, z) \leq 3\ell(c_i, v_i)$. □

Observation IV.19. *For each $i \in [N]$ and walk ρ in tree $\mathcal{T}$ originating from r and visiting some C_i-vertex, we have $\rho \cap E_i \neq \emptyset$.*

Proof. This follows from the fact that E_i cuts all paths from r to C_i in tree $\mathcal{T}$. So path ρ must contain some E_i-edge. □

Observation IV.20. *For any $v \in V$ and walk ρ in tree $\mathcal{T}$ from r to v, the length of ρ is at least $\sum_{i \in [N]: \rho \cap C_i \neq \emptyset} \ell(v_i, c_i)$.*

Proof. Let $I = \{i \in [N] : \rho \cap C_i \neq \emptyset\}$. By Observation IV.19, $\rho \cap E_i \neq \emptyset$ for each $i \in I$. Moreover, by Observation IV.17, the edges $\rho \cap E_i$ are disjoint for $i \in I$. So the length of ρ is at least $\sum_{i \in I} \ell(v_i, c_i)$ where $\ell(v_i, c_i)$ is the weight of any E_i-edge. □

Observation IV.21. *For any $i \in [N]$, the expected cost of tour τ_i restricted to active vertices is at most $12\ell(v_i, c_i)$.*

Proof. Consider the iteration when cluster i is formed. Recall that tour τ_i is on vertices $v_i \cup C_i$. By the structure of tree $\mathcal{T}$ based on Definition IV.15, the distance between any pair of vertices in $v_i \cup C_i$ is at most $4\ell(v_i, c_i)$. If $A \subseteq V$ denotes the set of active vertices, the cost of τ_i restricted to $A \cap (v_i \cup C_i)$ is at most $4\ell(v_i, c_i) \cdot (1 + |A \cap C_i|)$. So the expected cost restricted to active vertices is at most

$$4\ell(v_i, c_i) \left(1 + \sum_{z \in C_i} p_z\right) \leq 12\ell(v_i, c_i),$$

where we used the fact that the total probability in each cluster is at most two. □

Lemma IV.22. *The re-optimization cost of instance $\mathcal{J}$ is at most $7 \cdot \mathsf{ROPT}$.*

Proof. Let $A \subseteq V$ and $I \subseteq [N]$ denote the active subsets in the instances $\mathcal{I}$ and $\mathcal{J}$ respectively. For any $i \in [N]$, by definition of q_i we have $\Pr_I[i \in I] = q_i = \Pr_A[C_i \cap A \neq \emptyset]$. For any $A \subseteq V$ let $\gamma(A) = \{i \in [N] : C_i \cap A \neq \emptyset\}$. As the random set A is independent across vertices V, the distribution of $\gamma(A)$ is identical to that of I. So

$$\mathbb{E}_I[\text{optimal latency of } I \text{ in } \mathcal{T}'] = \mathbb{E}_A[\text{optimal latency of } \gamma(A) \text{ in } \mathcal{T}']. \tag{4.14}$$

For each $A \subseteq V$, let ρ_A denote the TRP tour corresponding to OPT_A; so the latency of ρ_A is OPT_A. Using this, we construct a TRP tour in $\mathcal{T}'$ that visits vertices $\{c_i : i \in \gamma(A)\}$. First, choose an arbitrary vertex $u_i \in C_i \cap A$ for each $i \in \gamma(A)$ and let ρ'_A denote tour ρ_A restricted to $\{u_i : i \in \gamma(A)\}$. By triangle inequality, the visit-time of each u_i in ρ'_A is at most that in ρ_A. Note that ρ'_A is also a valid tour in $\mathcal{T}'$ as tree $\mathcal{T} \subseteq \mathcal{T}'$. We now modify tour ρ'_A in tree $\mathcal{T}'$ by adding paths from u_i to c_i and c_i to u_i at the point when u_i is visited, for each

$i \in \gamma(A)$. Let ρ''_A denote the resulting tour. We will show that:

$$\text{the visit-time of } c_i \text{ in } \rho''_A \;\leq\; 7 \times (\text{visit-time of } u_i \text{ in } \rho'_A)\,, \qquad \forall i \in \gamma(A). \tag{4.15}$$

Fix any $i \in \gamma(A)$. Let $K_i \subseteq \gamma(A)$ denote the indices that are visited at or before u_i in ρ'_A. So all the vertices $\{u_j : j \in K_i\}$ are visited in tour ρ'_A before u_i. By Observation IV.20, the visit-time t'_i of u_i in ρ'_A is at least $\sum_{j \in K_i} \ell(v_j, c_j)$. By definition of tour ρ''_A, the visit-time t''_i of c_i is at most $t'_i + 2\sum_{j \in K_i} \ell(u_j, c_j) \leq t'_i + 6\sum_{j \in K_i} \ell(v_j, c_j)$ where the last inequality is by Observation IV.18. It now follows that $t''_i \leq 7 \cdot t'_i$ which proves (4.15).

Now, using (4.15), the latency of tour ρ''_A is at most 7 times that of tour ρ'_A (and also ρ_A). So the optimal latency for visiting $\gamma(A)$ in $\mathcal{T}'$ is at most $7 \cdot \mathsf{OPT}_A$. Combined with (4.14), the lemma follows. □

Lemma IV.23. *Let τ_1 and τ_2 be two tours from root r visiting disjoint subsets of vertices V_1 and V_2. Let L_1 (resp. L_2) denote the total latency of tour τ_1 (resp. τ_2). Then, there is a single tour τ from r visiting $V_1 \cup V_2$ of total latency at most $16(L_1 + L_2)$. Moreover, the tour τ can be found in polynomial time.*

Proof. For each $k \geq 0$, let $\tau_1(k)$ denote the maximal prefix of tour τ_1 of length at most 2^k. Define $\tau_2(k)$ similarly. Tour τ is obtained as follows. For each $k = 0, 1, \cdots$: follow $\tau_1(k)$ and return to r, then follow $\tau_2(k)$ and return to r. We now bound the latency of τ. Consider any vertex $v \in V_1$ (the case $v \in V_2$ is symmetric). Let k be the smallest value such that $v \in \tau_1(k)$. Then, the latency of v in τ_1 is at least 2^{k-1}. By construction, the latency of v in τ is at most $2\sum_{j=0}^{k} 2^{j+1} \leq 2^{k+3}$ which is at most 16 times v's latency in τ_1. Adding the

contribution over all $v \in V_1 \cup V_2$ yields the lemma. □

Lemma IV.24. *Consider two instances of* a priori *TRP on the same metric* (V, d) *but different probability distributions* $\{p'_v\}_{v\in V}$ *and* $\{p_v\}_{v\in V}$ *where* $0 \le p_v \le p'_v \le 1$ *for each* $v \in V$*. Then, the re-optimization cost under* $\{p_v\}_{v\in V}$ *is at most that under* $\{p'_v\}_{v\in V}$*.*

Proof. Let function $f(p_1, \cdots p_n)$ denote the re-optimization cost of the instance on metric (V, d) as a function of vertex probabilities $\{p_v\}_{v\in V}$. We can express f as a multilinear polynomial

$$f(\mathbf{p}) = \sum_{A\subseteq V} \left(\prod_{u\in A} p_u \prod_{w\in V\setminus A} (1 - p_w) \right) \cdot \mathsf{OPT}_A \ .$$

We will show that all partial derivatives of f are non-negative. This would imply the lemma. Now, the v^{th} partial derivative is:

$$\frac{\partial f}{\partial p_v} = \sum_{A\subseteq V\setminus v} \left(\prod_{u\in A} p_u \prod_{w\in V\setminus A\setminus v} (1 - p_w) \right) \left(\mathsf{OPT}_{A\cup v} - \mathsf{OPT}_A \right) .$$

For any $A \subseteq V \setminus v$, note that we can shortcut the solution corresponding to $\mathsf{OPT}_{A\cup v}$ over v, to obtain a feasible solution for subset A. By triangle inequality, the cost of this solution is at most $\mathsf{OPT}_{A\cup v}$. So $\mathsf{OPT}_A \le \mathsf{OPT}_{A\cup v}$. This shows that each term in the above summation is non-negative and so $\frac{\partial f}{\partial p_v} \ge 0$. □

Lemma IV.25. *The optimal cost of the deterministic TRP for TRP instance with demands* $\{c_i\}_{i=1}^N$ *is* $O(1) \cdot \mathsf{ROPT}$*.*

Proof. Let ROPT' denote the re-optimization cost of instance $\mathcal{J}$. By the construction of clusters C_i, we have $q_i = 1 - \prod_{v\in C_i}(1 - p_v) \ge 1 - e^{-\sum_{v\in C_i} p_v} \ge 1 - 1/e$ for all $i \in [N]$. Let

$\mathcal{J}'$ denote the *a priori* TRP instance on $\mathcal{T}'$ with activation probability $\frac{1}{2}$ for each of $\{c_i\}_{i=1}^N$. The re-optimization cost of $\mathcal{J}'$ is:

$$\frac{1}{2^N}\sum_{X\subseteq[N]} \mathsf{OPT}_X \leq \mathsf{ROPT}', \tag{4.16}$$

where the inequality is by Lemma IV.24. Above, OPT_X is the optimal cost for the TRP instance on $\mathcal{T}'$ with active vertices $\{c_i : i \in X\}$.

For any $X \subseteq [N]$, let $\tau_1(X)$ (resp. $\tau_2(X)$) denote the optimal TRP tour for active set X (resp. $[N] \setminus X$). Note that the total latency of $\tau_1(X)$ and $\tau_2(X)$ are OPT_X and $\mathsf{OPT}_{N\setminus X}$ respectively. By Lemma IV.23, there is a tour $\tau(X)$ that visits all the vertices $\{c_i\}_{i=1}^N$ with latency at most $16(\mathsf{OPT}_X + \mathsf{OPT}_{N\setminus X})$. The optimal value of the deterministic TRP instance on tree $\mathcal{T}'$ is clearly at most:

$$\min_{X\subseteq[N]} 16(\mathsf{OPT}_X + \mathsf{OPT}_{N\setminus X}) \leq \frac{16}{2^N}\sum_{X\subseteq[N]} (\mathsf{OPT}_X + \mathsf{OPT}_{N\setminus X}) = \frac{32}{2^N}\sum_{X\subseteq[N]} \mathsf{OPT}_X$$

Which is at most $32\cdot\mathsf{ROPT}'$ by inequality 4.16. Now, as $\mathsf{ROPT}' \leq 7\cdot\mathsf{ROPT}$ by Lemma IV.22, we can conclude the proof. □

Lemma IV.26. *The expected latency cost of* a priori *tour* π *is* $O(1)\cdot\mathsf{ROPT}$.

Proof. Recall that δ is an $O(1)$-approximately optimal deterministic TRP tour visiting $\{c_i\}_{i=1}^N$ in tree $\mathcal{T}'$. Fix any $i \in [N]$ and let $K_i \subseteq [N]$ index the vertices visited by δ at or before c_i, so we have $i \in K_i$. The visit-time of c_i in δ is $t_i \geq \sum_{j\in K_i} \ell(v_j, c_j)$. Note that the tour must visit vertex v_i immediately before visiting c_i (recall that c_i is a leaf node connected to v_i). Recall that π is the master tour that visits all the vertices V by visiting tours τ_i in

the same order as δ visits clusters c_is. Recall that τ_i is a tour on vertices $\{v_i\} \cup C_i$ originating from v_i. Note that π returns to v_i after traversing each τ_i.

For any subset $A \subseteq V$ of active vertices, the latency of π for A can be upper-bounded as follows:

For each $i \in [N]$, let T_i denote the visit-time of vertex v_i (after completing τ_i) in π. We can bound

$$\mathbb{E}[T_i] \leq t_i + \sum_{j \in K_i} \mathbb{E}_A[\text{cost of } \tau_j \text{ restricted to } A] \leq t_i + 12 \sum_{j \in K_i} \ell(v_j, c_j) \leq 13 t_i,$$

where the second inequality is by Observation IV.21.

Fix any $i \in [N]$. Note that the latency of each active vertex in τ_i is at most T_i. For any $v \in C_i$, the tours obtained by restricting π to active sets A and $A \setminus v$ differ by at most two edges: moreover, these edges are between vertices of $v_i \cup C_i$ and so each such edge has the cost at most $2\ell(v_i, c_i)$. Hence $\mathbb{E}[T_i | v \in A] \leq \mathbb{E}[T_i] + 4\ell(v_i, c_i)$ for any $v \in C_i$. We can now upper bound the expected latency from C_i-vertices as

$$\begin{aligned}\sum_{v \in C_i} \Pr[v \in A] \cdot \mathbb{E}[T_i | v \in A] &\leq \left(\sum_{v \in C_i} \Pr[v \in A] \right) (\mathbb{E}[T_i] + 4\ell(v_i, c_i)) \\ &\leq 2 \cdot (\mathbb{E}[T_i] + 4\ell(v_i, c_i)) \leq 26 t_i + 8\ell(v_i, c_i) \leq 34 t_i.\end{aligned}$$

The last inequality uses the fact that $t_i \geq \ell(v_i, c_i)$ as the edge (v_i, c_i) must be traversed to reach c_i.

Finally, the expected total latency of π is at most $34 \sum_{i=1}^{N} t_i$. Note that $\sum_{i=1}^{N} t_i$ is the total latency of the deterministic tour δ, which is $O(1) \cdot \mathsf{ROPT}$ by Lemma IV.25. This completes the proof. $\square$

4.3.3 Analysis for Light Clusters

Here we define a new instance $\mathcal{K}$ of *a priori* TRP on the star-metric with center r and leaves U with edge-weights $\{\ell(r,u) : u \in U\}$. Each vertex $u \in U$ is active with probability $q_u = 1 - \prod_{v \in B_u}(1-p_v)$. Note that q_u is exactly the probability that there is some active vertex in cluster B_u in the original instance $\mathcal{I}$. To reduce notation we also use $w_u := \ell(r,u)$ for $u \in U$.

Observation IV.27. *The re-optimization cost of $\mathcal{K}$ is at most* ROPT.

Proof. Let $A \subseteq V$ and $K \subseteq U$ denote the active subsets in instances $\mathcal{I}$ and $\mathcal{K}$. For any $A \subseteq V$ let $\gamma(A) = \{u \in U : B_u \cap A \neq \emptyset\}$. As the random set A is independent across vertices V, the distribution of $\gamma(A)$ is identical to that of K. So it suffices to bound the expected latency of $\gamma(A)$. Fix any $A \subseteq V$ and let ρ_A denote the TRP tour corresponding to OPT_A; so the latency of ρ_A is OPT_A. For each $u \in \gamma(A)$, note that ρ_A must visit u before the vertices $B_u \cap A$. So the total latency of $\gamma(A)$-vertices in this tour is at most OPT_A. Taking expectation over A, the claim follows. □

Let σ' be the *a priori* tour for instance $\mathcal{K}$ that visits vertices $u \in U$ in increasing order of $w_{r,u}$. For each $u \in U$ let $S_u \subseteq U$ be the vertices that appear before u in σ'.

Observation IV.28. *The expected latency of a priori tour σ' equals the re-optimization cost of $\mathcal{K}$. Hence,* $\sum_{u \in U} q_u \left(w_u + 2 \sum_{j \in S_u} q_j w_j \right) \leq \mathsf{ROPT}$

Proof. Let $K \subseteq U$ denote the active subset in $\mathcal{K}$. It is clear that the optimal latency tour in any star metric involves visiting the leaves in increasing order of distance from the root.

This is indeed the tour obtained by restricting σ' to K. This proves the first statement.

Note that the expected visit-time of vertex u in σ' *conditioned on* $u \in K$ is exactly $w_u + 2\sum_{j \in S_u} q_j w_j$: this is because all other vertices are active independently. So the expected latency of σ' is

$$\sum_{u \in U} q_u \left(w_u + 2 \sum_{j \in S_u} q_j w_j \right).$$

Combined with Observation IV.27 this proves the second statement. □

Observation IV.29. *For any $u \in U$ we have $p(B_u) \geq q_u \geq (1 - 1/e) \cdot p(B_u)$.*

Proof. The inequality $q_u \leq p(B_u)$ follows by union bound. For the other inequality,

$$q_u = 1 - \prod_{v \in B_u} (1 - p_v) \geq 1 - e^{-p(B_u)} \geq (1 - 1/e) \cdot p(B_u),$$

where the first inequality uses $1 + x \leq e^x$ (for all x) and the second inequality uses $1 - e^{-x} \geq (1 - 1/e) \cdot x$ for $x \in [0, 1]$ and the fact that $p(B_u) \leq 1$. □

Lemma IV.30. *The expected latency of tour σ, obtained in step 14 of Algorithm 11, is at most $\frac{6e}{e-1}\mathsf{ROPT}$.*

Proof. Consider any $u \in U$ and vertex $v \in B_u$. Conditioned on v being active, its expected visit-time in σ is at most $d(r, v) + 2\sum_{j \in (S_u \cup u)} \sum_{z \in B_j} p_z \cdot d(r, z)$. This follows from the fact that the only vertices visited before v are the active vertices in $\cup_{j \in (S_u \cup u)} B_j$ and these vertices are active independently. For each $z \in B_j$, by the structure of tree $\mathcal{T}$ it follows that $d(r, z) \leq 2w_j$. So the expected visit-time of v conditioned on being active is $t_v \leq$

$2w_u + 4\sum_{j\in(S_u\cup u)} p(B_j)\cdot w_j \le 6w_u + 4\sum_{j\in S_u} p(B_j)\cdot w_j$. By Observation IV.29,

$$t_v \le 6w_u + \frac{4e}{e-1}\sum_{j\in S_u} q_j \cdot w_j, \quad \forall v \in B_u,\, u \in U.$$

Recall from Observation IV.28 that the expected visit-time of vertex $u \in U$ in tour σ' (conditioned on being active) is $t'_u := w_u + 2\sum_{j\in S_u} q_j w_j$. So we have $t_v \le 6\cdot t'_u$ for all $v \in B_u$ and $u \in U$. So the expected latency of σ is at most:

$$\sum_{u\in U}\sum_{v\in B_u} p_v \cdot t_v \le 6\sum_{u\in U} p(B_u)\cdot t'_u \le \frac{6e}{e-1}\sum_{u\in U} q_u \cdot t'_u \le \frac{6e}{e-1}\cdot \mathsf{ROPT}.$$

The second inequality is by Observation IV.29 and the last one is by Observation IV.28. □

4.3.4 Combining the Two *A priori* Tours

Here we show that any two *a priori* tours can be combined to obtain a new *a priori* tour of expected cost $O(1)$ times the total. This is used in step 15 of Algorithm 11 to combine π and σ into one.

Lemma IV.31. *Let π and σ be two a priori tours visiting disjoint subsets of vertices V_1 and V_2. Let L_1 (resp. L_2) denote the expected latency of tour π (resp. σ). Then, there is a single a priori tour τ visiting $V_1 \cup V_2$ of expected latency at most $19(L_1 + L_2)$. Moreover, the tour τ can be found in polynomial time.*

Proof. For each $v \in V_1$, let π_v denote the prefix of *a priori* tour π from r until v and let E_v be the expected length of π_v restricted to active vertices (note that the end point need not be v). Note that E_v is monotonically non-decreasing along tour π. Moreover, the expected arrival-time of v (conditioned on being active) in π is at least E_v. Similarly, we define E_v for $v \in V_2$ based on tour σ.

For each $k \geq 0$, let $\pi(k)$ denote the maximal prefix of *a priori* tour π such that $E_v \leq 2^k$ for each $v \in \pi(k)$. Define $\sigma(k)$ similarly. *A priori* tour τ is obtained as follows. For each $k = 0, 1, \cdots$: follow $\pi(k)$ and return to r, then follow $\sigma(k)$ and return to r.

Consider any vertex $v \in V_1$ (the case $v \in V_2$ is symmetric). Let t_v be the expected arrival-time of v in π (conditioned on being active); note that $t_v \geq \max\{E_v, \ell(r, v)\}$. Let k be the smallest value such that $v \in \pi(k)$; so $E_v \geq 2^{k-1}$. We now bound the expected arrival-time t'_v of v (conditioned on being active) in τ. For any $j < k$, the expected length due to $\pi(j)$ (or $\sigma(j)$) is at most $2 \cdot 2^j$ as its vertices are active independent of v. Let f_v denote the expected length (conditioned on v being active) due to $\pi(k)$. Then $t'_v \leq 2\sum_{j=0}^{k} 2^{j+1} + f_v \leq 2^{k+3} + f_v$. Below, we will bound $f_v \leq 2^k + \ell(r, v)$, which would imply $t'_v \leq 9 \cdot 2^k + \ell(r, v) \leq 19 \cdot \max\{2^{k-1}, \ell(r, v)\} \leq 19 t_v$. Taking expectations and adding over all $v \in V_1 \cup V_2$ yields the lemma.

It remains to show $f_v \leq 2^k + \ell(r, v)$. Let A denote the active subset and $u \in A$ be the active vertex immediately before v in $\pi(k)$. Let F denote the path $\pi(k)$ until u, restricted to A; and let $\ell(F)$ be its length. Then, the length of $\pi(k)$ until v (again restricted to A) is:

$$L_v = \ell(F) + \ell(u, v) \leq 2 \cdot \ell(F) + \ell(r, v),$$

where we used the triangle inequality: the distance $\ell(u, v)$ is at most the length of path F in reverse (from u to r) plus edge (r, v). Taking expectations,

$$f_v = \mathbb{E}[L_v | v \in A] \leq 2 \cdot \mathbb{E}[\ell(F) | v \in A] + \ell(r, v) = \mathbb{E}[\ell(F)] + \ell(r, v) \leq 2^k + \ell(r, v).$$

The second equality is because path F is independent of v being active. The last inequality

is because F is a sub-path of $\pi(k)$ restricted to A, which has expected length at most 2^k. $\square$

This shows that τ gives a constant-factor approximation algorithm for *a priori* TRP with respect to the re-optimization, which is what we wantd to prove in Theorem IV.4.

4.4 Converting the Metric to a Relaxed 2-HST

In this section, we prove Theorem IV.5 and conclude the chapter:

Theorem IV.5. *If there is an $\mathcal{O}(\alpha)$-approximation algorithm for a priori TRP on a relaxed 2-HST w.r.t the re-optimization solution, then there is an $\mathcal{O}(\alpha \cdot \log n)$-approximation algorithm for a priori TRP on a general metric w.r.t the re-optimization solution.*

To do this, we explain how we convert any general metric (V, d) to a relaxed 2-HST $(\mathcal{T}, \ell_{\mathcal{T}})$. First, we use the algorithm from [32] to change metric (V, d) to a 2-HST $(\mathcal{T}', \ell_{\mathcal{T}'})$ and use Theorem 2 from [32] which is restated as follows:

Theorem IV.32. *For every metric (V, d), the algorithm in [32] finds a set of 2-HST metrics $\mathcal{S}'$, with distribution $\mathcal{D}$ such that:*

1. $\forall(\mathcal{T}', \ell) \in \mathcal{S}'$, *all vertices in V are leaves of $\mathcal{T}'$*
2. $\forall u, w \in V, \forall(\mathcal{T}', \ell_{\mathcal{T}'}) \in \mathcal{S}' : \ell_{\mathcal{T}'}(u, w) \geq d(u, w)$
3. $\forall u, w \in V : E_{\ell_{\mathcal{T}'}}[\ell_{\mathcal{T}'}(u, w)] = \mathcal{O}(\log n) d(u, w)$

Then, we change the set $\mathcal{S}'$ to a set $\mathcal{S}$ of relaxed 2-HSTs with the same distribution. The reason we do this is to make $r \in V$ which is a leaf in 2-HST $\mathcal{T}'$ the root of the relaxed 2-HST $\mathcal{T}$. So, we prove the following lemma:

Lemma IV.33. *For every metric (V, d) with root $r \in V$, we can find a set of relaxed 2-HST metrics $\mathcal{S}$, with distribution $\mathcal{D}$ such that:*

1. $\forall (\mathcal{T}, \ell) \in \mathcal{S}$, $\mathcal{T}$ *is rooted at* r *and all vertices in* $V \setminus \{r\}$ *are leaves of* $\mathcal{T}$
2. $\forall u, w \in V, \forall (\mathcal{T}, \ell_{\mathcal{T}}) \in \mathcal{S} : \ell_{\mathcal{T}}(u, w) \geq d(u, w)$
3. $\forall u, w \in V : E_{\ell_{\mathcal{T}}}[\ell_{\mathcal{T}}(u, w)] = \mathcal{O}(\log n) d(u, w)$

Proof. We apply the algorithm in [32] to find a set of 2-HST metrics $\mathcal{S}'$, with distribution $\mathcal{D}$. Recall that by Definitions IV.14 and IV.15, the main difference between a 2-HST and a relaxed 2-HST is that in a relaxed 2-HST the edges incident to root need not to have equal weights. Consider an arbitrary $(\mathcal{T}', \ell_{\mathcal{T}'}) \in \mathcal{S}'$ and let s denote the root of $\mathcal{T}'$. By Theorem 4.4 all vertices in V, including r, are leaves of $\mathcal{T}'$. We consider the $r - s$ path P in $\mathcal{T}'$ as a sequence of vertices $r = v_0, v_1, v_2, ..., v_{k-1}, v_k = s$. For every $i \in [k] \cup \{0\}$, let T'_{v_i} be the subtree rooted at v_i. We define $H_0 = T'_{v_0} = \{r\}$, and $H_i = T'_{v_i} \setminus T'_{v_{i-1}}$ $\forall i \in [k]$. So H_i is a subtree rooted at v_i. Now, we contract all vertices in P and let this new vertex be the root r of $\mathcal{T}'$. See Figure 4.5 for an illustration.

We multiply the weight of each edge by a factor 3 and define $(\mathcal{T}, \ell_{\mathcal{T}})$ to be the resulting metric. Note that r is the new root and every leaf of $\mathcal{T}'$ except for r is a leaf of $\mathcal{T}$. So all vertices in $V \setminus \{r\}$ are leaves of $\mathcal{T}$, and condition 1 holds. To prove condition 2, consider arbitrary vertices $w, u \in V$. If $u, w \in H_i$ for some i, then by our construction we have $\ell_{\mathcal{T}}(u, w) = 3\ell_{\mathcal{T}'}(u, w)$, and by Theorem 4.4 we have:

$$d(u, w) \leq \ell_{\mathcal{T}'}(u, w) \leq \ell_{\mathcal{T}}(u, w)$$

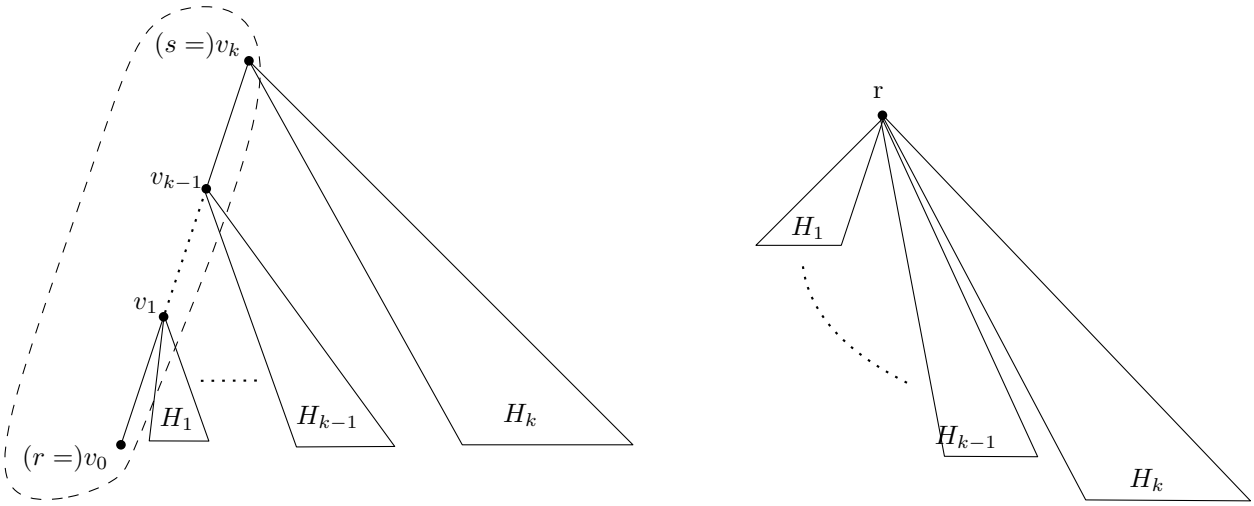

Figure 4.5: From left to right: the initial 2-HST and its corresponding relaxed 2-HST

Now, assume $u \in H_i$ and $w \in H_j$ for some $i, j : i < j$. We can write:

$$\begin{aligned} \ell_{\mathcal{T}'}(u,w) &= \ell_{\mathcal{T}'}(u,v_i) + \ell_{\mathcal{T}'}(v_i,v_j) + \ell_{\mathcal{T}'}(v_j,w) \\ &= \ell_{\mathcal{T}}(u,r)/3 + \ell_{\mathcal{T}'}(v_i,v_j) + \ell_{\mathcal{T}}(r,w)/3 \end{aligned} \tag{4.17}$$

Since $w \in \{r\} \cup (V \setminus P)$, $w \neq v_j$, and since all children of a node in $\mathcal{T}'$ have equal length, we have:

$$\ell_{\mathcal{T}'}(v_j, v_{j-1}) \leq \ell_{\mathcal{T}'}(v_j, w)$$

Consider the $v_j - w$ path in $\mathcal{T}'$ and let z be the child of v_j that lies on this path. We can write:

$$\ell_{\mathcal{T}'}(v_j,w) \geq \ell_{\mathcal{T}'}(v_j,z) = \ell_{\mathcal{T}'}(v_j,v_{j-1}) \tag{4.18}$$

Where the last inequality is based on Definition IV.14 that states all edges incident to a vertex in a 2-HST need to have equal weights.

Recall that distances are decreasing by a factor of 2, as we go deeper in tree $\mathcal{T}'$, and since

$i < j$:

$$\begin{aligned}\ell_{\mathcal{T}'}(v_j, v_i) &= \ell_{\mathcal{T}'}(v_j, v_{j-1}) + \ell_{\mathcal{T}'}(v_{j-1}, v_{j-2}) + ... + \ell_{\mathcal{T}'}(v_{i+1}, v_i) \\ &= \ell_{\mathcal{T}'}(v_j, v_{j-1}) + 1/2 \cdot \ell_{\mathcal{T}'}(v_j, v_{j-1}) + ... + 1/2^{j-i}\ell_{\mathcal{T}'}(v_j, v_{j-1}) \\ &\leq 2\ell_{\mathcal{T}'}(v_j, v_{j-1}) \leq 2\ell_{\mathcal{T}'}(v_j, w) \end{aligned} \tag{4.19}$$

Note that the last inequality is due to equation (4.18). Now, by combining equations (4.17) and (4.19), we obtain:

$$\begin{aligned}\ell_{\mathcal{T}'}(u, w) &\leq \ell_{\mathcal{T}}(u, r)/3 + 2\ell_{\mathcal{T}'}(v_j, w) + \ell_{\mathcal{T}}(r, w)/3 \\ &= \ell_{\mathcal{T}}(u, r)/3 + 2\ell_{\mathcal{T}}(r, w)/3 + \ell_{\mathcal{T}}(r, w)/3 \\ &\leq \ell_{\mathcal{T}}(u, r) + \ell_{\mathcal{T}}(r, w) = \ell_{\mathcal{T}}(u, w)\end{aligned}$$

Where the equality is obtained by the fact that $\ell_{\mathcal{T}}(r, w) = 3\ell_{\mathcal{T}'}(v_j, w)$. Now, by Theorem 4.4, this results in:

$$d(u, w) \leq \ell_{\mathcal{T}'}(u, w) \leq \ell_{\mathcal{T}}(u, w)$$

Which completes our proof for condition 2. Now we prove the last condition. Note that to construct $\mathcal{T}$ from $\mathcal{T}'$, first we contracted some vertices, which does not increase any distance in the metric. Then we multiplied each distance by a factor 3. So we for any $u, w \in V$ can write:

$$\ell_{\mathcal{T}}(u, w) \leq 3\ell_{\mathcal{T}'}(u, w)$$

Now, by Theorem 4.4 we can write:

$$E_{\ell_{\mathcal{T}}}[\ell_{\mathcal{T}}(u, w)] = \mathcal{O}(E_{\ell_{\mathcal{T}'}}[\ell_{\mathcal{T}'}(u, w)]) = \mathcal{O}(\log n)d(u, w)$$

which shows condition 3 and completes the proof of this lemma. □

Now, we are ready to complete the proof of Theorem IV.5.

Proof. Consider an instance $\mathcal{I}$ of *a priori* TRP on general metric (V, d) with probabilities $\{p_v\}_{v\in V}$. By Lemma 4.4 we can find a set of relaxed 2-HST metrics $\mathcal{S}$, with distribution $\mathcal{D}$ with the mentioned conditions. Fix some $(\mathcal{T}, \ell_{\mathcal{T}}) \in \mathcal{S}$, and consider an instance $\mathcal{I}_{\mathcal{T}}$ on $(\mathcal{T}, \ell_{\mathcal{T}})$ with probabilities $\{p_v\}_{v\in\mathcal{T}}$, where $p_v = 0\ \forall v \in \mathcal{T} \setminus V$. For any *a priori* TRP instance $\mathcal{J}$, let $\mathsf{ROPT}(\mathcal{J})$ denote the re-optimization solution. For any master tour τ, we define $\mathsf{LAT}_\tau(v)$ and $\overline{\mathsf{LAT}}_\tau(v)$ to be the latency of vertex v in tour τ with respect to metric d and $\ell_{\mathcal{T}}$ respectively. By the theorem's assumption, for any metric $(\mathcal{T}, \ell_{\mathcal{T}})$ there is some master tour $\pi(\mathcal{T})$ for which:

$$E_A\left[\sum_{v\in A} \overline{\mathsf{LAT}}_{\pi(\mathcal{T})_A}(v)\right] \le \mathcal{O}(\alpha)\cdot \mathsf{ROPT}(\mathcal{I}_{\mathcal{T}})$$

We take expectation from both sides with respect to $\ell_{\mathcal{T}}$ to obtain:

$$E_{\ell_{\mathcal{T}}}\left[E_A\left[\sum_{v\in A} \overline{\mathsf{LAT}}_{\pi(\mathcal{T})_A}(v)\right]\right] \le \mathcal{O}(\alpha)\cdot E_{\ell_{\mathcal{T}}}[\mathsf{ROPT}(\mathcal{I}_{\mathcal{T}})] \tag{4.20}$$

Consider the tour $\pi(\mathcal{T})_V$ on instance $\mathcal{I}$. Note that as we have $p_v = 0$ for every $v \in \mathcal{T} \setminus V$, shortcutting $\pi(\mathcal{T})$ and $\pi(\mathcal{T})_V$ on every set $A \leftarrow \Pi$ result in the same tour $\pi(\mathcal{T})_A$. By Theorem 4.4, we know that for every $u, w \in V$, we have $d(u, w) \le \ell_{\mathcal{T}}(u, w)$, and as a result $\mathsf{LAT}_{\pi(\mathcal{T})_A}(v) \le \overline{\mathsf{LAT}}_{\pi(\mathcal{T})_A}(v)$. So we can write:

$$E_A\left[\sum_{v\in A} \mathsf{LAT}_{\pi(\mathcal{T})_A}(v)\right] \le E_A\left[\sum_{v\in A} \overline{\mathsf{LAT}}_{\pi(\mathcal{T})_A}(v)\right]$$

And by taking expectation with respect to $\ell_{\mathcal{T}} \leftarrow \mathcal{D}$ from both sides we obtain:

$$E_{\ell_{\mathcal{T}}}\left[E_A\left[\sum_{v\in A}\mathsf{LAT}_{\pi(\mathcal{T})_A}(v)\right]\right] \leq E_{\ell_{\mathcal{T}}}\left[E_A\left[\sum_{v\in A}\overline{\mathsf{LAT}}_{\pi(\mathcal{T})_A}(v)\right]\right]$$

Which is true as $\mathsf{LAT}_{\pi(\mathcal{T})_A}(v)$ is independent of $\ell_{\mathcal{T}}$. Now, by combining the above inequality and equation (4.20), we get:

$$E_{\ell_{\mathcal{T}}}\left[E_A\left[\sum_{v\in A}\mathsf{LAT}_{\pi(\mathcal{T})_A}(v)\right]\right] \leq \mathcal{O}(\alpha)E_{\ell_{\mathcal{T}}}[\mathsf{ROPT}(\mathcal{I}_{\mathcal{T}})] \tag{4.21}$$

So to conclude the proof, it is enough to show:

$$E_{\ell_{\mathcal{T}}}[\mathsf{ROPT}(\mathcal{I}_{\mathcal{T}})] \leq \mathcal{O}(\log n)\mathsf{ROPT}(\mathcal{I}) \tag{4.22}$$

Let $\delta(A)$ be the optimal deterministic tour for vertices A on metric d; therefore the total latency of $\delta(A)$ is at most the optimal deterministic TRP solution for vertices A on metric $\ell_{\mathcal{T}}$ and we have:

$$\mathsf{ROPT}(\mathcal{I}_{\mathcal{T}}) \leq E_A\left[\sum_{v\in A}\overline{\mathsf{LAT}}_{\delta(A)}(v)\right]$$

By taking expectation with respect to $\ell_{\mathcal{T}}$ from both sides we get:

$$E_{\ell_{\mathcal{T}}}[\mathsf{ROPT}(\mathcal{I}_{\mathcal{T}})] \leq E_{\ell_{\mathcal{T}}}\left[E_A\left[\sum_{v\in A}\overline{\mathsf{LAT}}_{\delta(A)}(v)\right]\right]$$

$$E_{\ell_{\mathcal{T}}}[\mathsf{ROPT}(\mathcal{I}_{\mathcal{T}})] \leq E_A\left[\sum_{v\in A}E_{\ell_{\mathcal{T}}}\left[\overline{\mathsf{LAT}}_{\delta(A)}(v)\right]\right] \tag{4.23}$$

Also, by condition 3 of Lemma 4.4, we have $E_{\ell_{\mathcal{T}}}[\ell_{\mathcal{T}}(u,w)] = \mathcal{O}(\log n)d(u,w)\ \forall u,w \in V$. Therefore for every $A \subseteq V$ and $v \in A$:

$$E_{\ell_{\mathcal{T}}}[\overline{\mathsf{LAT}}_{\delta(A)}(v)] = \mathcal{O}(\log n)\mathsf{LAT}_{\delta(A)}(v)$$

Now, we combine this with equation (4.23):

$$E_{\ell_{\mathcal{T}}}[\mathsf{ROPT}(\mathcal{I}_{\mathcal{T}})] \leq \mathcal{O}(\log n) E_A \left[\sum_{v \in A} \mathsf{LAT}_{\delta(A)}(v) \right] = \mathcal{O}(\log n) \mathsf{ROPT}(\mathcal{I}) \tag{4.24}$$

And the proof is complete by using equations (4.21), (4.24) as follows:

$$\begin{aligned} E_{\ell_{\mathcal{T}}} \left[E_A \left[\sum_{v \in A} \mathsf{LAT}_{\pi(\mathcal{T})_A}(v) \right] \right] &\leq \mathcal{O}(\alpha) E_{\ell_{\mathcal{T}}}[\mathsf{ROPT}(\mathcal{I}_{\mathcal{T}})] \\ &\leq \mathcal{O}(\alpha \log n) \mathsf{ROPT}(\mathcal{I}) \end{aligned}$$

Note that we obtain a randomized master tour $\pi(\tau)$ as $\mathcal{T}$ is a random tree metric. As discussed in [32] the number of trees in $\mathcal{S}$ is $\mathcal{O}(n \log n)$. So we can enumerate all such trees in polynomial time and choose $\mathcal{T} \in \mathcal{S}$ that leads to the minimum objective for $\pi(\mathcal{T}) = \pi^*$ to obtain a deterministic algorithm. So we get:

$$E_A \left[\sum_{v \in A} \mathsf{LAT}_{\pi^*_A}(v) \right] \leq \mathcal{O}(\alpha \log n) \mathsf{ROPT}(\mathcal{I})$$

□

The results in this chapter appear in [42].

CHAPTER V

Conclusion

In this dissertation, we focused on three problems in stochastic optimization, where more information about the random data is revealed over time.

In Chapter II, we studied a general sequential decision making problem, Adaptive Submodular Ranking, where we repeatedly choose from a set of weighted decisions to cover a random scenario. We also studied the routing version of this problem, where the solution was an adaptive *path* of decisions. We obtained polylogarithmic-factor approximation algorithms for both versions. We also implemented our algorithm on real-world and synthetic data sets, and found good empirical performance. One possible future direction, is to consider the case where the feedback is noisy or missing. This is motivated by the fact that in many applications (such as search ranking), the feedback comes from users and it might be prone to error.

In Chapter III, we considered the Optimal Decision Tree problem, which is an special case of Adaptive Submodular Ranking, in presence of noise. In this problem, we choose tests sequentially to *identify* a random scenario. We gave the first approximation algorithms

for this problem that can handle any number of noisy outcomes in both the non-adaptive and adaptive settings. We also presented the result of implementing these algorithms and showed how close it is to an information theoretic lower bound. We found that in practice, our algorithms can do much better than the approximation guarantee. One way to extend this work, is by considering the case where we need to choose a batch of tests rather than a single test. So we receive feedback from the whole batch simultaneously after its selection, which can help us in choosing the next batch. This is motivated by the active learning application, where labels of data points are obtained manually - which is often done in batches.

In Chapter IV, we studied *A Priori* Traveling Repairman Problem, where a random subset of vertices is active. The goal was to find a master tour such that the expected cost of its shortcut tour over active vertices is minimized. To evaluate the performance of an algorithm, we compared to (1) the expected cost of an optimal master tour, and (2) the "re-optimization" solution which is the expected cost of the optimal tour over active vertices. We gave a constant-factor approximation algorithm with respect to the optimal master tour and a logarithmic-factor approximation algorithm with respect to the re-optimization solution. A natural future direction is to find a constant-factor approximation algorithm with respect to the re-optimization solution or provide a lower bound based on the gap between the two solutions.

BIBLIOGRAPHY

[1] M. Adler and B. Heeringa. Approximating optimal binary decision trees. *Algorithmica*, 62(3-4):1112–1121, 2012.

[2] F. N. Afrati, S. S. Cosmadakis, C. H. Papadimitriou, G. Papageorgiou, and N. Papakostantinou. The complexity of the travelling repairman problem. *Theoretical Informatics and Applications*, 20(1):79–87, 1986.

[3] E. M. Arkin, H. Meijer, J. S. Mitchell, D. Rappaport, and S. S. Skiena. Decision trees for geometric models. *International Journal of Computational Geometry & Applications*, 8(03):343–363, 1998.

[4] S. Arora and G. Karakostas. Approximation schemes for minimum latency problems. *SIAM Journal on Computing*, 32(5):1317–1337, 2003.

[5] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *SODA*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[6] Y. Azar and I. Gamzu. Ranking with submodular valuations. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1070–1079. SIAM, 2011.

[7] Y. Azar, I. Gamzu, and X. Yin. Multiple intents re-ranking. In *STOC*, pages 669–678, 2009.

[8] M. Balcan, A. Beygelzimer, and J. Langford. Agnostic active learning. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, pages 65–72, 2006.

[9] N. Bansal, A. Gupta, and R. Krishnaswamy. A constant factor approximation algorithm for generalized min-sum set cover. In *SODA*, pages 1539–1545, 2010.

[10] 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.

[11] G. Bellala, S. Bhavnani, and C. Scott. Active diagnosis under persistent noise with unknown noise distribution: A rank-based approach. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 155–163, 2011.

[12] G. Bellala, S. K. Bhavnani, and C. Scott. Group-based active query selection for rapid diagnosis in time-critical situations. *IEEE Trans. Information Theory*, 58(1):459–478, 2012.

[13] D. Bertsimas. *Probabilistic combinatorial optimization problems*. PhD thesis, Massachusetts Institute of Technology, 1988.

[14] D. J. Bertsimas. A vehicle routing problem with stochastic demand. *Operations Research*, 40(3):574–585, 1992.

[15] D. J. Bertsimas, P. Jaillet, and A. R. Odoni. A priori optimization. *Operations Research*, 38(6):1019–1033, 1990.

[16] 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. In *AMIA Annual Symposium Proceedings*, volume 2007, page 51. American Medical Informatics Association, 2007.

[17] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan. The minimum latency problem. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 163–171. ACM, 1994.

[18] G. Calinescu and A. Zelikovsky. The polymatroid steiner problems. *Journal of Combinatorial Optimization*, 9(3):281–294, 2005.

[19] A. Campbell and B. Thomas. Probabilistic traveling salesman problem with deadlines. *Transportation Science*, 42(1):1–21, 2008.

[20] 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.

[21] V. T. Chakaravarthy, V. Pandit, S. Roy, and Y. Sabharwal. Approximating decision trees with multiway branches. In *International Colloquium on Automata, Languages, and Programming*, pages 210–221. Springer, 2009.

[22] K. Chaudhuri, B. Godfrey, S. Rao, and K. Talwar. Paths, trees, and minimum latency tours. In *Proceedings of the 44th Symposium on Foundations of Computer Science (FOCS)*, pages 36–45, 2003.

[23] C. Chekuri and M. Pal. A recursive greedy algorithm for walks in directed graphs. In *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on*, pages 245–253. IEEE, 2005.

[24] Y. Chen, S. H. Hassani, and A. Krause. Near-optimal bayesian active learning with correlated and noisy tests. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, pages 223–231, 2017.

[25] Y. Chen, S. Javdani, A. Karbasi, J. A. Bagnell, S. S. Srinivasa, and A. Krause. Submodular surrogates for value of information. In *AAAI*, pages 3511–3518, 2015.

[26] F. Cicalese, E. S. Laber, and A. M. Saettler. Diagnosis determination: decision trees optimizing simultaneously worst and expected testing cost. In *ICML*, pages 414–422, 2014.

[27] S. Dasgupta. Analysis of a greedy active learning strategy. In *Advances in neural information processing systems*, pages 337–344, 2005.

[28] 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.

[29] A. Deshpande, L. Hellerstein, and D. Kletenik. Approximation algorithms for stochastic submodular set cover with applications to boolean function evaluation and min-knapsack. *ACM Transactions on Algorithms (TALG)*, 12(3):42, 2016.

[30] I. Dinur and D. Steurer. Analytical approach to parallel repetition. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 624–633, 2014.

[31] A. Erera, M. Savelsbergh, and E. Uyar. Fixed routes with backup vehicles for stochastic vehicle routing problems with time constraints. *Networks*, 54(4):270–283, 2009.

[32] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, 2004.

[33] M. Fischetti, G. Laporte, and S. Martello. The delivery man problem and cumulative matroids. *Operations Research*, 41(6):1055–1064, 1993.

[34] M. Garey and R. Graham. Performance bounds on the splitting algorithm for binary testing. *Acta Informatica*, 3:347–355, 1974.

[35] N. Garg, G. Konjevod, and R. Ravi. A Polylogarithmic Approximation Algorithm for the Group Steiner Tree Problem. *Journal of Algorithms*, 37(1):66–84, 2000.

[36] M. Goemans and J. Kleinberg. An improved approximation ratio for the minimum latency problem. *Mathematical Programming*, 82(1-2):111–124, 1998.

[37] M. Goemans and J. Vondrák. Stochastic covering and adaptivity. In *Latin American symposium on theoretical informatics*, pages 532–543. Springer, 2006.

[38] D. Golovin and A. Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *J. Artif. Intell. Res.*, 42:427–486, 2011.

[39] D. Golovin and A. Krause. Adaptive submodularity: A new approach to active learning and stochastic optimization. *CoRR*, abs/1003.3967, 2017.

[40] D. Golovin, A. Krause, and D. Ray. Near-optimal bayesian active learning with noisy observations. In *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada.*, pages 766–774, 2010.

[41] I. Gorodezky, R. D. Kleinberg, D. B. Shmoys, and G. Spencer. Improved lower bounds for the universal and a priori tsp. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 178–191. Springer, 2010.

[42] I. L. Gørtz, V. Nagarajan, and F. Navidi. Approximation algorithms for the a priori traveling repairman. *arXiv preprint arXiv:1901.06581*, 2019.

[43] 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.

[44] A. Guillory and J. Bilmes. Average-Case Active Learning with Costs. In *Algorithmic Learning Theory*, pages 141–155. Springer Berlin / Heidelberg, 2009.

[45] A. Guillory and J. Bilmes. Simultaneous learning and covering with adversarial noise. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 369–376, 2011.

[46] A. Guillory and J. A. Bilmes. Interactive submodular set cover. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 415–422, 2010.

[47] A. Gupta, A. Kumar, M. Pál, and T. Roughgarden. Approximation via cost sharing: Simpler and better approximation algorithms for network design. *ACM*, 54(3):11, 2007.

[48] A. Gupta, V. Nagarajan, and R. Ravi. Approximation algorithms for VRP with stochastic demands. *Operations Research*, 60(1):123–127, 2012.

[49] A. Gupta, V. Nagarajan, and R. Ravi. Approximation algorithms for optimal decision trees and adaptive tsp problems. *Mathematics of Operations Research*, 42(3):876–896, 2017.

[50] E. Halperin and R. Krauthgamer. Polylogarithmic inapproximability. In *Proceedings of the 35th Annual Symposium on Theory of Computing*, pages 585–594, 2003.

[51] S. Hanneke. A bound on the label complexity of agnostic active learning. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, pages 353–360, 2007.

[52] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4):19, 2015.

[53] L. Hyafil and R. L. Rivest. Constructing optimal binary decision trees is NP-complete. *Information Processing Lett.*, 5(1):15–17, 1976/77.

[54] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM (JACM)*, 22(4):463–468, 1975.

[55] S. Im, V. Nagarajan, and R. V. D. Zwaan. Minimum latency submodular cover. *ACM Transactions on Algorithms (TALG)*, 13(1):13, 2016.

[56] S. Im, M. Sviridenko, and R. Van Der Zwaan. Preemptive and non-preemptive generalized min sum set cover. *Mathematical Programming*, 145(1-2):377–401, 2014.

[57] P. Jaillet. *Probabilistic traveling salesman problems*. PhD thesis, Massachusetts Institute of Technology, 1985.

[58] 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.

[59] S. Jia, F. Navidi, V. Nagarajan, and R. Ravi. Optimal decision tree with noisy outcomes. In *Advances in Neural Information Processing Systems*, pages 3298–3308, 2019.

[60] P. Kambadur, V. Nagarajan, and F. Navidi. Adaptive submodular ranking. In *Integer Programming and Combinatorial Optimization - 19th International Conference, IPCO 2017, Waterloo, ON, Canada, June 26-28, 2017, Proceedings*, pages 317–329, 2017.

[61] H. Kellerer and U. Pferschy. A new fully polynomial time approximation scheme for the knapsack problem. *Journal of Combinatorial Optimization*, 3(1):59–71, 1999.

[62] D. Kempe, J. M. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. *Theory of Computing*, 11:105–147, 2015.

[63] S. R. Kosaraju, T. M. Przytycka, and R. S. Borgstrom. On an optimal split tree problem. In *Algorithms and Data Structures, 6th International Workshop, WADS '99, Vancouver, British Columbia, Canada, August 11-14, 1999, Proceedings*, pages 157–168, 1999.

[64] Z. W. Lim, D. Hsu, and W. S. Lee. Adaptive stochastic optimization: From sets to paths. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1585–1593, 2015.

[65] 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*, pages 510–520, 2011.

[66] 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 2008, Vancouver, BC, Canada, June 10-12, 2008*, pages 133–146, 2008.

[67] D. W. Loveland. Performance bounds for binary testing with arbitrary weights. *Acta Inform.*, 22(1):101–114, 1985.

[68] M. J. Moshkov. Greedy algorithm with weights for decision tree construction. *Fundam. Inform.*, 104(3):285–292, 2010.

[69] M. Naghshvar, T. Javidi, and K. Chaudhuri. Noisy bayesian active learning. In *50th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2012, Allerton Park & Retreat Center, Monticello, IL, USA, October 1-5, 2012*, pages 1626–1633, 2012.

[70] F. Nan and V. Saligrama. Comments on the proof of adaptive stochastic set cover based on adaptive submodularity and its implications for the group identification problem in "group-based active query selection for rapid diagnosis in time-critical situations". *IEEE Transactions on Information Theory*, 63(11):7612–7614, 2017.

[71] F. Navidi, P. Kambadur, and V. Nagarajan. Adaptive submodular ranking and routing. *To Appear in Operations Research*, arXiv preprint arXiv:1606.01530, 2016.

[72] R. D. Nowak. Noisy generalized binary search. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada.*, pages 1366–1374, 2009.

[73] J. C. Picard and M. Queyranne. The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. *Operations Research*, 26(1):86–110, 1978.

[74] A. Prasad, S. Jegelka, and D. Batra. Submodular meets structured: Finding diverse subsets in exponentially-large structured item sets. In *Advances in Neural Information Processing Systems*, pages 2645–2653, 2014.

[75] A. M. Saettler, E. S. Laber, and F. Cicalese. Trading off worst and expected cost in decision tree problems. *Algorithmica*, 79(3):886–908, 2017.

[76] M. Savelsbergh and M. Goetschalkx. A comparison of the efficiency of fixed versus variable vehicle routes. *J Business Logistics*, 16:163–187, 1995.

[77] F. Schalekamp and D. B. Shmoys. Algorithms for the universal and a priori tsp. *Operations Research Letters*, 36(1):1–3, 2008.

[78] L. S. Shapley. Cores of convex games. *International Journal of Game Theory*, 1(1):11–26, 1971.

[79] D. Shmoys and K. Talwar. A constant approximation algorithm for the a priori traveling salesman problem. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 331–343. Springer, 2008.

[80] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser. Efficient informative sensing using multiple robots. *J. Artif. Intell. Res.*, 34:707–755, 2009.

[81] R. Sitters. The minimum latency problem is np-hard for weighted trees. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 230–239. Springer, 2002.

[82] R. Sitters. Polynomial time approximation schemes for the traveling repairman and other minimum latency problems. In *Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 604–616, 2014.

[83] M. Skutella and D. P. Williamson. A note on the generalized min-sum set cover problem. *Oper. Res. Lett.*, 39(6):433–436, 2011.

[84] M. van Ee and R. Sitters. The a priori traveling repairman problem. *Algorithmica*, 80(10):2818–2833, 2018.

[85] A. Van Zuylen. Deterministic sampling algorithms for network design. *Algorithmica*, 60(1):110–151, 2011.

[86] D. P. Williamson and A. Van Zuylen. A simpler and better derandomization of an approximation algorithm for single source rent-or-buy. *Operations Research Letters*, 35(6):707–712, 2007.

[87] L. A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.

[88] B. Y. Wu. Polynomial time algorithms for some minimum latency problems. *Information Processing Letters*, 75(5):225–229, 2000.