# Exact and Approximation Algorithms for Entropy Sampling and Stochastic Optimization

by

Hessa K. Al-Thani

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
2025

Doctoral Committee:

Associate Professor Viswanath Nagarajan, Chair
Professor Marina Epelman
Assistant Professor Salar Fattahi
Assistant Professor Euiwong Lee

Hessa K. Al-Thani
hessakh@umich.edu
ORCID iD: 0000-0002-1777-1069

# ACKNOWLEDGEMENTS

Completing this doctoral thesis has been a challenging yet immensely rewarding journey, and it would not have been possible without the support and encouragement of many individuals.

First and foremost, I would like to express my deepest gratitude to my advisor, Viswanath Nagarajan. Your guidance and support have been pivotal in allowing me to complete the journey I started here. You consistently provided an objective perspective and offered valuable feedback.

I owe countless thanks to my family for their support and encouragement throughout this journey. To my parents, Johara and Khaled, thank you for instilling in me the values of independence and perseverance. To my siblings, Sarah, Loulwah, Maryam, and Hamad, your support and constant stream of live cat videos have been a source of comfort and motivation. A special thank you to my grandparents, who have been an endless source of solace during trying times.

I am also grateful for my friends, both near and far, who have been my cheerleaders and confidants. Your enthusiasm and kindness have kept me grounded and inspired. Special thanks to Ummkulthum Umlai, Anne Fitzpatrick, Narcis Jafarian, Jahan Gabayre, Alaa Khader, Fatima Amir, and Narjis Premjee for being a reliable source of support. I am especially thankful to my officemates and dear friends Zaira Pagan and Jiacheng Liu for their invaluable advice and for keeping me going when times were hard. Your presence made the daily struggles of pursuing a PhD easier to bear. Additionally, my heartfelt thanks go to Leena Ghrayeb for being a vision of strength during hard times. I extend my gratitude to my friends in CSE—Serafina Kamp, Gena Kim, and Snehal Prabhudesal—for your unwavering support and insightful discussions. Finally, I want to thank all the friends I've made along the way; your presence, whether brief or enduring, made this journey more surmountable and enjoyable.

A special shout out to my rock climbing crew, whose camaraderie and shared passion for challenging ourselves served as a much-needed outlet. Our weekly climbs not only built strength but also offered solidarity and joy at the end of rigorous weeks.

I am deeply thankful to the scholars, staff, and fellow researchers at ICERM for creating such an inspiring environment, which fueled my intellectual curiosity. The vibrant and

collaborative atmosphere has been instrumental in shaping my work and expanding my academic horizons.

I'm grateful to my sponsor the Qatar Research Development Institute, formerly Qatar National Research Fund for fundng my academic pursuits and for all the staff members who supported me through this time, Nuha, and Dr. Ayman.

Finally, I would like to acknowledge everyone who has been part of this journey, including those whose paths crossed mine, leaving lasting impressions and sparking enriching discussions.

Thank you all for being a part of this chapter in my life.

# TABLE OF CONTENTS

# LIST OF FIGURES

FIGURE

# LIST OF TABLES

TABLE

# ABSTRACT

In this thesis, we study three combinatorial problems. In the first half, we focus on a deterministic combinatorial optimization problem called the maximum-entropy sampling problem (MESP). The MESP seeks to find the maximum log-determinant principal submatrix of a given positive-semidefinite matrix $C$. For cases where $C$ (or its inverse) is tridiagonal, we develop an efficient dynamic-programming algorithm and extend this approach to matrices where the graph support of $C$ is a spider graph with a bounded number of legs. We also introduce the concept of using a tridiagonal mask $M$ to obtain a fast upper-bounding method for the MESP. To complement this, we provide an R package tailored for handling multivariate precipitation chemistry data. This package facilitates temporal modeling of data from the National Atmospheric Deposition Program / National Trends Network, yielding covariance matrices essential for the evaluation of MESP algorithms.

The second half of the thesis addresses stochastic combinatorial optimization problems, where uncertainty is introduced to some input parameters. We present new approximation algorithms for the adaptive submodular cover and stochastic minimum query problems. We address the problem of minimizing the cost to cover adaptive-submodular functions and present a $4(1 + \ln Q)$-approximation algorithm, where $Q$ represents the goal value of the covering functions. We also show a guarantee for a more general objective, the $p-$th moment of the cost of our policy. For the $p-$th moment, we show a $(p+1)^{(p+1)}(1+\ln(Q))^p$-approximation.

In the stochastic minimum query problem, we seek to find a value within $\delta$ of the minimum (or maximum) over $n$ independent random variables. We call this value the $\delta-$minimum, and we are also interested in a variant of the problem where we seek to identify the $\delta-$minimizer, which is the interval that contains a $\delta-$minimum. For each random variable we query, we incur some cost, and the goal is to minimize the expected query costs while identifying the $\delta-$minimum (or $\delta-$minimizer). For uniform query costs, we develop a 4-approximation algorithm applicable to both $\delta$-minimum and $\delta$-minimizer variants. For non-uniform costs, we achieve a 5.83-approximation and 7.47-approximation for the problem of finding the $\delta$-minimum and $\delta$-minimizer, respectively. Our algorithms, based on non-adaptive querying, effectively upper bound the adaptivity gap.

# CHAPTER 1

# Introduction

> The essence of combinatorial optimization is choosing wisely from a vast but discrete set of possibilities.
>
> *László Lovász*

The overarching theme of this thesis is combinatorial optimization (CO) [Coo98]. As the name suggests, this is a class of optimization problems defined over combinatorial objects (i.e. finite and countable objects). Some well-studied combinatorial optimization problems are scheduling [RV09], knapsack [Mar90], and graph-based problems like traveling salesman problem [App06, TV22], network flow, and shortest path [AMO93]. In addition to combinatorial optimization problems, this thesis also studies *stochastic* variants where certain input parameters are uncertain. The first half of this thesis is based on our work on a deterministic problem called the *maximum-entropy sampling problem* [FL22]. The second half of this thesis presents our work in approximation algorithms [WS11] for stochastic combinatorial optimization problems.

**Maximum-entropy sampling problem (MESP).** This is a problem that arises in the evaluation, design, and redesign of environmental monitoring networks [ZSL00, BLZ94, GLSZ93, LZ06]. In particular, it models the problem of evaluating which sensors are the "most informative" in a given sensor network. A widely used sensor network is the National Trends Network maintained by the National Atmospheric Deposition Program (see Figure 1.1). This sensor network has 300+ sensors and tracks chemistry deposits in precipitation across the United States.

Assuming that the data set is Gaussian, this application was formulated as the following submodular optimization problem in [SW87a]. Given an $n$-by-$n$ positive-definite matrix $C$ and a size $s < n$, the MESP seeks to find the maximum subdeterminant of size $s$-by-$s$. Formally, $N = \{1, 2, ..., n\}$, $s < n$ and $n$-by-$n$ matrix $C \succeq 0$,

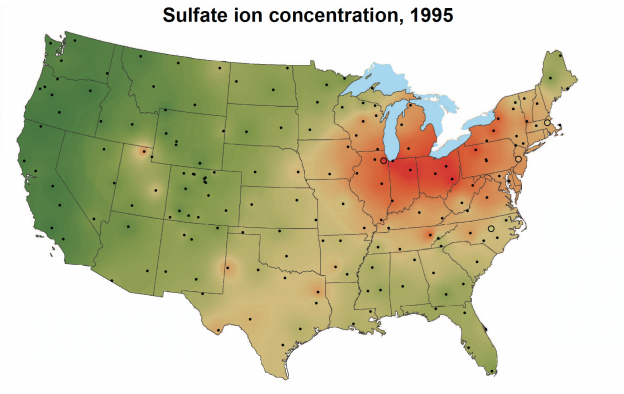



Figure 1.1: National Atmospheric Deposition Program/ National Trends Network

$$z := \max \ \{\log(\det C[S,S]) \ : \ |S| = s, \ S \subseteq N\}, \tag{MESP}$$

where $C[S,S]$ is the principal submatrix of $C$ indexed by the elements in $S$.

**Chapter 2 contributions.** The first half of Chapter 2 is on exact solutions for sparse variants of the MESP. This direction was motivated by the NP-Hard reduction of the MESP from the independent set problem. In the reduction [KLQ95], the authors construct an instance of the MESP by using a graph $G$ from an instance of the independent set, and let $C = A_G + 3nI$ where $A_G$ is the adjacency matrix of $G$. The construction adds sufficiently high values to the diagonal of $C$ to ensure it is positive definite. We then aimed to investigate whether exact solutions could be found for sparse instances of the MESP that correspond to sparse graph classes where finding an independent set is known to be in $P$.

$$C = \begin{pmatrix} \times & \times & 0 & 0 & 0 & 0 & 0 \\ \times & \times & \times & 0 & 0 & 0 & 0 \\ 0 & \times & \times & \times & 0 & 0 & 0 \\ 0 & 0 & \times & \times & \times & 0 & 0 \\ 0 & 0 & 0 & \times & \times & \times & 0 \\ 0 & 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & 0 & 0 & \times & \times \end{pmatrix}$$

Figure 1.2: Tridiagonal

$$C = \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & 0 & 0 & 0 & 0 \\ \times & 0 & \times & 0 & 0 & 0 \\ \times & 0 & 0 & \times & 0 & 0 \\ \times & 0 & 0 & 0 & \times & 0 \\ \times & 0 & 0 & 0 & 0 & \times \end{pmatrix}$$

Figure 1.3: Arrowhead

The graph classes we considered are:

1. $G$ is a path which corresponds to $C$ being tridiagonal,
2. $G$ is a star which corresponds to $C$ being an arrowhead matrix,
3. $G$ is a spider, where the number of legs is bounded (i.e. the degree of the center vertex

is bounded). In this case, $C$ is tridiagonal but also has some non-zero entries in the first row and column.

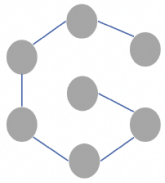


Figure 1.4: Path

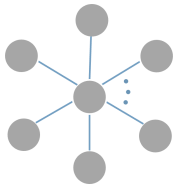


Figure 1.5: Star

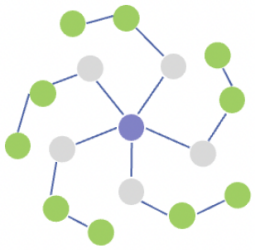


Figure 1.6: Spider

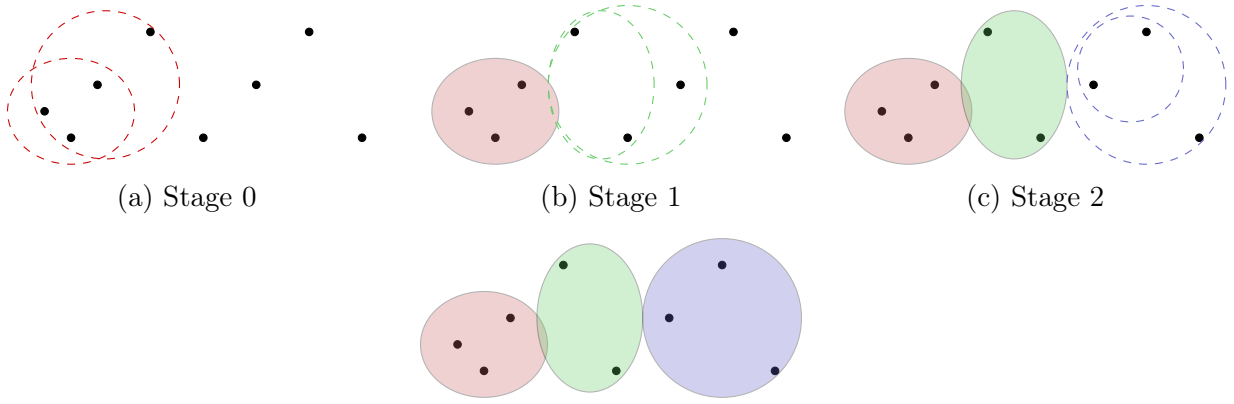
In Chapter 2 we develop an algorithm that solves the MESP in polynomial time, if the input matrix $C$ or its inverse $C^{-1}$ is tridiagonal. We also show the same when $G(C)$ or $G(C^{-1})$ has the sparsity structure of a spider with a bounded number of legs, where $G(C)$ is the graph formed by the non-zero entries of $C$.

In Chapter 2 we also show that under certain conditions a greedy algorithm is optimal for when $C$ is an arrowhead (see Figure 1.3). However, the MESP has been resolved to be NP-Hard for this class of matrices [Ohs24]. This hardness result also rules out polynomial-time algorithms for more general sparsity structures like when $G(C)$ is a tree.

We also use the algorithm developed for tridiagonal matrices to upper-bound more general instances of $C$. We apply a tridiagonal mask $M$ to $C$ using an element-wise product and make a detailed analysis of how changing key components in the mask leads to tighter bounds. There are multiple ways to choose a tridiagonal mask, in Chapter 2 we run a series of experiments to show how a local search on a mask can improve a generic tridiagonal mask bound. A masked bound of this type is fast to compute and provides a fast upper-bounding technique that can improve the performance of the branch-and-bound algorithm which solves the MESP exactly for general $C$ [KLQ95].

**Chapter 3 contributions.** Prior to our work, the number of instances generated from real-world data for the MESP was very limited. This motivated the creation of our R package which generates an arbitrary number of instances from a chemistry dataset. In Chapter 3 we systemitize the process of creating valid instances of the MESP from National Atmospheric Deposition Program/National Trends Network data. Since the raw data is not guaranteed to be Gaussian, the R package takes the raw data stored from the NADP's National Trends Network and applies the methods used by [GLSZ93] to deseasonalize and detrend the data, so that the resulting dataset is Gaussian. Access to more instances of the MESP allows for more reliable evaluations of the performance of MESP algorithms.

**Stochastic combinatorial optimization (SCO).** This is a class that generalizes CO to situations where certain input parameters are uncertain and modeled by probability distributions. The setting of any stochastic variant of a combinatorial problem is more general and has a larger solution space than its deterministic counter-part. Since a stochastic problem adds elements of uncertainty, we now have the opportunity to make observations based on prior decisions and *adaptively* use observations to make the next best decision (or an approximate one). For example, consider a telecommunications company (A-mobile) that must choose which mobile edge servers to deploy. The goal is for all customers to be covered by a server. At each stage in Figure 1.7, A-mobile makes a choice of which server location to deploy, each deployment has the potential to cover the customers (black dots) within the dashed line but only when the server is deployed can we see the result of choosing a specific deployment. This example motivates a natural question, how do we design a policy that makes use of observations in a meaningful way?



(a) Stage 0 (b) Stage 1 (c) Stage 2

(d) Stage 3

Figure 1.7: Adaptive Set Cover: Demonstrating the coverage of customers.

First, this question highlights that we are no longer concerned with *fixed* subsets as was the case in deterministic CO. In particular, for a groundset of size $n$ , the number of solutions in CO is at most exponential in $n$, whereas the number of adaptive policies in SCO can be doubly exponential in $n$.

Second, the same question highlights that there are two possible types of policies: adaptive and non-adaptive. An adaptive policy uses observations from prior decisions, while a non-adaptive policy is a fixed permutation of decisions. We call the ratio of the best non-adaptive solution to the best adaptive solution the *adaptivity gap* [DGV08]. A small adaptivity gap implies that a well-selected non-adaptive policy can perform nearly as well as an adaptive policy for a given problem. Whenever possible, we prefer a non-adaptive policy because

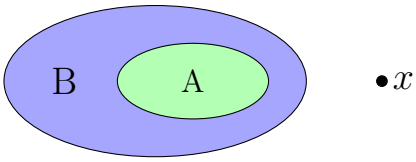
the space of non-adaptive solutions is smaller, which may enable better algorithms. Non-adaptive policies are also faster to implement in practice because they do not need to adapt to incremental observations.

Stochastic combinatorial optimization problems turn out to be significantly harder than their deterministic counterparts. Take for example, the problem of finding a maximum weight matching on a bipartite graph. This is a known problem with many polynomial-time algorithms for stochastic bipartite matching [EK72, HK73]. Despite a long line of work we still only have a constant factor approximation [BGL+12, ASW16]. Nevertheless, several papers have designed algorithms for stochastic variants of classic CO problems such as knapsack, orienteering, and set cover [DGV08, BGL+12, GKNR15, INvdZ16, JLLS20, HKP21, HLS24]. For an SCO minimization problem, an $\alpha$-approximation finds a policy with expected value or cost of at most $\alpha$ times the expected value or cost of the optimal policy. A similar definition follows for maximization problems in SCO.

This thesis studies two stochastic combinatorial optimization problems: the stochastic minimum query and the adaptive submodular cover. The stochastic minimum query is concerned with identifying a minimum/maximum value from a set of bounded random variables. The adaptive submodular cover is a generalization of the stochastic set cover problem.

**Chapter 4 contributions.** In Chapter 4 we study the stochastic minimum query problem (SMQ). Given a set of bounded random variables we seek to find the minimum value over all intervals within a given precision $\delta$, we call this the $\delta$-minimum. Probing an interval incurs a query cost. Our goal is to find the $\delta-$minimum while minimizing the query cost. All the algorithms we consider for this problem are non-adaptive. A non-adaptive policy can utilize a stopping rule to truncate the permutation of random variables, and a stopping rule can use observations made from prior decisions. For example, in the stochastic set cover problem the stopping rule is whether all elements are covered. This stopping rule makes use of which elements were covered by the chosen sets up to a certain stage (see Figure 1.7). In the SMQ our stopping rule is whether we have observed a value that is within $\delta$ of the minimum lower bound of any unprobed random variable.

We show a 4-approximation algorithm for the SMQ with unit costs and a $(3 + 2\sqrt{2} + \epsilon)-$approximation algorithm for the SMQ with general costs. We also study a variant of the problem where we seek to identify the *interval* that contains a $\delta-$minimum (SMQI) while minimzing the query cost. We show a similar set of results for the SMQI, namely a 4-approximation algorithm for the uniform cost and $(4+2\sqrt{3}+\epsilon)$-approximation algorithm for the general cost case. Given that all the algorithms we consider are non-adaptive, our results also provide an upper bound to the adaptivity gap for the SMQ and SMQI respectively.




For every $A \subseteq B \subseteq N$ and $x \notin B$, $f(B \cup \{x\}) - f(B) \leq f(A \cup \{x\}) - f(A)$

Figure 1.8: Submodularity

We empirically evaluate the performance of our policies for the SMQ on a synthetic dataset. We consider instances of size $n = 5, 10$, and 15. We limit the number of intervals because we use a dynamic programming algorithm of exponential size to find the expected cost of the optimal policy. We randomly choose $n$ intervals of size six on average, and we consider instances where all intervals have a uniform distribution and where all intervals have a normal distribution. We also create instances that have a greater degree of overlap (dense) and instances where the overlap of lower bounds is sparse. A total of 480 instances are run, revealing that the ratio of the expected cost of our policies to that of the optimal policy is close to 1, with a maximum value of 1.21.

**Chapter 5 contributions.** The adaptive submodular cover problem we study in Chapter 5 is a generalization of the stochastic set cover problem [GV06] and the submodular cover problem [Wol82]. In Figure 1.7 we give an example of the stochastic set cover problem, where the uncertainty is around which elements are covered by each set. The submodular cover problem is a generalization of the set cover problem, where we wish to cover a submodular function $f : 2^N \to \mathbb{R}$ by selecting $S \subseteq N$ such that $f(S) = f(N)$. Submodularity is the property of diminishing marginal gains (see Figure 1.8). The objective is to minimize the cost of the selected subset $S$ such that $f(S) = f(N)$.

A special case of the adaptive submodular cover problem is the stochastic submodular cover problem [INvdZ16, AAK19, GGN21, HKP21]. In this setting, the random variables are assumed to be independent. Adaptive submodularity is a more general property that allows for correlated random variables [GK11]. The adaptive submodularity property follows the same intuition of diminishing marginal gain in the same way as regular submodularity, but it also depends on the probability distributions of the given random variables. This setting captures well-studied problems such as influence maximization [KKT15] and optimal decision tree [GG74, HR76, KPB99, AH12, GB09, GNR17]. In Chapter 5, we consider this problem for the natural expectation objective as well as the more general $p-$th moment objective. We show that the natural greedy algorithm has an approximation of $(p+1)^p(1+\ln(Q))^p$.

For $p = 1$, this result is tight up to a small constant factor, where $Q$ is the target value of the covering function. In fact, our result is the first $O(\log(Q))$-approximation for the first-order objective ($p = 1$).

We run a series of experiments to evaluate the performance of our greedy algorithm on instances of the Optimal Decision Tree (ODT) problem, an application of the adaptive submodular cover problem. Using the National Institute of Health WISER dataset (http://wiser.nlm.nih.gov/), we constructed five instances, each of which randomly fills in a positive or negative result for symptoms where it is unknown if the toxin exhibits the symptom. We also develop a new lower bound that allows us to more effectively evaluate the empirical performance of our policy without calculating the cost of the optimal policy. Furthermore, we develop this lower bound such that it can lower bound the cost of the $p-$th moment of the optimal policy. Finally, we show that for the five instances of the WISER dataset, the cost of our greedy policy matches the lower bound. To better evaluate the performance of our greedy policy we also studied five more modified instances and show that the ratio of the cost of our policy to the optimal policy is at most 1.04, 1.09, and 1.18 for $p = 1, 2, 3$, respectively.

# CHAPTER 2

# Tridiagonal Maximum-Entropy Sampling and Tridiagonal Masks

## 2.1 Introduction

Let $C$ be an order-$n$ symmetric positive semidefinite real matrix, and let $s$ be an integer satisfying $1 \leq s \leq n$. In applications, $C$ is the covariance matrix for a multivariate Gaussian random vector $Y_{N_n}$, where $N_n := [1, n] := \{1, 2, \ldots, n\}$ — note that we regard $N_n$ as an ordered set, which will be important later. For nonempty $S \subseteq N_n$, let $C[S, S]$ denote the principal submatrix of $C$ indexed by $S$. We denote $\log\det(\cdot)$ by $\mathrm{ldet}(\cdot)$. Up to constants, $\mathrm{ldet}\, C[S, S]$ is the (differential) entropy associated with the subvector $Y_S$ (see [SW87b], for example). The *maximum-entropy sampling problem*, defined by [SW87b], is

$$z := \max \ \{\mathrm{ldet}\, C[S, S] \ : \ |S| = s, \ S \subseteq N_n\} \tag{MESP}$$

MESP corresponds to choosing a maximum-entropy subvector $Y_S$ from $Y_{N_n}$, subject to $|S| = s$. Later, it will sometimes be convenient to emphasize the dependence of MESP on the data, and in such situations we will write MESP$(C, s)$ and $z(C, s)$.

MESP was introduced in [SW87b], it was established to be NP-Hard in [KLQ95] (via reduction from the NP-Complete problem: does an $n$-vertex graph $G$ have a vertex packing of cardinality $s$), and there has been considerable work on algorithms. Viable approaches aimed at exact solution of moderate-sized instances employ branch-and-bound (see [KLQ95]). For use in branch-and-bound, we have many methods for efficiently calculating good upper bounds; see [KLQ95, AFLW96, AFLW99, HLW01, LW03a, AL04, BL07, Ans18b, Ans20, CFLL21, Nik15, LX24, CFL21] and the survey [Lee12]. Notably, we have developed an `R` package providing an easy means for instantiating MESP from raw environmental-monitoring data (see [ATL20a, ATL20b]).

In what follows, "$\circ$" denotes Hadamard (i.e., elementwise) product of a pair of matrices of

the same dimensions. Given a positive integer $n$, we define a *mask* as any $n \times n$ symmetric positive semidefinite matrix $M$ with all ones on the diagonal. Masks are better known as "correlation matrices". These were introduced for MESP in [AL04], where it was observed that for any $S$ and mask $M$, $\operatorname{ldet} C[S,S] \leq \operatorname{ldet}(C \circ M)[S,S]$, and so any upper bound on MESP for $C \circ M$ is valid for the MESP on $C$. It is a challenge to find a good mask with respect to a particular MESP upper-bounding method — i.e., a mask $M$ that minimizes the upper bound on $z(C \circ M, s)$. This topic was investigated in [AL04] and [BL07]. We define a *combinatorial mask* as any block-diagonal mask $M$ where the diagonal blocks (which may vary in size) are matrices of all ones. At the extremes, we have $M := I_n$ (an identity matrix) and $M =: J_n$ (an all ones matrix). As observed in [AL04], we can view some of [HLW01] and [LW03a] in this way; e.g., the "spectral partition bound" of [HLW01]. But there are many possibilities for masks that are not combinatorial masks. For example, a (tridiagonal) $\frac{1}{2}$-*mask* $M$ has $M_{i,i+1} = M_{i+1,i} := \frac{1}{2}$ for $i = 1, \ldots, n-1$ (and all other off-diagonal entries are 0). One goal of ours is to investigate computational aspects of tridiagonal masks $M$, so as to take advantage of the fact that most bounds can be calculated much faster (than on dense matrices) for tridiagonal matrices (in our case, $M \circ C$).

In Section 2.2, we give an $\mathcal{O}(n^5)$ dynamic-programming algorithm for MESP when $C$ is tridiagonal. This also solves the problem in the same complexity when instead $C^{-1}$ is tridiagonal. Furthermore, this covers the cases where $C$ or $C^{-1}$ are "$k$-tridiagonal" (see [dY15]). This is the first progress on identifying significant polynomially-solvable cases of MESP. We extend our result to the case where the support graph of $C$ is a spider with a constant number of legs, and we indicate how it can be further extended when the number of connected subgraphs of the support graph of $C$ is polynomial in $n$. Finally, we present the results of computational experiments on matrices having support graphs that are spiders, indicating the superiority of a parallel implementation of our dynamic-programming algorithm as compared with branch-and-bound.

In Section 2.3, we characterize a class of positive-semidefinite "arrowhead matrices" (i.e., having the support graph of $C$ being a star), such that a certain natural greedy algorithm optimally solves MESP.

In Section 2.4, we characterize for each $n$, masks that differ from $\frac{1}{2}$-masks in two (symmetric) pairs of off-diagonal positions. These results are useful in identifying good masks for MESP bounds, in the context of local search in the space of masks.

In Section 2.5, we develop a combinatorial local-search algorithm that seeks a good tridiagonal mask $M$ for the so-called 'linx' upper bound (which is one of the best known upper-bounding method for MESP). Some computational experiments validate our approach.

## 2.2 Tridiagonal Covariance Matrices

The inverse covariance matrix, known as the *precision matrix*, captures the conditional covariances between pairs of random variables, conditioning on the remaining $n-2$ random variables. For a spatial process with random variables placed in a line, a reasonable model may have the precision matrix being tridiagonal, when the random variables are numbered in a natural order (along the line); intuitively, such a model would assume that no extra information can be obtained from a non-neighbor of a random variable $Y_i$, over and above the information obtainable from the neighbors of $Y_i$.

To see how the inverse covariance matrix plays a role for MESP, we employ the identity

$$\det C[S,S] = \det C \times \det C^{-1}[N_n \setminus S, N_n \setminus S]$$

(see [HJ85, Section 0.8.4]). With this identity, we have $z(C,s) = \operatorname{ldet} C + z(C^{-1}, n-s)$, and so the MESP for choosing $s$ elements with respect to $C$ is equivalent to the MESP for choosing $n-s$ elements with respect to $C^{-1}$.

The determinant of a tridiagonal matrix can be calculated in linear time, via a simple recursion (see [Dem97]), which we write for the symmetric case as that is our need. Let $T_1 = (a_1)$, and for $r \geq 2$, let

$$T_r := \begin{pmatrix} a_1 & b_1 & & & \\ b_1 & a_2 & b_2 & & \\ & b_2 & \ddots & \ddots & \\ & & \ddots & \ddots & b_{r-1} \\ & & & b_{r-1} & a_r \end{pmatrix}.$$

**Lemma 1.** *Defining* $\det T_0 := 1$, *we have* $\det T_r =: a_r \det T_{r-1} - b_{r-1}^2 \det T_{r-2}$, *for* $r \geq 2$.

**Theorem 2.** *MESP is polynomially solvable when $C$ or $C^{-1}$ is tridiagonal, or when there is a symmetric permutation of $C$ or $C^{-1}$ so that it is tridiagonal.*

*Proof.* Suppose that $C$ is tridiagonal. Let $S$ be an ordered subset of $N_n$. Then we can write $C[S,S]$ uniquely as $C[S,S] = \operatorname{Diag}(C[S_1,S_1], C[S_2,S_2], \ldots, C[S_p,S_p])$, with $p \geq 1$, where each $S_i$ is a *maximal ordered contiguous subset* of $S$, and for all $1 \leq i < j \leq p$, all elements of $S_i$ are less than all elements of $S_j$. We call the $S_i$ the *pieces* of $S$, and in particular $S_p$ is the *last piece*. It is easy to see that

$$\det C[S,S] = \textstyle\prod_{i=1}^{p} \det C[S_i,S_i] = \det C[S_p,S_p] \times \det C[S \setminus S_p, S \setminus S_p] \ .$$

Of course every $S$ has a last piece, and for an optimal $S$ to MESP, if the last piece is $S_p =: [k, \ell]$, then we have the "principle of optimality":

$$\operatorname{ldet} C[S \setminus [k,\ell], S \setminus [k,\ell]] = z(C[N_{k-2}, N_{k-2}], s - (\ell - k + 1)).$$

With this way of thinking, we define

$$f(k,\ell,t) := \max \Big\{ \operatorname{ldet} C[S,S] \ : \\ |S| = t,\ S \subset N_n\ ,\ \text{and the last piece of } S \text{ is } [k,\ell] \Big\},$$

for $1 \le \ell - k + 1 \le t \le s$. Note that the last block of $S$ (in the definition of $f$) has $\ell - k + 1$ elements, which is why $t$ must be at least that large. We have that

$$z(C,s) = \max_{k,\ell} \{ f(k,\ell,s) \ : \ 1 \le k \le \ell \le n,\ \ell - k + 1 \le s \},$$

where we are simply maximizing over the possible (quadratic number of) last pieces.

Our dynamic-programming recursion is

$$f(k,\ell,t) = \operatorname{ldet} C[[k,\ell],[k,\ell]] + \max_{i,j} \{ f(i,j,t-(\ell-k+1)) \ : \\ 1 \le i \ \le j \le k-2,\ j - i + 1 \le t - (\ell - k + 1) \},$$

The idea is that if $[k, \ell]$ is the last block of an optimal selection of $t$ elements, then we have to pick $t - (\ell - k + 1)$ more elements, and *element* $k - 1$ *is ruled out* (because $[k, \ell]$ is maximal).

To get the recursion started, we calculate

$$f(k,\ell,\ell-k+1) = \operatorname{ldet} C[[k,\ell],[k,\ell]],$$

for $1 \le k \le \ell \le n$, $\ell - k + 1 \le s$, observing that in such a boundary case, there is only one feasible solution. Already, it appears that the initialization requires $\mathcal{O}(n^5)$ basic arithmetic operations, but in fact we can do this part in $\mathcal{O}(n^2)$ operations, using the tridiagonal-determinant formula (Lemma 1).

Next, we compute, using the recursion, for $t = 1, 2, \ldots, s$, $f(k, \ell, t)$ for all $1 \le k \le \ell \le n$ such that $\ell - k + 1 < t$. It is not hard to see that this gives an $\mathcal{O}(n^5)$ algorithm for MESP, when $C$ is tridiagonal, and by earlier observations, we also get an $\mathcal{O}(n^5)$ algorithm for MESP, when $C^{-1}$ is tridiagonal. □

For an arbitrary symmetric $C$, with rows and columns indexed from $N_n$, we consider the

*support graph* $G(C)$, with node set $V(G[C]) := N_n$, and edge set $E(G(C)) := \{(i,j) \ : \ i,j \in N_n\,,\ i<j,\ C[i,j] \neq 0\}$. If $C$ is a generic tridiagonal matrix (i.e., $C[i,i+1] \neq 0$ for $i \in N_{n-1}$), then $G(C)$ is the path $P_n := \textcircled{1}\text{–}\textcircled{2}\text{–}\cdots\text{–}\textcircled{n}$. If $C$ is tridiagonal but not generic, then $G(C)$ is a subgraph of $P_n$. Because we can efficiently solve MESP when $C$ is tridiagonal (and also when $C$ is tridiagonal after symmetric permutation), it is natural to consider broader classes of $C$, and natural exploitable structure is encoded in the support graph $G(C)$.

We are interested in "spiders" with $r \geq 1$ legs on an n-vertex set (see [MMW08], for example): for convenience, we let the vertex set be $N_n$, and let vertex 1 be the *body* of the spider; the non-body vertex set $V_i$ of *leg* $i$, is a non-empty contiguously numbered subset of $N_n \setminus \{1\}$, such that distinct $V_i$ do not intersect, and the union of all $V_i$ is $N_n \setminus \{1\}$; we number the legs $i$ in such a way that: (i) the minimum element of $V_1$ is 2, and (ii) the minimum element of $V_{i+1}$ is one plus the maximum element of $V_i$, for $i \in [1, r-1]$. A small example clarifies all of this; see Figures 2.1–2.2. Note that with one leg, the spider is the path $P_n$, and with two legs, the vertices can be re-numbered so that the the spider is the path $P_n$. So, we may as well assume that the spider has $r \geq 3$ legs.

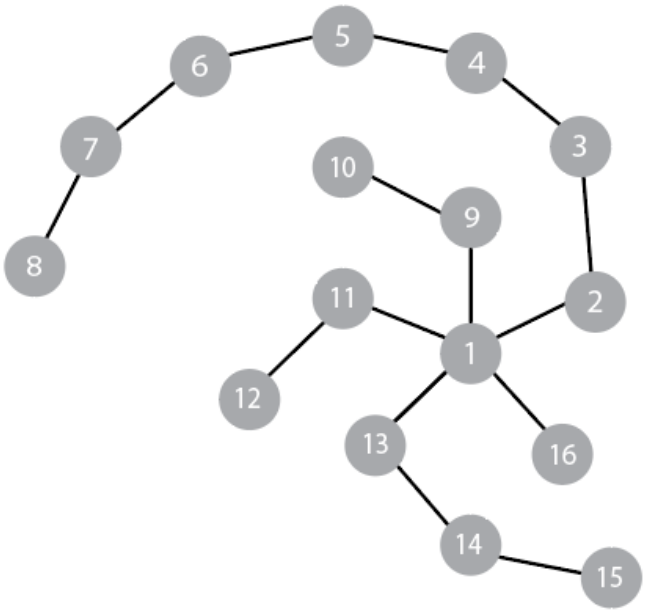



Figure 2.1: Spider with five legs

Consider how a MESP solution $S$ intersects with the vertices of the spider. As before, the solution has pieces. Note how at most one piece contains the body, and every other piece is a contiguous set of vertices of a leg. The number of distinct possible pieces containing the body is $\mathcal{O}(n^r)$. And the number of other pieces is $\mathcal{O}(n^2)$. Overall, we have $\mathcal{O}(n^r)$ pieces. In

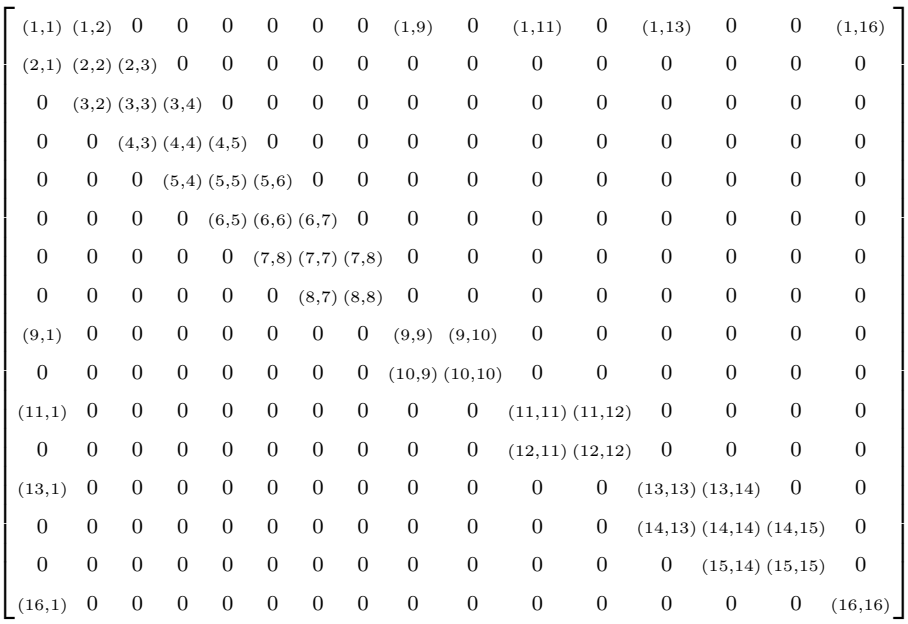



Figure 2.2: Matrix structure corresponding to Figure 2.1

any solution, we can order the pieces by the minimum vertex in each piece. Based on this, we have a well defined last piece. From this, we can devise an efficient dynamic-programming algorithm, when we consider $r$ to be constant.

**Theorem 3.** *MESP is polynomially solvable when $G(C)$ or $G(C^{-1})$ is a spider with a constant number of legs.*

In fact, we can easily organize a dynamic-programming scheme to exploit parallel computation. We can compute $\operatorname{ldet} C[S_1, S_1]$ for all possible pieces containing the body 1. In parallel, we can compute optimal MESP solution values for all possible budgets $t \leq s$ for each leg, keeping track of the minimum vertex used for each such solution. With all of that information, we can then calculate an overall optimal solution.

We conducted some experiments to get some evidence that our dynamic-programming algorithm can be practical, compared to branch-and-bound. For each of $k = 13, 18, 23, 28, 33, 38, 43$, we constructed ten positive-semidefinite matrices $C$, with $G(C)$ being a spider with three legs and $k$ vertices per leg. So we have $n = 3k+1$, and we chose $s \sim \ell n/4$, for $\ell = 1, 2, 3$. For the matrix constructions, we built a `cvx` (see [GB17]) semidefinite-programming model that takes a random choice of positive diagonal elements (constructed using `rand` of `Matlab`). The objective of the semidefinite-program was to maximize the sum of the off-diagonal elements of $C$, subject to $C$ is positive semidefinite, and $G(C)$ being the spider described above.

In Table 2.1, we report on our numerical experiments. We compare our parallel dynamic-programming algorithm, implemented in `Matlab`, with the serial branch-and-bound `Matlab` code of [Ans20], which in turn uses the conic solver `SDPT3` (see [TTT12, TTT99]). For our dynamic-programming algorithm, we use the `Matlab` parallel for-loop instruction `parfor`, for easy parallelization across spider legs. Parallelizing branch-and-bound would be a much more difficult task, and load balancing is highly nontrivial. We note that the running time of `Matlab` is highly variable (particularly for branch-and-bound), even with the same data and deterministic implementations of the algorithms. But because we average over ten experiments for each choice of $n$ and $s$, our results are meaningful. In the individual experiments it is clear that the dynamic-programming algorithm has more consistent time performance, while the time taken by the branch-and-bound fluctuates considerably. For most the experiments with $n \geq 70$, we set an upper limit of 2 hours (7200 seconds), except for the hardest one, the $n = 130$, $s = 97$ experiment, where we increased the time limit to 4 hours. In the table, $*$ indicates that the time limit was reached. Overall the dynamic-programming algorithm performed much better than branch-and-bound, and it scales much better.

| n (k) | s | No. of trials | Avg. time for DP (sec.) | Avg. time for B&B (sec.) |
|---|---|---|---|---|
| 40 (13) | 10 | 10 | $< 1$ | 90 |
| 40 (13) | 20 | 10 | 5 | 21 |
| 40 (13) | 30 | 10 | 7 | 678 |
| 55 (18) | 13 | 10 | 1 | 5642 |
| 55 (18) | 27 | 10 | 21 | 173 |
| 55 (18) | 41 | 10 | 33 | 8150 |
| 70 (23) | 17 | 10 | 4 | 7200* |
| 70 (23) | 35 | 10 | 86 | 241 |
| 70 (23) | 52 | 10 | 146 | 7200* |
| 85 (28) | 21 | 10 | 12 | 7200* |
| 85 (28) | 42 | 10 | 260 | 347 |
| 85 (28) | 63 | 10 | 519 | 7200* |
| 100 (33) | 25 | 10 | 32 | 7200* |
| 100 (33) | 50 | 10 | 858 | 7200* |
| 100 (33) | 75 | 10 | 1364 | 7200* |
| 115 (38) | 28 | 10 | 68 | 7200* |
| 115 (38) | 57 | 10 | 1654 | 7200* |
| 115 (38) | 86 | 10 | 3250 | 7200* |
| 130 (43) | 32 | 10 | 163 | 7200* |
| 130 (43) | 65 | 10 | 4086 | 7200* |
| 130 (43) | 97 | 10 | 7093 | 14400* |

Table 2.1: DP vs B&B on spiders

Finally, we observe that for any class of $n$-vertex graphs for which the number of *connected subgraphs* is bounded above by a polynomial in $n$, we can use the same ideas as above to build an efficient dynamic-programming algorithm, assuming that we can enumerate the connected subgraphs in polynomial time. We do note that the class of $n$-vertex "stars" (i.e., spiders with $n-1$ single-edge legs) has $2^{n-1}$ subtrees containing the body, and so we do not in this way get an efficient algorithm for MESP when $G(C)$ is a star. But see the next section for an efficient algorithm for a subclass of the positive-semidefinite $C$ for which $G(C)$ is a star.

We could perhaps hope that what is generally needed for a tree (to lead to an efficient dynamic-programming algorithm of this type) is a degree bound. But even for a degree bound of three, we get bad behavior: for binary trees, the number of subtrees can be exponential in the number of vertices.[1]

## 2.3 Stars

In this section, we present an algorithm that computes the exact optimum of MESP, when $C$ is an arrowhead matrix (i.e., when the support graph $G(C)$ is a star with center 1) under an easily-checkable sufficient condition.

We define the *arrowhead* matrix

$$A(\alpha_1, \alpha, D) := \begin{pmatrix} \alpha_1 & \alpha^\top \\ \alpha & D \end{pmatrix},$$

with $\alpha_1 \in \mathbb{R}$, $\alpha := (\alpha_2, \ldots, \alpha_n) \in \mathbb{R}^{n-1}$, $d := (d_2, \ldots, d_n) \in \mathbb{R}^{n-1}_+$, and $D := \mathrm{Diag}(d) \in \mathbb{R}^{(n-1)\times(n-1)}$; see [OS90], for example.

**Lemma 4.** *$A(\alpha_1, \alpha, D) \succeq 0$ if and only if $\alpha_1 \geq \sum_{i=2}^n \frac{\alpha_i^2}{d_i}$.*

*Proof.* By symmetric row/column scaling, it is easy to see that $A(\alpha_1, \alpha, D) \succeq 0$ if and only if $A(\alpha_1, \tilde{\alpha}, \alpha_1 I) \succeq 0$, where $\tilde{\alpha}_i := \alpha_i \sqrt{\frac{\alpha_1}{d_i}}$, for $i = 2, \ldots, n$. From [AG01], we have that $A(\alpha_1, \tilde{\alpha}, \alpha_1 I) \succeq 0$ if and only if $\alpha_1 \geq \|\tilde{\alpha}\|$ — this is how the "ice-cream cone" constraint is typically modeled as a semidefinite-programming constraint. Now plugging in the definition of $\tilde{\alpha}$ and simplifying, we obtain our result. □

We consider now MESP$(A(\alpha_1, \alpha, D), s)$, where we assume that $\alpha_1 \geq \sum_{i=2}^n \frac{\alpha_i^2}{d_i}$, so that $A(\alpha_1, \alpha, D) \succeq 0$. We will employ a greedy algorithm to solve

[1] Consider a full binary tree with $\ell$ levels, and hence $2^\ell - 1$ vertices. Such a graph has $2^{\ell-1}$ chains (on $\ell - 1$ edges) from the top to the bottom. Taking the union of any subset of these chains gives a distinct subtree, so we have at least $2^{2^{\ell-1}}$ subtrees, which is exponential in the number of vertices.

MESP$(A(\alpha_1, \alpha, D), s)$, but we do not simply apply such an algorithm directly. Rather, we will branch on element 1, and then apply a greedy algorithm to the two subproblems, selecting the best solution so found.

Our greedy algorithm is a manifestation of a generic greedy maximization algorithm for the more general problem: Given $f : 2^{N_n} \mapsto \mathbb{R}$, find an $s$ element set $S \subset N_n$ with maximum value of $f(S)$.

**Algorithm 1** Generic greedy for $\max\{f(S) \ : \ |S| = s\}$

1: **Input:** $f : 2^{N_n} \mapsto \mathbb{R}$; $1 < s < n$
2: **Output:** $S$
3: $S \leftarrow \emptyset$
4: **while** $|S| < s$ **do**
5: let $j^* \leftarrow \arg\max\{f(S + \{j\}) \ : \ j \in N_n \setminus S\}$
6: let $S \leftarrow S + \{j^*\}$

We let $S_0 = \emptyset, S_1, \ldots, S_s$, be the sequence of iterates $S$ that are created, iteration by iteration, in Algorithm 1. We have the following nice property that we can exploit.

**Lemma 5.** *Suppose that $f : 2^{N_n} \mapsto \mathbb{R}$ satisfies the following stability property: $f(T \cup \{i\}) \leq f(T \cup \{j\})$ for all $T \subset N_n \setminus \{i, j\}$, whenever distinct $i, j$ in $N_n$ satisfy $f(i) \leq f(j)$. Then the iterates of Algorithm 1 are maximizing sets for each cardinality.*

*Proof.* Without loss of generality, we assume that $f(1) \geq f(2) \geq \cdots \geq f(n)$. Under our hypothesis, the sequence of sets $S_k := \{1, 2, \ldots, k\}$, $k = 0, 1, 2, \ldots, s$, is a valid sequence of sets to be produced by the greedy algorithm. By way of a proof by contradiction, suppose that some $S_k$, with $k \geq 2$, is not a maximizing set of its cardinality. Among all maximizing solutions of cardinality $k$, let $S^*$ be a maximizer that has the maximum number of elements in common with $S_k$. We have $f(S^*) > f(S_k)$, and so we can choose $j \in S^* \setminus S_k$ and $i \in S_k \setminus S^*$. Clearly, by how $S_k$ is chosen and by the hypothesis of the lemma, we have $f(j) \leq f(i)$. But now the hypothesis of the lemma give us that $f((S^* \setminus \{j\}) \cup \{i\}) \geq f((S^* \setminus \{j\}) \cup \{j\})$. Therefore $S^* \setminus \{j\} \cup \{i\}$ is also optimal, but it has more elements in common with $S_k$ than $S^*$ does — a contradiction. □

In fact, we are interested in MESP$(C, s)$, whereupon Algorithm 1 particularizes as follows:

**Algorithm 2** Greedy for MESP$(C, s)$

**Input:** $C \in \mathbb{S}_+^n$; $1 < s < n$
**Output:** $S$
$S \leftarrow \emptyset$
**while** $|S| < s$ **do**
    let $j^* \leftarrow \arg\max \{\log\det C[S \cup \{j\}, S \cup \{j\}] \ : \ j \in N_n \setminus S\}$
    let $S \leftarrow S \cup \{j^*\}$

**Remark 1.** *In fact, using the Schur complement of $C[S,S]$ in $C[S+j, S+j]$, we have*

$$\operatorname{ldet} C[S+j, S+j] = \operatorname{ldet} C[S,S] + \log\left(C[j,j] - C[j,S]\left(C[S,S]\right)^{-1} C[S,j]\right).$$

*So, to calculate $j^*$ in Algorithm 2, we simply find the largest diagonal element from the Schur complement of $C[S,S]$ in $C$:*

$$C[N_n \setminus S, N_n \setminus S] - C[N_n \setminus S, S]\left(C[S,S]\right)^{-1} C[S, N_n \setminus S].$$

**Branching on element 1.**

- If 1 is not in some optimal solution of MESP$(A(\alpha_1, \alpha, D), s)$, then

$$z(A(\alpha_1, \alpha, D), s) = z(D, s) = \sum_{i=1}^{s} \log d_{[i]} \, .$$

  That is, if 1 is not in some optimal solution, then such an optimal solution contains $s$ values of $i$ (from $\{2, \ldots, n\}$) with largest $d_i$. Or, to put it another way, an optimal solution is found by applying Algorithm 2 to MESP$(D, s)$.

- Alternatively, if 1 is in some optimal solution of MESP$(A(\alpha_1, \alpha, D), s)$, then

$$z(A(\alpha_1, \alpha, D), s) = \log \alpha_1 + z(D - \tfrac{1}{\alpha_1}\alpha\alpha^\top, s-1).$$

  So, then

$$z(A(\alpha_1, \alpha, D), s) = \max \left\{ \sum_{i=1}^{s} \log d_{[i]}, \ \log \alpha_1 + z(D - \tfrac{1}{\alpha_1}\alpha\alpha^\top, s-1) \right\},$$

  and it remains only to calculate $z(D - \frac{1}{\alpha_1}\alpha\alpha^\top, s-1)$. So in what follows, we employ Algorithm 2, although we will apply it to MESP$(D - \frac{1}{\alpha_1}\alpha\alpha^\top, s-1)$.

Now, we are prepared to describe our sufficient condition for Algorithm 2 to correctly solve MESP$(D - \frac{1}{\alpha_1}\alpha\alpha^\top, s-1))$. What we will show is that as we take successive Schur complements in $D - \frac{1}{\alpha_1}\alpha\alpha^\top$, the ordering of the remaining diagonal elements is not affected.

**Definition 1.** *Let* $r_k := \frac{\alpha_k^2}{d_k}$ *for* $2 \leq k \leq n$*, and we choose a (sorting) bijection* $\pi : \{1, \ldots, n-1\} \to \{2, \ldots, n\}$*, such that* $r_{\pi(1)} \geq r_{\pi(2)} \geq \cdots r_{\pi(n-1)}$ *. Let* $\Phi(s-1) := \{\pi(k) \ : \ 1 \leq k \leq s-1\}$.

So, $\pi$ sorts the ratios $r_i$ , and then $\Phi$ selects the original indices of the $s-1$ largest.

**Theorem 6.** *Let*

$$\hat{\alpha}_1 := \sum_{k \in \Phi(s-1)} r_k + \max_{i,j \in \{2,\ldots,n\} \setminus \Phi(s-1)} \left\{ \frac{\alpha_i^2 - \alpha_j^2}{d_i - d_j} \ : \ d_i > d_j, \ \alpha_i^2 > \alpha_j^2 \right\}.$$

*If* $A(\alpha_1, \alpha, D) \succeq 0$*,* $\alpha_1 \geq \hat{\alpha}_1$ *, then Algorithm 2 produces an optimal solution of MESP*$(D - \frac{1}{\alpha_1}\alpha\alpha^\top, s-1)$.

*Proof.* Let $\bar{D} := D - \frac{1}{\alpha_1}\alpha\alpha^\top$. *If the Schur complement of* $\bar{D}[T, T]$ *in* $\bar{D}$ *has the same ordering of its diagonal elements as* $\bar{D}[N_n \setminus T, N_n \setminus T]$*, for all* $T \subset N_n \setminus \{1\}$, then Algorithm 2 applied

to MESP$(\bar{D}, s-1)$ will choose $s-1$ values of $i$ corresponding to the $s-1$ greatest diagonal elements of $\bar{D}$, and that will be optimal for MESP$(\bar{D}, s-1)$, by Lemma 5. Hence Algorithm 2, generates the optimal solution.

So, it remains to demonstrate that if $\alpha_1 \geq \hat{\alpha}_1$, then the Schur complement of $\bar{D}[T,T]$ in $\bar{D}$ has the same ordering of its diagonal elements as $\bar{D}[N_n \setminus T, N_n \setminus T]$, for all $T \subset N_n \setminus \{1\}$. In what follows, it is notationally easier to work with $D$ rather than $\bar{D}$, so we consider $1 \in T \subset N_n$ rather than $T \subset N_n \setminus \{1\}$.

For $1 \in T \subset N_n$, with $|T| = t$, the Schur complement of $A[T,T]$ in $A := A(\alpha_1, \alpha, D)$ is

$$\begin{aligned}
&A[N_n \setminus T, N_n \setminus T] - A[N_n \setminus T, T]\,(A[T,T])^{-1}\,A[T, N_n \setminus T] \\
&\quad = D[N_n \setminus T, N_n \setminus T] - \left(\alpha[N_n \setminus T], 0_{(n-t)\times t}\right)(A[T,T])^{-1}\left(\alpha[N_n \setminus T], 0_{(n-t)\times t}\right)^{\top} \\
&\quad = D[N_n \setminus T, N_n \setminus T] - (A[T,T])^{-1}_{11}\,\alpha[N_n \setminus T]\alpha[N_n \setminus T]^{\top}.
\end{aligned}$$

Now, for $i \in N_n \setminus T$, the diagonal entry indexed by $i$ of this Schur complement is

$$d_i - \left(\frac{1}{\alpha_1 - \sum_{k \in T\setminus\{1\}} \frac{\alpha_k^2}{d_k}}\right)\alpha_i^2\,,$$

where we have extracted $(A[T,T])^{-1}_{11}$ using the standard block-matrix inverse formula on

$$A[T,T] = \left(\begin{array}{c|c} \alpha_1 & \alpha[T\setminus\{1\}]^{\top} \\ \hline \alpha[T\setminus\{1\}] & D[T\setminus\{1\}, T\setminus\{1\}] \end{array}\right).$$

If $T = \{1\}$, then diagonal element $i$ of the Schur complement is $d_i - \frac{1}{\alpha_1}\alpha_i^2$. So we want to demonstrate that, for all $T \ni 1$ and $i, j \in N_n \setminus T$, if

$$d_i - \frac{1}{\alpha_1}\alpha_i^2 \;\geq\; d_j - \frac{1}{\alpha_1}\alpha_j^2\,, \tag{$*$}$$

then

$$d_i - \frac{\alpha_i^2}{\alpha_1 - \sum_{k \in T\setminus\{1\}} \frac{\alpha_k^2}{d_k}} \;\geq\; d_j - \frac{\alpha_j^2}{\alpha_1 - \sum_{k \in T\setminus\{1\}} \frac{\alpha_k^2}{d_k}}\,. \tag{$**$}$$

Note that ($*$) implies that either $d_i \geq d_j$ and $\alpha_i^2 \leq \alpha_j^2$, in which case ($**$) is trivially true, or $d_i > d_j$ and $\alpha_i^2 > \alpha_j^2$. In this latter case, ($**$) reduces to

$$\alpha_1 \geq \sum_{k \in T\setminus\{1\}} \frac{\alpha_k^2}{d_k} + \frac{\alpha_i^2 - \alpha_j^2}{d_i - d_j}\,.$$

The result now follows by considering the maximum of the right-hand side of this last inequality, over appropriate $T$, $i$ and $j$. □

**Example 1.** *If the sufficient condition of Theorem 6 does not hold, then indeed the greedy algorithm may not find an optimum. For example, Let*

$$A = \begin{pmatrix} 12 & 3.5 & 1.9 & 0.04 & 4.9 \\ 3.5 & 4 & 0 & 0 & 0 \\ 1.9 & 0 & 3 & 0 & 0 \\ 0.04 & 0 & 0 & 2.5 & 0 \\ 4.9 & 0 & 0 & 0 & 5 \end{pmatrix}$$

*and take $s = 3$. It is easy to check that*

$$\alpha_1 = 12 < \hat{\alpha} = 15.0831 = \frac{4.9^2}{5} + \frac{3.5^2}{4} + \frac{1.9^2 - 0.04^2}{3 - 2.5}.$$

*The diagonal elements of the Schur complements of $A[T,T]$ for $T := \{1\}$ and $T := \{1,5\}$ respectively are*

$$\begin{matrix} 2 \\ 3 \\ 4 \\ 5 \end{matrix}\begin{pmatrix} 4 - \frac{3.5^2}{12} \\ 3 - \frac{1.9^2}{12} \\ 2.5 - \frac{0.04^2}{12} \\ 5 - \frac{4.9^2}{12} \end{pmatrix} = \begin{pmatrix} 2.9792 \\ 2.6992 \\ 2.4999 \\ 2.9992 \end{pmatrix} \quad \textit{and} \quad \begin{matrix} 2 \\ 3 \\ 4 \end{matrix}\begin{pmatrix} 4 - \frac{3.5^2}{12 - 4.9^2/5} \\ 3 - \frac{1.9^2}{12 - 4.9^2/5} \\ 2.5 - \frac{0.04^2}{12 - 4.9^2/5} \end{pmatrix} = \begin{pmatrix} 2.2981 \\ 2.4985 \\ 2.4998 \end{pmatrix}.$$

*The ordering implied by the first Schur complement is $5, 2, 3, 4$ and for the second it is $4, 3, 2$. The greedy solution appends 5 and then 4 to $\{1\}$, but the optimal solution turns out to be $S^* = \{1, 2, 3\}$.*

## 2.4 Tridiagonal Masks

It may well be that neither $G(C)$ nor $G(C^{-1})$ is a spider with a constant number of legs. Even then, we can use our dynamic-programming algorithm to get a bound on $z(C, s)$, because $\operatorname{ldet} C[S,S] \leq \operatorname{ldet}(C \circ M)[S,S]$, for all $S \subset N_n$. It is evident that when $M$ is sparse and $C$ is fully dense, $G(C \circ M) = G(M)$, and so when $G(M)$ is a spider with a constant number of legs, we can apply our dynamic-programming algorithm to $C \circ M$ to efficiently get an upper bound on $z(C, s)$. Furthermore, upper bounds for $z(C^{-1}, n-s)$ yield upper bounds for

$z(C,s)$, shifting by $\operatorname{ldet} C$, and upper bounds (under this complementation) are *not* always equivalent. So we can as well profitably apply masking to $C^{-1}$.

For simplicity of exposition and because we have developed the theory in more detail for tridiagonal masks $M$ (i.e., when $G(M)$ is a collection of disjoint paths), we confine our attention to tridiagonal masks $M$. To set some notation, a *tridiagonal mask* $M \in \mathbb{R}^{n\times n}$ has the form

$$M := \begin{pmatrix} 1 & \mu_1 & & & \\ \mu_1 & 1 & \mu_2 & & \\ & \mu_2 & \ddots & \ddots & \\ & & \ddots & \ddots & \mu_{n-1} \\ & & & \mu_{n-1} & 1 \end{pmatrix},$$

with $M_{ij} := 0$ when $|i-j| > 1$.

For $M$ to be a mask, we need it to be positive semidefinite. For example, we can check that if we have all $\mu_i := 1$, then $M$ is not positive semidefinite (for all $n > 2$); but if we have all $|\mu_i| \leq \frac{1}{2}$, then $M$ is diagonally dominant and hence positive semidefinite for all $n$. A $\frac{1}{2}$-*mask* is such an $M$ with all $|\mu_i| = \frac{1}{2}$. In fact, we can do better than this, in the sense that we can increase *some* entries from $\frac{1}{2}$, which seems empirically to be valuable for getting better bounds. For example: for $n = 2$, we can set $\mu_1 = 1$; and for $n = 3$, we could set $\mu_1 = \frac{1}{2}$ and $\mu_2 = \sqrt{\frac{3}{4}}$. For practical purposes, we will limit the number of pairs of symmetric entries that are increased from $\frac{1}{2}$ to two, and we want to see how much we can increase such entries up from $\frac{1}{2}$. Clearly an upper bound on the maximum value of each $\mu_i$ is 1, because $\left(\begin{smallmatrix} [c]1 & \mu_i \\ \mu_i & 1 \end{smallmatrix}\right)$ is always a principal submatrix.

For $1 \leq p < q < n$, and $a, b \in \mathbb{R}$, let $M := M(n,p,a,q,b)$ be an order-$n$ matrix that differs from the order-$n$ $\frac{1}{2}$-mask in that $M_{p+1,p} = M_{p,p+1} = \mu_p := a$ and $M_{q+1,q} = M_{q,q+1} = \mu_q := b$. We will also denote $M(n,p,a) := M(n,p,a,p,a)$ and $M_{1/2}(n) := M(n,p,1/2)\ \forall 1 \leq p \leq n-1$.

The following lemma has a somewhat lengthy technical proof, which can be found in Appendix 1.

**Lemma 7.** *For $1 \leq p < q < n$, and $a, b \in \mathbb{R}$,*

$$\begin{aligned} \det M(n,p,a,q,b) =& \frac{1}{2^n}(n-q+1)\Big((p+1)(q-p+1) - 4a^2p(q-p)\Big) \\ & - \frac{1}{2^n}(n-q)4b^2\left((p+1)(q-p) - 4a^2p(q-p-1)\right). \end{aligned}$$

Our proof is by induction on $n$, and our base cases are $n = 3$ and $n = 4$. For $n = 3$, we have

$$\begin{aligned}\det M(3,p,a,q,b) &= \det M(3,1,a,2,b) = (1-a^2) - b^2 \\ &= \frac{1}{2^3}(3-2+1)\Big((1+1)(2-1+1) - 4a^2(2-1)\Big) \\ &\quad - \frac{1}{2^3}(3-2)4b^2\left((1+1)(2-1) - 4a^2(2-1-1)\right).\end{aligned}$$

For $n = 4$, we have three combinations of $p$ and $q$ to consider.

Case 1: $p = 1$, $q = 2$.

$$\begin{aligned}\det M(4,1,a,2,b) &= \frac{3}{4}(1-a^2) - b^2 \\ &= \frac{1}{2^4}(4-2+1)\Big((1+1)(2-1+1) - 4a^2(2-1)\Big) \\ &\quad - \frac{1}{2^4}(4-2)4b^2\left((1+1)(2-1) - 4a^2(2-1-1)\right).\end{aligned}$$

Case 2: $p = 1$, $q = 3$.

$$\begin{aligned}\det M(4,1,a,3,b) &= (1-a^2)(1-b^2) - \frac{1}{4} \\ &= \frac{1}{2^4}(4-3+1)\Big((1+1)(3-1+1) - 4a^2(3-1)\Big) \\ &\quad - \frac{1}{2^4}(4-3)4b^2\left((1+1)(3-1) - 4a^2(3-1-1)\right).\end{aligned}$$

Case 3: $p = 2$, $q = 3$.

$$\begin{aligned}\det M(4,2,a,3,b) &= \frac{3}{4}(1-b^2) - a^2 \\ &= \frac{1}{2^4}(4-3+1)\Big((2+1)(3-2+1) - 4a^2(2)(3-2)\Big) \\ &\quad - \frac{1}{2^4}(4-3)4b^2\left((2+1)(3-2) - 4a^2(2)(3-2-1)\right).\end{aligned}$$

Next, we suppose that $n > 4$ and that the result holds for matrices of order less than $n$. We apply the recursive determinant formula for tridiagonal matrices (Lemma 1) to $M(n,p,a,q,b)$. Because the recursion has a depth of two, we need to consider six combinations of $p$ and $q$.

Recall our short notation $M(n,p,a) := M(n,p,a,q,1/2) = M(n,p_1,1/2,p,a)$.

$$\begin{aligned}\det M(n,p,a) &= \frac{1}{2^n}(n-p)\Big((p+1)(n-p) - 4a^2p(n-p-1)\Big)\\ &\quad - \frac{1}{2^n}(n-p-1)4(1/2)^2\Big((p+1)(n-p) - 4a^2p(n-p-1)\Big)\\ &= \frac{1}{2^n}\Big((p+1)(n-p+1) - (n-p)4a^2p\Big).\end{aligned}$$

Case 1: $p < q < n-2$.

$$\begin{aligned}\det M(n,p,a,q,b) &= \det M(n-1,p,a,q,b) - \frac{1}{2^2}\det M(n-2,p,a,q,b)\\ &= \frac{1}{2^{n-1}}(n-q)\Big((p+1)(q-p+1) - 4a^2p(q-p)\Big)\\ &\quad - \frac{1}{2^{n-1}}(n-q-1)4b^2\left((p+1)(q-p) - 4a^2p(q-p-1)\right)\\ &\quad - \frac{1}{2^n}(n-q-1)\Big((p+1)(q-p+1) - 4a^2p(q-p)\Big)\\ &\quad + \frac{1}{2^n}(n-q-2)4b^2\left((p+1)(q-p) - 4a^2p(q-p-1)\right)\\ &= \frac{1}{2^n}(n-q+1)\Big((p+1)(q-p+1) - 4a^2p(q-p)\Big)\\ &\quad - \frac{1}{2^n}(n-q)4b^2\Big((p+1)(q-p) - 4a^2p(q-p-1)\Big).\end{aligned}$$

Case 2: $p < n-3$, $q = n-1$.

$$\begin{aligned}\det M(n,p,a,n-1,b) &= \det M(n-1,p,a) - b^2\det M(n-2,p,a)\\ &= \frac{1}{2^{n-1}}\Big((p+1)(n-p) - (n-p-1)4a^2p\Big)\\ &\quad - \frac{b^2}{2^{n-2}}\Big((p+1)(n-p-1) - (n-p-2)4a^2p\Big)\\ &= \frac{1}{2^n}2\Big((p+1)(n-p) - (n-p-1)4a^2p\Big)\\ &\quad - \frac{1}{2^n}4b^2\Big((p+1)(n-p-1) - (n-p-2)4a^2p\Big).\end{aligned}$$

Case 3: $p = n-3$, $q = n-1$.

$$\begin{aligned}&\det M(n,n-3,a,n-1,b) = \det M(n-1,n-3,a) - b^2\det M(n-2,n-3,a)\\ &= \frac{1}{2^{n-1}}\Big((n-2)(3) - (2)4a^2(n-3)\Big) - \frac{b^2}{2^{n-2}}\Big((n-2)(2) - 4a^2(n-3)\Big)\\ &= \frac{1}{2^n}2\Big((n-2)(3) - (2)4a^2(n-3)\Big) - \frac{1}{2^n}4b^2\Big((n-2)(2) - 4a^2(n-3)\Big).\end{aligned}$$

Case 4: $p = n-2$, $q = n-1$.

$$\begin{aligned}\det M(n, n-2, a, n-1, b) &= \det M(n-1, n-2, a) - b^2 \det M(n-2, n-3, 1/2) \\ &= \frac{1}{2^{n-1}}\Big((n-1)(2) - 4a^2(n-2)\Big) - \frac{b^2}{2^{n-2}}\Big((n-2)(2) - (n-3)\Big) \\ &= \frac{1}{2^n}2\Big((n-1)(2) - 4a^2(n-2)\Big) - \frac{1}{2^n}4b^2\Big(n-1\Big).\end{aligned}$$

Case 5: $p < n-3$, $q = n-2$.

$$\begin{aligned}\det M(n, p, a, n-2, b) &= \det M(n-1, p, a, n-2, b) - (1/2)^2 \det M(n-2, p, a) \\ &= \frac{1}{2^{n-1}}3\Big((p+1)(n-p-1) - (n-p-2)4a^2p\Big) \\ &\quad - \frac{b^2}{2^{n-2}}(2)\Big((p+1)(n-p-2) - (n-p-3)4a^2p\Big) \\ &\quad - \frac{1}{2^n}\Big((p+1)(n-p-1) - (n-p-2)4a^2p\Big) \\ &= \frac{1}{2^n}3\Big((p+1)(n-p-1) - (n-p-2)4a^2p\Big) \\ &\quad - \frac{1}{2^n}(2)4b^2\Big((p+1)(n-p-2) - (n-p-3)4a^2p\Big).\end{aligned}$$

Case 6: $p = n-3$, $q = n-2$.

$$\begin{aligned}&\det M(n, n-3, a, n-2, b) \\ &\quad = \det M(n-1, n-3, a, n-2, b) - (1/2)^2 \det M(n-2, n-3, a) \\ &\quad = \frac{1}{2^{n-1}}(2)\Big((n-2)(2) - 4a^2(n-3)\Big) - \frac{1}{2^{n-1}}4b^2\,(n-2) \\ &\qquad - \frac{1}{2^n}\Big((n-2)(2) - 4a^2(n-3)\Big) \\ &\quad = \frac{1}{2^n}(3)\Big((n-2)(2) - 4a^2(n-3)\Big) - \frac{1}{2^n}(2)4b^2\Big((n-2)\Big).\end{aligned}$$

□

From this, we can see how large we can make $a$, when $b$ is held at $1/2$. In particular, we will see that if $b$ is held at $1/2$, we can make $a$ at least $\sqrt{2}/2$ by positioning $a$ at the first off-diagonal pair.

**Proposition 8.** *The maximum value of $a$ such that $M(n, p, a) \succeq 0$ is*

$$a^*(n, p) := \frac{1}{2}\sqrt{\left(1 + \frac{1}{p}\right)\left(1 + \frac{1}{n-p}\right)}.$$

*Furthermore, $a^*(n, p)$ is convex and decreasing in $n$ and $\lim_{n\to\infty} a^*(n, p) = \frac{1}{2}\sqrt{1 + 1/p}$. In particular, $\max_{p\,:\,1\le p<n-1} \lim_{n\to\infty} a^*(n, p) = \lim_{n\to\infty} a^*(n, 1) = \sqrt{2}/2$.*

*Proof.* Note that $M(n,p,\frac{1}{2}) = M_{1/2}(n) \succeq 0$, because it is diagonally dominant. Also, we have that $\det M(n,p,\frac{1}{2}) > 0$ (by Lemma 7). Using Lemma 7, we can see that

$$\det M(n,p,a) = \frac{1}{2^n}\Big((p+1)(n-p+1) - 4a^2p(n-p)\Big)$$

is decreasing in $a$. Now, solving $\det M(n,p,a) = 0$ for $a$, and carrying out some algebraic manipulations, we obtain the formula for $a^*(n,p)$ stated in the result. What we can conclude, at this point, is that $a^*(n,p)$ is an upper bound on the maximum value of $a$ such that $M(n,p,a) \succeq 0$.

To see that this upper bound is in fact the true maximum, we need to verify that $M(n,p,a^*) \succeq 0$. To do this, we will check that all leading principal submatrices of order $1 \leq k < n$ have positive determinant. This, together with the nonnegativity of the determinant of $M(n,p,a^*)$ (it is in fact 0) is sufficient to establish that $M(n,p,a^*) \succeq 0$ (see, for example, [HJ85, Exercise at the bottom of p. 404]).

Now, for $k \leq p$, the $k$-th leading principal submatrix is an order-$k$ $\frac{1}{2}$-mask, which has positive determinant by Lemma 7. For $p < k < n$, the $k$-th leading principal submatrix is $M(k,p,a^*)$. After some algebraic manipulations, we arrive at

$$\det M(k,p,a^*) = \frac{p+1}{2^k}(k-p)\left(\frac{1}{k-p} - \frac{1}{n-p}\right),$$

which is clearly positive for $p < k < n$.

Thus, we have established that $a^*(n,p)$ is the maximum value of $a$ such that $M(n,p,a) \succeq 0$. The rest of the result is easy to verify. □

Knowing the possible increases of $a$ from $1/2$, we can determine the maximum of $b$ (given $a$) and the general behavior of the maximum as we vary $p$ and $q$.

**Theorem 9.** *For $1 \leq p < q < n$ and $a \in [\frac{1}{2}, a^*(n,p)]$, we have that $M(n,p,a,q,b) \succeq 0$ if and only if $b \in [\frac{1}{2}, b^*(n,p,a,q)]$, where*

$$b^*(n,p,a,q) := \frac{1}{2}\sqrt{\frac{(n-q+1)\Big((p+1)(q-p+1) - 4a^2p(q-p)\Big)}{(n-q)\Big((p+1)(q-p) - 4a^2p(q-p-1)\Big)}}.$$

*Proof.* It is easy to verify, from Lemma 7, that the formula given for $b^*(n,p,a,q)$ is the unique nonnegative solution of $\det M(n,p,a,q,b) = 0$.

We claim that $\det M(n,p,a,q,b)$ is decreasing in $b$ when $a \in [\frac{1}{2}, a^*(n,p)]$ (i.e., the the

range of $a$ that ensures $M(n,p,a) \succeq 0$). It is evident, from Lemma 7, that this is when

$$(p+1)(q-p) - 4a^2p(q-p-1) > 0,$$

or simply when

$$a < \tilde{a}(q,p) := \frac{1}{2}\sqrt{\left(1+\frac{1}{p}\right)\left(1+\frac{1}{(q-1)-p}\right)}.$$

Notice that with $q < n$, we clearly have $\tilde{a}(q,p) > a^*(n,p)$, and therefore $\det M(n,p,a,q,b)$ is decreasing in $b$ when $a \in [\frac{1}{2}, a^*(n,p)]$.

We can conclude that for $a \in [\frac{1}{2}, a^*(n,p)]$, the maximum value of $b$ such that $\det(M(n,p,a,q,b)) \geq 0$ is $b^*(n,p,a,q)$.

Now, it only remains to demonstrate that $M(n,p,a,q,b) \succeq 0$, for all $b \in [\frac{1}{2}, b^*(n,p,a,q)]$. It suffices to demonstrate that all proper leading principal submatrices of $M(n,p,a,q,b^*)$ have a positive determinant. Because $\det M(n,p,a,q,b)$ is decreasing in $b$ (for the relevant $n$, $p$, $a$, $q$), we will then have that all proper leading principal submatrices of $M(n,p,a,q,b)$ have a positive determinant, for all $b \in [\frac{1}{2}, b^*(n,p,a,q)]$. This will then imply that $M(n,p,a,q,b) \succeq 0$, for all $b \in [\frac{1}{2}, b^*(n,p,a,q)]$.

From Proposition 8, we have that principal submatrices that take the form $M_{1/2}(k)$ and $M(k,p,a)$ with $p < k < n$ are positive semidefinite, for $a \in [\frac{1}{2}, a^*]$. So it remains to show that $M(k,p,a,q,b^*)$ has a positive determinant, for $1 \leq p < q < k < n$.

Plugging in the formula for $b^*(n,p,a,q)$ into the formula for $\det M(k,p,a,q,b^*)$, after some simplifications we have

$$\frac{1}{2^k}\underbrace{\left((p+1)(q-p+1) - 4a^2p(q-p)\right)}_{(*)}\underbrace{\left((k-q+1)(n-q) - (n-q+1)(k-q)\right)}_{(**)}.$$

It is easy to check that $(*)$ is positive because $a < \tilde{a}(q,p)$, and $(**)$ is positive because $(k-q+1)/(k-q) > (n-q+1)/(n-q)$, for $1 < q < k < n$. □

With the next two results (proofs deferred to Appendix 1), we see that insofar as maximizing the value of $b$ in $M(n,p,a,q,b)$, if $n, q, a$ are fixed, we should set $p = 1$; and then with $p = 1$, we should set $q = n-1$.

**Theorem 10.** *For each $1 \leq p < q < n$, $a \in (\frac{1}{2}, a^*(n,p)]$, and $n \geq 4$, we have that $b^*(n,1,a,q) \geq b^*(n,p,a,q)$.*

*Proof.* First, we note that it is easy to check that $a^*(n,p)$ is convex and symmetric about $n/2$ in $p$. So $a^*(n,p)$ is maximized on $[1, n-2]$ at $p = 1$. Therefore, for $a \in (\frac{1}{2}, a^*(n,p)]$, we

also have $a \in (\frac{1}{2}, a^*(n,1)]$. Hence, $b^*(n,1,a,q)$ is well defined (for $a \in (\frac{1}{2}, a^*(n,p)]$).

We begin by investigating where the continuous function $\beta(p) := (b^*(n,p,a,q))^2$ is decreasing, for (continuous) $p \in (1, n-2)$.

$$\frac{\partial \beta(p)}{\partial p} = \frac{(4a^2-1)\,(n-q+1)}{4(n-q)} \frac{((2a+1)p+1)\,((2a-1)p-1)}{\left((4a^2-1)\,p(p-q+1)+q\right)^2}.$$

It is clear that all factors in the numerator and denominator are positive except for $(2a-1)p-1$, which we need to analyze. We can easily see that this is negative, precisely when $a < \frac{1}{2}\left(1+\frac{1}{p}\right)$. So, we only need check that $a^*(n,p) < \frac{1}{2}\left(1+\frac{1}{p}\right)$; that is, we need to check that

$$\frac{1}{2}\sqrt{\left(1+\frac{1}{p}\right)\left(1+\frac{1}{n-p}\right)} < \frac{1}{2}\left(1+\frac{1}{p}\right).$$

But this easily reduces to $1+\frac{1}{n-p} < 1+\frac{1}{p}$, which is true precisely when $p < n/2$. In particular, $\beta(p)$ is decreasing on $(1, n/2)$ and increasing on $(n/2, n-2)$. So it is quasiconvex, and has a maximum on $[1, q-1]$ at an endpoint.

Next, we will demonstrate that $\Delta b^*(q) := b^*(n,1,a,q) - b^*(n,q-1,a,q) \geq 0$, which will complete our proof. After some algebraic manipulations, we have that $\Delta b^*(q) \geq 0$ simplifies to

$$\frac{(1-2a^2)q+2a^2}{2((1-2a^2)q+(4a^2-1))} \geq \frac{(1-2a^2)q+2a^2}{q}. \tag{2.1}$$

Claim 1: $(1-2a^2)q+2a^2 \geq 0$. To see this, first note that it is clear when $a \leq \sqrt{2}/2$. If $a > \sqrt{2}/2$, then the claim is equivalent to $q \leq \frac{a^2}{a^2-1/2}$. The left-hand side of this last expression is trivially increasing in $q$ and the right-hand side is decreasing in $a$, so it suffices to check that

$$n-1 \leq \frac{(a^*(n,1))^2}{(a^*(n,1))^2-1/2}.$$

Using Proposition 8, we can check that the right-hand side of this last expression is precisely $n$, and so the claim is verified.

Claim 2: $(1-2a^2)q+(4a^2-1) > 0$. It is clearly true when $1/2 \leq a \leq \sqrt{2}/2$. For $a > \frac{\sqrt{2}}{2}$, the claim is equivalent to $q \leq \frac{4a^2-1}{2a^2-1}$. Similarly to Claim 1, it suffices to check that

$$n < \frac{4(a^*(n,1))^2-1}{2(a^*(n,1))^2-1}.$$

Using Proposition 8, we can check that the right-hand side of this last expression is precisely $n+1$, and so the claim is verified.

Combining Eq. (2.1), and the two claims, it remains to show that $2((1-2a^2)q+(4a^2-1)) \leq$

$q$. But this reduces to showing that $(1-4a^2)(q-2) \leq 0$, which immediately follows from $a \geq 1/2$ and $q \geq 2$.

□

**Theorem 11.** *$b^*(n,1,a,q)$ is maximized over $q$ for $1 < q < n$ at $q^*(n,1,a) = n-1$.*

*Proof.* Let $\beta(q) := (b^*(n,1,a,q))^2$. We first demonstrate that $\beta(q)$ is quasiconvex in $q$. This implies that the maximum is attained a boundary point, that is $q = p+1$ or $q = n-1$. First, we calculate the unique stationary point

$$\bar{q}(a) := \frac{n}{2} + 1 + \frac{1}{2(2a^2-1)}.$$

Next, we demonstrate that $\bar{q} \notin [\frac{n}{2}+1, n-1]$. It is easy to see that $\bar{q}$ is decreasing in $a$. Then, for $a \in [1/2, \sqrt{2}/2]$, $\bar{q}$ is maximized at $a = 1/2$, and we have $\bar{q}(a) \leq \bar{q}(1/2) = n/2$. For $a \in (\sqrt{2}/2, a^*(n,1)]$, $\bar{q}$ is minimized at $a = a^*(n,1)$, and we have $\bar{q}(a) \geq \bar{q}(a^*(n,1)) = n+1/2$. This implies that for all $a \in [1/2, a^*(n,1)]$, we have $\bar{q} \notin [\frac{n}{2}+1, n-1]$.

Next, we will demonstrate that $\frac{\partial \beta}{\partial q} > 0$ for $q \geq \frac{n}{2}+1$. We consider the case of $q = n-1$, the maximum value for $q$. Then we can calculate $\frac{\partial \beta}{\partial q}$ evaluated at $q = n-1$:

$$\frac{((2a^2-1)(n-4)-1)\left((2a^2-1)(n-1)-1\right)}{4\left((2a^2-1)(n-3)-1\right)^2} > 0.$$

Note that is easy to check that the numerator is positive for $n \geq 4$, and $a \in [\frac{1}{2}, a^*(n,1)]$. We do this by observing that the two positive roots of the numerator are $\sqrt{\frac{n}{2(n-1)}}$ and $\sqrt{\frac{n-3}{2(n-4)}}$, both of which are greater than or equal to $a^*(n,1) = \sqrt{\frac{n}{2(n-1)}}$. Then we can plug in any relevant $a$ and observe that both the factors in the numerator are negative. This also implies that if $\bar{q} \in [2, \frac{n}{2}+1]$, then $\beta(q)$ is nondecreasing for the interval $[\bar{q}, n-1]$ and either nonincreasing or nondecreasing for $q \in [2, \bar{q}]$. For $\bar{q} \notin [2, n/2+1]$, $\beta(q)$ is strictly increasing. Because $\beta(q)$ has only one stationary point, at $\bar{q}$, we can conclude that $\beta(q)$ (and hence $b^*(n,1,a,q)$) is quasiconvex in $q$, and therefore the maximum is either at $q = 2$ or $q = n-1$.

It remains to demonstrate that $b^*(n,1,n-1,a) \geq b^*(n,1,2,a)$, which reduces to demonstrating that

$$\frac{a^2(n-3)\left(2a^2(n-1)-n\right)}{2(n-2)\left(2a^2(n-3)-n+2\right)} \geq 0.$$

The positive root of the numerator is $\sqrt{\frac{n}{2(n-1)}}$, which is greater than $a^*(n,1)$, and then it is easy to check then that the numerator is nonpositive for all $a \in [1/2, a^*(n,1)]$. Similarly,

the positive root of the denominator is $\sqrt{\frac{n-2}{2(n-3)}}$, which is greater than $a^*(n,1)$, and then it is easy to check then that the numerator is negative for all $a \in [1/2, a^*(n,1)]$. The result follows. □

Note that by Theorems 10 and 11, $b^*(n,p,a,n-1) \geq b^*(n,p,a,q)$. Symmetrically, corresponding to the last four results, we can establish: (i) a formula for the maximum value of $b$ when $a = 1/2$, (ii) a formula for the maximum value $a^*$ of $a$, (iii) that the $a^*$ is increasing in $q$, and (iv) that $a^*$ is maximized over $p$ at $p^* = 1$.

## 2.5 Local Search on Tridiagonal Masks

Our goal is to do a local search in the space of tridiagonal masks, so as to get a (tridiagonal) mask $M$ that leads to a low value of $f(C \circ M, s)$ for some upper-bounding method $f(\cdot, s)$ applied to MESP. Besides exact calculation by dynamic programming of $z(C \circ M)$, we work with the following additional bounding methods:

- The (masked) *spectral bound* for $z(C \circ M, s)$:

$$\textstyle\sum_{\ell=1}^{s} \log \lambda_\ell(C \circ M),$$

  where $\lambda_\ell(\cdot)$ denotes the $\ell$-th greatest eigenvalue When $M = I$, the masked spectral bound is the *diagonal bound*: the sum of the logs of the $s$ biggest diagonal components of $C$. The spectral bound was first presented in [KLQ95], and its masked versions in [HLW01, AL04, BL07].

- The (masked) *linx bound* for $z(C \circ M, s)$ is the optimal value of the following convex-optimization relaxation:

$$\begin{aligned} \max \ & \tfrac{1}{2}\left(\operatorname{ldet}(\gamma(C \circ M)\operatorname{Diag}(x)(C \circ M) + \operatorname{Diag}(\mathbf{e} - x)) - s \log \gamma\right) \\ \text{s.t.} \ & \mathbf{e}^\top x = s, \ 0 \leq x \leq \mathbf{e}, \end{aligned}$$

  where $\gamma > 0$ is a scaling parameter that must be judiciously selected, and $\mathbf{e}$ is an all-ones vector. The linx bound was first presented in [Ans20], and its optimal scaling was studied in [Ans20, CFLL21].

Note that $z(C,s)$ is permutation invariant. That is, if $\Pi$ is an $n \times n$ permutation matrix, then $z(C,s) = z(\Pi C \Pi^\top, s)$. Moreover the value of most bounds applied to $C$ and $\Pi C \Pi^\top$ are identical. But these observations are *not* at all generally true for $C \circ M$ compared to

$(\Pi C \Pi^\top) \circ M$ (unless $M$ is permutation invariant, for example $M = I_n$ or $M = J_n$), and so we can try to optimize the bound produced by a bounding method, even for a fixed mask $M$, by varying the permutation matrix $\Pi$. As a first phase, we start with $M$ equal to the $\frac{1}{2}$-mask of order $n$, and we do a local search, using minimization of the spectral bound as a criterion, applied to $(\Pi C \Pi^\top) \circ M$. Each permutation matrix $\Pi$ is in one-to-one correspondence with a permutation $\pi$ of $N_n$ (i.e., each of the $n!$ permutations $\pi$ is an ordered set $(\pi_1, \pi_2, \ldots, \pi_n)^\top$ of the $n$ distinct elements of $N_n = \{1, 2, \ldots, n\}$, so that $\Pi(1, 2, \ldots, n)^\top = (\pi_1, \pi_2, \ldots, \pi_n)^\top)$). Our local moves correspond to choosing an $i, j$ with $1 \leq i < j \leq n$, and then replacing $\pi_i, \pi_{i+1}, \ldots, \pi_{j-1}, \pi_j$ with $\pi_j, \pi_{j-1}, \ldots, \pi_{i+1}, \pi_i$. We do a best-improvement local search; upon convergence, we replace $C$ with $\Pi C \Pi^\top$.

Next, we proceed to our second phase. As a general step, we have a *blocked tridiagonal mask* $M := \mathrm{Diag}(M[S_1, S_1], M[S_2, S_2], \ldots, M[S_p, S_p])$, with $p \geq 1$, where the $S_i$ partition $N_n$, each $S_i$ is a nonempty ordered contiguous set, each $M[S_i, S_i]$ is a tridiagonal mask (of order $|S_i|$), and for all $1 \leq i < j \leq p$, all elements of $S_i$ are less than all elements of $S_j$. In one-to-one correspondence with the partitioning is its *signature* $(|S_1|, |S_2|, \ldots, |S_i|)$, a sequence of positive integers summing to $n$. Let $(s_1, s_2, \ldots, s_p)$ be the current signature. We do local search on the sets underlying the signature. We want to choose a rich set of local moves that we can reasonably search over for a best-improving move. The moves that we consider are:

- Merge a pair of adjacent blocks: If $p > 1$, choose some $1 \leq i < p$, and replace $s_i, s_{i+1}$ with $s_i + s_{i+1}$.
- Split a block: If $p < n$, choose some $i$ with $s_i > 1$, choose some $1 \leq t < s_i$, and replace $s_i$ with $t, s_i - t$.
- Interchange: If $p > 1$ and not all $s_i$ are equal, choose some $1 \leq i < j \leq p$, with $s_i \neq s_j$, and swap $s_i$ with $s_j$.

The number of available moves at each step of the local search is $\mathcal{O}(n^2)$, which we can reasonably search over.

Considering Theorem 11 (and its analogue for $a^*$), for the purpose of our local search, we will restrict our attention to $M[S_i, S_i]$ that are of the form $M(|S_i|, 1, a, |S_i| - 1, b)$. So given an $S_i$, the only flexibility that we will take is in choosing $a$ and $b$, only differing from

an order-$|S_i|$ mask in the "end pairs":

$$M[S_i, S_i] := \begin{pmatrix} 1 & a & 0 & & & & & & \\ a & 1 & \frac{1}{2} & & & & & & \\ 0 & \frac{1}{2} & 1 & & & & & & \\ & \ddots & \ddots & \ddots & & & & & \\ & & \frac{1}{2} & 1 & \frac{1}{2} & & & & \\ & & & \ddots & \ddots & \ddots & & & \\ & & & & & 1 & \frac{1}{2} & 0 & \\ & & & & & \frac{1}{2} & 1 & b & \\ & & & & & 0 & b & 1 & \end{pmatrix},$$

For any new block $M[S_i, S_i] = M(|S_i|, 1, a, |S_i| - 1, b)$ created in the local search over signatures, we consider the following choices of $(a, b)$: $(\frac{1}{2}, b)$ with $b$ maximized and $(a, \frac{1}{2})$ with $a$ maximized. We calculate these maximum values using Theorem 9 (and its analogue for $a^*$).

We take these local search moves, and we perform a best-improvement local search on $f(C \circ M, s)$, where $f$ is the spectral bound. Note that calculation of the spectral bound is quite fast and also easily parallelizes across the blocks. Then, we switch to the linx bound (generally a better bound, but slower to calculate — and not easily parallelized), continuing the local search. Finally, we further continue the local search, using our dynamic-programming recursion to exactly calculate $z(C \circ M, s)$ at each step. We note that linx solves are much faster than dynamic-programming solves (nearly twice as fast in our experiments), justifying the linx phase in our search.

We implemented our algorithm in `Matlab`, and we carried out some experiments designed to demonstrate the potential of our approach, compared to the masked linx bound using the $\frac{1}{2}$-mask. Our test instances are described in Subsection 2.5.1. In our experiments presented in this section, we took $s \sim n/2$.[2]

### 2.5.1 Test instances

Referring to the first column of Table 2.2, we explain how we built our 20 test instances:

- $C_{63}$ is a well-known benchmark covariance matrix generated from 63 chemical data sensors (see [KLQ95, Ans20]), for example). To get some idea of how our results can

[2]Results of further experiments, with $s \sim n/4$ and $s \sim 3n/4$, are presented in Figures 2.4a and 2.4b and Tables **??** and 2.4 and 2.5.

| $C$ | $n$ | $s$ | linx($\frac{1}{2}$) | $\lambda(\frac{1}{2})$ | $\lambda(\Pi)$ | $\lambda(\Sigma)$ | linx($\Sigma$) | DP($\Sigma$) |
|---|---|---|---|---|---|---|---|---|
| $C_{63}^{-1}$ | 63 | 32 | 3.2512 | 3.2469 | 2.9521 | 2.7197 | 2.4828 | 2.4828 |
| $C_{63}^{-1} + 0.125I$ | 63 | 32 | 3.1841 | 3.1801 | 2.8821 | 2.6584 | 2.4205 | 2.4205 |
| $C_{63}^{-1} + 0.25I$ | 63 | 32 | 3.1182 | 3.1144 | 2.8302 | 2.5994 | 2.3649 | 2.3649 |
| $C_{63}^{-1} + 0.375I$ | 63 | 32 | 3.0523 | 3.0488 | 2.7585 | 2.5404 | 2.3092 | 2.3092 |
| $C_{63}^{-1} + 0.5I$ | 63 | 32 | 2.9891 | 2.9858 | 2.6992 | 2.4838 | 2.2558 | 2.2558 |
| $C_1$ | 100 | 50 | 5.7979 | 5.9629 | 5.4154 | 5.2334 | 4.8091 | 4.8091 |
| $C_2$ | 100 | 50 | 5.6529 | 5.8427 | 5.1636 | 4.8221 | 4.3299 | 4.3299 |
| $C_3$ | 100 | 50 | 6.0613 | 6.1358 | 5.6738 | 5.3606 | 4.8532 | 4.8532 |
| $C_4$ | 100 | 50 | 5.2033 | 5.4311 | 4.8394 | 4.6112 | 4.2871 | 4.2871 |
| $C_5$ | 100 | 50 | 4.835 | 4.8861 | 4.4351 | 4.1941 | 3.8815 | 3.8815 |
| $C_6$ | 100 | 50 | 0.511 | 1.6225 | 0.54995 | 0.4410 | 0.1263 | 0.1263 |
| $C_7$ | 100 | 50 | 0.3095 | 1.3056 | 0.33 | 0.1901 | 0.1332 | 0.1332 |
| $C_8$ | 100 | 50 | 0.3921 | 1.3292 | 0.4362 | 0.3298 | 0.2312 | 0.2312 |
| $C_9$ | 100 | 50 | 0.3477 | 1.598 | 0.3945 | 0.2713 | 0.1984 | 0.1984 |
| $C_{10}$ | 100 | 50 | 0.2422 | 1.4178 | 0.2591 | 0.1735 | 0.133 | 0.133 |
| $C_{11}$ | 100 | 50 | 0.3715 | 1.5703 | 0.382 | 0.2607 | 0.1818 | 0.1818 |
| $C_{12}$ | 100 | 50 | 0.3114 | 1.3027 | 0.3289 | 0.1818 | 0.1289 | 0.1289 |
| $C_{13}$ | 100 | 50 | 0.3859 | 1.3201 | 0.4287 | 0.30696 | 0.215 | 0.215 |
| $C_{14}$ | 100 | 50 | 0.3447 | 1.5994 | 0.3923 | 0.2725 | 0.2182 | 0.2182 |
| $C_{15}$ | 100 | 50 | 0.2579 | 1.4405 | 0.2839 | 0.2072 | 0.1635 | 0.1635 |
| $C_{16}$ | 100 | 50 | 0.7153 | 10.628 | 0.8203 | 0.8202 | 0.8202 | 0.8201 |
| $C_{17}$ | 100 | 50 | 0.7309 | 10.641 | 0.8124 | 0.8123 | 0.8123 | 0.8122 |
| $C_{18}$ | 100 | 50 | 0.8497 | 10.788 | 0.9727 | 0.9727 | 0.9727 | 0.9724 |
| $C_{19}$ | 100 | 50 | 0.9758 | 10.907 | 1.1094 | 1.1093 | 1.1093 | 1.1090 |
| $C_{20}$ | 100 | 50 | 0.7178 | 10.647 | 0.8106 | 0.8105 | 0.8105 | 0.8103 |

Table 2.2: Entropy gaps: $s \sim n/2$

change when uncorrelated noise is added, we also experimented with adding different positive multiples of the identity matrix. We can see that the behavior is similar, for different multiples, while the gaps are smaller.

- $C_1$ through $C_5$ were generated using the `Matlab sprandsym` function. This function can take eigenvalues as input; for each randomly generated positive-definite matrix we set $\lambda_i := 4^{\frac{n+1-2i}{n-i}}$, for each $i \in N$. This gives a nice concave decreasing sequence of eigenvalues that is preserved under matrix inversion (see [CFLL21]).

- Each of $C_6$ through $C_{10}$ were generated as follows. First, we randomly generated a diagonally dominant tridiagonal matrix $\bar{C}$ with:

$$\bar{C}_{i,i} \sim U[2,5], \quad \text{for } i = 1,\ldots,n;$$
$$\bar{C}_{i,i+1} = \bar{C}_{i+1,i} \sim U[-1,1], \quad \text{for } i = 1,\ldots,n-1,$$

independently generated. Next, we made $m = 100,000$ independent samples from the multivariate normal distribution $N(\mathbf{0}, \bar{C})$. From these $m$ samples, we calculated the sample covariance matrix $C$. In this way, we get $C$ as a dense noisy version of the tridiagonal $\bar{C}$.

- $C_{11}$ through $C_{15}$ are generated in a similar way to $C_6,\ldots,C_{10}$ but with a different $\bar{C}$. In this case tridiagonal $\bar{C}$ is made up $n/2$ blocks of $J_2 = \left(\begin{smallmatrix}1&1\\1&1\end{smallmatrix}\right)$ plus independent random samples from $U[0,0.05]$ added to each diagonal element. Here we have samples from $n/2$ pairs of bivariate normals, with no correlation between pairs and high correlation within a pair. Again, we made $m = 100,000$ independent samples from the multivariate normal distribution $N(\mathbf{0}, \bar{C})$, and from these $m$ samples, we calculated the sample covariance matrix $C$.

- For $C_{16}$ through $C_{20}$, we randomly and independently generated

$$R_i \sim N(0,0.2), \quad \text{for } i = 1,\ldots,n;$$
$$E_i \sim N(0,1), \quad \text{for } i = 0,\ldots,n+1.$$

Then, with $d = 0.2$, let

$$Y_i := R_i + E_i + d \times (E_{i-1} + E_{i+1}), \quad \text{for } i = 1,\ldots,n.$$

In this way, we have significant correlation between "neighbors". We made $m = 100,000$ independent samples from this distribution, we calculated the sample covariance matrix, and we used this as $C$.

### 2.5.2 Results of experiments

In Table 2.3, we summarize some statistics for our test matrices. The ratios $\lambda_1/\lambda_n$ clearly indicate that all of our matrices are not nearly diagonal. The column 'abs sum' tabulates $\Sigma_{i,j}|C[i,j]|$. The column 'abs sum tridiag' tabulates $\Sigma_{i-j\le 1}|C[i,j]|$. Hence, the ratios between these two, the tri-ratio' values, measures how far from tridiagonal these matrices are. We can see that the first five have only a very modest degree of tridiagonality. $C_1$–$C_5$ are not tridiagonal at all. Finally, $C_6$–$C_{20}$ are decidedly not tridiagonal, but not extremely far from being tridiagonal, as desired.

| $C$ | $n$ | $\lambda_n(C)$ | $\lambda_1(C)$ | $\lambda_1/\lambda_n$ | abs sum | abs sum tridiag | tri-ratio |
|---|---|---|---|---|---|---|---|
| $C_{63}^{-1}$ | 63 | 0.65 | 31.50 | 48.42 | 1780.66 | 920.04 | 1.94 |
| $C_{63}^{-1} + 0.125I$ | 63 | 0.78 | 31.63 | 40.78 | 1788.54 | 927.92 | 1.93 |
| $C_{63}^{-1}1 + 0.25I$ | 63 | 0.90 | 31.75 | 35.26 | 1796.41 | 935.79 | 1.92 |
| $C_{63}^{-1} + 0.375I$ | 63 | 1.03 | 31.98 | 31.18 | 1804.29 | 943.67 | 1.91 |
| $C_{63}^{-1} + 0.5I$ | 63 | 1.15 | 32.00 | 27.81 | 1812.16 | 951.54 | 1.90 |
| $C_1$ | 100 | 0.25 | 4.00 | 16.00 | 582.96 | 146.62 | 3.98 |
| $C_2$ | 100 | 0.25 | 4.00 | 16.00 | 620.53 | 143.78 | 4.32 |
| $C_3$ | 100 | 0.25 | 4.00 | 16.00 | 543.63 | 146.14 | 3.72 |
| $C_4$ | 100 | 0.25 | 4.00 | 16.00 | 580.94 | 144.61 | 4.02 |
| $C_5$ | 100 | 0.25 | 4.00 | 16.00 | 574.24 | 143.15 | 4.01 |
| $C_6$ | 100 | 1.16 | 5.74 | 4.97 | 546 | 459.67 | 1.19 |
| $C_7$ | 100 | 1.41 | 5.74 | 4.07 | 520.34 | 435.59 | 1.19 |
| $C_8$ | 100 | 1.38 | 5.91 | 4.27 | 544.32 | 459.57 | 1.18 |
| $C_9$ | 100 | 1.17 | 5.78 | 4.94 | 537.27 | 451.79 | 1.19 |
| $C_{10}$ | 100 | 1.58 | 5.47 | 3.45 | 534.84 | 451.56 | 1.18 |
| $C_{11}$ | 100 | 1.11 | 5.88 | 5.31 | 524.93 | 442.32 | 1.19 |
| $C_{12}$ | 100 | 1.42 | 5.83 | 4.09 | 518.17 | 435.43 | 1.19 |
| $C_{13}$ | 100 | 1.38 | 5.90 | 4.29 | 546.19 | 459.26 | 1.19 |
| $C_{14}$ | 100 | 1.16 | 5.74 | 4.93 | 535.49 | 452.04 | 1.18 |
| $C_{15}$ | 100 | 1.59 | 5.45 | 3.43 | 536.84 | 450.93 | 1.19 |
| $C_{16}$ | 100 | 0.49 | 2.51 | 5.12 | 274.69 | 212.78 | 1.29 |
| $C_{17}$ | 100 | 0.49 | 2.53 | 5.14 | 276.40 | 213.01 | 1.30 |
| $C_{18}$ | 100 | 0.49 | 2.52 | 5.13 | 275.30 | 213.04 | 1.29 |
| $C_{19}$ | 100 | 0.49 | 2.52 | 5.11 | 275.19 | 212.97 | 1.29 |
| $C_{20}$ | 100 | 0.49 | 2.51 | 5.09 | 275.68 | 212.94 | 1.29 |

Table 2.3: Test-matrix statistics

Our results appear in Table 2.2, where we tabulated (difference) gaps to a heuristically-generated lower bound on $z(C, s)$ (see [KLQ95, Section 4]). "$C$" indicates the covariance matrix that we used. "linx($\frac{1}{2}$)" is the gap value of the masked linx bound using the $\frac{1}{2}$-mask. "$\lambda(\frac{1}{2})$" is the spectral-bound gap using the $\frac{1}{2}$-mask. Each remaining column starts from the solution of the previous column. "$\lambda(\Pi)$" is the spectral-bound gap after performing a first-phase (permutation-based local search). "$\lambda(\Sigma)$" is the spectral-bound gap after performing a second phase (signature-based local search). "linx($\Sigma$)" is the linx-bound gap after performing a second phase (signature-based local search). "DP($\Sigma$)" is the DP-bound gap after performing a second phase (signature-based local search). In our experiments, we can see clear and consistent improvements from $\lambda(\frac{1}{2})$ to $\lambda(\Pi)$ to $\lambda(\Sigma)$ to linx($\Sigma$). Furthermore, we can see that linx($\Sigma$) improves on linx($\frac{1}{2}$), except for $C_{16}$ to $C_{20}$. Interestingly, for those we

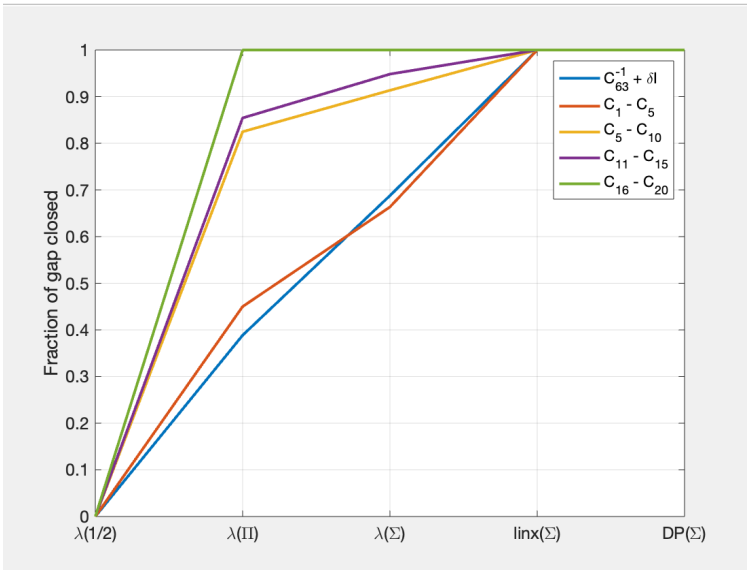
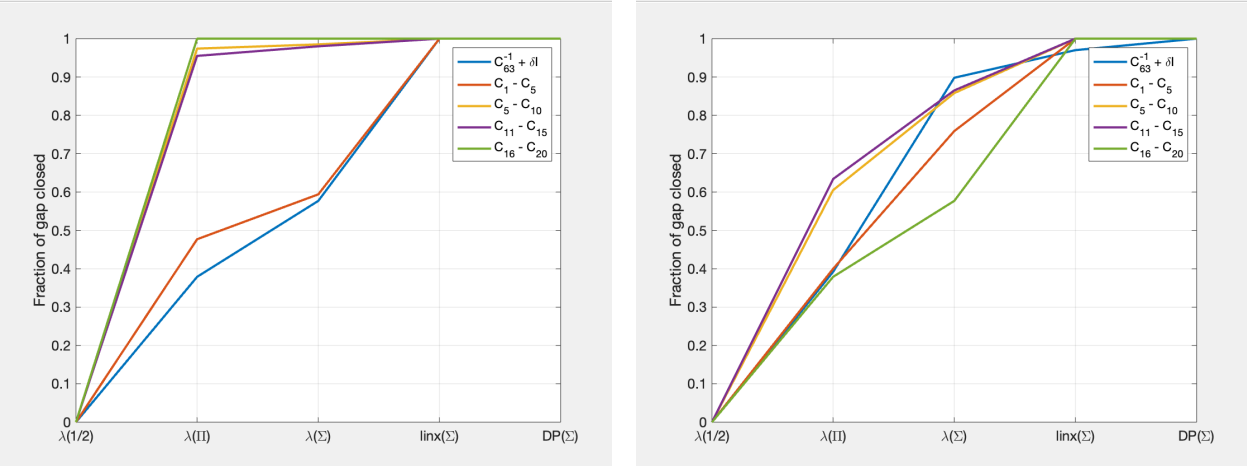



Figure 2.3: Average fraction of gap closed by each masking phase

can see a small improvement in $\mathrm{DP}(\Sigma)$ compared to $\mathrm{linx}(\Sigma)$ (in contrast to the other test problems). Overall, our method gives a good way to get a better mask for the linx bound than the $\frac{1}{2}$-mask. Considering the final column, $\mathrm{DP}(\Sigma)$, we can see that the masked linx bound using the final mask $M$, consistently gives almost exactly $z(C \circ M, s)$. That is, the dynamic program establishes that the masked linx bound with the final mask $M$ nearly gives the optimal value of $z(C \circ M, s)$.

In Figure 2.4, and tables 2.2, 2.4 and 2.5 we present additional experiments to showcase the performance of the masking and permutation steps. The plot of Figure 2.3 shows the average portion of the gap between $\lambda(\frac{1}{2})$ and $\mathrm{DP}(\Sigma)$ that is bridged by each $\Delta \in \{\lambda(\frac{1}{2}), \lambda(\Pi), \lambda(\Sigma), \mathrm{linx}(\Sigma), \mathrm{DP}(\Sigma)\}$. That is, we plot averages of $\frac{\lambda(\frac{1}{2})-\Delta}{\lambda(\frac{1}{2})-\mathrm{DP}(\Sigma)}$; averaged over each of the five types of instances. In this way, we can understand the improvements obtained by successively adding more bounding effort.

Figure 2.4 is of the same type as Figure 2.3, but now with $s \sim n/4$ and $s \sim 3n/4$. A more detailed view of these experiments, following what we presented in Table 2.2, is in Tables 2.4 and 2.5. While the general trends seen for $s \sim n/2$ persist, we do see now several instances where $\mathrm{DP}(\Sigma)$ improves upon $\mathrm{linx}(\Sigma)$.




(a) (b)

Figure 2.4: Average fraction of gap closed by each masking phase

| $C$ | $n$ | $s$ | linx($\frac{1}{2}$) | $\lambda(\frac{1}{2})$ | $\lambda(\Pi)$ | $\lambda(\Sigma)$ | linx($\Sigma$) | DP($\Sigma$) |
|---|---|---|---|---|---|---|---|---|
| $C_{63}^{-1}$ | 63 | 15 | 0.9839 | 0.9880 | 0.8745 | 0.8150 | 0.6880 | 0.6880 |
| $C_{63}^{-1} + 0.125I$ | 63 | 15 | 0.9696 | 0.9739 | 0.8616 | 0.8027 | 0.6772 | 0.6772 |
| $C_{63}^{-1} + 0.25I$ | 63 | 15 | 0.9558 | 0.9601 | 0.8490 | 0.7908 | 0.6667 | 0.2334 |
| $C_{63}^{-1} + 0.375I$ | 63 | 15 | 0.9423 | 0.9467 | 0.8368 | 0.7791 | 0.6565 | 0.2262 |
| $C_{63}^{-1} + 0.5I$ | 63 | 15 | 0.9292 | 0.9336 | 0.8248 | 0.7678 | 0.6466 | 0.2192 |
| $C_1$ | 100 | 25 | 1.0899 | 1.2264 | 0.9922 | 0.9616 | 0.8183 | 0.8183 |
| $C_2$ | 100 | 25 | 1.0936 | 1.1161 | 00.9034 | 0.7943 | 0.5885 | 0.5885 |
| $C_3$ | 100 | 25 | 0.8229 | 0.8743 | 0.7404 | 0.7144 | 0.6201 | 0.6201 |
| $C_4$ | 100 | 25 | 0.9130 | 0.9969 | 0.7969 | 0.7766 | 0.5616 | 0.5616 |
| $C_5$ | 100 | 25 | 0.8480 | 0.8597 | 0.7065 | 0.6521 | 0.4981 | 0.4981 |
| $C_6$ | 100 | 25 | 0.0592 | 0.9574 | 0.0775 | 0.0763 | 0.0656 | 0.0656 |
| $C_7$ | 100 | 25 | 0.0277 | 0.6608 | 0.0193 | 0.0147 | 0.0026 | 0.0026 |
| $C_8$ | 100 | 25 | 0.1028 | 0.9697 | 0.1175 | 0.1168 | 0.0752 | 0.0752 |
| $C_9$ | 100 | 25 | 0.0424 | 0.7216 | 0.0509 | 0.0426 | 0.0426 | 0.0426 |
| $C_{10}$ | 100 | 25 | 0.0447 | 0.6762 | 0.0435 | 0.0236 | 0.0236 | 0.0236 |
| $C_{11}$ | 100 | 25 | 0.0574 | 0.6121 | 0.0505 | 0.0092 | 0.0057 | 0.0057 |
| $C_{12}$ | 100 | 25 | 0.0298 | 0.6763 | 0.0195 | 0.0137 | 0.0026 | 0.0023 |
| $C_{13}$ | 100 | 25 | 0.0853 | 0.9386 | 0.0955 | 0.0930 | 0.0514 | 0.0514 |
| $C_{14}$ | 100 | 25 | 0.0443 | 0.7356 | 0.0463 | 0.0375 | 0.0146 | 0.0139 |
| $C_{15}$ | 100 | 25 | 0.0534 | 0.6781 | 0.0541 | 0.0341 | 0.0341 | 0.0341 |
| $C_{16}$ | 100 | 25 | 0.3970 | 7.2480 | 0.0462 | 0.0461 | 0.0461 | 0.0461 |
| $C_{17}$ | 100 | 25 | 0.3929 | 7.2532 | 0.0499 | 0.0498 | 0.0498 | 0.0498 |
| $C_{18}$ | 100 | 25 | 0.3823 | 7.2506 | 0.0590 | 0.0590 | 0.0588 | 0.0588 |
| $C_{19}$ | 100 | 25 | 0.3769 | 7.2391 | 0.0497 | 0.0497 | 0.0497 | 0.0496 |
| $C_{20}$ | 100 | 25 | 0.3949 | 7.2591 | 0.0439 | 0.0438 | 0.0438 | 0.0438 |

Table 2.4: Entropy gaps: $s \sim n/4$

| $C$ | $n$ | $s$ | $\mathrm{linx}(\frac{1}{2})$ | $\lambda(\frac{1}{2})$ | $\lambda(\Pi)$ | $\lambda(\Sigma)$ | $\mathrm{linx}(\Sigma)$ | $\mathrm{DP}(\Sigma)$ |
|---|---|---|---|---|---|---|---|---|
| $C_{63}^{-1}$ | 63 | 47 | 5.1504 | 5.1546 | 4.7257 | 4.1712 | 4.0911 | 4.0582 |
| $C_{63}^{-1} + 0.125I$ | 63 | 47 | 5.0078 | 5.0120 | 4.5895 | 4.0436 | 3.9649 | 3.9323 |
| $C_{63}^{-1} + 0.25I$ | 63 | 47 | 4.8736 | 4.8777 | 4.4614 | 3.9239 | 3.8467 | 3.8142 |
| $C_{63}^{-1} + 0.375I$ | 63 | 47 | 4.7471 | 4.7512 | 4.3409 | 3.8115 | 3.7358 | 3.7035 |
| $C_{63}^{-1} + 0.5I$ | 63 | 47 | 4.6268 | 4.6308 | 4.2265 | 3.7050 | 3.6307 | 3.6040 |
| $C_1$ | 100 | 75 | 14.1800 | 14.2440 | 12.9740 | 11.7390 | 11.2780 | 11.2780 |
| $C_2$ | 100 | 75 | 12.8690 | 12.9300 | 11.8830 | 11.0540 | 10.4730 | 10.4730 |
| $C_3$ | 100 | 75 | 12.0690 | 12.1220 | 10.8660 | 9.6651 | 8.9006 | 8.9006 |
| $C_4$ | 100 | 75 | 13.5630 | 13.5810 | 12.2950 | 11.1710 | 10.2000 | 10.2000 |
| $C_5$ | 100 | 75 | 12.9400 | 13.0260 | 11.7660 | 10.6340 | 9.6674 | 9.6674 |
| $C_6$ | 100 | 75 | 1.1464 | 2.2101 | 1.2271 | 0.7309 | 0.6028 | 0.6028 |
| $C_7$ | 100 | 75 | 0.8360 | 1.6736 | 0.8169 | 0.4710 | 0.3384 | 0.3384 |
| $C_8$ | 100 | 75 | 0.9222 | 1.9064 | 0.9357 | 0.5663 | 0.4240 | 0.4240 |
| $C_9$ | 100 | 75 | 1.1488 | 2.1226 | 1.1956 | 0.8275 | 0.4835 | 0.4835 |
| $C_{10}$ | 100 | 75 | 1.0013 | 1.8240 | 1.0901 | 0.7925 | 0.4964 | 0.4964 |
| $C_{11}$ | 100 | 75 | 0.9316 | 2.3084 | 0.9966 | 0.6492 | 0.4991 | 0.0584 |
| $C_{12}$ | 100 | 75 | 0.8318 | 1.6619 | 0.8143 | 0.4657 | 0.3406 | 0.3406 |
| $C_{13}$ | 100 | 75 | 0.9199 | 1.9245 | 0.9319 | 0.5658 | 0.4228 | 0.4228 |
| $C_{14}$ | 100 | 75 | 1.1488 | 2.1226 | 1.2121 | 0.8175 | 0.5496 | 0.5496 |
| $C_{15}$ | 100 | 75 | 0.9909 | 1.8258 | 1.0688 | 0.7949 | 0.4852 | 0.4852 |
| $C_{16}$ | 100 | 75 | 6.8324 | 14.6930 | 8.4079 | 8.4078 | 8.4072 | 8.4072 |
| $C_{17}$ | 100 | 75 | 6.8630 | 14.7360 | 8.4535 | 8.4533 | 8.4526 | 8.4526 |
| $C_{18}$ | 100 | 75 | 6.8601 | 14.7380 | 8.4570 | 8.4570 | 8.4567 | 8.4562 |
| $C_{19}$ | 100 | 75 | 6.8393 | 14.7240 | 8.4577 | 8.4573 | 8.4567 | 8.4567 |
| $C_{20}$ | 100 | 75 | 6.8605 | 14.7390 | 8.4501 | 8.4500 | 8.4493 | 8.4493 |

Table 2.5: Entropy gaps: $s \sim 3n/4$

**Acknowledgments.** The results in this chapter are based on the paper [ATL23].

# CHAPTER 3

# Generating Test Instances for Maximum-Entropy Sampling

## 3.1 Introduction

The MESP (maximum-entropy sampling problem) (see [SW87a, SW00, FL00, Lee12]) has been applied to many domains where the objective is to determine a "most informative" subset $Y_S$, of pre-specified size $s = |S| > 0$, from a Gaussian random vecor $Y_N$, $|N| = n > s$. Information is typically measured by (differential) entropy. Generally, we assume that $Y_N$ has a joint Gaussian distribution with mean vector $\mu$ and covariance matrix $C$. Up to constants, the entropy of $Y_S$ is the log of the determinant of the principle submatrix $C[S, S]$. So, the MESP seeks to maximize the (log) determinant of $C[S, S]$, for some $S \subseteq N$ with $|S| = s$.

The MESP is NP-hard (see [KLQ95]), and there has been considerable work on algorithms aimed at exact solutions for problems of moderate size; see [KLQ95, Lee98, AFLW99, LW03b, HLW01, AL04, BL07, Ans18c, Ans18a, CFLL21]. All of this algorithmic work is based on a branch-and-bound framework introduced in [KLQ95], and the bulk of the contributions in these references is on different methods for upper bounding the optimal value. This work has been developed and validated in the context of a very small number of data sets, despite the fact that of course multivariate data is widely available. The reason for this shortcoming is that despite all of the raw multivariate data that is available, it is not a simple matter to turn this data into meaningful covariance matrices for Gaussian random variables.

Our goal with the R package (**MESgenCov** v 0.1.0) that we have developed is to provide such a link — between readily available raw environmental-monitoring data and covariance matrices suitable for the MESP — in the context of environmental monitoring. Our work fits squarely into recent efforts to better exploit massive amounts of available data for mathematical-programming approaches to decision problems. Even if we have reliable raw data, we can only make good decisions if we have a means to prepare that data so that we can populate our mathematical-programming models in such a was as to meet required

assumptions.

We note that another R package of interest is [LZWC19]: "EnviroStat provides functions for spatio-temporal modeling of environmental processes and designing monitoring networks for them based on an approach described in [LZ06]".

In Section refsec:EnvMon, we discuss application of the MESP to environmental monitoring and the NADP/NTN (National Atmospheric Deposition Program / National Trends Network) data environment. In Section 3.3, we describe our methodology. In Section 3.4, we describe our R package (**MESgenCov** v 0.1.0).

## 3.2 Environmental Monitoring and NADP/NTN Data

A key area of application for the MESP has been in environmental monitoring (see [ZSL00, BLZ94, GLSZ93], for example). The idea is that precipitation is collected at many sites, and its chemistry is analyzed. This is costly, and it is a natural question as to whether a subset of the sites might yield data without much loss of information (as measured by entropy). But it is a challenge to process the raw data in such a way that multivariate normality is achieved, because only then are the model of the MESP and its related algorithms applicable.

The NADP maintains the NTN (see [NAD18]); this network measures the chemistry (i.e., ammonium, calcium, chloride, hydrogen, magnesium, nitrate, pH, potassium, sodium, and sulfate) of precipitation at 379 monitoring sites across the US, with some data available as far back as 1978; at present, 255 sites are active.

Our R package is tightly coupled with this precipitation and chemistry data. We are interested in instances of the MESP where $n$ *user-specified* site/chemical pairs comprise $N$. Precipitation data (measured in L) are available on a daily basis, and chemical concentrations (measured in mg/L) are available on a weekly basis. Both datasets are available in our R package and can be loaded respectively as

```
#load package's internal data
 >   data("weeklyConc")
 >   data("preDaily")
```

A full description of the daily and weekly precipitation data appears in Figure 3.1 and Figure 3.2, derived from http://NADP.slh.wisc.edu/, courtesy of the NADP[1]

Small snapshots of the data can easily be viewed. For example, we can output the first 6 rows and first 5 columns of the weekly raw data.

[1]National Atmospheric Deposition Program (NRSP-3). 2019. NADP Program Office, Wisconsin State Laboratory of Hygiene, 465 Henry Mall, Madison, WI 53706.

**NADP/NTN Daily Data**

| Column number | Field | Data type | Description |
|---|---|---|---|
| 1 | SiteID | Char(4) | Site Identifier |
| 2 | StartTime | Char(16) | Period start, reported in Greenwich Mean Time (GMT) YYYY-MM-DD hh:mm format |
| 3 | EndTime | Char(16) | Period end, reported in Greenwich Mean Time (GMT) YYYY-MM-DD hh:mm format |
| 4 | Amount | Integer | Precipitation depth, inches<br>Missing = -9,<br>Trace precipitation amount = -7 |

Figure 3.1: NADP/NTN Daily Data Descriptions

**NADP/NTN Weekly Data**

| Column number | Field | Data type | Description |
|---|---|---|---|
| 1 | SiteID | Char(4) | Site Identifier |
| 2 | DateOn | Char(16) | Date on which the sample bucket was installed on the collector, reported in Greenwich Mean Time (GMT) YYYY-MM-DD hh:mm format |
| 3 | DateOff | Char(16) | Date on which the sample bucket was removed from the collector, reported in Greenwich Mean Time (GMT) YYYY-MM-DD hh:mm format |
| 4 | yrMonth | Integer | Year and Month of sample midpoint, in YYYYMM format |
| 5 | ph | Decimal | Negative log of the hydrogen ion concentration as measured at the CAL, in pH units |
| 6 | Ca | Decimal | Ca concentration, mg/L |
| 7 | Mg | Decimal | Mg concentration, mg/L |
| 8 | K | Decimal | K concentration, mg/L |
| 9 | Na | Decimal | Na concentration, mg/L |
| 10 | NH4 | Decimal | NH4 concentration, mg/L |
| 11 | NO3 | Decimal | NO3 concentration, mg/L |
| 12 | Cl | Decimal | Cl concentration, mg/L |
| 13 | SO4 | Decimal | SO4 concentration, mg/L |
| 14 | Br | Decimal | Br concentration, mg/L |

Figure 3.2: NADP/NTN Weekly Data Descriptions

```
#display part of the weeklyConc data frame
 weeklyConc[1:6,1:5]

  siteID              dateon             dateoff yrmonth    ph
1   AB32 2016-09-13 18:40:00 2016-09-20 15:10:00  201609 -9.00
2   AB32 2016-09-20 15:15:00 2016-09-28 16:00:00  201609 -9.00
3   AB32 2016-09-28 16:00:00 2016-10-05 16:55:00  201610  6.56
4   AB32 2016-10-05 16:55:00 2016-10-11 17:00:00  201610 -9.00
5   AB32 2016-10-11 17:00:00 2016-10-18 20:00:00  201610 -9.00
6   AB32 2016-10-18 20:00:00 2016-10-25 18:00:00  201610  4.73
```

Note that missing concentration values are coded as -9.00.

## 3.3 Our Methodology

### 3.3.1 NADP/NTN data processing

We process the raw NADP/NTN data in a similar way to earlier uses in the context of the MESP in the field of environmental statistics (see [GLSZ93]).

We calculate the level of a chemical's concentration by summing weekly quantities (mg) of the chemical, over a month, and dividing the monthly total by total precipitation (L), over dates in that month, to get monthly values of chemical concentration (mg/L). We use monthly concentrations instead of the available weekly concentrations because there is a large proportion of missing data for individual weeks compared to full months. Furthermore the univariate models were better at predicting average monthly concentrations than they were are at predicting weekly concentrations.

For a given monitoring site, chemical, and month $t = 0, 1, \ldots, T-1$, let

$$
\begin{aligned}
W(t) &:= \text{set of weeks in month } t,\\
D(w) &:= \text{set of days in week } w,\\
c_w &:= \text{recorded chemical concentration (mg/L) for week } w\\
&\qquad (c_w = * \text{ denotes an unrecorded value}),\\
p_d &:= \text{recorded precipitation quantity (L) for day } d,\\
p_w &:= \text{precipitation quantity (L) for week } w;\ p_w = \textstyle\sum_{d \in D(w)} p_d.
\end{aligned}
$$

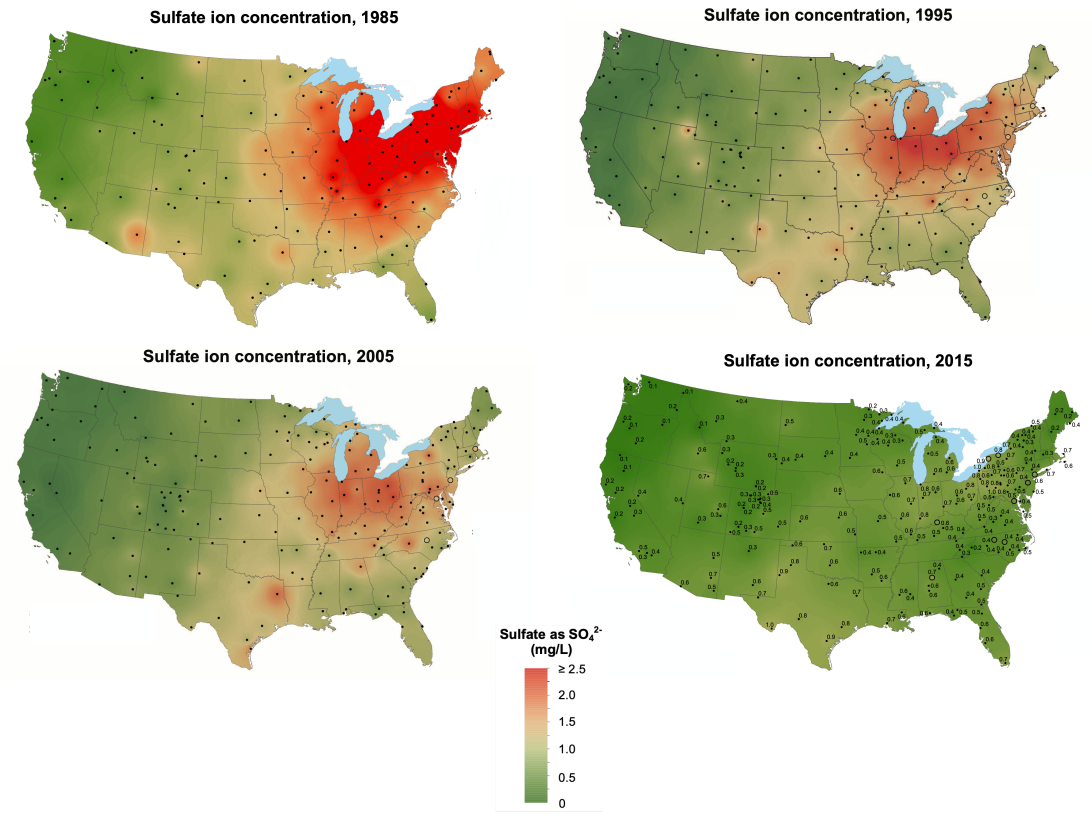



Figure 3.3: Sulfate concentration over time

Then the chemical concentration (mg/L) for month $t$ is calculated as

$$y(t) := \frac{\sum_{w \in W(t): c_w \neq *} p_w c_w}{\sum_{w \in W(t): c_w \neq *} \sum_{d \in w} p_d}.$$

It should be noted that when there is no weekly value available for the chemical quantity, we do not use the precipitation values for any of the days in such a week (so as to not artificially dilute the chemical concentration level for the month).

Next, we fit a temporal model to $\log(y(t))$, which is a rather standard method for handling heavy-tailed distributions. We note that our data has no zero concentrations, so there is no issue of "log(0)".

A quick look at some graphics indicates that there are clear long-term trends; see Figure 3.3[2], from which we can see that sulfate concentrations are generally trending downward over time. Again, looking at some data, we can easily see periodic trends; see Figure 3.4, where we can easily see a yearly periodicity.

[2]Reprinted with the kind permission of the National Atmospheric Deposition Program (NRSP-3). 2019. NADP Program Office, Wisconsin State Laboratory of Hygiene, 465 Henry Mall, Madison, WI 53706.

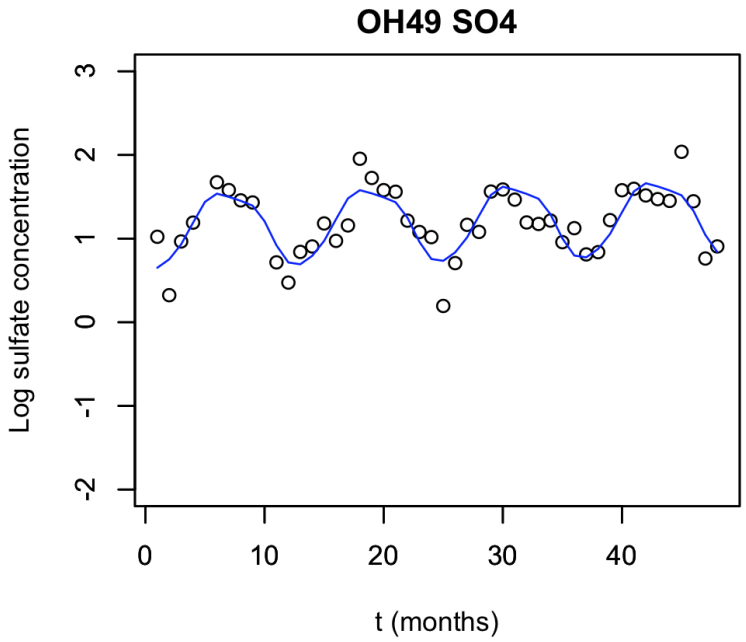



Figure 3.4: Log sulfate concentration over a four-year period at a site

The general model that we provide is

$$\widehat{\log(y(t))} = \sum_{i=0}^{r} \beta_i t^i + \sum_{j=1}^{k} \left[ a_j \cos\left(\frac{2\pi j t}{12}\right) + b_j \sin\left(\frac{2\pi j t}{12}\right) \right], \tag{3.1}$$

with the parameters $\beta_i$, $a_i$, and $b_i$ fit by ordinary linear regression. The user can specify the degree $r$ for the polynomial part of the model which we think of as a truncated Taylor series, aimed at capturing aperiodic trends. Periodic trends are captured via a truncated Fourier series, truncated at level $k$.

We note that [GLSZ93] used the following model to de-seasonalize and de-trend the log-transformed monthly sulfate concentration values:

$$\widehat{\log(y(t))} = \beta_1 + \beta_2 t + a_1 \cos\left(\frac{2\pi t}{12}\right) + b_1 \sin\left(\frac{2\pi t}{12}\right). \tag{3.2}$$

This is just an affine model $\beta_1 + \beta_2 t$ plus a sinusoidal model with monthly periodicity and intercept $a_1$. The simple model (3.2) is (3.1) with $r = 1$ and $k = 1$. We found that (3.2) did well at normalizing the errors for certain sites, but some sites, such as "AL10" and "IN41", (3.2) did not do so well. So rather than fix (3.2) as the model for our R package, we provide the flexibility of (3.1).

In Figure 3.5, we provide an example where (3.1) with $r = 1$ and $k = 3$ is a much better

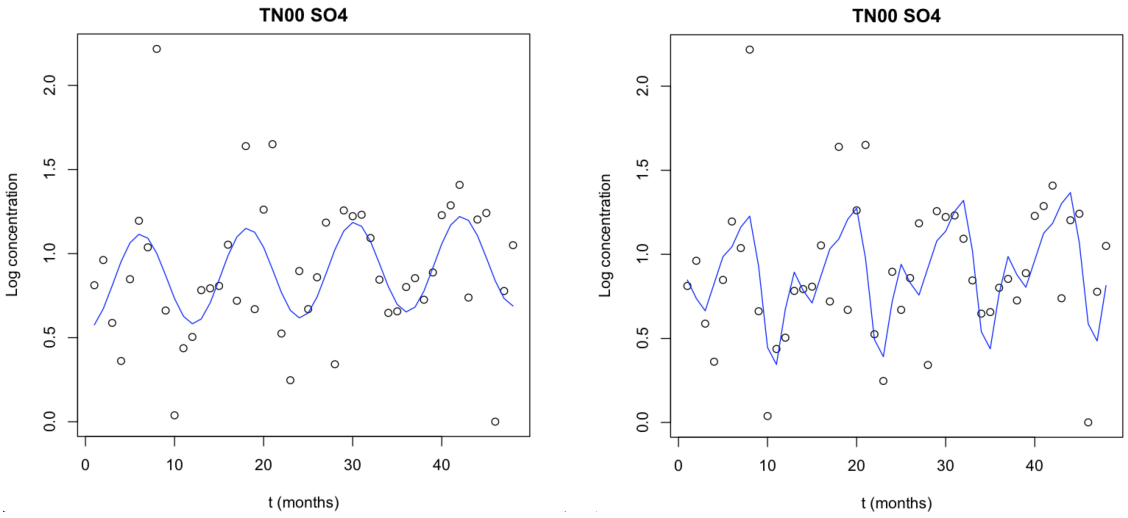



Figure 3.5: Log sulfate concentrations at site "TN00"

fit for the log concentration of sulfate at site "TN00" than (3.2). The plot on the left shows the fit of model (3.2), and the plot on the right shows the fit of our model (3.1).

To produce the covariance matrix, we need error values for each time point. Missing values in the NADP/NTN data set means that each set of error values produced by the model may vary in size. So we have filled in missing values for each site and time point by sampling from a normal distribution with the mean being the predicted value by the univariate model and the standard deviation being the standard error of the univariate model.

## 3.4 MESgenCov

Our R package **MESgenCov** can be obtained from https://github.com/hessakh/MESgenCov and installed using the `devtools` library. If not already installed then the `devtools` library can be installed and loaded as follows:

```
#Install and load devtools
>   install.packages("devtools")
>   library(devtools)
```

Then we can use the `devtools` function `install_github()` to download the **MESgenCov** package from GitHub.

```
#Install and load MESgenCov
>   install_github("hessakh/MESgenCov")
>   library(MESgenCov)
```

**MESgenCov** contains functions in the `S3` class to create a covariance matrix from the desired subset of NADP/NTN data. The function `getCov()` returns a covariance matrix, a list of univariate model summaries, and a table of normality tests produced by the MVN R package (see [KGZ14]). The multivariate analysis is used to asses the validity of the covariance matrix to be used as input for the MESP. The user can make adjustments to the input of `getCov()` so as to obtain a covariance matrix for a multivariate Gaussian vector, which is then valid for use in the MESP.

To avoid sites with a small sample size for the specified time-frame, the function `getSites()` outputs a vector of the sites with the largest sample of data for a given time-frame and measured chemical (see Section 3.4.2). To find sites that are spatially "spread out" but have at least some specified sample size, the function `maxDistSites()` can be used to obtain a list of geographically sparse sites (see Section 3.4.2). Finally, in the case where the residuals from a univariate model do not appear to be normally distributed, the function `lambertWtransform()` allows the user to transform the residuals (from a univariate model) using the R package LambertW: Probabilistic Models to Analyze and Gaussianize Heavy-Tailed, Skewed Data (see [Goe16]). This can be very effective in situations where the distributions seems to have heavy tails and some skewness (see [Goe11] and Section 3.4.3).

### 3.4.1 `getCov`

`getCov()` takes a 14 column data frame as input where each column corresponds to one of the user-specifications shown in Figure 3.6. The 14 specifications in the input allow the user to specify the subset of data to analyze and gives the user options in displaying different parts of the analysis.

#### 3.4.1.1 Input

| **Arguments** | Definition |
|---|---|
| **startdateStr** | Date and time of when to start analyzing the data, in the format = m/d/y H:M |
| **enddateStr** | Date and time of when to stop analyzing the data, in the format = m/d/y H:M |
| **comp** | String of pollutant or acidity level to be analyzed, the pollutants name should be used as it appears in weeklyConc |
| **use36** | TRUE if default 36 sites should be added, FALSE otherwise |
| **siteAdd** | List of strings of siteIDs that should be analyzed |
| **outlierDatesbySite** | List of sites where outliers should be analyzed |
| **siteOutliers** | List of sites where outliers should be removed |
| **removeOutliers** | Specify siteID string for outlier analysis |
| **plotMulti** | TRUE if multivariate analysis plots should be displayed, FALSE otherwise |
| **sitePlot** | Specify list of siteIDs to be plotted |
| **plotAll** | TRUE if plots for all sites should be displayed, FALSE otherwise |
| **writeMat** | TRUE if .mat file of the resulting covariance matrix should be written in the working directory |
| **r** | Integer <=5, see univariate model |
| **k** | Integer <= 5, see univariate model |

Figure 3.6: Input parameters for `getCov()`

A default set of inputs can be found in the stored data frame "defaultInput". Each column of "defaultInput" is an argument in the function `getCov()`. After storing "defaultInput" in a variable in the user's workspace, the input can be changed. For example, below we store the "defaultInput" data frame in a variable "df", and then change the end date:

```
#Load defaultInput data frame and store in df
 >   data("defaultInput")
 >   df <- defaultInput
 >   df

      startdateStr   enddateStr      use36   siteAdd
   1  01/01/83 00:00 12/31/86 00:00  TRUE    NULL
     outlierDatesbySite siteOutliers comp plotMulti sitePlot
   1 NULL               NULL         SO4  FALSE     NULL
     plotAll writeMat r k
     FALSE   FALSE    1 1

#Change the end date to extend the sample of data taken from
 weeklyConc
 df$enddateStr   <- "12/31/88 00:00"
```

#### 3.4.1.2 Output

The function `getCov()` produces a list with the the following elements:

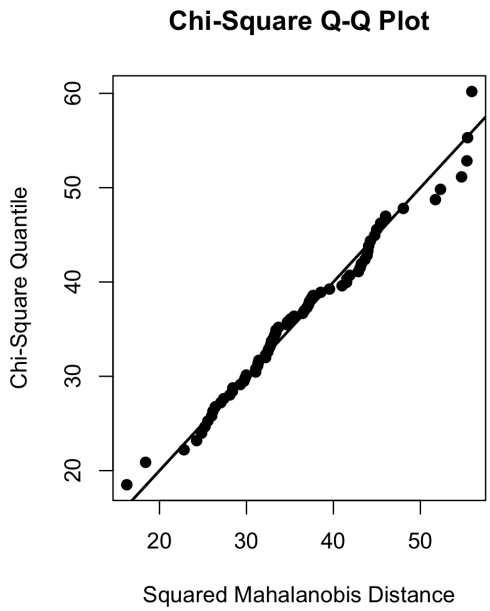
Figure 3.7: `getCov()` function output

| **Output** | Definition |
|---|---|
| **cov** | Covariance matrix produced by univariate model residuals |
| **listMod** | List of univariate model summaries produced by lm() |
| **sites** | List of sites that were analyzed |
| **mvn** | Output of the MVN package |
| **univariateTest** | Univariate test output, also by the MVN package |
| **residualData** | Data frame of residuals produced by the univariate model |
| **residualDataNA** | Data frame of residuals, where missing values are left as NA |
| **rosnerTest** | Output of the Rosner test for outlier analysis produced by the EnvStats package; see [Mil13] |
| **pred** | List of predicted values produced by the univariate model for each site |

Next, we demonstrate how to access these elements and certain plots after running the function `getCov()`.

1. Multivariate and univariate normality

```
#Change part of the input data frame df
>   df$plotMulti <- TRUE
#Change univariate model parameter from 1 to 3
>   df$k <- 3
#Store output in variable g so that the list of outputs given
 by getCov() can be called
>   g <- getCov(df)
```




```
>  g$univariateTest

           Test  Variable Statistic   p value Normality
 1 Shapiro-Wilk  AL10SO4     0.9347     0.001        NO
 2 Shapiro-Wilk  IL11SO4     0.9894    0.8121       YES
 3 Shapiro-Wilk  IL18SO4     0.9909    0.8854       YES
 4 Shapiro-Wilk  IL19SO4     0.9183     2e-04        NO
 5 Shapiro-Wilk  IL35SO4     0.8709    <0.001        NO
```

2. Output all MVN package analysis

The following output is from a call to the MVN package that produces multivariate analysis based on the Mardia method, univariate analysis based on the Shapiro-Wilson method, and a multivariate outlier test that is presented as a plot and not as an output in the user's R console.

```
>  g <- getCov(df)
#Display full output of the MVN package
>  g$mvn

 $multivariateNormality
              Test Statistic p value Result
 1 Mardia Skewness    14.019  0.1721    YES
 2 Mardia Kurtosis   -0.1465  0.8835    YES
 3             MVN      <NA>    <NA>    YES

 $univariateNormality
           Test  Variable Statistic   p value Normality
 1 Shapiro-Wilk  NY52SO4     0.9821    0.3981    YES
 2 Shapiro-Wilk  TN11SO4     0.9830    0.4385    YES
 3 Shapiro-Wilk  IL63SO4     0.9873    0.6858    YES

 $Descriptives
          n     Mean   Std.Dev  Median     Min    Max
 NY52SO4 72  1.1709e-17 0.2812  0.0333 -0.8815 0.6170
 TN11SO4 72 -1.4991e-02 0.4666 -0.0020 -0.9826 1.4243
 IL63SO4 72  4.8127e-18 0.3016 -0.0373 -0.7353 0.7899
               25th      75th       Skew   Kurtosis
 NY52SO4 -0.2211160 0.1840066 -0.3702252 0.08937981
 TN11SO4 -0.3258896 0.2130045  0.4513955 0.29813810
 IL63SO4 -0.1836875 0.1724338  0.3287155 0.06092834
```

The specific call of the MVN package is

```
>  mvn(dfRes[,-1], subset = NULL, mvnTest = "mardia",
 covariance = TRUE, tol = 1e-25, alpha = 0.5, scale = FALSE,
 desc = TRUE, transform = "none",univariateTest = "SW",
 univariatePlot = "none", multivariatePlot ="none",
 multivariateOutlierMethod = "none", bc = FALSE, bcType =
 "rounded", showOutliers = FALSE, showNewData = FALSE).
```

See [KGZ14] for details on the MVN package.

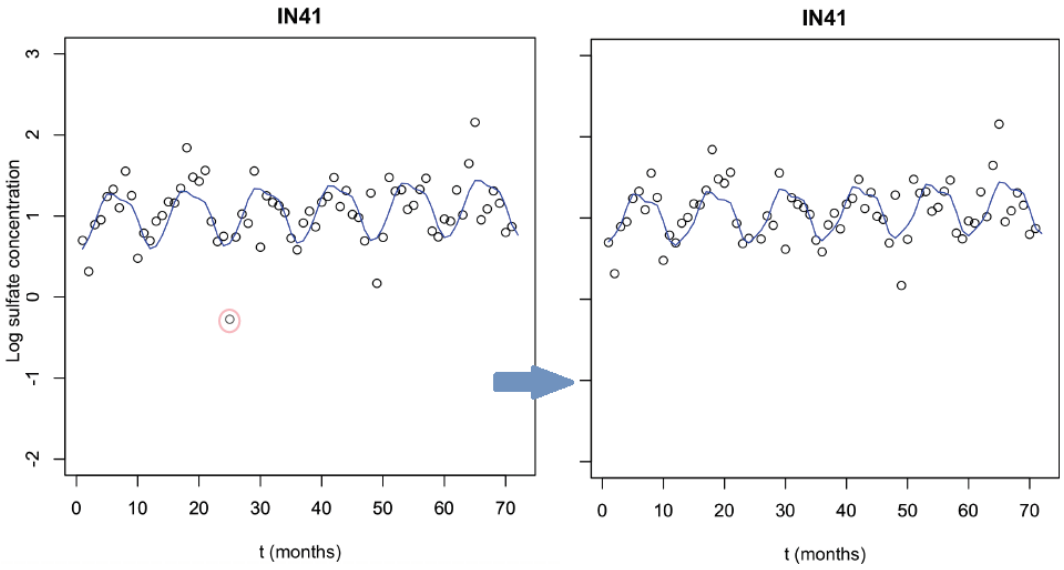
3. Outlier test for specific tests

```
 df$siteOutliers <- list(c("IN41"))
 df$sitePlot <- list(c("IN41"))
 g <- getCov(df)
 i <- match("IN41",g$sites)
 g$rosnerTest[[i]]$all.stats

  i  Mean.i   SD.i   Value Obs.Num  R.i+1 lambda.i+1 Outlier
1 0 -0.0069 0.2814 -0.9359      25 3.3013     3.2680    TRUE
2 1  0.0062 0.2604 -0.7194      30 2.7862     3.2628   FALSE
3 2  0.0165 0.2471  0.7215      66 2.8533     3.2576   FALSE
```

By changing the input, we can remove the outliers detected by the Rosner test. Note that the plots are generated after running `getCov()`. Furthermore, `getCov()` does not need to be stored in a variable to generate the plots.

```
#Remove month 25 from site IN41's pollutant concentration data
>  df$outlierDatesbySite <- c("IN41",25)
>  getCov(df)
```

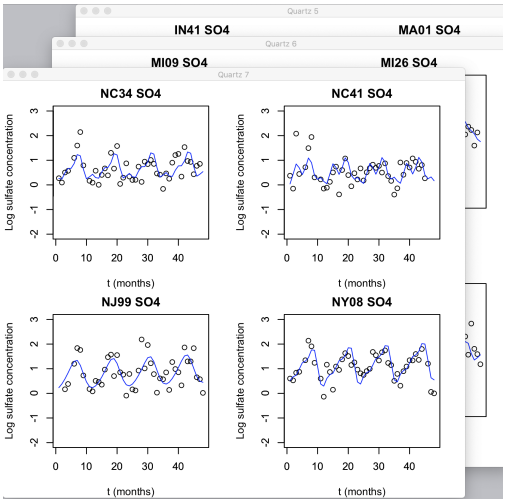



4. Outlier test for all sites

```
#take sites used in analysis in g and run outlier test
 df$siteOutliers <- list(g$sites)
#remove data points identified as outliers from these sites
 df$removeOutliers <- list(g$sites)
 g <- getCov(df)
```

5. Plot all sites

```
>  df$plotAll <- TRUE
>  getCov(df)
```



6. Covariance matrix

```
#Remove default list of sites so that their data is not
 analyzed
>  df$use36 <- FALSE
#Add new site list
>  df$siteAdd <- list(c("NY52", "TN11", "IL63"))
#Remove any set of sites and pollutant combinations
 that had been previously added
>  df$siteOutliers <- NULL
>  df$outlierDatesbySite <- NULL
>  df$removeOutliers <- NULL
>  g <- getCov(df)
#Print covariance matrix
>  round(g$cov,digits = 4)

         NY52SO4 TN11SO4 IL63SO4
 NY52SO4  0.0791  0.0047  0.0009
 TN11SO4  0.0047  0.2177  0.0185
 IL63SO4  0.0009  0.0185  0.0909
```

7. Save covariance matrix as a .mat file (to populate an instance of the MESP, for example).

   This is done by simply setting the input data frame attribute `writeMat` to TRUE. The .mat file will be saved to the user's current working directory as covSites.mat. For processing further with Matlab, use the (Matlab) 'load' command.

```
#Write cov into .mat file in current directory
>  df$writeMat <- TRUE
>  g           <- getCov(df)
```

   In the case that the user has already generated an output by the function `getCov()`, it is possible to also create the .mat file in the following manner.

```
>  library(rmatio)
>  write.mat(g$cov,filename = "covariance1.mat")
```

8. Univariate model summaries

```
>  result <- getCov(df)
#Store site list in sites variable
>  sites  <- result$sites
#Find site OH71 index in the list
>  i = match(c("NY52"),sites)
#Use site index to find model summary for NY52
>  result$listMod[i]


    [[1]]
    Call:
    lm(formula = y1 ~ I(cos(t*(2*pi/s))) + I(sin(t*(2 *
        pi/s))) + I(cos(t*(2*pi/s)*2)) + I(sin(t*
        (2*pi/s)*2)) + I(cos(t*(2*pi/s)*3)) +
        I(sin(t*(2*pi/s)*3)) + I(t), data = df)

    Residuals:
        Min      1Q  Median      3Q     Max
    -0.5236 -0.1677 -0.0189  0.1818  0.9087

    Coefficients:
                        Estimate Std. Error t value Pr(>|t|)
    (Intercept)           1.1110     0.0712   14.40  < 2e-16***
    I(cos(t*(2*pi/s)))   -0.3541     0.0494   -9.75  2.8e-14***
    I(sin(t*(2*pi/s)))   -0.1109     0.0498   -2.55     0.013*
    I(cos(t*(2*pi/s)*2))  0.0500     0.0494   -0.37     0.716
    I(sin(t*(2*pi/s)*2))  0.0797     0.0494    2.13     0.037*
    I(cos(t*(2*pi/s)*3))  0.0391     0.0494   -0.20     0.845
    I(sin(t*(2*pi/s)*3))  0.0536     0.0494   -0.52     0.604
    I(t)                 -0.0010     0.0017   -0.61     0.546
    ---
 Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

    Residual standard error: 0.2961 on 64 degrees of freedom
    Multiple R-squared:  0.6265, Adjusted R-squared:  0.5857
    F-statistic: 15.34 on 7 and 64 DF,  p-value: 1.327e-11
```

9. Output data frame of residuals

```
>   g <- getCov(df)
#Display dataframe containing the residuals
 from the fitted univariate model
>    g$residualData[1:5,]

      NY52SO4     TN11SO4    IL63SO4
 1  0.1959813 -0.44377735  0.2824803
 2  0.6170340 -0.38791938 -0.3495080
 3 -0.4510128  1.05519620 -0.1158200
 4 -0.1580145 -0.01789642 -0.1297776
 5 -0.3218159 -0.47833685 -0.3415119
```

### 3.4.2 Functions for getting a vector of sites

`getCov()` takes site lists as input. The function `getSites()` produces a list of sites with available data for a specified time frame. The code below produces a list of 36 sites with the most weekly data between the years 1983–1986.

```
>  result <- getSites("01/01/83 00:00","12/31/86 00:00",36,104,
            "SO4","")
>  result$finalList

   [1] "OH71" "NY08" "WV18" "MI53" "NH02" "OH49" "PA42" "ME09"
   [9] "IN34" "MA13" "NY52" "NY10" "WA14" "NY20" "OH17" "ME00"
  [17] "TN00" "IL63" "MI99" "WI28" "IN41" "PA29" "WI36" "ME02"
  [25] "MI09" "MO05" "NC03" "NJ99" "PA15" "CO19" "MN18" "WI37"
  [33] "AR27" "KS31" "ME98" "MO03"
```

The 4th input specifies the minimum sample of weekly data required to be included in the produced list, and the last input tells the function to only look at sites in the Northern region of the US. The options for region are "W", "S", and "N"; see Figure 3.8 for the geographic split.


Figure 3.8: Geographic split

```
> NSites <- getSites("01/01/83 00:00","12/31/86 00:00",36,104,
            "SO4","N")
> NSites$finalList

  [1] "OH71" "NY08" "MI53" "NH02" "OH49" "PA42" "ME09" "IN34"
  [9] "MA13" "NY52" "NY10" "NY20" "OH17" "ME00" "MI99" "WI28"
 [17] "IN41" "PA29" "WI36" "ME02" "MI09" "NJ99" "PA15" "MN18"
 [25] "WI37" "ME98" "IL11" "IL18" "MN16" "MI26" "NE15" "VT01"
 [33] "NY99" "MA01" "MA08" "MN27"
```

The function `maxDistSites()` prioritizes sites that are farther away from each other. This function takes the same arguments as input as `getSites()`, except for the last argument where instead of specifying a region, the user can specify which site should be included first. If the last argument is 1, then the site with the greatest amount of data for the specified time period will be chosen; if the last argument is 2, then the site with the second greatest amount of data will be chosen; etc.

```
>   maxdist <- maxDistSites("01/01/83 00:00","12/31/86 00:00",36,
              104,"SO4",1)
>   maxdist$finalList

   [1] "OH71" "WA14" "TX04" "FL11" "ME00" "WY06" "MN27" "LA12"
   [9] "CA45" "OK00" "NY99" "GA41" "MI99" "AZ03" "MT05" "NC35"
  [17] "MO05" "CO00" "WY99" "IN34" "KY03" "MI09" "FL03" "MA01"
  [25]  "OR10" "PA42" "AR27" "MN16" "TX21" "VT99" "NE15" "VA13"
  [33]  "CO15" "CO22" "NY52" "AR02"
```

### 3.4.3 Lambert W transformation on univariate data

For a number of sites, the residuals produced by our univariate model have skewed distributions with heavy tails. In particular, this is the case for many sites when the sample of data is taken over a period greater than 4 years. To deal with this issue, we have incorporated functions from the LambertW package (see [Goe16]) in the function `lambertWtransform()` that will allow a user to transform the residuals produced by the deterministic univariate model. The LambertW package estimates the parameters that fit a Lambert W distribution on the given univariate data. Then the underlying Gaussian distribution implied by the Lambert W distribution is extracted and is used for the multivariate analysis in the function `lambertWtransform()`. The `lambertWtransform()` function takes the following as input: a data frame of residuals, and two logical inputs specifying whether to plot the multivariate qq plot and whether to produce the .mat file containing the covariance matrix with the Lambert W transformed residuals. Details on the algorithms that perform the transformation can be found in [Goe11]. Here we provide an example where we transform the residuals of 50 sites stored in an internal dataset, named "dfRes50".

```
>  data("dfRes50")
>  loutput <- lambertWtransform(dfRes=dfRes50, plotMulti=FALSE,
             writeMat=FALSE)
>  loutput$mvn$multivariateNormality

              Test  Statistic   p value    Result
  1 Mardia Skewness      22800    0.0004        NO
  2 Mardia Kurtosis      0.418    0.6763       YES
  3             MVN       <NA>      <NA>        NO
```

This function produces a list of four outputs:

1. `loutput$mvn` contains the results of applying the multivariate analysis by the MVN package
2. `loutput$cov` contains the covariance matrix produced by the transformed residuals
3. `loutput$newResiduals` contains the data frame of Lambert W transformed residuals
4. `loutput$univariateTest` contains the univariate tests produced by the MVN function for the transformed residuals

```
> data("dfRes50")
> dfRes50 <- dfRes50
> loutput <- lambertWtransform(dfRes=dfRes50, plotMulti=FALSE,
              writeMat=FALSE)
> loutput$mvn
> loutput$cov
> loutput$newResiduals
> loutput$univariateTest
```

Next, we present an example where we use `maxDistSites()` to get a list of 50 sites that is geographically sparse and has at least 200 weeks of data between 1986 and 1994. From this list of sites, a covariance and its corresponding multivariate normality test is generated and compared to the Lambert W transformed output.

```
#get list of sites
>  maxd  <- maxDistSites("01/01/86 00:00","12/31/94 00:00",50,
           200,"SO4",1)
#create input data frame
>  df <- defaultInput
#use list of sites and specification in maxd
>  df$siteAdd <- list(maxd$finalList)
>  df$startdateStr <- maxd$startDate
>  df$use36 <- FALSE
>  df$comp <- maxd$comp
>  df$enddateStr <- maxd$endDate
>  df$writeMat <- TRUE
>  output <- getCov(df)
>  output$mvn$multivariateNormality

               Test   Statistic  p value    Result
  1 Mardia Skewness       22962   2.511e-05     NO
  2 Mardia Kurtosis      0.2408   0.8097       YES
  3             MVN        <NA>     <NA>        NO

loutput <- lambertWtransform(g$residualDataNA, TRUE,FALSE)
loutput$mvn$multivariateNormality

##              Test         Statistic           p value Result
## 1 Mardia Skewness  22266.3107494977 0.214106607512284    YES
## 2 Mardia Kurtosis -1.59221962100786 0.111335366028036    YES
## 3             MVN              <NA>              <NA>    YES
```

### 3.4.4 Internal datasets and their properties

We provide five internal covariance matrices that we have produced from geographically sparse lists of sites. The exact specifications used to produce the sites can be found in the following chunk of code:

```
#sites with maximum distance data sets, get 50 sites
> maxd1 <- maxDistSites("01/01/86 00:00","12/31/94 00:00",50,
          200,"SO4",1)
> maxd2 <- maxDistSites("01/01/07 00:00","12/31/14 00:00",50,
          230,"SO4",1)
> maxd3 <- maxDistSites("01/01/07 00:00","12/31/14 00:00",50,
          230,"NO3",1)
> maxd4 <- maxDistSites("01/01/07 00:00","12/31/14 00:00",50,
          230,"Na",1)
> maxd5 <- maxDistSites("01/01/07 00:00","12/31/14 00:00",50,
          230,"ph",1)
```

We offer these covariance matrices for the convenience of the user. We note that although the list of sites are quite spread out geographically, they are not independent. We test the independence of the covariance matrices using a likelihood ratio test (see [RC12, p. 275]) The test statistic is

$$u := -\left(\nu - \frac{2m+5}{6}\right)\log(\det(R)),$$

where $m :=$ number of sites, $\nu := m(m+1)/2$, and $R$ is the sample correlation matrix. The null hypothesis $H_0$ is that the variates are independent, and we reject $H_0$ if $u > \chi^2_{m(m-1)/2,\alpha}$ where for our analysis $\alpha = 0.05$. The five covariance matrices are named "maxd1Cov", "maxd2Cov", "maxd3Cov", "maxd4Cov", and "maxd5Cov", and the corresponding test statistics $u$ are 20182, 25851, 26133, 28331, and 24898, all comfortably giving evidence to reject $H_0$ at the $\alpha = 0.05$ level (we reject when $u > 1308$).

We have made a function available that performs this independence test. Here we indicate how it can be used on a data frame of residuals produced by `getCov()`.

```
> maxd1  <- maxDistSites("01/01/86 00:00","12/31/94 00:00",50,
            200,"SO4",1)
> df$comp <- maxd1$comp
> df$enddateStr <- maxd1$endDate
> df$startdateStr <- maxd1$startDate
> df$siteAdd <- list(maxd1$finalList)
> result  <- getCov(df)
> indp  <- independenceTest(result$residualData)
> indp$test

    chisq dist likelihood ratio   tchisq  independent
  1                       20182  1307.54        FALSE
```

**Acknowledgments.** The results in this chapter are based on the paper [ATL20b].

# CHAPTER 4

# Stochastic Minimum Query

## 4.1 Introduction

We study a natural stochastic selection problem that involves querying a set of random variables so as to identify their minimum (or maximum) value within a desired precision. Consider a car manufacturer who wants to choose one design from $n$ options so as to optimize some attribute (e.g., maximum velocity or energy efficiency). Each option $i$ corresponds to an attribute value $X_i$ which is uncertain and drawn from a known probability distribution. It is possible to determine the *exact* value of $X_i$ by further testing— but this incurs some cost $c_i$. Identifying the exact minimum (or maximum) value among the $X_i$s might be too expensive. Instead, our goal is to identify an *approximately* minimum (or maximum) value, within a prescribed tolerance level. For example, we might be satisfied with a value (and corresponding option) that is within 10% of the true minimum. The objective is to minimize the expected cost. In this chapter, we provide the first constant-factor approximation algorithm for this problem.

Our problem falls under the umbrella of stochastic combinatorial optimization, where a solution makes selections incrementally and adaptively (i.e., the next selection can depend on previously observed random outcomes). Such problems are significantly harder than deterministic optimization problems because an optimal solution here may require exponential space to describe. Nevertheless, there has been much recent success in obtaining (efficient) approximation algorithms for such problems, e.g., [DGV08, GM07, BGL+12, GKNR15, GGHK18, INvdZ16, JLLS20, HKP21, HLS24].

Related to the minimum (or maximum) value problem that we consider, there have been some alternative formulations to trade off test/query costs with the attribute value [AN16, GGM10, WGW22]. The common aspect in these models is that there is a given *budget* $B$ on the query costs and the goal is to select queries $S \subseteq [n]$ of cost at most $B$ so as to minimize (or maximize) $X_i$ over $i \in S$, that is, *only among the selected* random variables. In

contrast, our goal is to achieve a value close to the true minimum/maximum taken over *all* random variables $X_1, \cdots X_n$. Moreover, we want to find an approximately min/max value with probability one, as opposed to optimizing the expected min/max value. So, our results are incomparable to these prior results: see Section 4.1.3 for more details.

### 4.1.1 Problem definition

In the *stochastic minimum query* (SMQ) problem, there are $n$ independent discrete random variables $X_1, ..., X_n$ that lie in intervals $I_1, ..., I_n$ respectively. The random variables (r.v.s) may be negative. We assume that each interval is bounded and closed, i.e., $I_j = [\ell_j, r_j]$ for each $j \in [n]$. We also assume (without loss of generality) that each r.v. has non-zero probability at the endpoints of its interval, i.e., $\Pr[X_j = \ell_j] > 0$ and $\Pr[X_j = r_j] > 0$ for each $j \in [n]$.[1] We will use the terms random variable (r.v.) and interval interchangeably. The exact value of any r.v. $X_j$ can only be determined by querying it, which incurs some cost $c_j \geq 1$. Additionally, we are given a "precision" value $\delta \geq 0$, where the goal is to identify the minimum value over *all* r.v.s up to an additive precision of $\delta$. Formally, if $\mathsf{MIN} = \min_{j=1}^{n} X_j$ then we want to find a deterministic value $\mathsf{VAL}$ such that $\mathsf{MIN} \leq \mathsf{VAL} \leq \mathsf{MIN} + \delta$. Such a value $\mathsf{VAL}$ is called a *$\delta$-minimum* value. The objective in SMQ is to minimize the expected cost of the queried intervals. Note that it may be sufficient to probe only a (small) subset of intervals before stopping.

We also consider a related, but harder, problem where the goal is to *identify* some $\delta$-minimizer $i^* \in [n]$, i.e., an interval that satisfies $X_{i^*} \leq \mathsf{MIN} + \delta$. We refer to this problem as *stochastic minimum query for identification* (SMQI). If a $\delta$-minimum value is found then it also provides a $\delta$-minimizer (see Section 4.1.4). However, the converse is not true. So, an SMQI solution may return an un-queried a $\delta$-minimizer $i^*$ without determining a $\delta$-minimum value.

Although our formulation above uses *additive* precision (we aim to find a value that is at most $\mathsf{MIN} + \delta$), we can also handle *multiplicative* precision where the goal is to find a value that is at most $\alpha \cdot \mathsf{MIN}$. This just requires a simple logarithmic transformation; see Section 4.1.5. We can also handle the goal of finding the *maximum* value by working with negated r.v.s $\{-X_i\}_{i=1}^{n}$.

Throughout, we use $N := [n] = \{1, 2, \cdots n\}$ to denote the index set of the r.v.s.

**Adaptive and Non-adaptive policies** Any solution to SMQ involves querying r.v.s sequentially until a $\delta$-minimum value is found. In general, the sequence of queries may depend

[1] Otherwise, we can just work with a smaller interval representing the same r.v.

on the realizations of previously queried r.v.s. We refer to such solutions as *adaptive* policies. Formally, such a solution can be described as a decision tree where each node corresponds to the next r.v. to query and the branches out of a node represent the realization of the queried r.v. *Non-adaptive* policies are a special class of solutions where the sequence of queries is fixed upfront: the policy then performs queries in this order until a $\delta$-minimum value is found. A central notion in stochastic optimization is the *adaptivity gap* [DGV08], which is the worst-case ratio between the optimal non-adaptive value and the optimal adaptive value. All our algorithms produce non-adaptive policies and hence also bound the adaptivity gap.

### 4.1.2 Results

Our first result is on the SMQ problem with unit costs, for which we provide a 4-approximation algorithm. Moreover, we achieve this result via a non-adaptive policy, which also proves an upper bound of 4 on the adaptivity gap. This algorithm relies on combining two natural policies. The first policy simply queries the r.v. with the smallest left-endpoint. The second policy queries the r.v. that maximizes the probability of stopping in the very next step. When used in isolation, both these policies have unbounded approximation ratios. However, interleaving the two policies leads to a constant-factor approximation algorithm.

We also consider the (harder) unit-cost SMQI problem and show that the same policy leads to a 4-approximation algorithm: the only change is in the criterion to stop, which is now more relaxed. While the algorithm is the same as SMQ, the analysis for SMQI is significantly more complex due to the new stopping criterion, which allows us to infer a $\delta$-minimizer $i^*$ even when it has not been queried. Specifically, we prove a stochastic dominance property between r.v.s in our algorithm and the optimum (conditioned on the SMQ stopping criterion not occurring), and use this in relating the SMQI stopping-probability in the algorithm and the optimum.

Our next result is for the SMQ problem with non-uniform costs. We obtain a constant-factor approximation again, with a slightly worse ratio of 5.83. This is based on combining ideas from the unit-cost algorithm with a "power-of-two" approach. In particular, the algorithm proceeds in several iterations, where the $i^{th}$ iteration incurs cost roughly $2^i$. In each iteration $i$, the algorithm selects a subset of r.v.s with cost $O(2^i)$ based on the following two criteria (i) smallest left-endpoint and (ii) maximum probability of stopping in one step. In order to select the r.v.s for criterion (ii) we need to use a PTAS for an appropriate version of the knapsack problem.

Finally, we consider the SMQI problem with non-uniform costs. Directly using the SMQ algorithm for SMQI (as in the unit-cost case) does not work here: it leads to a poor approxi-

mation ratio. However, a modification of the SMQ algorithm works. Specifically, we modify step (i) above: instead of just selecting a prefix of intervals with the smallest left-endpoints, we select an "almost prefix" set by skipping some expensive intervals. We show that this approach leads to an approximation ratio of 7.47, which is slightly worse than what we obtain for SMQ. The analysis combines aspects of unit-cost SMQI and SMQ with non-uniform costs.

### 4.1.3 Related work

Computing an approximately minimum or maximum value by querying a set of random variables is a central question in stochastic optimization. Most of the prior works on this topic have focused on *budgeted* variants. Here, one wants to select a subset of queries of total cost within some budget so as to maximize or minimize the value among the queried r.v.s. The results for the minimization and maximization versions are drastically different. A $1 - \frac{1}{e}$ approximation algorithm for the budgeted max-value problem follows from results on stochastic submodular maximization [AN16]; more complex "budget" constraints can also be handled in this setting [ASW16, GNS17]. These results also bound the adaptivity gap. In addition, PTASes are known for non-adaptive and adaptive versions of budgeted max-value [FLX18, SS21]. For the budgeted min-value problem, it is known that the adaptivity gap is unbounded and results for the non-adaptive and adaptive versions are based on entirely different techniques. [GGM10] obtained a bi-criteria approximation algorithm for the non-adaptive problem (the queried subset must be fixed upfront) that achieves a $1+\epsilon$ approximation to the optimal value while exceeding the budget by at most an $O(\log \log m)$ factor, where each r.v. takes an integer value in the range $\{0, 1, \cdots m\}$. Subsequently, [WGW22] studied the adaptive setting (the queried subset may depend on observed realizations) and obtained a 4-approximation while exceeding the budget by at most an $O(\log \log m)$ factor.

Closely related to our work, [CHdT21] studied the SMQI problem with exact precision, i.e., $\delta = 0$. In particular, their goal is to identify an *interval* that is an exact minimizer. [CHdT21] obtained a 1.45-approximation ratio for general query costs. The SMQI problem that we study allows for arbitrary precision $\delta$, and is significantly more complex than the setting in [CHdT21]. One indication of the difficulty of handling arbitrary $\delta$ is that the simpler SMQ problem with $\delta = 0$ (where we want to find the exact minimum value) admits a straightforward exact algorithm that queries by increasing left-endpoint; however, this algorithm has an unbounded ratio for SMQ with arbitrary $\delta$ (see Section4.2 for an example).

The SMQ problem is also related to optimization problems under explorable uncertainty, which typically involve adversarial (i.e., not stochastic) values drawn from each

interval, though some results on stochastic inputs are known [BDE+21, MS23]. The key difference from our work is that all these results focus on the *competitive ratio*, which relates the algorithm's (expected) query cost to the optimum query-cost in hindsight. This differs significantly from our setting, where we compare to the optimal policy that is limited in the same manner as the algorithm. Apart from the minimum-value problem [Kah91, CHdT21], various other problems like computing the median [FMP+00], minimum spanning tree [EHK+08, MMS17] and set selection [EHK16, BDE+21, MS23] have been studied in this setting. We note that there is an $\tilde{\Omega}(n)$ lower bound on the competitive ratio for SMQ and SMQI; see Section 4.1.8. Our results show that much better (constant) approximation ratios are achievable for SMQ and SMQI in the stochastic setting, relative to an optimal policy.

### 4.1.4 Stopping rule for SMQ

Even without querying any interval, we know that the minimum value is at most $R := \min_{i \in N}\{r_i\}$, the minimum right-endpoint. In order to simplify notation, we incorporate this information using a dummy r.v. $X_0 = [R, R]$ that is queried at the start of any policy and incurs no cost. We now formally define the condition under which a policy for SMQ is allowed to stop. We will refer to the partial observations at any point in a policy (i.e., values of r.v.s queried so far) as the *state*. Consider any state, given by a subset $S \subseteq N$ of queried r.v.s along with their observations $\{x_i\}_{i \in S}$. The minimum observed value is $\min_{i \in S} x_i$ and the minimum possible value among the un-queried r.v.s is $\min_{j \in N \setminus S} \ell_j$. The stopping criterion is:

$$\min_{i \in S} x_i \quad \leq \quad \min_{j \in N \setminus S} \ell_j \; + \; \delta. \tag{4.1}$$

If this criterion is met then $\mathsf{VAL} = \min_{i \in S} x_i$ is guaranteed to satisfy $\mathsf{MIN} \leq \mathsf{VAL} \leq \mathsf{MIN} + \delta$, where $\mathsf{MIN} = \min_{j \in N} X_j$. Also, $\arg\min_{i \in S} x_i$ is a $\delta$-minimizer. On the other hand, if this criterion is not met then there is no value $v$ that guarantees $\mathsf{MIN} \leq v \leq \mathsf{MIN} + \delta$: there is a non-zero probability that the minimum value is $\min_{j \in N \setminus S} \ell_j$ or $\min_{i \in S} x_i$ (and these values are more than $\delta$ apart). So,

**Proposition 4.1.1.** *A policy for* SMQ *can stop if and only if criterion* (4.1) *holds.*

The stopping rule for SMQI is described in Section 4.2.2. An SMQI policy can stop either due to the SMQ stopping rule (above) or by inferring an un-queried interval $i^*$ as a $\delta$-minimizer.

### 4.1.5 Multiplicative precision

Given an instance with non-negative r.v.s $\{X_i\}_{i=1}^n$ and multiplicative precision $\alpha \geq 1$, consider a new instance of $\mathsf{SMQ}$ with r.v.s $\{X'_i := \ln(X_i)\}_{i=1}^n$ and additive precision $\delta := \ln \alpha$. Note that

$$\mathsf{MIN}' = \min_{i=1}^{n} X'_i = \min_{i=1}^{n} \ln(X_i) = \ln\left(\min_{i=1}^{n} X_i\right) = \ln(\mathsf{MIN}).$$

An $\alpha$-approximately minimum value $W$ for the original instance satisfies $\mathsf{MIN} \leq W \leq \alpha \cdot \mathsf{MIN}$, where $\mathsf{MIN} = \min_{i=1}^{n} X_i$. Then, $\mathsf{VAL} = \ln(W)$ satisfies $\mathsf{MIN}' = \ln(\mathsf{MIN}) \leq \mathsf{VAL} \leq \ln(\mathsf{MIN}) + \ln \alpha = \mathsf{MIN}' + \delta$, i.e., $\mathsf{VAL}$ is a $\delta$-minimum value for the new instance. Similarly, if $\mathsf{VAL}$ is a $\delta$-minimum value for the new instance then $W := e^{\mathsf{VAL}}$ is an $\alpha$-approximately minimum value for the original instance.

### 4.1.6 Adaptivity gap

We show that the adaptivity gap for the $\mathsf{SMQ}$ problem is more than one: so adaptive policies may indeed perform better. This example also builds some intuition for the problem. Consider an instance $\mathcal{I}$ with three intervals as shown in Figure 4.1. In particular, $X_1 \in \{0, 3, \infty\}$, $X_2 \in \{1, \infty\}$, $X_3 \in \{2, \infty\}$ and $\delta = 1$. Let $\Pr(X_1 = 0) = \frac{1}{3}, \Pr(X_1 = 3) = \frac{1}{3}, \Pr(X_1 = \infty) = \frac{1}{3}, \Pr(X_2 = 1) = \epsilon_2, \Pr(X_2 = \infty) = 1 - \epsilon_2, \Pr(X_3 = 2) = 1 - \epsilon_3, \Pr(X_3 = \infty) = \epsilon_3$. An adaptive policy is shown in Figure 4.2, which has cost at most $1 + \frac{2}{3} + \frac{\epsilon_3}{3} = \frac{5+\epsilon_3}{3}$. Setting $\epsilon_3 = \frac{1}{2}$, and as $\epsilon_2 \to 0$ we obtain an adaptivity gap of $\frac{12}{11}$.

We consider the cost of all possible non-adapative policies:

1. The cost of policy $\{1, 2, 3\}$ is,

$$NA \geq 1 + \frac{2}{3} + \frac{1 - \epsilon_2}{3} = \frac{6 - \epsilon_2}{3}.$$

   We query $X_1$ w.p. 1, $X_2$ w.p. $2/3$ and $X_3$ w.p. $(1 - \epsilon_2)/3$.

2. The cost of policy $\{1, 3, 2\}$ is,

$$NA \geq 1 + \frac{2}{3} + \frac{2\epsilon_3}{3} = \frac{5 + 2\epsilon_3}{3}.$$

   We query $X_1$ w.p. 1, $X_3$ w.p. $2/3$ and $X_2$ w.p. $2\epsilon_3/3$.

3. The cost of policy $\{2, 1, 3\}$ is,

$$NA \geq 2 - \epsilon_2 + \frac{1 - \epsilon_2}{3} = \frac{7 - 4\epsilon_2}{3}.$$

We query $X_2$ w.p. 1, $X_1$ w.p. $1-\epsilon_2$ and $X_3$ w.p. $(1-\epsilon_2)/3$.

4. The cost of policy $\{2,3,1\}$ is,

$$NA \geq 2-\epsilon_2+1-\epsilon_2 = 3-2\epsilon_2.$$

We query $X_2$ w.p. 1, $X_3$ w.p. $1-\epsilon_2$ and $X_1$ w.p. $1-\epsilon_2$.

5. The remaining non-adaptive policies start with 3. Any such policy costs at least 2 because even if $X_3 = 2$ (its lowest value) we cannot stop.

So, the optimal non-adaptive value is $\frac{1}{3}\cdot\min\{6-\epsilon_2, 5+2\epsilon_3, 7-4\epsilon_2, 9-6\epsilon_2, 6\}$. Setting $\epsilon_3 = \frac{1}{2}$, and $\epsilon_2 \to 0$ then non-adaptive optimum is at least $\frac{12}{6}$, whereas the adaptive optimum is $\frac{11}{6}$.

Hence, the adaptivity gap for SMQ is at least $\frac{12}{11}$.

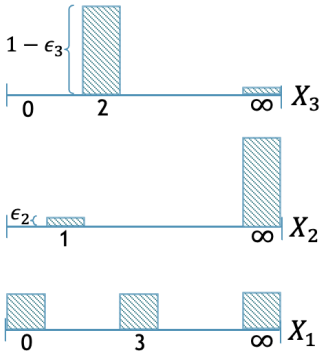



Figure 4.1: Adaptivity gap instance for SMQ.

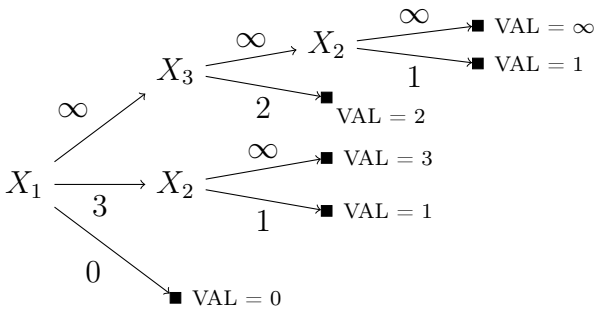



Figure 4.2: Optimal Adaptive policy

### 4.1.7 Fixed threshold problem

In our analysis, we relate SMQ to the following simpler problem. Given $n$ r.v.s $\{X_i : i \in N\}$ with costs as before, a *fixed* threshold $\theta$ and budget $k$, find a policy having query-cost at most $k$ that maximizes the probability of observing a realization less than $\theta$. A useful property of this fixed threshold problem is that it has adaptivity gap one.

**Proposition 4.1.2.** *Consider any instance of the fixed threshold problem. Let $V^*$ and $F^*$ denote the maximum success probabilities over adaptive and non-adaptive policies respectively. Then, $V^* = F^*$*

*Proof.* We proceed by induction on the budget $k$. For any set $S$ of r.v.s and budget $k$, let

$$V(S,k) := \max_{\mathcal{A}\subseteq S, c(\mathcal{A})\le k} \Pr_{\mathcal{A},X}\left[\min_{j\in\mathcal{A}} X_j \le \theta\right],$$

denote the maximum success probability over adaptive policies (having cost at most $k$). Similarly,

$$F(S,k) = \max_{T\subseteq S: c(T)\le k} \Pr_{X}\left[\min_{j\in T} X_j \le \theta\right]$$

be the maximum over non-adaptive policies. We will show that $V(S,k) = F(S,k)$, which would prove Proposition 4.1.2. It suffices to show $V(S,k) \le F(S,k)$. (Clearly, $V(S,k) \ge F(S,k)$ as adaptive policies capture all non-adaptive policies.)

The base case ($k = 1$) is trivial because any policy is non-adaptive (it selects a single r.v.).

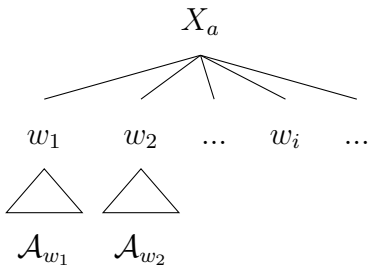



Figure 4.3: Adaptive policy $\mathcal{A}$ for the fixed threshold problem.

For the inductive step, we fix some budget $\ell$ and want to show $V(S,\ell) \le F(S,\ell)$. For any policy $\pi$, we will use $prob(\pi) := \Pr[\min_{i\in\pi} X_i \le \theta]$ to denote its success probability. Let $\mathcal{A}$ denote the optimal adaptive policy, which has $prob(\mathcal{A}) = V(S,\ell)$. Let $a \in S$ denote the first query in policy $\mathcal{A}$. Let $T^+$ (resp. $T^-$) represent all realizations of $X_a$ that are at most (resp. more than) threshold $\theta$. For any realization $w$ of $X_a$, let $\mathcal{A}_w$ denote the rest of policy $\mathcal{A}$ *conditioned* on $X_a = w$; note that the cost $c(\mathcal{A}_w) \le \ell - c_a$ because policy $\mathcal{A}$ always has cost at most $\ell$. See Figure 4.3. Below, we use $p_a := \Pr[X_a \le \theta] = \sum_{w\in T^+} \Pr[X_a = w]$; so $\sum_{w\in T^-} \Pr[X_a = w] = 1 - p_a$. We now have:

$$\begin{aligned} V(S,\ell) = prob(\mathcal{A}) &= p_a + \sum_{w\in T^-} \Pr[X_a = w]\cdot prob(\mathcal{A}_w) \\ &\le p_a + \sum_{w\in T^-} \Pr[X_a = w]\cdot V(S\setminus a, \ell - c_a) = p_a + (1-p_a)\cdot V(S\setminus a, \ell - c_a) \quad (4.2) \\ &\le p_a + (1-p_a)\cdot F(S\setminus a, \ell - c_a) \le F(S,\ell) \quad (4.3) \end{aligned}$$

The inequality in (4.2) uses the fact that each $\mathcal{A}_w$ is a feasible adaptive policy for the smaller instance on r.v.s $S \setminus a$ and budget $\ell - c_a$. The first inequality in (4.3) is by induction. The second inequality in (4.3) is by the following observation. Let $T \subseteq S \setminus a$ be an optimal non-adaptive policy for the instance $F(S \setminus a, \ell - c_a)$; then $T \cup a$ is a feasible non-adaptive policy for the instance $F(S, \ell)$ with success probability $p_a + (1-p_a) \cdot prob(T) = p_a + (1-p_a) \cdot F(S \setminus a, \ell - c_a)$. □

### 4.1.8 Bad example for competitive ratio

We provide an example that rules out any reasonable *competitive ratio* bound for SMQ and SMQI with precision $\delta > 0$. This is in sharp contrast to the corresponding problem with exact precision ($\delta = 0$) for which a constant competitive ratio is known [Kah91]. We note that results in the online setting assume open intervals, which in our setting (with discrete r.v.s) corresponds to all left-endpoints being distinct.[2] The benchmark in the online setting is the *hindsight optimum*, which is the minimum number (or cost) of queries that are needed to verify a $\delta$-minimum value *conditioned* on the realizations $\{x_i\}_{i=1}^n$ of the r.v.s.

Consider an instance with $n$ r.v.s with $\Pr[X_i = i] = \frac{\ln n}{n}$ and $\Pr[X_i = n^2] = 1 - \frac{\ln n}{n}$ for all $i \in [n]$. All costs are unit and the precision $\delta = n$. We refer to the values $\{1, 2, \cdots n\}$ as *low* values: note that any low value is a $\delta$-minimum value for this instance.

We first consider the hindsight optimum. If any of the $n$ r.v.s (say $k$) realizes to a low value then verifying the $\delta$-minimum value just requires querying $k$, which has cost 1. On the other hand, the probability that none of the $n$ r.v.s realizes to a low value is $\left(1 - \frac{\ln n}{n}\right)^n \leq \frac{1}{n}$: in this case the optimal verification cost is $n$ (querying all r.v.s). So the expected optimal cost is at most 2.

Now, consider an SMQ policy: this does not know the realizations. It is easy to see that the only way to stop querying is when some low value is observed (or all $n$ r.v.s are queried). So, the expected cost of any policy is at least $\frac{n}{\ln n}$. Hence the competitive ratio for SMQ is $\Omega(\frac{n}{\ln n})$.

## 4.2 Algorithm for Unit Costs

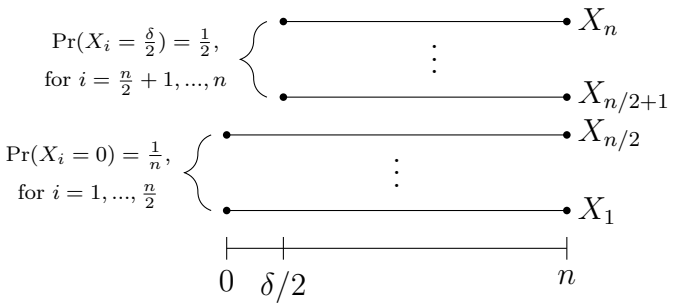
Before presenting our algorithm, we discuss two simple greedy policies and show why they fail to achieve a good approximation.

1. A natural approach is to select intervals by increasing left-endpoint. Indeed, [Kah91]

[2] Alternatively, our example can be modified into one with open intervals where the competivity ratio is $\tilde{\Omega}(n)$.

shows that this algorithm is optimal when $\delta = 0$, even in an online setting. Consider the instance with two types of intervals as shown in Figure 4.4. The r.v.s $X_1, \cdots X_{n/2}$ are identically distributed with $X_i = 0$ w.p. $\frac{1}{n}$ and $X_i = n$ otherwise. The remaining r.v.s $X_{n/2+1}, \cdots X_n$ are identically distributed with $X_i = \frac{\delta}{2}$ w.p. $\frac{1}{2}$ and $X_i = n$ otherwise. The greedy policy queries r.v.s in the order $1, 2, \cdots n$, resulting in an expected cost of $\Omega(n)$ as it can stop only when it observes a "low" realization for some r.v. However, the policy that probes in the reverse order $n, n-1, \cdots 1$ has constant expected cost: the policy can stop upon observing *any* "low" realization (even if a value of $\delta/2$ is observed, it is guaranteed to be within $\delta$ of the true minimum). So the approximation ratio of this greedy policy is $\Omega(n)$.

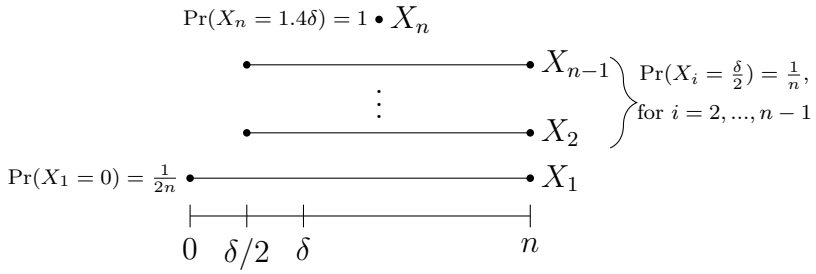
Figure 4.4: Bad example for greedy by left-endpoint.




2. A different greedy policy (based on the instance in Figure 4.4) is to always select the interval that maximizes the likelihood of stopping in one step. Now consider another instance with three types of intervals; see Figure 4.5. The r.v. $X_n$ is always $1.4\delta$. The r.v. $X_1$ takes value 0 w.p. $\frac{1}{2n}$ and has value $n$ otherwise. The remaining r.v.s $X_2, \cdots X_{n-1}$ are identically distributed with $X_i = \frac{\delta}{2}$ w.p. $\frac{1}{n}$ and $X_i = n$ otherwise. As long as $X_1$ is not queried, the probability of stopping (in one step) is as follows: $\frac{1}{2n}$ for $X_1$, $\frac{1}{n}$ for $X_2, \cdots X_{n-1}$ and zero for $X_n$. So this greedy policy will query in the order $2, 3, \cdots n-1, 1, n$ resulting in an $\Omega(n)$ expected cost. On the other hand, querying the r.v.s $X_1$ and $X_n$ guarantees that the policy can stop. So the optimal cost is at most 2, implying an $\Omega(n)$ approximation ratio.

Our approach is to interleave the above two greedy criteria. In particular, each iteration of our algorithm makes two queries: the interval with the smallest left-endpoint and the interval that maximizes the probability of stopping in one step. We will show that this leads to a constant-factor approximation. We first re-number intervals by increasing order of their

Figure 4.5: Bad example for greedy by stopping probability.




left-endpoint, i.e., $\ell_1 \leq \ell_2 \leq \cdots \ell_n$. For each $k \in N$, let $\theta_k := \ell_{k+1} + \delta$. Algorithm 3 describes our algorithm formally.

**Algorithm 3** Non-Adaptive Double Greedy

Let $\ell^* = \min_{i \in N} \ell_i$, $m^* = R := \min_{i=1}^{n} r_i$, and $\pi \leftarrow \emptyset$.
**for** $j = 1, \ldots, n$ **do** ▷ iterations
  Query interval $j$ (if not already in $\pi$).
  Query interval $b(j) = \text{argmax}_{i \in N \setminus (\pi \circ j)} \Pr[X_i \leq \theta_j]$.
  Update list $\pi \leftarrow \pi \circ j \circ b(j)$. ▷ skip $j$ if it was already in $\pi$
  Update $m^* = \min\{m^*, X_j, X_{b(j)}\}$ and $\ell^* = \min_{i \in N \setminus \pi}\{\ell_i\}$.
  **if** $m^* - \ell^* \leq \delta$ **then** stop.

Equivalently, we can view Algorithm 3 as first computing the permutation $\pi$ (without querying) and then performing queries in the order given by $\pi$ until the stopping criterion is met. Note that Algorithm 3 is non-adaptive because it uses observations only to determine when to stop. So, our analysis also upper bounds the adaptivity gap.

We overload notation slightly and use $\pi$ to also denote the non-adaptive policy given in Algorithm 3. Note that each *iteration* in this policy involves *two* queries. We use $\sigma$ to denote the optimal (adaptive) policy. Let $c_{exp}(\pi)$ and $c_{exp}(\sigma)$ denote the expected number of queries in policies $\pi$ and $\sigma$, respectively. The key step in the analysis is to relate the termination probabilities in these two policies, formalized below.

**Lemma 4.2.1.** *For any $k \geq 1$, we have*

$$\Pr[\sigma \textit{ finishes in } k \textit{ queries}] \leq \Pr[\pi \textit{ finishes in } 2k \textit{ iterations}].$$

We will prove this lemma in the next subsection. First, we complete the analysis using this.

**Theorem 4.2.2.** *We have $c_{exp}(\pi) \leq 4 \cdot c_{exp}(\sigma)$.*

*Proof.* Let $C_\sigma$ denote the random variable that captures the number of queries made by the optimal policy $\sigma$. Similarly, let $C_\pi$ denote the number of queries made by our policy. Using Lemma 4.2.1 and the fact that policy $\pi$ makes two queries in each iteration, for any $k \geq 1$ we have

$$\Pr[C_\sigma \leq k] = \Pr[\sigma \text{ finishes in } k \text{ queries}] \leq \Pr[\pi \text{ finishes in } 2k \text{ iterations}] \leq \Pr[C_\pi \leq 4k] \tag{4.4}$$

Hence,

$$\begin{aligned} c_{exp}(\sigma) &= \int_0^\infty \Pr[C_\sigma > t]dt = \int_0^\infty (1 - \Pr[C_\sigma \leq t])\, dt \geq \int_0^\infty (1 - \Pr[C_\pi \leq 4t])dt \\ &= \frac{1}{4}\int_0^\infty (1 - \Pr[C_\pi \leq y])dy = \frac{1}{4}\int_0^\infty \Pr[C_\pi > y]dy = \frac{1}{4}c_{exp}(\pi) \end{aligned} \tag{4.5}$$

The first equality in (4.5) is by a change of variables $y = 4t$. □

### 4.2.1 Proof of key lemma

We now prove Lemma 4.2.1. Fix any $k \geq 1$ and define threshold $\theta := \theta_k = \ell_{k+1} + \delta$.

Let $T^* \subseteq N$ denote the optimal solution to the non-adaptive "fixed threshold" problem:

$$\max_{T \subseteq N, |T| \leq k} \quad \Pr\left[\min_{i \in T} X_i \leq \theta\right]. \tag{4.6}$$

We then proceed in two steps, as follows.

$$\Pr[\sigma \text{ finishes in } k \text{ queries}] \leq \Pr\left[\min_{i \in T^*} X_i \leq \theta\right] \leq \Pr[\pi \text{ finishes in } 2k \text{ iterations}]$$

The first inequality is shown in Lemma 4.2.3: this uses the fact that the fixed-threshold problem has adaptivity gap one (Proposition 4.1.2). The second inequality is shown in Lemma 4.2.4: this relies on the greedy criteria used in our algorithm.

**Lemma 4.2.3.** $\Pr[\sigma$ *finishes in* $k$ *queries*$] \leq \Pr\left[\min_{i \in T^*} X_i \leq \theta\right]$.

*Proof.* Let $\sigma^k$ denote the optimal policy truncated after $k$ queries: so the cost of $\sigma^k$ is always at most $k$. Let $L(\sigma^k) = \min_{i \in N \setminus \sigma^k}\{\ell_i\}$ denote the smallest un-queried left-endpoint at the

end of $\sigma^k$. Then,

$$\Pr[\sigma \text{ finishes in } k \text{ queries}] = \Pr\left[\min_{i\in\sigma^k} X_i \;\le\; L(\sigma^k) + \delta\right] \tag{4.7}$$

$$\le \Pr\left[\min_{i\in\sigma^k} X_i \;\le\; \ell_{k+1} + \delta\right] = \Pr\left[\min_{i\in\sigma^k} X_i \;\le\; \theta\right] \tag{4.8}$$

$$\le \Pr\left[\min_{i\in T^*} X_i \le \theta\right] \tag{4.9}$$

(4.7) is by the stopping criterion for SMQ. The inequality in (4.8) uses the observation that after *any* $k$ queries, the smallest un-queried left-endpoint must be at most $\ell_{k+1}$: so $L(\sigma^k) \le \ell_{k+1}$. The equality in (4.8) is by definition of the threshold $\theta$. Inequality (4.9) follows from Proposition 4.1.2 note that $\sigma^k$ is a feasible adaptive policy for the fixed-threshold problem and $T^*$ is the optimal non-adaptive policy. □

**Lemma 4.2.4.** $\Pr\left[\min_{i\in T^*} X_i \le \theta\right] \le \Pr[\pi \textit{ finishes in } 2k \textit{ iterations}]$.

*Proof.* Recall that each iteration $j$ of Algorithm 3 selects two intervals: $j$ in Step 3 and $b(j)$in Step 4. Let $B = \{b(1), \cdots b(2k)\}$ be the set of intervals chosen by our policy $\pi$ in Step 4 of the first $2k$ iterations. We partition $B$ into $B' = \{b(1), \cdots b(k)\}$ and $B'' = \{b(k+1), \cdots b(2k)\}$. Let $d^* = \operatorname{argmin}_{d\in T^*\setminus B} \Pr(X_d > \theta_k) = \operatorname{argmax}_{d\in T^*\setminus B} \Pr(X_d \le \theta_k)$.

$$\Pr\left[\min_{i\in T^*} X_i \;>\; \theta_k\right] = \prod_{i\in T^*} \Pr[X_i > \theta_k]$$

$$= \prod_{i\in T^*\cap B} \Pr[X_i > \theta_k] \cdot \prod_{i\in T^*\setminus B} \Pr[X_i > \theta_k]$$

$$\ge \prod_{i\in T^*\cap B} \Pr[X_i > \theta_k] \;\cdot\; (\Pr[X_{d^*} > \theta_k])^{|T^*\setminus B|} \tag{4.10}$$

$$\ge \prod_{i\in T^*\cap B} \Pr[X_i > \theta_{2k}] \cdot \prod_{i\in B''\setminus T^*} \Pr[X_i > \theta_{2k}] \tag{4.11}$$

$$\ge \prod_{i\in B} \Pr[X_i > \theta_{2k}] = \Pr\left[\min_{i\in B} X_i \;>\; \theta_{2k}\right] \tag{4.12}$$

(4.10) follows from the definition of $d^*$. (4.12) just uses that $T^* \cap B$ and $B'' \setminus T^*$ are disjoint subsets of $B$. The key step above is (4.11), which we prove using two cases:

- Suppose that $T^* \setminus B = \emptyset$. Then, using $\theta_k \le \theta_{2k}$ we obtain $\Pr[X_i > \theta_k] \ge \Pr[X_i > \theta_{2k}]$, which proves (4.11) for this case.

- Suppose that $T^* \setminus B \neq \emptyset$. In this case, $d^*$ is well-defined. We now claim that:

$$\text{For each } j = k+1, \cdots 2k, \text{ either } b(j) \in T^* \text{ or } \Pr[X_{b(j)} > \theta_{2k}] \leq \Pr[X_{d^*} > \theta_k]. \tag{4.13}$$

Indeed, consider any such $j$ and suppose that $b(j) \notin T^*$. As $d^*$ is a valid choice for $b(j)$, the greedy rule implies:

$$\Pr\left[X_{d^*} > \theta_j\right] \geq \Pr\left[X_{b(j)} > \theta_j\right].$$

Further, using the fact that $\theta_k \leq \theta_j \leq \theta_{2k}$, we get

$$\Pr\left[X_{d^*} > \theta_k\right] \geq \Pr\left[X_{d^*} > \theta_j\right] \geq \Pr\left[X_{b(j)} > \theta_j\right] \geq \Pr\left[X_{b(j)} > \theta_{2k}\right],$$

which proves (4.13). Let $h$ denote the number of iterations $j \in \{k+1, \cdots 2k\}$ where $b(j) \notin T^*$. Note that $h = |B''| - |T^* \cap B''| = k - |T^* \cap B''| \geq |T^* \setminus B|$, where we used $|T^*| = k$. Using (4.13), it follows that $\Pr[X_i > \theta_{2k}] \leq \Pr[X_{d^*} > \theta_k]$ for all $i \in B'' \setminus T^*$. Hence,

$$\Pr[X_{d^*} > \theta_k]^{|T^* \setminus B|} \geq \Pr[X_{d^*} > \theta_k]^h \geq \prod_{i \in B'' \setminus T^*} \Pr[X_i > \theta_{2k}],$$

Combined with the fact that $\theta_k \leq \theta_{2k}$ (as before), we obtain (4.11).

We are now ready to complete the proof. Using the SMQ stopping criterion and the fact that $\pi$ queries all the intervals in $B$ within $2k$ iterations,

$$\Pr[\pi \text{ finishes in } 2k \text{ iterations}] \geq \Pr\left[\min_{i \in B} X_i \leq \theta_{2k}\right] = 1 - \Pr\left[\min_{i \in B} X_i > \theta_{2k}\right].$$

Combined with (4.12),

$$\Pr[\pi \text{ finishes in } 2k \text{ iterations}] \geq 1 - \Pr\left[\min_{i \in T^*} X_i > \theta_k\right] = \Pr\left[\min_{i \in T^*} X_i \leq \theta\right],$$

where we use the definition $\theta = \theta_k$. □

### 4.2.2 Finding the minimum interval

In this section, we consider the SMQI problem, where the goal is to identify *an interval* that is guaranteed to be a $\delta$-minimizer. Unlike the previous SMQ setting (where we find a $\delta$-minimum value), for SMQI we just want to identify some interval $i^* \in N$ such that $X_{i^*} \leq \mathsf{MIN} + \delta$. Recall that $\mathsf{MIN} = \min_{i \in N} X_i$. It is important to note that the interval $i^*$

may not have been queried. It is easy to see that any SMQ policy is also feasible to SMQI. Indeed, by the stopping rule (4.1) for SMQ, the $\delta$-minimum value returned is always the minimum value of a queried interval: so we also identify $i^*$. However, an SMQI policy may return an interval $i^*$ without querying it. So the optimal value of SMQI may be smaller than SMQ.

**Stopping criteria for SMQI** Consider any state, given by a subset $S \subseteq N$ of queried r.v.s along with their observations $\{x_i\}_{i \in S}$. There are two conditions under which the SMQI policy can stop.

- The first stopping rule is just the one for SMQ (4.1), which corresponds to the situation that interval $i^*$ is queried. We restate this rule below for easy reference:

$$\min_{i \in S} x_i \quad \leq \quad \min_{j \in N \setminus S} \ell_j \; + \; \delta. \tag{4.14}$$

  In this case, we return $i^* = \arg\min_{i \in S} x_i$. We refer to this as the old stopping rule.

- The second stopping rule handles the situation where an un-queried interval $i^*$ is returned. For any $i \in N$, define the "almost prefix" set $P_i := \{j \in N \setminus i : \ell_j < r_i - \delta\}$. Note that either $P_i$ or $P_i \cup i$ is a *prefix* of $[n]$. (As before, we assume that intervals are indexed by increasing order of their left-endpoint, i.e., $\ell_1 \leq \ell_2 \leq \cdots \ell_n$.) The new rule is:

$$\exists\, i \in N \text{ such that } \; P_i \subseteq S \text{ and } \min_{j \in P_i} x_j \; \geq \; r_i - \delta. \tag{4.15}$$

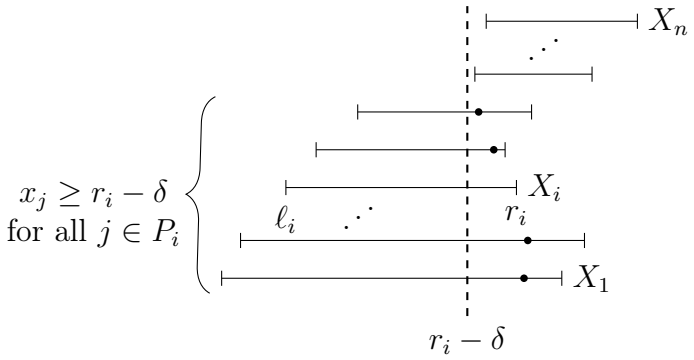
  In other words, there is some interval $i$ where (1) all intervals $j \neq i$ with left-endpoint $\ell_j < r_i - \delta$ have been queried, and (2) the minimum value of these r.v.s is at least $r_i - \delta$. In this case, we return $i^* = i$ (we may not know a $\delta$-minimum value). We refer to this as the new stopping rule. See Figure 4.6 for an example.

**Proposition 4.2.5.** *A policy for* SMQI *can stop if and only if either criterion* (4.14) *or* (4.15) *holds.*

Our algorithm for SMQI with unit costs remains the same as for SMQ (Algorithm 3). The only difference is in the new stopping criterion (described above). Recall that $\pi$ is the permutation used by our non-adaptive policy. When it is clear from the context, we will also use $\pi$ to denote our SMQI policy that performs queries in the order of $\pi$ until stopping criteria (4.14) or (4.15) applies.

**Theorem 4.2.6.** *The non-adaptive policy $\pi$ is a 4-approximation algorithm for* SMQI.

Figure 4.6: Illustration of new SMQI stopping criterion.




We now prove this result. Let $\sigma$ denote an optimal adaptive policy for SMQI. For any $k \geq 1$, we will show:

$$\Pr[\sigma \text{ finishes in } k \text{ queries}] \leq \Pr[\pi \text{ finishes in } 2k \text{ iterations}]. \tag{4.16}$$

This would suffice to prove the 4-approximation, exactly as in Theorem 4.2.2.

In order to prove (4.16), we fix some $k \geq 1$. As in the previous proof, let $\theta = \theta_k = \ell_{k+1} + \delta$ and let $T^*$ be defined as in (4.6). To reduce notation, define the following events.

$$\mathcal{A}_1 : \text{ our policy } \pi \text{ finishes within } 2k \text{ iterations due to (4.14).}$$

$$\mathcal{A}_2 : \text{ our policy } \pi \text{ finishes within } 2k \text{ iterations due to (4.15).}$$

$$\mathcal{O}_1 : \text{ optimal policy } \sigma \text{ finishes within } k \text{ queries due to (4.14).}$$

$$\mathcal{O}_2 : \text{ optimal policy } \sigma \text{ finishes within } k \text{ queries due to (4.15).}$$

**Handling the old stopping criterion.** Let $L$ denote the smallest un-queried left-endpoint at the end of iteration $2k$ in $\pi$. Note that $L$ is a deterministic value as $\pi$ is a non-adaptive policy. Moreover, $L \geq \ell_{2k+1}$ as $\pi$ would have queried the first $2k$ r.v.s. Let $\mathcal{G}$ be the event that $X_i > L + \delta$ for all intervals $i$ queried by $\pi$ in its first $2k$ iterations. In other words, $\mathcal{G}$ is precisely the event that stopping criterion (4.14) *does not* apply at the end of iteration $2k$ in $\pi$, i.e., $\mathcal{G} = \neg\mathcal{A}_1$. By Lemma 4.2.4,

$$\Pr[\neg\mathcal{G}] = \Pr[\mathcal{A}_1] \geq \Pr\left[\min_{i \in T^*} X_i \leq \theta\right].$$

Similarly, let $\mathcal{G}^*$ be that event that $X_i > \theta$ for all intervals $i$ in the first $k$ queries of $\sigma$. From the proof of Lemma 4.2.3, we obtain $\mathcal{O}_1 \subseteq \neg\mathcal{G}^*$ and

$$\Pr[\neg\mathcal{G}^*] \leq \Pr\left[\min_{i \in T^*} X_i \leq \theta\right].$$

Combining the above two inequalities, we have

$$\Pr[\neg\mathcal{G}^*] \leq \Pr[\neg\mathcal{G}]. \tag{4.17}$$

**Handling the new stopping criterion.** Let $\mathcal{G}_A$ be the event that $X_j > L + \delta$ for *all* r.v.s $j \in N$. Similarly, let $\mathcal{G}_A^*$ be the event that $X_j > \theta$ for *all* $j \in N$. Clearly,

$$\Pr\left[\mathcal{A}_2 \,|\, \mathcal{G}\right] \;=\; \Pr\left[\mathcal{A}_2 \,|\, \mathcal{G}_A\right] \quad \text{and} \quad \Pr\left[\mathcal{O}_2 \,|\, \mathcal{G}^*\right] \;=\; \Pr\left[\mathcal{O}_2 \,|\, \mathcal{G}_A^*\right]. \tag{4.18}$$

We will now prove that

$$\Pr\left[\mathcal{A}_2 \,|\, \mathcal{G}_A\right] \;\geq\; \Pr\left[\mathcal{O}_2 \,|\, \mathcal{G}_A^*\right]. \tag{4.19}$$

If $\sigma$ finishes due to (4.15) in $k$ queries then $P_{i^*} \subseteq [k+1]$: otherwise $|P_{i^*}| > k$ which contradicts with the fact that all r.v.s in $P_{i^*}$ must be queried. Let $R = \{i \in N : P_i \subseteq [k+1]\}$ be all such intervals. It now follows that the event $\mathcal{O}_2$ (which corresponds to policy $\sigma$) is a subset of the event

$$\mathcal{E} := \bigvee_{i \in R} \left(\wedge_{j \in P_i}(X_j \geq r_i - \delta)\right). \tag{4.20}$$

Note that $\mathcal{E}$ is independent of the policy: it only depends on the realizations of the r.v.s. Moreover, our policy $\pi$ queries all the r.v.s in $[2k] \supseteq [k+1]$ within $2k$ iterations. So, for all $i \in R$, the r.v.s in $P_i \subseteq [k+1]$ are queried by $\pi$ in $2k$ iterations. Hence, event $\mathcal{A}_2$ (which corresponds to policy $\pi$) contains event $\mathcal{E}$.

Recall that the event $\mathcal{G}_A$ (resp. $\mathcal{G}_A^*$) in policy $\pi$ (resp. $\sigma$) means that every r.v. is more than $L + \delta$ (resp. $\theta$). Also, $\theta \leq L + \delta$, which means

$$\Pr[X_j \geq u | X_j > L + \delta] \geq \Pr[X_j \geq u | X_j > \theta], \quad \forall u \in \mathbb{R}, \forall j \in N.$$

In other words, for any $j \in N$, if $Y_j$ (resp. $Z_j$) is the r.v. $X_j$ conditioned on $\mathcal{G}_A$ (resp. $\mathcal{G}_A^*$) then $Y_j$ *stochastically dominates* $Z_j$.[3] Note also that the r.v.s $Y_j$s (resp. $Z_j$s) are independent. Using the fact that event $\mathcal{E}$ corresponds to a *monotone* function, we obtain:

**Lemma 4.2.7.** *Let $\{Y_j : j \in N\}$ and $\{Z_j : j \in N\}$ be independent r.v.s such that $Y_j$*

[3]We say that r.v. $Y$ stochastically dominates $Z$ if $\Pr[Y \geq u] \geq \Pr[Z \geq u]$ for all $u \in \mathbb{R}$.

*stochastically dominates $Z_j$ for each $j \in N$. Then, $\Pr[\mathcal{E}(Y_1, ..., Y_n)] \geq \Pr[\mathcal{E}(Z_1, ..., Z_n)]$ where event $\mathcal{E}$ is a function of independent r.v.s as defined in (4.20).*

*Proof.* It suffices to prove the following.

$$\Pr[\mathcal{E}(Y_1, ..., Y_k, Z_{k+1}, ..., Z_n)] \geq \Pr[\mathcal{E}(Y_1, ..., Y_{k-1}, Z_k, ..., Z_n)], \quad \forall k \in [n].$$

Note that the r.v.s above only differ at position $k$. To keep notation simple, for any $j \in [n] \setminus k$ let $X'_j = Y_j$ if $j < k$ and $X'_j = Z_j$ if $j > k$. So, we need to show $\Pr[\mathcal{E}(X', Y_k)] \geq \Pr[\mathcal{E}(X', Z_k)]$. We *condition* on the realizations of the $X'$ r.vs. For each $j \in [n] \setminus k$ let $t_j$ denote the realization of the r.v. $X'_j$. Having conditioned on these r.v.s, the only randomness is in $Y_k$ and $Z_k$. We will show:

$$\Pr\left[\mathcal{E}(X', Y_k) | X' = t\right] \geq \Pr\left[\mathcal{E}(X', Z_k) | X' = t\right]. \tag{4.21}$$

Using the definition of the event $\mathcal{E}$ from (4.20), let $R(t) = \{i \in R : k \in P_i \text{ and } t_j > r_i - \delta \text{ for all } j \in P_i \setminus k\}$. In other words, $R(t) \subseteq R$ corresponds to those "clauses" in (4.20) that have not evaluated to true or false based on the realizations $\{X'_j = t_j : j \in [n] \setminus k\}$. If there is some clause in (4.20) that already evaluates to true (based on $t$) then $\mathcal{E}$ holds regardless of $Y_k$ or $Z_k$. So, (4.21) holds in this case (both terms are one). Now, we assume that no clause in (4.20) that already evaluates to true. We can write

$$\{\mathcal{E}(X', Y_k) | X' = t\} \quad = \quad \bigvee_{i \in R(t)} (Y_k \geq r_i - \delta) \quad = \quad \{Y_k \geq f\},$$

where $f = \min_{i \in R(t)} r_i - \delta$ is a deterministic value.[4] Similarly, we have

$$\{\mathcal{E}(X', Z_k) | X' = t\} \quad = \quad \{Z_k \geq f\}.$$

Using the fact that $Y_k$ (resp. $Z_k$) is independent of $X'$ and that $Y_k$ stochastically dominates $Z_k$,

$$\Pr\left[\mathcal{E}(X', Y_k) | X' = t\right] = \Pr[Y_k \geq f] \geq \Pr[Z_k \geq f] = \Pr\left[\mathcal{E}(X', Z_k) | X' = t\right].$$

This completes the proof of (4.21). De-conditioning the $X'$ r.v.s, we obtain $\Pr[\mathcal{E}(X', Y_k)] \geq \Pr[\mathcal{E}(X', Z_k)]$ as desired. □

Using Lemma 4.2.7, we obtain $\Pr[\mathcal{E}|\mathcal{G}_A] \geq \Pr[\mathcal{E}|\mathcal{G}^*_A]$, which proves (4.19). Combined with (4.18),

$$\Pr\left[\mathcal{A}_2 \,|\, \mathcal{G}\right] \;\geq\; \Pr\left[\mathcal{O}_2 \,|\, \mathcal{G}^*\right]. \tag{4.22}$$

[4] If $R(t) = \emptyset$ then we set $f = \infty$.

**Wrapping up.** We have

$$\begin{aligned}
\Pr[\mathcal{A}_1 \vee \mathcal{A}_2] &= \Pr[\mathcal{A}_1] + \Pr[\mathcal{A}_2 \wedge \neg\mathcal{A}_1] = \Pr[\neg\mathcal{G}] + \Pr[\mathcal{A}_2 \wedge \mathcal{G}] \\
&= \Pr[\neg\mathcal{G}] + \Pr[\mathcal{A}_2|\mathcal{G}] \cdot \Pr[\mathcal{G}] = 1 - (1 - \Pr[\mathcal{A}_2|\mathcal{G}]) \cdot \Pr[\mathcal{G}] \\
&\geq 1 - (1 - \Pr[\mathcal{O}_2|\mathcal{G}^*]) \cdot \Pr[\mathcal{G}^*] \qquad \text{by (4.17) and (4.22)} \\
&= \Pr[\neg\mathcal{G}^*] + \Pr[\mathcal{O}_2 \wedge \mathcal{G}^*] \\
&\geq \Pr[\mathcal{O}_1] + \Pr[\mathcal{O}_2 \wedge \neg\mathcal{O}_1] \qquad \text{using } \mathcal{O}_1 \subseteq \neg\mathcal{G}^* \\
&= \Pr[\mathcal{O}_1 \vee \mathcal{O}_2].
\end{aligned}$$

This completes the proof of (4.16) and the theorem.

## 4.3 Algorithms for General Costs

We now consider the SMQ problem with non-uniform query costs. The high-level idea is similar to the unit-cost case: interleaving the two greedy criteria of smallest left-endpoint and highest probability of stopping. However, we need to incorporate the costs carefully. To this end, we use an iterative algorithm that in every iteration $g$, makes a *batch* of queries having total cost about $2^g$. (In order to optimize the approximation ratio, we use a generic base $y$ for the exponential costs.)

For any subset $S \subseteq N$, let $c(S) := \sum_{j \in S} c_j$ denote the cost of querying all intervals in $S$. Again, we renumber intervals so that $\ell_1 \leq \ell_2 \leq \cdots \ell_n$.

**Definition 4.3.1.** *For any $g \geq 0$, let $T_g$ be the maximal prefix of intervals having cost at most $y^g$.*

**Algorithm 4** Double Greedy for General Cost

Let $\ell^* = \min_{j \in N} \ell_j$, $m^* = R := \min_{j \in N} r_j$, and $\pi \leftarrow \emptyset$.
**for** $g = 0, 1, 2, \cdots$ **do** ▷ iteration
  Query intervals $T_g \setminus \pi$ and update list $\pi \leftarrow \pi \circ T_g$.
  Update $\ell^* = \min_{j \in N \setminus \pi}\{\ell_j\}$ and let $\theta_g = \ell^* + \delta$.
  Compute a $(1, 1+\epsilon)$ bicriteria approximate solution $U_g$ for:

$$p_g^* = \min_{T \subseteq N \setminus \pi} \left\{ \Pr\left[\min_{j \in T} X_j > \theta_g\right] : c(T) \leq y^g \right\}. \tag{KP}$$

  Query intervals $U_g$ and update list $\pi \leftarrow \pi \circ U_g$.
  Update $\ell^* = \min_{j \in N \setminus \pi}\{\ell_j\}$ and $m^* = \min\left\{m^*, \min_{j \in T_g \cup U_g} X_j\right\}$.
  **if** $m^* - \ell^* \leq \delta$ **then** stop.

The complete algorithm is given in Algorithm 4. The optimization problem (KP) solved in Step 5 is a variant of the classic knapsack problem: in Theorem 4.3.7 (see Section 4.3.2) we provide a $(1, 1+\epsilon)$ bicriteria approximation algorithm for (KP) for any constant $\epsilon > 0$. In particular, this ensures that $c(U_g) \leq y^g(1+\epsilon)$ and

$$\Pr\left[\min_{j \in U_g} X_j > \theta_g\right] \leq p_g^*.$$

Note that the left-hand-side above equals $\prod_{j \in U_g} \Pr[X_j > \theta_g]$ as all r.v.s are independent.

Furthermore, just like Algorithm 3, we can view Algorithm 4 as first computing the permutation $\pi$ (without querying) and then performing queries in that order until the stopping criterion. So, our algorithm is a non-adaptive policy and our analysis also upper-bounds the adaptivity gap.

### 4.3.1 Analysis

We use $\sigma$ to denote the optimal (adaptive) policy and $\pi$ to denote our non-adaptive policy.

**Definition 4.3.2.** *For any $g \geq 0$, let*

$$o_g := \Pr[\sigma \textit{ does not finish by cost } y^g].$$

*Similarly, for our policy we define*

$$v_g := \Pr[\pi \textit{ does not finish by iteration } g].$$

*We also define $\sigma_g$ to be the optimal policy truncated at cost $y^g$, i.e., the total cost of queried intervals is always at most $y^g$. Similarly, we define $\pi_g$ to be our policy truncated at the end of iteration $g$.*

The key part of the analysis lies in relating the non-stopping probabilities $o_g$ and $a_g$ in the optimal and algorithmic policies: see Lemma 4.3.4. Our first lemma bounds the (worst-case) cost incurred in $g$ iterations of our policy.

**Lemma 4.3.3.** *The cost of our policy until the end of iteration $g$ is*

$$c(\pi_g) \leq (1+\epsilon)\left(1 + \frac{y}{y-1}\right) y^g.$$

*Proof.* We handle separately the costs of intervals queried in Steps 3 and 6. The total cost

incurred in Step 3 of the first $g$ iterations is $c(T_g) \le y^g$: this uses $\cup_{k=0}^{g} T_k = T_g$ because $T_g$ are prefixes. The total cost due to Step 6 can be bounded using a geometric series:

$$\sum_{k=0}^{g} c(U_k) \le (1+\epsilon) \sum_{k=0}^{g} y^k = (1+\epsilon) \cdot \frac{y^{g+1}-1}{y-1}.$$

The inequality above is by the cost guarantee for (KP). The lemma now follows. □

**Lemma 4.3.4.** *For all $g \ge 0$, we have $a_g \le o_g$.*

*Proof.* Recall that $\sigma_g$ denotes the optimal policy truncated at cost $y^g$. We let $L(\sigma_g) = \min_{j \in N \setminus \sigma_g} \{\ell_j\}$ be the smallest un-queried left-endpoint: this is a random value as $\sigma_g$ is adaptive. In the algorithm, consider iteration $g$ and let $L(T_g) = \min_{j \in N \setminus T_g} \{\ell_j\}$; note that the threshold $\theta_g \ge L(T_g) + \delta$ in Step 4. Let $\pi' = \pi_{g-1} \circ T_g$ denote the list after Step 3 in iteration $g$. Note that the optimization in (KP) of iteration $g$ is over $T \subseteq N \setminus \pi'$, which yields $U_g$. Also, $\pi_g = \pi' \cup U_g$.

$$\begin{aligned}
o_g &= \Pr\left[\text{OPT does not finish within cost } y^g\right] \\
&= \Pr\left[\min_{j \in \sigma_g} X_j > L(\sigma_g) + \delta\right] \ge \Pr\left[\min_{j \in \sigma_g} X_j > L(T_g) + \delta\right] && (4.23) \\
&\ge \Pr\left[\min_{j \in \sigma_g} X_j > \theta_g\right] = 1 - \Pr\left[\min_{j \in \sigma_g} X_j \le \theta_g\right] && (4.24) \\
&\ge 1 - \max_{T \subseteq N, c(T) \le y^g} \Pr\left[\min_{j \in T} X_j \le \theta_g\right] = \min_{T \subseteq N, c(T) \le y^g} \Pr\left[\min_{j \in T} X_j > \theta_g\right] && (4.25) \\
&= \min_{T \subseteq N, c(T) \le y^g} \prod_{j \in T} \Pr\left[X_j > \theta_g\right] \ge \prod_{j \in \pi'} \Pr[X_j > \theta_g] \cdot \min_{T \subseteq N \setminus \pi', c(T) \le y^g} \prod_{j \in T} \Pr[X_j > \theta_g] && (4.26) \\
&= \prod_{j \in \pi'} \Pr[X_j > \theta_g] \cdot p_g^* = \Pr\left[\min_{j \in \pi'} X_j > \theta_g\right] \cdot p_g^* && (4.27) \\
&\ge \Pr\left[\min_{j \in \pi'} X_j > \theta_g\right] \cdot \Pr\left[\min_{j \in U_g} X_j > \theta_g\right] = \Pr\left[\min_{j \in \pi_g} X_j > \theta_g\right] \ge v_g && (4.28)
\end{aligned}$$

The equality in (4.23) is given by the definition of $L(\sigma_g)$ and the stopping rule. The inequality in (4.23) uses the fact that $L(\sigma_g) \le L(T_g)$ always, which in turn is because $\sigma_g$ has cost at most $y^g$ and $T_g$ is the maximal prefix within this cost. The inequality in (4.24) uses $\theta_g \ge L(T_g) + \delta$. The inequality in (4.25) is by Proposition 4.1.2: we view $\sigma_g$ as a feasible adaptive policy for the fixed-threshold problem with threshold $\theta_g$ and budget $y^g$. The equality in (4.26) follows from independence of the random variables. The first equality in (4.27) uses the definition of $p_g^*$ from (KP) and independence. The first inequality in (4.28) uses the choice of $U_g$ and Theorem 4.3.7. The equality in (4.28) is by $\pi_g = \pi' \cup U_g$. To see the last inequality in (4.28),

note that if $\min_{j\in\pi_i}\{X_j\} \le \theta_g$ then $\pi$ finishes by iteration $g$. □

In Lemma 4.3.5 we lower bound the expected cost of the optimal policy. Let $c_{exp}(\pi)$ and $c_{exp}(\sigma)$ denote the expected cost of our greedy policy and the optimal policy, respectively.

**Lemma 4.3.5.** *For any base $y \ge 1$, we have $\sum_{g\ge 0} y^g \cdot o_g \le \frac{y}{y-1} c_{exp}(\sigma) - \frac{1}{y-1}$.*

*Proof.* Let $Z$ denote the random variable that represents the cost of the optimal policy $\sigma$: so $c_{exp}(\sigma) = \mathbb{E}[Z]$. Let $\mathbf{1}(Z > y^g)$ be the indicator variable for when $Z > y^g$; so $\mathbb{E}[I(Z > y^g)] = o_g$. We now show that:

$$\sum_{g\ge 0} y^g \cdot \mathbf{1}(Z > y^g) \le \frac{y}{y-1} Z - \frac{1}{y-1} \tag{4.29}$$

To see this, suppose that $y^k < Z \le y^{k+1}$ for some integer $k \ge 0$. Then the left-hand-side of (4.29) equals

$$\sum_{g=0}^{k} y^g = \frac{y^{k+1}-1}{y-1} \le Z\frac{y}{y-1} - \frac{1}{y-1},$$

which proves (4.29). Taking the expectation of (4.29) proves the lemma. □

**Theorem 4.3.6.** *There is a $(3+2\sqrt{2}+\epsilon)$-approximation for the* SMQ *problem with general costs.*

*Proof.* By Lemma 4.3.3, we have $c_{exp}(\pi) \le (1+\epsilon)\left(1+\frac{y}{y-1}\right)\sum_{g\ge 1} y^g(v_{g-1}-v_g)$. Now,

$$\sum_{g\ge 1} y^g(v_{g-1}-v_g) = v_0 + (y-1)\sum_{g\ge 1} y^g v_g \le 1 + (y-1)\sum_{g\ge 1} y^g o_g \tag{4.30}$$

$$\le 1 + (y-1)\sum_{g\ge 0} y^g o_g \le 1 + y\cdot c_{exp}(\sigma) - 1 = y\cdot c_{exp}(\sigma) \tag{4.31}$$

The inequality in (4.30) is by Lemma 4.3.3 and $v_0 = 1$. The first inequality in (4.31) uses $y \ge 1$ and the second inequality is by Lemma 4.3.5.

Hence, we obtain $c_{exp}(\pi) \le (1+\epsilon)y\cdot\left(1+\frac{y}{y-1}\right)\cdot c_{exp}(\sigma)$. Now, optimizing for $y$, we obtain the stated approximation ratio. □

### 4.3.2 The knapsack subroutine (KP)

We now provide a bi-criteria approximation algorithm for the knapsack instance (KP).

**Theorem 4.3.7.** *Given discrete random variables $\{X_i\}_{i=1}^n$ with costs $\{c_i\}_{i=1}^n$, budget $d$*

*and threshold* $\theta \in \mathbb{R}$*, there is an* $n^{O(1/\epsilon)}$ *time algorithm that finds* $T \subseteq N$ *such that* $\Pr\left[\min_{j \in T} X_j \geq \theta\right] \leq p^*$ *and* $c(T) \leq (1+\epsilon)d$*, for any* $\epsilon > 0$*. Here,*

$$p^* = \min_{T \subseteq N} \left\{ \Pr\left[\min_{j \in T} X_j > \theta\right] : c(T) \leq d \right\}. \tag{*}$$

*Proof.* First, we re-write (*) as a weighted knapsack problem, using

$$\Pr\left[\min_{j \in T} X_j > \theta\right] = \prod_{j \in T} \underbrace{\Pr[X_j > \theta]}_{q_j}.$$

Then, we take an inverse (which converts the min objective to max) and the logarithm (which makes the objective linear).

$$\log(\frac{1}{p^*}) = \max_{T \subseteq N} \left\{ \sum_{j \in T} \log\left(\frac{1}{q_j}\right) : c(T) \leq d \right\}.$$

Let $r_j := \log(1/q_j) \geq 0$ be the "reward" of each item. Then the above problem is just the usual knapsack problem. We now provide a bicriteria approximation algorithm using standard enumeration techniques combined with a greedy algorithm. (We provide the full proof because we did not find a reference to the precise bi-criteria guarantee that is needed here.)

**Algorithm: Bicriteria Knapsack**

1. Order items greedily such that $\frac{r_1}{c_1} \geq \frac{r_2}{c_2} \geq ... \geq \frac{r_n}{c_n}$.
2. Let $B = \{j \in [n] : c_j > \epsilon d\}$ be the set of "large" items.
3. For each $S \subseteq B$ with $c(S) \leq d$:
   (a) Initialize solution $T_S \leftarrow S$.
   (b) Add items from $[n] \setminus B$ to $T_S$ in the greedy order until $c(T_S)$ exceeds $d$ for the first time.
4. Return the best solution $T_S$ obtained above.

We first show that the runtime is $n^{O(1/\epsilon)}$. The key observation is that the number of distinct subsets $S \subseteq B$ considered in Step 3 is at most $n^{1/\epsilon}$: this is because each item in $B$ has cost more that $\epsilon d$.

Let $T$ be the solution found at the end of the algorithm. It is clear that $d \le c(T) \le d + c_{max}$ where $c_{max} = \max_{i \in [n] \setminus B} c_i$. Note that $c_{max} \le \epsilon d$ by definition of the large items $B$. So, $c(T) \le (1+\epsilon)d$.

We now bound the total "reward" $\sum_{i \in T} r_i$. We will use the following well-known fact about the greedy algorithm for knapsack.

> Consider any knapsack instance with items $I$, rewards $\{r_i\}_{i \in I}$, costs $\{c_i\}_{i \in I}$ and budget $d'$. Let $G \subseteq I$ be the "greedy" solution obtained by including items in decreasing order of the ratio $\frac{r_i}{c_i}$ until $c(G) > d'$ for the first time. Then $r(G) \ge \max_{K \subseteq I : c(K) \le d'} \sum_{i \in K} r_i$.

Let $T^*$ be the optimal solution to (*). Then, $S = T^* \cap B$ will be one of the choices for $S$ considered in Step 3. Moreover, the set $T_S \setminus S$ added in Step 3b is precisely the greedy solution for the knapsack instance with items $I = [n] \setminus B$ and budget $d' = d - c(S) = d - c(T^* \cap B)$. The above fact implies that $r(T_S \setminus S) \ge r(T^* \setminus B)$ because $T^* \setminus B$ is a feasible solution to this knapsack instance. It follows that $r(T_S) = r(S) + r(T_S \setminus S) \ge r(T^*) = \log(\frac{1}{p^*})$. □

## 4.4 SMQI under Non-uniform Costs

We now consider the (harder) problem of identifying a $\delta$-minimum interval. Recall that the goal here is to identify some interval $i^* \in N$ such that $X_{i^*} \le \mathsf{MIN} + \delta$ where $\mathsf{MIN} = \min_{i \in N} X_i$. The interval $i^*$ may not have been queried by the policy. Unlike the unit-cost case, we can no longer rely on the SMQ algorithm itself (see the example below).

**Bad example for the SMQ policy.** Consider an instance with the following r.v.s.

- $X_1$ is distributed over the interval $[0, 1.5\,\delta]$. (The exact distribution is irrelevant.)
- $X_2$ has $\Pr[X_2 = 0.3\,\delta] = \frac{1}{n^2}$ and $\Pr[X_2 = 2\,\delta] = 1 - \frac{1}{n^2}$.
- $X_3, \cdots X_n$ are identically distributed with $\Pr[X_i = 0.7\,\delta] = \frac{1}{n}$ and $\Pr[X_i = 1.5\,\delta] = 1 - \frac{1}{n}$.

The cost $c_1 = n \gg 1$ and all other costs are unit. The SMQ policy from Algorithm 4 will first select at least $\Omega(n)$ r.v.s among $X_3, \cdots X_n$ because these will optimize (KP). Crucially, the policy will not query $X_1$ or $X_2$ for a long time. Consequently, the expected cost of this policy is $\Omega(n)$. On the other hand, an optimal policy just queries $X_2$: if $X_2 = 2\,\delta$ then it returns $i^* = 1$ (stopping rule (4.15) applies); if $X_2 = 0.3\,\delta$ then it returns $i^* = 2$ (stopping

rule (4.14) applies). So the SMQ algorithm has an $\Omega(n)$ approximation ratio when applied directly to SMQI.

This example shows that in the presence of non-uniform costs, additional work is needed to handle the new stopping criterion (4.15). In particular, we need to skip expensive intervals while querying in the order of left-endpoints. The SMQI algorithm has the same high-level structure as the one for SMQ (Algorithm 4). The only change is in Step 3 where we modify the queried set $T_g$ by skipping some expensive intervals, as formalized next.

**Definition 4.4.1.** *For any iteration $g \geq 0$, an interval $j \in N$ is called* **$g$-big** *if its cost $c_j > y^g$ (otherwise, $j$ is called $g$-small).*

Recall that the intervals are numbered according to their left-endpoint, i.e., $\ell_1 \leq \ell_2 \leq \cdots \ell_n$.

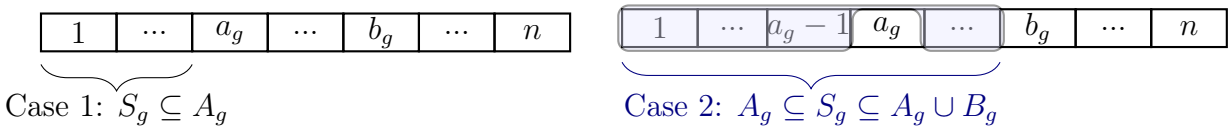
**Definition 4.4.2.** *Consider any iteration $g \geq 0$.*

- *Let $a_g$ and $b_g$ denote the first and second $g$-big intervals, respectively.*
- *$A_g := \{1, 2, \cdots a_g - 1\}$ is the maximal prefix of $[n]$ that does not contain any $g$-big interval.*
- *$B_g := \{a_g + 1, \cdots b_g - 1\}$ is the segment of $[n]$ between the first and second $g$-big intervals.*

*We now define the almost-prefix query set $S_g$ as follows:*

1. *If $c(A_g) > y^g$ then $S_g$ is the maximal prefix of $A_g$ having cost at most $2y^g$. Here, $S_g \subseteq A_g$.*
2. *If $c(A_g) \leq y^g$ then $S_g$ is the maximal prefix of $A_g \cup B_g$ having cost at most $y^g$. Here, $S_g \supseteq A_g$.*

Figure 4.7: Illustration of Definition 4.4.2.



**SMQI algorithm.** This involves replacing the "prefix set" $T_g$ in Step 3 of the SMQ algorithm (Algorithm 4) by the almost-prefix set $S_g$ defined above. The other steps remain the same as in Algorithm 4. The stopping criterion also changes: we will perform queries in the order of $\pi$ until either (4.14) or (4.15) applies. We start with a useful lemma showing that the almost-prefix sets $S_g$ are nested (as was the case for the sets $T_g$). We note that this is just needed to obtain a tighter constant factor.

**Lemma 4.4.3.** *For each $g \geq 0$, we have $S_g \subseteq S_{g+1}$.*

*Proof.* First, suppose that $c(A_g) > y^g$ (corresponds to case 1 in Definition 4.4.2). Then, set $S_g \subseteq A_g$ is a prefix of cost at most $2y^g$. Clearly, the first $(g+1)$-big interval $a_{g+1} \geq a_g$, so $A_g \subseteq A_{g+1}$.

- If $c(A_{g+1}) \leq y^{g+1}$ then $S_{g+1} \supseteq A_{g+1} \supseteq A_g \supseteq S_g$.
- If $c(A_{g+1}) > y^{g+1}$ then $S_{g+1}$ is the maximal prefix of $A_{g+1} \supseteq A_g$ of cost at most $2y^{g+1} > 2y^g$: so we must have $S_g \subseteq S_{g+1}$.

Now, suppose that $c(A_g) \leq y^g$ (corresponds to case 2 in Definition 4.4.2). Here, $S_g \supseteq A_g$ and is a prefix of $A_g \cup B_g$ with $c(S_g) \leq y^g$. If $a_g$ is also $(g+1)$-big then $A_{g+1} = A_g$ and $c(A_{g+1}) \leq y^{g+1}$: so $S_{g+1}$ corresponds to case 2 in Definition 4.4.2. Also, the second $(g+1)$-big interval $b_{g+1} \geq b_g$: so, $B_{g+1} \supseteq B_g$. Hence, $S_{g+1}$ is the maximal prefix of $A_g \cup B_{g+1} \supseteq A_g \cup B_g$ of cost at most $y^{g+1}$. So, we must have $S_{g+1} \supseteq S_g$. If $a_g$ is not $(g+1)$-big then we have $y^g < c(a_g) \leq y^{g+1}$ and the first $(g+1)$-big interval $a_{g+1} \geq b_g$, i.e., $A_{g+1} \supseteq A_g \cup \{a_g\} \cup B_g$.

- If $c(A_{g+1}) > y^{g+1}$ then $S_{g+1}$ corresponds to case 2 in Definition 4.4.2. Consider the prefix $\{a_g\} \cup S_g$: it has cost at most $y^{g+1} + y^g < 2\,y^{g+1}$. So, we must have $S_{g+1} \supseteq \{a_g\} \cup S_g$.
- If $c(A_{g+1}) \leq y^{g+1}$ then $S_{g+1}$ corresponds to case 2 in Definition 4.4.2. Here, $S_{g+1} \supseteq A_{g+1} \supseteq A_g \cup \{a_g\} \cup B_g \supseteq S_g$.

In all cases, we have $S_g \subseteq S_{g+1}$. □

The rest of the analysis combines ideas from the non-uniform cost SMQ and the uniform cost SMQI. Let $\pi$ denote our (non-adaptive) policy and $\sigma$ the optimal adaptive policy. We re-use the terms from Definition 4.3.2:

$$o_g := \Pr[\sigma \text{ does not finish by cost } y^g].$$

$$v_g := \Pr[\pi \text{ does not finish by iteration } g].$$

$\sigma_g$ is the optimal policy truncated at cost $y^g$

$\pi_g$ is our policy truncated at the end of iteration $g$.

**Lemma 4.4.4.** *The cost of our policy until the end of iteration $g$ is*

$$c(\pi_g) \leq (1+\epsilon)\left(2+\frac{y}{y-1}\right) y^g.$$

*Proof.* We handle separately the costs of intervals queried in the (modified) Step 3 and Step 6 of Algorithm 4. By Lemma 4.4.3, we have $\cup_{k=0}^{g} S_k = S_g$. So, the total cost incurred in the modified Step 3 of the first $g$ iterations is $c(S_g) \leq 2 \cdot y^g$. The total cost due to Step 6 is exactly as in Lemma 4.3.3, which is at most $(1+\epsilon) \cdot \dfrac{y^{g+1}}{y-1}$. The lemma now follows. □

Lemma 4.3.5 continues to hold here as well; so:

$$\sum_{g\geq 0} y^g o_g \leq \frac{y}{y-1} c_{exp}(\sigma) - \frac{1}{y-1}. \tag{4.32}$$

The key step is the analogue of Lemma 4.3.4, which we prove in the next subsection.

**Lemma 4.4.5.** *For all $g \geq 0$, we have $v_g \leq o_g$.*

We can now prove the main result.

**Theorem 4.4.6.** *There is a $(4+2\sqrt{3}+\epsilon)$-approximation for* SMQI *with general costs.*

*Proof.* By Lemma 4.4.4, we have $c_{exp}(\pi) \leq (1+\epsilon)\left(2+\frac{y}{y-1}\right)\sum_{g\geq 1} y^g(v_{g-1}-v_g)$. Exactly as in the proof of Theorem 4.3.6, using (4.32), we get $\sum_{g\geq 1} y^g(v_{g-1}-v_g) \leq y \cdot c_{exp}(\sigma)$. Therefore, $c_{exp}(\pi) \leq (1+\epsilon)y \cdot \left(2+\frac{y}{y-1}\right) \cdot c_{exp}(\sigma)$. Now, optimizing for $y$, we obtain the stated approximation ratio. □

### 4.4.1 Proof of Lemma 4.4.5

Fix any iteration $g$. As in the unit-cost SMQI proof, we define the following events.

$\mathcal{A}_1$ : our policy $\pi$ finishes within $g$ iterations due to (4.14).

$\mathcal{A}_2$ : our policy $\pi$ finishes within $g$ iterations due to (4.15).

$\mathcal{O}_1$ : optimal policy $\sigma$ finishes by cost $y^g$ due to (4.14).

$\mathcal{O}_2$ : optimal policy $\sigma$ finishes by cost $y^g$ due to (4.15).

Clearly, $1-v_g = \Pr[\mathcal{A}_1 \vee \mathcal{A}_2]$ and $1-o_g = \Pr[\mathcal{O}_1 \vee \mathcal{O}_2]$.

**Handling the old stopping criterion.** Let $L$ denote the smallest un-queried left-endpoint at the end of iteration $g$ in $\pi$. Note that $L$ is a deterministic value. Let $\mathcal{G}$ be the event that $X_j > L + \delta$ for all intervals $j$ queried by $\pi_g$. Note that $\mathcal{G} = \neg\mathcal{A}_1$, i.e., criterion (4.14) does *not* apply by the end of iteration $g$. Recall that threshold $\theta_g$ (Step 4 in Algorithm 4) is $\delta$ more that the smallest un-queried left-endpoint in that step. Clearly, $\theta_g \leq L + \delta$ in iteration $g$.

Now consider the truncated optimal policy. Let $L(\sigma_g)$ be its smallest un-queried left-endpoint; this is a random value as $\sigma_g$ is an adaptive policy. We claim that

$$L + \delta \;\geq\; \theta_g \;\geq\; \min_{j \in N \setminus S_g} \ell_j + \delta \;\geq\; L(\sigma_g) + \delta. \tag{4.33}$$

Above, the second inequality uses the fact that $S_g$ is queried before Step 4. To see the last inequality, note that $\sigma_g$ cannot query any $g$-big interval: so $L(\sigma_g) \leq \ell_{a_g}$. We have two cases depending on the definition of $S_g$:

- If $S_g \supseteq A_g$ then clearly $\min_{j \in N \setminus S_g} \ell_j \geq \ell_{a_g} \geq L(\sigma_g)$.
- If $S_g \subseteq A_g$ then we must have $c(A_g) > y^g$, which means that $S_g$ contains the maximal prefix of cost at most $y^g$. Again, this implies $\min_{j \in N \setminus S_g} \ell_j \geq L(\sigma_g)$.

This proves (4.33).

Now, let $\mathcal{G}^*$ be the that event that $X_j > \theta_g$ for all intervals $j$ in $\sigma_g$. Using (4.33) it follows that $\mathcal{O}_1 \subseteq \neg\mathcal{G}^*$. We now obtain:

$$\Pr[\mathcal{G}^*] = \Pr\left[\min_{j \in \sigma_g} X_j > \theta_g\right] \geq \Pr\left[\min_{j \in \pi_g} X_j > \theta_g\right] \geq \Pr\left[\min_{j \in \pi_g} X_j > L + \delta\right] = \Pr[\mathcal{G}]. \tag{4.34}$$

The first inequality follows from the proof of Lemma 4.3.4: see (4.24) - (4.28). The second inequality above uses $\theta_g \leq L + \delta$.

**Handling the new stopping criterion.** Let $\mathcal{G}_A$ be the event that $X_j > L + \delta$ for *all* r.v.s $j \in N$. Similarly, let $\mathcal{G}_A^*$ be the event that $X_j > \theta_g$ for *all* $j \in N$. Clearly, $\Pr[\mathcal{A}_2 \,|\, \mathcal{G}] = \Pr[\mathcal{A}_2 \,|\, \mathcal{G}_A]$ and $\Pr[\mathcal{O}_2 \,|\, \mathcal{G}^*] = \Pr[\mathcal{O}_2 \,|\, \mathcal{G}_A^*]$. We will now prove that

$$\Pr[\mathcal{A}_2 \,|\, \mathcal{G}] \;=\; \Pr[\mathcal{A}_2 \,|\, \mathcal{G}_A] \;\geq\; \Pr[\mathcal{O}_2 \,|\, \mathcal{G}_A^*] \;=\; \Pr[\mathcal{O}_2 \,|\, \mathcal{G}^*]. \tag{4.35}$$

Using (4.34), exactly as in the proof of Theorem 4.2.6, this implies $\Pr[\mathcal{A}_1 \vee \mathcal{A}_2] \geq \Pr[\mathcal{O}_1 \vee \mathcal{O}_2]$. We repeat the argument below for completeness.

$$\begin{aligned}
\Pr[\mathcal{A}_1 \vee \mathcal{A}_2] &= \Pr[\mathcal{A}_1] + \Pr[\mathcal{A}_2 \wedge \neg\mathcal{A}_1] = \Pr[\neg\mathcal{G}] + \Pr[\mathcal{A}_2 \wedge \mathcal{G}] \\
&= \Pr[\neg\mathcal{G}] + \Pr[\mathcal{A}_2|\mathcal{G}] \cdot \Pr[\mathcal{G}] = 1 - (1 - \Pr[\mathcal{A}_2|\mathcal{G}]) \cdot \Pr[\mathcal{G}] \\
&\geq 1 - (1 - \Pr[\mathcal{O}_2|\mathcal{G}^*]) \cdot \Pr[\mathcal{G}^*] \qquad \text{by (4.34) and (4.35)} \\
&= \Pr[\neg\mathcal{G}^*] + \Pr[\mathcal{O}_2 \wedge \mathcal{G}^*] \\
&\geq \Pr[\mathcal{O}_1] + \Pr[\mathcal{O}_2 \wedge \neg\mathcal{O}_1] \qquad \text{using } \mathcal{O}_1 \subseteq \neg\mathcal{G}^* \\
&= \Pr[\mathcal{O}_1 \vee \mathcal{O}_2].
\end{aligned}$$

This proves Lemma 4.4.5.

**Proving (4.35).** The key property here is the following.

**Lemma 4.4.7.** *If $\sigma$ finishes due to criterion (4.15) and identifies $i^*$ by cost $y^g$ then $P_{i^*} \subseteq S_g$.*

*Proof.* Clearly $c(P_{i^*}) \leq y^g$ as $\sigma$ finishes by cost $y^g$. Recall that $P_{i^*} = \{j \in N \setminus i^* : \ell_j < r_{i^*} - \delta\}$ is an almost-prefix set. We consider two cases:

- $P_{i^*}$ is itself a prefix. In this case, the first $g$-big interval $a_g$ must occur after $P_{i^*}$, i.e., $P_{i^*} \subseteq A_g$. By the definition of $S_g$, it either contains $A_g \supseteq P_{i^*}$ or $S_g$ is a maximal prefix of cost at most $2y^g$ (which also contains $P_{i^*}$).

- $P_{i^*}$ is not a prefix, but $P_{i^*} \cup i^*$ is a prefix.

  If $i^*$ is not $g$-big then $P_{i^*} \cup i^*$ has cost at most $2y^g$ and is a subset of $A_g$ (as $P_{i^*} \cup i^*$ cannot contain any $g$-big interval). So $P_{i^*} \cup i^*$ is contained in the maximal prefix of $A_g$ having cost at most $2y^g$, which is always contained in $S_g$.

  If $i^*$ is $g$-big then $i^* = a_g$ the first $g$-big interval (otherwise $P_{i^*}$ would contain some $g$-big interval). This also means that $c(A_g) \leq y^g$: so $S_g$ is the maximal prefix in $[n] \setminus \{a_g\}$ of cost at most $y^g$. Clearly, we must then have $P_{i^*} \subseteq S_g$.

□

Let $R = \{h \in N : P_h \subseteq S_g\}$. By Lemma 4.4.7 it follows that if event $\mathcal{O}_2$ occurs then $i^* \in R$. Hence, $\mathcal{O}_2$ is a subset of the event

$$\mathcal{E} := \bigvee_{h \in R} \left(\wedge_{j \in P_h}(X_j > r_h - \delta)\right).$$

Moreover, our policy $\pi_g$ queries all the r.v.s in $S_g$. So, for all $h \in R$, the r.v.s in $P_h \subseteq S_g$ are queried by $\pi_g$. Hence, event $\mathcal{A}_2$ contains event $\mathcal{E}$.

Recall that the event $\mathcal{G}_A$ (resp. $\mathcal{G}_A^*$) in policy $\pi$ (resp. $\sigma$) means that every r.v. is more than $L + \delta$ (resp. $\theta$). Also, $\theta \leq L + \delta$, which means

$$\Pr[X_j > t | X_j > L + \delta] \geq \Pr[X_j > t | X_j > \theta], \quad \forall t \in \mathbb{R}, \forall j \in N.$$

In other words, for any $j \in N$, r.v. $X_j$ conditioned on $\mathcal{G}_A$ stochastically dominates $X_j$ conditioned on $\mathcal{G}_A^*$. Using Lemma 4.2.7 (which deals with the same event $\mathcal{E}$) with $Y_j = X_j | \mathcal{G}_A$ and $Z_j = X_j | \mathcal{G}_A^*$, we obtain $\Pr[\mathcal{E}|\mathcal{G}_A] \geq \Pr[\mathcal{E}|\mathcal{G}_A^*]$, which proves (4.35).

## 4.5 Experiments

The goal of our experiments is to compare the expected cost of our greedy policy to the expected cost of the optimal policy. In subsection 4.5.1 we describe how we compute the cost of the optimal policy using a dynamic program. Since we compute the optimal policy for each instance, this limits the size of instances we can consider. Here we create instances of size $5, 10$ and $15$.

### 4.5.1 Computing the optimal policy

We develop a dynamic programming algorithm that solves the SMQ problem exactly in exponential time. The dynamic program (DP) allows us to compare the expected cost of the optimal policy with the expected cost of our policy for general costs and our policy for unit costs. This algorithm maintains a DP table that stores $O(|M| \cdot 2^n)$ values, where $M$ is the set of discrete values over all intervals. The table $V$ is indexed by the set of probed indices $S \subseteq N$, and the current minimum value observed $m$, and then $V(S, m)$ is the minimum expected cost to terminate conditioned on having probed $S$ and observed $m = \min_{i \in S} X_i$. The recurrence for $V(S, m)$ is given by :

$$\begin{aligned} V(S, m) &= \min_{j \in N \setminus S} \{\mathbb{E}_{X_j}[V(S \cup j, \min(m, X_j))] + c_j\} \\ &= \min_{j \in N \setminus S} \{c_j + \sum_{w \in M} \Pr[X_j = w] \cdot V(S \cup j, \min\{m, w\})\} \end{aligned}$$

The base case is then $V(S, m^*) = 0$ whenever $m^* \leq \min_{j \in N \setminus S} \ell_j + \delta$. The expected cost of the optimal policy is given by $V(\emptyset, \min_{j \in N} r_j)$.

### 4.5.2 Instance generation

We create a synthetic data set of $n$ intervals for each instance. We consider instance sizes equal to $\{5, 10, 15\}$ intervals since we are limited by the time it takes to find the optimal policy on larger instances. The average time taken for the optimal policy to terminate in instances of size $n = 15$ and $|M| = 150$ is a little over 2 minutes. The synthetic dataset was constructed such that for each instance, we generate a set of $n$ intervals. We fix the number of discrete points per interval to be ten, set $\delta$ to 0.1 and for general cost instances, $c_j$ is a uniformly selected integer between 1 and 5.

Each instance can be characterized by the degree of overlap in the intervals (sparse vs. dense) and the distribution over random discrete values (uniform vs. normal). Then the four types of instances we generate are all possible combinations of the two defining characteristics. Intervals in sparse instances are ensured to have only a meaningful amount of overlap i.e. the left-endpoint $\ell_i$ of each interval is within at most $\delta$ of the next. Additionally, we were interested in dense instances where nearly all left-endpoints are within at most $\delta$ of each other. Furthermore, we consider instances with a uniform distribution over all intervals and instances with discretized normal distribution over all intervals.

**Sparse and dense instances.** For all instances we set the first left-endpoint $\ell_1 = 0$. For a sparse instance, a left-endpoint $\ell_i$ is drawn from a uniform distribution over $[\ell_{i-1}, \ell_{i-1} + \delta]$. For dense instances we draw the left-endpoints $\ell_i$ from a uniform distribution over $[\ell_{i-1}, \ell_{i-1} + \frac{\delta}{n/2}]$. This increases the degree of overlap in left-endpoints in dense instances. For all instances (sparse and dense) a right-endpoint $r_j$ is drawn from a uniform distribution over $[\ell_j + 2, \ell + 10]$. Finally, each discrete point for all intervals over all instances is drawn from a uniform distribution over $[\ell_j, r_j]$.

**Uniform and normal instances.** For instances where all intervals have a uniform distribution, all values on the interval are assigned equal probability. For instances with a normal distribution, the distribution over each interval $i$ is now a discretized normal distribution with mean $\frac{\ell_i + u_i}{2}$ and standard deviation 1.

### 4.5.3 Computational results

We run Algorithm 3, 4 and the optimal policy on the synthetic instances described in Section 4.5.2. The experiments were conducted on a 2020 MacBook Pro equipped with an Apple M1 chip, featuring an 8-core CPU, 8-core GPU, 16-core Neural Engine, and 8GB of memory, running macOS Ventura version 13.4. In Tables 4.1, 4.2, 4.3 and 4.4 we present a summary

of our experiments, where each row in each table shows the average ratio over twenty instances. Each table has a column that describes the type of instance (sparse vs. dense) and displays the ratio of the cost of our policy to the cost of the optimal policy. Tables 4.1 and 4.2 display the results for sparse and dense instances with a uniform distribution over the interval and Tables 4.3 and 4.4 display the results for sparse and dense instances generated with a discretized normal distribution over the interval. The full set of results that make up the averages in the summary tables are displayed in Figures 4.8, 4.9, 4.10, 4.11, 4.12, 4.13, 4.14, and 4.15.

In the detailed tables we display the expected cost of the optimal policy as **E[OPT]**, the time taken (in seconds) by the dynamic program to calculate the expected cost of the optimal policy, the expected cost of our policy **E[GRD]**, and the ratio of **E[GRD]** to **E[OPT]**. The expected cost of our policy is calculated using the permutation ($\pi$) of intervals given by our policy without using the stopping rule. This permutation is the same for each set of realizations because our policies are non-adaptive. Then at each iteration of our policy $i$, we calculate the probability of querying an interval in position $i$ of the permutation to get the expected cost of our policy. Formally, if we let $\pi_i$ be the permutation truncated at iteration $i$ we have:

$$\textbf{E[GRD]} = \sum_{i \in [n]} i \cdot \Pr\left[\min_{j \in \pi_{i-1}} X_j > \min_{j \in N \setminus \pi_{i-1}} \ell_j + \delta\right].$$

We observe that our policies performed better when the instances had a uniform distribution. Furthermore, we note that for unit cost instances the ratio is higher for dense instances than it is for sparse instances, this is possibly because probing the intervals with the smallest left-endpoint in a dense instance will change the stopping threshold by an amount that is at most $\delta = 0.1$. Our policy will probe such an interval in each iteration, while the optimal policy is more likely to make more probes to intervals with high likelihoods of stopping earlier on in its sequence of probed intervals.

Tables 4.1 and 4.2: Instances with uniform distribution.

Table 4.1: Unit cost policy ratio for instances with a uniform distribution

| n | Density | Ratio |
|---|---|---|
| 15 | sparse | 1.003 |
| 10 | sparse | 1.003 |
| 5 | sparse | 1.002 |
| 15 | dense | 1.023 |
| 10 | dense | 1.008 |
| 5 | dense | 1.005 |

Table 4.2: General cost policy ratio for instances with a uniform distribution

| n | Density | Ratio |
|---|---|---|
| 15 | sparse | 1.053 |
| 10 | sparse | 1.063 |
| 5 | sparse | 1.047 |
| 15 | dense | 1.055 |
| 10 | dense | 1.034 |
| 5 | dense | 1.010 |

Tables 4.3 and 4.4: Instances with a discretized normal distribution.

Table 4.3: Unit cost policy ratio for instances with a normal distribution

| n | Density | Ratio |
|---|---|---|
| 15 | sparse | 1.013 |
| 10 | sparse | 1.013 |
| 5 | sparse | 1.011 |
| 15 | dense | 1.040 |
| 10 | dense | 1.028 |
| 5 | dense | 1.017 |

Table 4.4: General cost policy ratio instances with a normal distribution

| n | Density | Ratio |
|---|---|---|
| 15 | sparse | 1.018 |
| 10 | sparse | 1.023 |
| 5 | sparse | 1.013 |
| 15 | dense | 1.031 |
| 10 | dense | 1.032 |
| 5 | dense | 1.018 |

| E[OPT] | Time | E[GRD] | Ratio |
|---|---|---|---|
| 7.122 | 209.389 | 7.222 | 1.014 |
| 5.574 | 152.942 | 6.265 | 1.124 |
| 6.531 | 166.078 | 6.783 | 1.039 |
| 7.170 | 155.244 | 7.270 | 1.014 |
| 6.770 | 225.758 | 6.828 | 1.009 |
| 6.336 | 172.504 | 6.436 | 1.016 |
| 6.627 | 129.722 | 6.652 | 1.004 |
| 7.170 | 203.062 | 7.270 | 1.014 |
| 6.232 | 203.332 | 6.509 | 1.044 |
| 6.948 | 162.730 | 7.048 | 1.014 |
| 5.972 | 229.473 | 6.036 | 1.011 |
| 5.878 | 150.765 | 5.878 | 1.000 |
| 5.687 | 213.785 | 5.985 | 1.052 |
| 7.170 | 163.293 | 7.270 | 1.014 |
| 7.250 | 202.615 | 7.290 | 1.006 |
| 7.459 | 226.549 | 7.459 | 1.000 |
| 5.752 | 174.204 | 5.798 | 1.008 |
| 5.871 | 231.711 | 5.907 | 1.006 |
| 6.338 | 232.277 | 6.590 | 1.040 |
| 5.409 | 236.218 | 5.625 | 1.040 |
| | | AVG ratio: | 1.023 |

(a) Table 1: $n = 15$

| E[OPT] | Time | E[GRD] | Ratio |
|---|---|---|---|
| 5.901 | 2.925 | 6.001 | 1.017 |
| 5.437 | 2.054 | 5.537 | 1.018 |
| 6.091 | 2.295 | 6.172 | 1.013 |
| 5.901 | 1.621 | 6.001 | 1.017 |
| 5.720 | 3.460 | 5.773 | 1.009 |
| 5.697 | 2.415 | 5.697 | 1.000 |
| 5.739 | 1.708 | 5.739 | 1.000 |
| 6.513 | 2.536 | 6.513 | 1.000 |
| 5.385 | 2.645 | 5.485 | 1.019 |
| 6.369 | 1.922 | 6.369 | 1.000 |
| 5.081 | 3.173 | 5.145 | 1.013 |
| 5.255 | 2.067 | 5.255 | 1.000 |
| 5.153 | 2.375 | 5.318 | 1.032 |
| 6.513 | 2.474 | 6.513 | 1.000 |
| 6.206 | 2.555 | 6.206 | 1.000 |
| 6.182 | 3.304 | 6.182 | 1.000 |
| 5.267 | 2.894 | 5.267 | 1.000 |
| 5.047 | 3.951 | 5.047 | 1.000 |
| 5.739 | 3.117 | 5.839 | 1.017 |
| 4.770 | 3.079 | 4.770 | 1.000 |
| | | AVG ratio: | 1.008 |

(b) Table 2: $n = 10$

| E[OPT] | Time | E[GRD] | Ratio |
|---|---|---|---|
| 3.751 | 0.025 | 3.851 | 1.027 |
| 3.751 | 0.026 | 3.851 | 1.027 |
| 4.095 | 0.027 | 4.095 | 1.000 |
| 3.751 | 0.018 | 3.851 | 1.027 |
| 3.751 | 0.035 | 3.751 | 1.000 |
| 3.941 | 0.041 | 3.941 | 1.000 |
| 3.751 | 0.019 | 3.751 | 1.000 |
| 4.095 | 0.029 | 4.095 | 1.000 |
| 3.812 | 0.027 | 3.812 | 1.000 |
| 4.095 | 0.022 | 4.095 | 1.000 |
| 3.534 | 0.030 | 3.534 | 1.000 |
| 3.557 | 0.023 | 3.557 | 1.000 |
| 3.778 | 0.023 | 3.778 | 1.000 |
| 4.095 | 0.029 | 4.095 | 1.000 |
| 4.022 | 0.034 | 4.022 | 1.000 |
| 4.022 | 0.037 | 4.022 | 1.000 |
| 3.751 | 0.035 | 3.751 | 1.000 |
| 3.413 | 0.036 | 3.413 | 1.000 |
| 3.751 | 0.029 | 3.851 | 1.027 |
| 3.477 | 0.031 | 3.477 | 1.000 |
| | | AVG ratio: | 1.005 |

(c) Table 3: $n = 5$

Figure 4.8: Dense instances with unit cost and uniform distribution over all intervals

| E[OPT] | Time | E[GRD] | Ratio |
|---|---|---|---|
| 13.823 | 209.701 | 14.775 | 1.069 |
| 9.865 | 154.831 | 11.315 | 1.147 |
| 14.739 | 168.436 | 15.495 | 1.051 |
| 15.273 | 160.511 | 16.460 | 1.078 |
| 14.597 | 227.660 | 15.141 | 1.037 |
| 11.180 | 175.643 | 11.841 | 1.059 |
| 14.381 | 129.983 | 14.658 | 1.019 |
| 17.242 | 210.160 | 18.057 | 1.047 |
| 17.250 | 669.914 | 17.915 | 1.039 |
| 14.510 | 165.488 | 14.794 | 1.020 |
| 12.666 | 229.157 | 13.355 | 1.054 |
| 14.343 | 144.433 | 15.341 | 1.070 |
| 12.171 | 202.638 | 12.776 | 1.050 |
| 14.848 | 164.924 | 15.730 | 1.059 |
| 18.453 | 2034.251 | 19.450 | 1.054 |
| 13.867 | 3865.168 | 14.588 | 1.052 |
| 14.074 | 173.985 | 14.618 | 1.039 |
| 16.019 | 229.809 | 16.271 | 1.016 |
| 15.318 | 229.916 | 16.623 | 1.085 |
| 10.346 | 241.415 | 10.866 | 1.050 |
| | | AVG ratio: | 1.055 |

(a) Table 1: $n = 15$

| E[OPT] | Time | E[GRD] | Ratio |
|---|---|---|---|
| 14.340 | 2.903 | 15.625 | 1.090 |
| 12.392 | 2.005 | 12.835 | 1.036 |
| 13.086 | 2.272 | 14.010 | 1.071 |
| 13.868 | 1.628 | 14.428 | 1.040 |
| 13.133 | 3.504 | 13.304 | 1.013 |
| 11.344 | 2.375 | 11.514 | 1.015 |
| 12.882 | 1.684 | 13.005 | 1.010 |
| 14.661 | 2.543 | 14.785 | 1.008 |
| 13.116 | 2.667 | 13.848 | 1.056 |
| 13.490 | 1.943 | 13.490 | 1.000 |
| 11.469 | 3.124 | 11.872 | 1.035 |
| 14.687 | 2.052 | 14.771 | 1.006 |
| 13.641 | 2.386 | 14.688 | 1.077 |
| 11.952 | 2.470 | 12.154 | 1.017 |
| 16.084 | 2.645 | 16.647 | 1.035 |
| 15.955 | 3.622 | 16.365 | 1.026 |
| 11.830 | 3.017 | 12.144 | 1.026 |
| 10.422 | 4.094 | 10.830 | 1.039 |
| 11.892 | 3.152 | 12.351 | 1.039 |
| 10.637 | 3.081 | 11.043 | 1.038 |
| | | AVG ratio: | 1.034 |

(b) Table 2: $n = 10$

| E[OPT] | Time | E[GRD] | Ratio |
|---|---|---|---|
| 10.254 | 0.017 | 10.254 | 1.000 |
| 11.685 | 0.017 | 11.865 | 1.015 |
| 6.136 | 0.017 | 6.136 | 1.000 |
| 11.109 | 0.012 | 11.649 | 1.049 |
| 11.685 | 0.024 | 11.785 | 1.009 |
| 13.389 | 0.031 | 13.489 | 1.007 |
| 8.286 | 0.013 | 8.746 | 1.056 |
| 7.846 | 0.025 | 7.846 | 1.000 |
| 10.377 | 0.023 | 10.467 | 1.009 |
| 6.946 | 0.015 | 7.036 | 1.013 |
| 9.858 | 0.021 | 9.858 | 1.000 |
| 8.754 | 0.014 | 8.769 | 1.002 |
| 10.848 | 0.014 | 10.905 | 1.005 |
| 10.770 | 0.017 | 10.770 | 1.000 |
| 9.131 | 0.021 | 9.141 | 1.001 |
| 8.331 | 0.023 | 8.421 | 1.011 |
| 6.149 | 0.022 | 6.149 | 1.000 |
| 6.314 | 0.022 | 6.314 | 1.000 |
| 8.317 | 0.019 | 8.317 | 1.000 |
| 7.844 | 0.019 | 8.006 | 1.021 |
| | | AVG ratio: | 1.010 |

(c) Table 3: $n = 5$

Figure 4.9: Dense instances with general cost and uniform distribution over all intervals

| E[OPT] | Time | E[GRD] | Ratio |
|---|---|---|---|
| 6.362 | 198.154 | 6.362 | 1.000 |
| 5.761 | 176.567 | 5.784 | 1.004 |
| 7.308 | 206.071 | 7.308 | 1.000 |
| 7.233 | 150.473 | 7.314 | 1.011 |
| 5.484 | 226.963 | 5.484 | 1.000 |
| 5.665 | 222.676 | 5.716 | 1.009 |
| 5.117 | 135.723 | 5.117 | 1.000 |
| 6.799 | 263.597 | 6.820 | 1.003 |
| 5.119 | 202.725 | 5.119 | 1.000 |
| 6.073 | 151.746 | 6.073 | 1.000 |
| 5.614 | 244.025 | 5.639 | 1.004 |
| 6.049 | 167.078 | 6.080 | 1.005 |
| 5.574 | 227.538 | 5.574 | 1.000 |
| 5.987 | 178.633 | 5.995 | 1.001 |
| 6.543 | 228.199 | 6.543 | 1.000 |
| 5.699 | 200.327 | 5.699 | 1.000 |
| 5.570 | 219.733 | 5.570 | 1.000 |
| 5.130 | 271.736 | 5.130 | 1.000 |
| 5.198 | 245.099 | 5.327 | 1.025 |
| 4.441 | 224.021 | 4.441 | 1.000 |
| | | AVG ratio: | 1.003 |

(a) Table 1: $n = 15$

| E[OPT] | Time | E[GRD] | Ratio |
|---|---|---|---|
| 5.686 | 1.784 | 5.686 | 1.000 |
| 5.271 | 1.340 | 5.294 | 1.004 |
| 6.422 | 1.692 | 6.422 | 1.000 |
| 6.091 | 1.036 | 6.172 | 1.013 |
| 5.130 | 2.066 | 5.130 | 1.000 |
| 5.213 | 1.915 | 5.265 | 1.010 |
| 4.874 | 1.177 | 4.874 | 1.000 |
| 6.277 | 2.099 | 6.277 | 1.000 |
| 4.761 | 1.724 | 4.761 | 1.000 |
| 5.609 | 1.270 | 5.609 | 1.000 |
| 5.145 | 2.143 | 5.170 | 1.005 |
| 5.255 | 1.340 | 5.255 | 1.000 |
| 5.263 | 1.708 | 5.263 | 1.000 |
| 5.486 | 1.679 | 5.486 | 1.000 |
| 5.861 | 1.885 | 5.861 | 1.000 |
| 5.271 | 1.840 | 5.271 | 1.000 |
| 5.120 | 2.075 | 5.120 | 1.000 |
| 4.745 | 2.500 | 4.745 | 1.000 |
| 4.882 | 2.082 | 4.982 | 1.020 |
| 4.220 | 1.835 | 4.220 | 1.000 |
| | | AVG ratio: | 1.003 |

(b) Table 2: $n = 10$

| E[OPT] | Time | E[GRD] | Ratio |
|---|---|---|---|
| 3.851 | 0.017 | 3.851 | 1.000 |
| 3.851 | 0.016 | 3.851 | 1.000 |
| 4.095 | 0.016 | 4.095 | 1.000 |
| 3.941 | 0.011 | 4.022 | 1.021 |
| 3.751 | 0.022 | 3.751 | 1.000 |
| 3.714 | 0.025 | 3.714 | 1.000 |
| 3.434 | 0.012 | 3.434 | 1.000 |
| 4.095 | 0.019 | 4.095 | 1.000 |
| 3.680 | 0.017 | 3.680 | 1.000 |
| 4.022 | 0.014 | 4.022 | 1.000 |
| 3.534 | 0.020 | 3.534 | 1.000 |
| 3.557 | 0.014 | 3.557 | 1.000 |
| 3.893 | 0.014 | 3.893 | 1.000 |
| 3.941 | 0.018 | 3.941 | 1.000 |
| 3.941 | 0.021 | 3.941 | 1.000 |
| 3.751 | 0.021 | 3.751 | 1.000 |
| 3.751 | 0.022 | 3.751 | 1.000 |
| 3.413 | 0.023 | 3.413 | 1.000 |
| 3.534 | 0.019 | 3.634 | 1.028 |
| 3.347 | 0.019 | 3.347 | 1.000 |
| | | AVG ratio: | 1.002 |

(c) Table 3: $n = 5$

Figure 4.10: Sparse instances with unit cost and uniform distribution over all intervals

| E[OPT] | Time | E[GRD] | Ratio |
|---|---|---|---|
| 14.902 | 111.472 | 15.190 | 1.019 |
| 14.840 | 96.599 | 16.170 | 1.090 |
| 17.540 | 114.174 | 18.406 | 1.049 |
| 17.191 | 81.592 | 17.878 | 1.040 |
| 12.412 | 125.191 | 12.846 | 1.035 |
| 15.451 | 125.851 | 16.750 | 1.084 |
| 14.654 | 74.145 | 15.564 | 1.062 |
| 21.287 | 148.380 | 22.921 | 1.077 |
| 11.047 | 112.016 | 12.084 | 1.094 |
| 17.998 | 84.567 | 19.112 | 1.062 |
| 15.837 | 133.625 | 17.098 | 1.080 |
| 18.587 | 90.034 | 19.802 | 1.065 |
| 11.139 | 134.945 | 11.399 | 1.023 |
| 11.904 | 178.427 | 12.160 | 1.022 |
| 16.905 | 225.683 | 17.769 | 1.051 |
| 15.414 | 203.268 | 16.334 | 1.060 |
| 14.472 | 222.974 | 15.302 | 1.057 |
| 10.971 | 276.847 | 11.261 | 1.026 |
| 8.247 | 237.789 | 8.519 | 1.033 |
| 9.143 | 129.428 | 9.388 | 1.027 |
| | | AVG ratio: | 1.053 |

(a) Table 1: $n = 15$

| E[OPT] | Time | E[GRD] | Ratio |
|---|---|---|---|
| 19.274 | 1.801 | 19.517 | 1.013 |
| 17.362 | 1.357 | 18.887 | 1.088 |
| 16.148 | 1.655 | 17.113 | 1.060 |
| 17.745 | 1.027 | 18.478 | 1.041 |
| 14.246 | 2.076 | 15.342 | 1.077 |
| 15.554 | 1.898 | 17.075 | 1.098 |
| 11.423 | 1.139 | 12.406 | 1.086 |
| 18.399 | 2.028 | 18.472 | 1.004 |
| 12.094 | 1.768 | 13.852 | 1.145 |
| 13.410 | 1.263 | 13.855 | 1.033 |
| 14.306 | 2.036 | 15.606 | 1.091 |
| 13.237 | 1.351 | 13.918 | 1.051 |
| 14.509 | 1.615 | 15.492 | 1.068 |
| 12.762 | 1.629 | 13.667 | 1.071 |
| 14.187 | 1.724 | 14.942 | 1.053 |
| 11.731 | 1.776 | 12.039 | 1.026 |
| 10.334 | 1.988 | 10.671 | 1.033 |
| 13.348 | 2.480 | 14.349 | 1.075 |
| 11.769 | 2.054 | 12.280 | 1.043 |
| 10.556 | 1.843 | 11.618 | 1.101 |
| | | AVG ratio: | 1.063 |

(b) Table 2: $n = 10$

| E[OPT] | Time | E[GRD] | Ratio |
|---|---|---|---|
| 8.574 | 0.017 | 9.316 | 1.086 |
| 9.394 | 0.016 | 10.380 | 1.105 |
| 8.558 | 0.017 | 8.566 | 1.001 |
| 12.307 | 0.011 | 12.307 | 1.000 |
| 7.654 | 0.022 | 7.654 | 1.000 |
| 6.414 | 0.025 | 6.594 | 1.028 |
| 10.150 | 0.012 | 11.201 | 1.104 |
| 7.765 | 0.019 | 7.765 | 1.000 |
| 9.132 | 0.017 | 10.216 | 1.119 |
| 9.715 | 0.014 | 10.124 | 1.042 |
| 9.054 | 0.020 | 9.960 | 1.100 |
| 10.634 | 0.014 | 11.992 | 1.128 |
| 6.971 | 0.015 | 7.108 | 1.020 |
| 8.877 | 0.018 | 9.086 | 1.024 |
| 7.818 | 0.021 | 8.008 | 1.024 |
| 7.214 | 0.021 | 7.214 | 1.000 |
| 11.245 | 0.022 | 11.525 | 1.025 |
| 10.727 | 0.022 | 10.907 | 1.017 |
| 7.787 | 0.019 | 7.991 | 1.026 |
| 9.494 | 0.019 | 10.398 | 1.095 |
| | | AVG ratio: | 1.047 |

(c) Table 3: $n = 5$

Figure 4.11: Sparse instances with general cost and uniform distribution over all intervals

| E[OPT] | Time | E[GRD] | Ratio |
|---|---|---|---|
| 9.231 | 111.131 | 9.578 | 1.038 |
| 10.906 | 81.272 | 11.008 | 1.009 |
| 12.447 | 89.932 | 12.738 | 1.023 |
| 11.865 | 84.010 | 11.991 | 1.011 |
| 10.751 | 121.236 | 11.207 | 1.042 |
| 12.829 | 92.692 | 13.166 | 1.026 |
| 11.416 | 68.525 | 11.521 | 1.009 |
| 11.990 | 112.207 | 12.440 | 1.037 |
| 12.118 | 108.483 | 12.459 | 1.028 |
| 10.095 | 87.617 | 10.933 | 1.083 |
| 9.456 | 124.432 | 10.596 | 1.121 |
| 12.014 | 80.260 | 12.275 | 1.022 |
| 9.708 | 114.071 | 10.683 | 1.100 |
| 11.675 | 87.173 | 11.785 | 1.009 |
| 12.191 | 108.535 | 12.521 | 1.027 |
| 9.698 | 125.741 | 9.942 | 1.025 |
| 13.010 | 96.863 | 13.198 | 1.014 |
| 12.316 | 127.597 | 12.566 | 1.020 |
| 11.363 | 139.050 | 11.682 | 1.028 |
| 9.084 | 129.973 | 10.169 | 1.119 |
| | | AVG ratio: | 1.040 |

(a) Table 1: $n = 15$

| E[OPT] | Time | E[GRD] | Ratio |
|---|---|---|---|
| 6.729 | 1.752 | 6.969 | 1.036 |
| 7.735 | 1.237 | 7.801 | 1.008 |
| 8.636 | 1.384 | 8.832 | 1.023 |
| 8.987 | 0.999 | 8.988 | 1.000 |
| 7.925 | 2.007 | 8.216 | 1.037 |
| 9.081 | 1.702 | 9.205 | 1.014 |
| 8.028 | 1.017 | 8.057 | 1.004 |
| 8.649 | 1.602 | 8.829 | 1.021 |
| 8.527 | 1.618 | 8.680 | 1.018 |
| 8.324 | 1.188 | 8.584 | 1.031 |
| 7.411 | 1.864 | 8.015 | 1.081 |
| 8.539 | 1.238 | 8.688 | 1.018 |
| 7.770 | 1.425 | 8.192 | 1.054 |
| 7.988 | 1.491 | 8.096 | 1.014 |
| 8.408 | 1.551 | 8.650 | 1.029 |
| 7.098 | 1.986 | 7.236 | 1.019 |
| 9.751 | 1.764 | 9.789 | 1.004 |
| 9.192 | 2.377 | 9.315 | 1.013 |
| 7.958 | 1.861 | 8.214 | 1.032 |
| 7.119 | 1.844 | 7.855 | 1.103 |
| | | AVG ratio: | 1.028 |

(b) Table 2: $n = 10$

| E[OPT] | Time | E[GRD] | Ratio |
|---|---|---|---|
| 4.323 | 0.016 | 4.323 | 1.000 |
| 4.082 | 0.017 | 4.140 | 1.014 |
| 4.848 | 0.052 | 4.891 | 1.009 |
| 4.572 | 0.020 | 4.572 | 1.000 |
| 4.535 | 0.025 | 4.668 | 1.030 |
| 4.982 | 0.029 | 4.983 | 1.000 |
| 4.290 | 0.013 | 4.290 | 1.000 |
| 4.737 | 0.042 | 4.832 | 1.020 |
| 4.435 | 0.018 | 4.547 | 1.025 |
| 4.416 | 0.014 | 4.577 | 1.036 |
| 4.801 | 0.019 | 4.840 | 1.008 |
| 4.411 | 0.014 | 4.545 | 1.030 |
| 4.542 | 0.014 | 4.651 | 1.024 |
| 4.264 | 0.017 | 4.346 | 1.019 |
| 4.911 | 0.020 | 4.937 | 1.005 |
| 4.133 | 0.024 | 4.232 | 1.024 |
| 4.973 | 0.022 | 4.982 | 1.002 |
| 4.766 | 0.022 | 4.843 | 1.016 |
| 4.577 | 0.018 | 4.692 | 1.025 |
| 4.320 | 0.019 | 4.564 | 1.057 |
| | | AVG ratio: | 1.017 |

(c) Table 3: $n = 5$

Figure 4.12: Dense instances with unit cost and normal distribution over all intervals

| E[OPT] | Time | E[GRD] | Ratio |
|---|---|---|---|
| 20.837 | 115.018 | 21.594 | 1.036 |
| 30.262 | 84.329 | 30.725 | 1.015 |
| 28.906 | 92.056 | 29.194 | 1.010 |
| 28.660 | 86.304 | 28.733 | 1.003 |
| 21.199 | 125.511 | 21.902 | 1.033 |
| 31.134 | 94.348 | 32.738 | 1.052 |
| 28.916 | 69.399 | 29.445 | 1.018 |
| 28.532 | 112.599 | 29.145 | 1.021 |
| 29.332 | 108.785 | 30.286 | 1.033 |
| 23.285 | 87.929 | 24.007 | 1.031 |
| 22.602 | 124.432 | 24.367 | 1.078 |
| 28.831 | 81.543 | 29.464 | 1.022 |
| 19.827 | 117.619 | 21.297 | 1.074 |
| 34.459 | 90.947 | 35.117 | 1.019 |
| 30.436 | 111.004 | 30.971 | 1.018 |
| 20.670 | 126.303 | 20.896 | 1.011 |
| 27.634 | 95.980 | 27.784 | 1.005 |
| 27.815 | 129.493 | 28.306 | 1.018 |
| 27.892 | 124.537 | 28.763 | 1.031 |
| 20.908 | 130.935 | 22.801 | 1.091 |
| | | AVG ratio: | 1.031 |

(a) Table 1: $n = 15$

| E[OPT] | Time | E[GRD] | Ratio |
|---|---|---|---|
| 15.542 | 1.804 | 15.776 | 1.015 |
| 15.232 | 1.223 | 15.262 | 1.002 |
| 17.230 | 2.697 | 17.304 | 1.004 |
| 23.510 | 2.056 | 23.673 | 1.007 |
| 20.512 | 4.615 | 21.489 | 1.048 |
| 24.422 | 436.913 | 24.614 | 1.008 |
| 25.054 | 1.741 | 25.517 | 1.019 |
| 22.972 | 2.626 | 23.164 | 1.008 |
| 20.219 | 2.836 | 20.621 | 1.020 |
| 16.888 | 2.089 | 18.092 | 1.071 |
| 16.989 | 3.082 | 17.680 | 1.041 |
| 23.258 | 2.125 | 24.000 | 1.032 |
| 25.753 | 2.395 | 28.111 | 1.092 |
| 22.029 | 2.491 | 22.795 | 1.035 |
| 22.556 | 2.589 | 22.802 | 1.011 |
| 17.282 | 3.300 | 18.706 | 1.082 |
| 29.263 | 2.947 | 29.457 | 1.007 |
| 25.624 | 4.073 | 26.003 | 1.015 |
| 14.801 | 3.095 | 15.325 | 1.035 |
| 18.724 | 2.985 | 20.451 | 1.092 |
| | | AVG ratio: | 1.032 |

(b) Table 2: $n = 10$

| E[OPT] | Time | E[GRD] | Ratio |
|---|---|---|---|
| 11.188 | 0.025 | 11.383 | 1.017 |
| 12.034 | 0.025 | 12.261 | 1.019 |
| 13.579 | 0.026 | 13.676 | 1.007 |
| 11.171 | 0.018 | 11.510 | 1.030 |
| 7.209 | 0.034 | 7.440 | 1.032 |
| 15.944 | 0.040 | 15.957 | 1.001 |
| 10.209 | 0.019 | 10.353 | 1.014 |
| 14.241 | 0.029 | 14.485 | 1.017 |
| 14.272 | 0.027 | 14.574 | 1.021 |
| 13.399 | 0.021 | 14.177 | 1.058 |
| 12.553 | 0.030 | 12.673 | 1.010 |
| 9.527 | 0.023 | 9.575 | 1.005 |
| 12.640 | 0.022 | 12.676 | 1.003 |
| 12.607 | 0.027 | 12.607 | 1.000 |
| 13.742 | 0.033 | 13.809 | 1.005 |
| 12.094 | 0.037 | 12.175 | 1.007 |
| 12.926 | 0.035 | 12.933 | 1.001 |
| 17.185 | 0.036 | 17.493 | 1.018 |
| 11.044 | 0.033 | 11.137 | 1.008 |
| 13.951 | 0.031 | 15.157 | 1.086 |
| | | AVG ratio: | 1.018 |

(c) Table 3: $n = 5$

Figure 4.13: Dense instances with general cost and normal distribution over all intervals

| E[OPT] | Time | E[GRD] | Ratio |
|---|---|---|---|
| 9.842 | 109.966 | 9.921 | 1.008 |
| 9.631 | 97.017 | 9.757 | 1.013 |
| 13.518 | 113.740 | 13.602 | 1.006 |
| 12.664 | 86.915 | 12.744 | 1.006 |
| 11.640 | 124.232 | 11.858 | 1.019 |
| 13.915 | 123.691 | 14.007 | 1.007 |
| 10.460 | 73.542 | 10.629 | 1.016 |
| 13.171 | 143.332 | 13.280 | 1.008 |
| 12.103 | 111.121 | 12.318 | 1.018 |
| 11.140 | 82.924 | 11.348 | 1.019 |
| 11.904 | 136.130 | 12.094 | 1.016 |
| 12.715 | 94.317 | 12.779 | 1.005 |
| 11.739 | 131.239 | 12.039 | 1.026 |
| 10.496 | 103.168 | 10.683 | 1.018 |
| 12.641 | 136.246 | 12.769 | 1.010 |
| 8.882 | 118.435 | 9.018 | 1.015 |
| 14.628 | 128.511 | 14.698 | 1.005 |
| 13.183 | 168.577 | 13.332 | 1.011 |
| 11.350 | 137.979 | 11.529 | 1.016 |
| 10.742 | 134.422 | 10.971 | 1.021 |
| | | AVG ratio: | 1.013 |

(a) Table 1: $n = 15$

| E[OPT] | Time | E[GRD] | Ratio |
|---|---|---|---|
| 7.738 | 1.779 | 7.807 | 1.009 |
| 7.087 | 1.356 | 7.213 | 1.018 |
| 9.487 | 1.665 | 9.530 | 1.004 |
| 9.054 | 1.043 | 9.078 | 1.003 |
| 8.537 | 2.039 | 8.755 | 1.025 |
| 9.625 | 1.887 | 9.704 | 1.008 |
| 7.797 | 1.116 | 7.827 | 1.004 |
| 9.459 | 2.005 | 9.541 | 1.009 |
| 8.437 | 1.691 | 8.583 | 1.017 |
| 8.011 | 1.235 | 8.207 | 1.024 |
| 8.764 | 1.994 | 8.882 | 1.013 |
| 8.783 | 1.365 | 8.783 | 1.000 |
| 8.632 | 1.630 | 8.829 | 1.023 |
| 7.583 | 1.638 | 7.769 | 1.024 |
| 9.026 | 1.734 | 9.119 | 1.010 |
| 7.000 | 1.771 | 7.073 | 1.010 |
| 9.890 | 1.993 | 9.896 | 1.001 |
| 9.203 | 2.411 | 9.307 | 1.011 |
| 8.346 | 2.016 | 8.486 | 1.017 |
| 8.192 | 1.843 | 8.345 | 1.019 |
| | | AVG ratio: | 1.013 |

(b) Table 2: $n = 10$

| E[OPT] | Time | E[GRD] | Ratio |
|---|---|---|---|
| 4.349 | 0.017 | 4.392 | 1.010 |
| 4.113 | 0.017 | 4.157 | 1.011 |
| 4.857 | 0.017 | 4.897 | 1.008 |
| 4.607 | 0.011 | 4.630 | 1.005 |
| 4.567 | 0.022 | 4.689 | 1.027 |
| 4.978 | 0.025 | 4.978 | 1.000 |
| 4.156 | 0.012 | 4.156 | 1.000 |
| 4.933 | 0.020 | 4.956 | 1.005 |
| 4.456 | 0.017 | 4.557 | 1.023 |
| 4.390 | 0.014 | 4.536 | 1.033 |
| 4.932 | 0.020 | 4.935 | 1.001 |
| 4.565 | 0.014 | 4.565 | 1.000 |
| 4.771 | 0.015 | 4.863 | 1.019 |
| 4.228 | 0.018 | 4.304 | 1.018 |
| 4.910 | 0.021 | 4.928 | 1.004 |
| 4.064 | 0.021 | 4.131 | 1.016 |
| 4.989 | 0.022 | 4.995 | 1.001 |
| 4.773 | 0.022 | 4.847 | 1.016 |
| 4.589 | 0.019 | 4.681 | 1.020 |
| 4.580 | 0.019 | 4.586 | 1.002 |
| | | AVG ratio: | 1.011 |

(c) Table 3: $n = 5$

Figure 4.14: Sparse instances with unit cost and normal distributions over all intervals

| E[OPT] | Time | E[GRD] | Ratio |
|---|---|---|---|
| 26.980 | 198.814 | 28.084 | 1.041 |
| 27.153 | 172.252 | 28.032 | 1.032 |
| 28.856 | 202.041 | 29.203 | 1.012 |
| 37.999 | 146.366 | 38.517 | 1.014 |
| 29.289 | 223.054 | 29.866 | 1.020 |
| 38.620 | 222.472 | 39.027 | 1.011 |
| 29.361 | 133.783 | 29.945 | 1.020 |
| 34.768 | 258.178 | 35.119 | 1.010 |
| 29.340 | 200.893 | 29.566 | 1.008 |
| 25.766 | 150.631 | 26.323 | 1.022 |
| 24.341 | 243.214 | 24.890 | 1.023 |
| 33.855 | 168.590 | 34.311 | 1.013 |
| 28.990 | 230.071 | 29.729 | 1.025 |
| 27.921 | 173.098 | 28.485 | 1.020 |
| 36.249 | 227.644 | 36.683 | 1.012 |
| 23.620 | 200.364 | 24.249 | 1.027 |
| 39.100 | 225.135 | 39.336 | 1.006 |
| 40.348 | 280.686 | 40.860 | 1.013 |
| 27.742 | 244.587 | 28.102 | 1.013 |
| 22.237 | 227.154 | 22.757 | 1.023 |
| | | AVG ratio: | 1.018 |

(a) Table 1: $n = 15$

| E[OPT] | Time | E[GRD] | Ratio |
|---|---|---|---|
| 21.787 | 3.106 | 22.925 | 1.052 |
| 18.357 | 2.901 | 19.241 | 1.048 |
| 20.065 | 3.078 | 20.433 | 1.018 |
| 15.390 | 1.832 | 15.501 | 1.007 |
| 23.005 | 3.675 | 23.357 | 1.015 |
| 27.873 | 3.558 | 28.213 | 1.012 |
| 17.722 | 1.979 | 18.011 | 1.016 |
| 30.267 | 3.472 | 30.861 | 1.020 |
| 19.668 | 3.093 | 20.540 | 1.044 |
| 19.392 | 2.568 | 19.796 | 1.021 |
| 18.127 | 4.277 | 18.361 | 1.013 |
| 29.728 | 2.438 | 29.761 | 1.001 |
| 16.964 | 2.825 | 17.264 | 1.018 |
| 18.008 | 2.847 | 18.367 | 1.020 |
| 25.126 | 3.174 | 25.639 | 1.020 |
| 16.865 | 3.133 | 17.652 | 1.047 |
| 20.748 | 3.556 | 20.774 | 1.001 |
| 25.936 | 4.308 | 26.208 | 1.010 |
| 14.275 | 3.560 | 14.551 | 1.019 |
| 23.738 | 3.149 | 24.914 | 1.050 |
| | | AVG ratio: | 1.023 |

(b) Table 2: $n = 10$

| E[OPT] | Time | E[GRD] | Ratio |
|---|---|---|---|
| 9.112 | 0.017 | 9.452 | 1.037 |
| 8.829 | 0.017 | 8.829 | 1.000 |
| 13.608 | 0.017 | 13.801 | 1.014 |
| 11.930 | 0.011 | 11.976 | 1.004 |
| 11.694 | 0.022 | 11.936 | 1.021 |
| 9.958 | 0.026 | 9.967 | 1.001 |
| 10.353 | 0.012 | 10.353 | 1.000 |
| 16.774 | 0.020 | 16.793 | 1.001 |
| 13.414 | 0.018 | 13.993 | 1.043 |
| 7.853 | 0.014 | 7.866 | 1.002 |
| 9.893 | 0.020 | 9.916 | 1.002 |
| 8.107 | 0.015 | 8.235 | 1.016 |
| 10.302 | 0.015 | 10.307 | 1.000 |
| 8.396 | 0.018 | 8.615 | 1.026 |
| 14.770 | 0.021 | 14.850 | 1.005 |
| 8.042 | 0.023 | 8.359 | 1.039 |
| 14.957 | 0.023 | 14.958 | 1.000 |
| 12.294 | 0.023 | 12.389 | 1.008 |
| 12.991 | 0.020 | 13.471 | 1.037 |
| 13.800 | 0.019 | 13.951 | 1.011 |
| | | AVG ratio: | 1.013 |

(c) Table 3: $n = 5$

Figure 4.15: Sparse instances with general cost and normal distribution over all intervals

# CHAPTER 5

# Minimum Cost Adaptive Submodular Cover

## 5.1 Introduction

Adaptive stochastic optimization, where an algorithm makes sequential decisions while partially observing uncertainty, arises in numerous applications such as active learning ([Das04]), sensor placement ([GKS05]) and viral marketing ([TWTD17]). Often, these applications involve an underlying submodular function, and the framework of adaptive-submodularity, introduced by [GK11], has been widely used to solve these problems. In this chapter, we study a basic problem in this context: covering an adaptive-submodular function at the minimum expected cost.

In some applications, such as sensor placement or stochastic set cover, studied in [GV06], the uncertainty can be captured by an *independent* random variable associated with each decision. However, there are also a number of applications where the random variables associated with different decisions are correlated. The adaptive-submodularity framework that we consider is also applicable in certain applications involving correlations.

One such application is the viral marketing problem, where we are given a social network and target $Q$, and the goal is to influence at least $Q$ users to adopt a new product. A user can be influenced in two ways (i) directly because the user is offered a promotion, or (ii) indirectly because some friend of the user was influenced *and* the friend influenced this user. We incur a cost only in case (i), which accounts for the promotional offer. A widely-used model for influence behavior is the *independent cascade model* [KKT15]. Here, each arc $(u, v)$ has a value $p_{uv} \in [0, 1]$ that represents the probability that user $u$ will influence user $v$ (if $u$ is already influenced). A solution is a sequential process that in each step, selects one user $w$ to influence directly, after which we get to observe which of $w$'s friends were influenced (indirectly), which of their friends were influenced, and so on. So, the solution can utilize these partial observations to make decisions *adaptively*. Such an adaptive solution can be represented by a decision tree; however, it may require exponential space to store

explicitly. We will analyze simple solutions that can be implemented in polynomial time (and space), but our performance guarantees are relative to an optimal solution that can be very complex. Also, note that the random observations associated with different decisions in the viral marketing problem are highly correlated: the set of nodes that get (indirectly) influenced by any node $w$ depends on the entire network (not just $w$).

Another application involving correlations is in hypothesis identification ([Das04]). Here, there are $m$ possible hypotheses, out of which exactly one is correct. We need to determine the (unknown) correct hypothesis by performing binary tests: each test has a positive outcome on some subset of hypotheses and a negative outcome on the others. Notice that the random observations associated with different decisions (i.e., test outcomes) are highly correlated. This problem is also known as optimal decision tree, and has been studied extensively, see e.g., [GG74, HR76, KPB99, AH12, GB09, GNR17].

While there has been extensive prior work on minimum cost cover of adaptive submodular functions ([GV06, GK11, INvdZ16, HKP21, EKM21]), all these results focus on minimizing the *expected cost*, which is a risk-neutral objective. However, one may also be interested in minimizing a higher moment of the random cost, which corresponds to a *risk-averse* objective. We note that the quality of a solution may vary greatly depending on the chosen objective. For example, consider two solutions $A$ and $B$. Solution $A$ has cost 1 with probability (w.p.) $1 - \frac{1}{M}$ and cost $M$ w.p. $\frac{1}{M}$. Solution $B$ has cost $M^{1/3}$ w.p. 1. The expected cost (i.e., first moment) of $A$ is $2 - \frac{1}{M}$, whereas that of $B$ is $M^{1/3}$. On the other hand, the second moment of $A$ is $\approx M$, whereas that of $B$ is $M^{2/3}$. Clearly, solution $A$ is much better in terms of the expected cost, whereas solution $B$ is much better in terms of the second moment.

Motivated by this, we consider the adaptive submodular cover problem under the more general objective of minimizing the $p^{th}$ moment cost, for any $p \geq 1$. Somewhat surprisingly, we show that there is a *universal* algorithm for adaptive-submodular cover that approximately minimizes all moments simultaneously. We note that our result is the first approximation algorithm for higher moments ($p > 1$), even for widely studied special cases such as (independent) stochastic submodular cover ([INvdZ16, AAK19, GGN21, HKP21]) and optimal decision tree ([KPB99, AH12, GB09, GNR17]).

### 5.1.1 Problem definition

**Random items.** Let $E$ be a finite set of $n$ *items*. Each item $e \in E$ corresponds to a random variable $\Phi_e \in \Omega$, where $\Omega$ is a finite *outcome space* (for a single item). We use $\Phi = \langle \Phi_e : e \in E \rangle$ to denote the vector of all random variables (r.v.s). The r.v.s may be arbitrarily correlated across items. We use upper-case letters to represent r.v.s and the

corresponding lower-case letters to represent realizations of the r.v.s. Thus, for any item $e$, $\phi_e \in \Omega$ is the realization of $\Phi_e$; and $\phi = \langle \phi_e : e \in E \rangle$ denotes the realization of $\Phi$. Equivalently, we can represent the realization $\phi$ as a subset $\{(e, \phi_e) : e \in E\} \subseteq E \times \Omega$ of item-outcome pairs.

A *partial realization* $\psi \subseteq E \times \Omega$ refers to the realizations of any *subset* of items; $\mathsf{dom}(\psi) \subseteq E$ denotes the items whose realizations are represented in $\psi$, and $\psi_e$ denotes the realization of any item $e \in \mathsf{dom}(\psi)$. Note that a partial realization contains at most one pair of the form $(e, *)$ for any item $e \in E$. The (full) realization $\phi$ corresponds to a partial realization with $\mathsf{dom}(\phi) = E$. For two partial realizations $\psi, \psi' \subseteq E \times \Omega$, we say that $\psi$ is a *subrealization* of $\psi'$ (denoted $\psi \preccurlyeq \psi'$) if $\psi \subseteq \psi'$; in other words, $\mathsf{dom}(\psi) \subseteq \mathsf{dom}(\psi')$ and $\psi_e = \psi'_e$ for all $e \in \mathsf{dom}(\psi)$. (We use the notation $\psi \preccurlyeq \psi'$ instead of $\psi \subseteq \psi'$ in order to be consistent with prior works.) Two partial realizations $\psi, \psi' \subseteq E \times \Omega$ are said to be *disjoint* if there is no full realization $\phi$ with $\psi \preccurlyeq \phi$ and $\psi' \preccurlyeq \phi$; in other words, there is some item $e \in \mathsf{dom}(\psi) \cap \mathsf{dom}(\psi')$ such that the realization of $\Phi_e$ is different under $\psi$ and $\psi'$.

We assume that there is a prior probability distribution $p(\phi) = \Pr[\Phi = \phi]$ over realizations $\phi$. Moreover, for any partial realization $\psi$, we assume that we can compute the posterior distribution $p(\phi|\psi) = \Pr(\Phi = \phi | \psi \preccurlyeq \Phi)$.

**Utility function.** In addition to the random items (described above), there is a *utility function* $f : 2^{E \times \Omega} \to \mathbb{R}_{\geq 0}$ that assigns a value to any partial realization. We will assume that this function is monotone, i.e., having more realizations can not reduce the value. Formally,

**Definition 2** (Monotonicity)**.** *A function $f : 2^{E \times \Omega} \to \mathbb{R}_{\geq 0}$ is* **monotone** *if*

$$f(\psi) \leq f(\psi') \quad \text{for all partial realizations } \psi \preccurlyeq \psi'.$$

We also assume that the function $f$ can always achieve its maximal value, i.e.,

**Definition 3** (Coverable)**.** *Let $Q$ be the maximal value of function $f$. Then, function $f$ is said to be* **coverable** *if this value $Q$ can be achieved under every (full) realization, i.e.,*

$$f(\phi) = Q \text{ for all realizations } \phi \text{ of } \Phi.$$

Furthermore, we will assume that the function $f$ along with the probability distribution $p(\cdot)$ satisfies a submodularity-like property. Before formalizing this, we need the following definition.

**Definition 4** (Marginal benefit)**.** *The* **conditional expected marginal benefit** *of an item*

*$e \in E$ conditioned on observing the partial realization $\psi$ is:*

$$\Delta(e|\psi) := \mathbb{E}\left[f(\psi \cup (e, \Phi_e)) - f(\psi) \,|\, \psi \preccurlyeq \Phi\right] \;=\; \sum_{\omega \in \Omega} \Pr[\Phi_e = \omega | \psi \preccurlyeq \Phi] \cdot (f(\psi \cup (e, \omega)) - f(\psi)) \,.$$

We will assume that function $f$ and distribution $p(\cdot)$ jointly satisfy the adaptive-submodularity property, defined as follows.

**Definition 5** (Adaptive submodularity)**.** *A function $f : 2^{E \times \Omega} \to \mathbb{R}_{\geq 0}$ is* **adaptive submodular** *w.r.t. distribution $p(\phi)$ if for all partial realizations $\psi \preccurlyeq \psi'$, and all items $e \in E \setminus \mathsf{dom}(\psi')$,*

$$\Delta(e|\psi) \geq \Delta(e|\psi').$$

In other words, this property ensures that the marginal benefit of an item never increases as we condition on more realizations. Given any function $f$ satisfying Definitions 2, 3 and 5, we can pre-process $f$ by subtracting $f(\emptyset)$, to get an equivalent function (that maintains these properties), and has a smaller $Q$ value. So, we may assume that $f(\emptyset) = 0$.

**Min-cost adaptive-submodular cover (ASC).** In this problem, each item $e \in E$ has a positive cost $c_e \geq 1$. The goal is to select items (and observe their realizations) sequentially until the observed realizations have function value $Q$. The objective is to minimize the expected cost of selected items.

Due to the stochastic nature of the problem, the solution concept here is much more complex than in the deterministic setting (where we just select a static subset). In particular, a solution corresponds to a "policy" that maps observed realizations to the next selection decision. The observed realization at any point corresponds to a partial realization (namely, the realizations of the items selected so far). Formally, a *policy* is a mapping $\pi : 2^{E \times \Omega} \to E$, which specifies the next item $\pi(\psi)$ to select when the observed realizations are $\psi$. Policies and utility functions are not necessarily defined over all subsets $2^{E \times \Omega}$, but only over partial realizations; recall that a partial realization is of the form $\{(e, \phi_e) : e \in S\}$ where $\phi$ is some full-realization and $S \subseteq E$. The policy $\pi$ terminates at the first point when $f(\psi) = Q$, where $\psi \subseteq E \times \Omega$ denotes the observed realizations so far. For any policy $\pi$ and full realization $\phi$, let $C(\pi, \phi)$ denote the total cost of items selected by policy $\pi$ under realization $\phi$. Then, the expected cost of policy $\pi$ is:

$$c_{exp}(\pi) \;=\; \mathbb{E}_{\Phi}\left[C(\pi, \Phi)\right] \;=\; \sum_{\phi} p(\phi) \cdot C(\pi, \phi).$$

While minimizing the expected cost is the primary objective, we are also interested in min-

imizing higher moments of the cost. For any $p \geq 1$ and policy $\pi$, the $p^{th}$ moment of the policy's cost is:

$$c_p(\pi) \;=\; \mathbb{E}_{\Phi}\left[C(\pi,\Phi)^p\right] \;=\; \sum_{\phi} p(\phi)\cdot C(\pi,\phi)^p.$$

At any point in policy $\pi$, we refer to the cumulative cost incurred so far as the *time*. If $J_1, J_2, \cdots J_k$ denotes the (random) sequence of items selected by $\pi$ then for each $i \in \{1,2,\cdots k\}$, we view item $J_i$ as being selected during the time interval $[\sum_{h=1}^{i-1} c(J_h)\,,\, \sum_{h=1}^{i} c(J_h))$ and the realization of $J_i$ is only observed at time $\sum_{h=1}^{i} c(J_h)$. For any time $t \geq 0$, we use $\Psi(\pi,t) \subseteq E \times \Omega$ to denote the (random) realizations that have been observed by time $t$ in policy $\pi$. We note that $\Psi(\pi,t)$ only contains the realizations of items that have been *completely* selected by time $t$. Note that the policy terminates at the earliest time $t$ where $f(\Psi(\pi,t)) = Q$.

Given any policy $\pi$, we define its *cost $k$ truncation* by running $\pi$ and stopping it just before the cost of selected items exceeds $k$. That is, we stop the policy as late as possible while ensuring that the cost of selected items never exceeds $k$ (for any realization).

**Remark:** Our definition of the utility function $f$ is slightly more restrictive than the original definition by [GK11]. In particular, the utility function in [GK11] is of the form $g : 2^E \times \Omega^E \to \mathbb{R}_{\geq 0}$, where the function value $g(\mathsf{dom}(\psi), \Phi)$ for any partial realization $\psi$ is still random and can depend on the outcomes of unobserved items, i.e., those in $E \setminus \mathsf{dom}(\psi)$. Nevertheless, our formulation (ASC) still captures most applications of the formulation studied in [GK11]. See Section 5.3 for details.

### 5.1.2 Adaptive greedy policy

Algorithm 5 describes a natural greedy policy for min-cost adaptive-submodular cover, which has also been studied in prior works by [GK17, EKM21, HKP21].

**Algorithm 5** Adaptive Greedy Policy $\pi$.

selected items $A \leftarrow \emptyset$, observed realizations $\psi \leftarrow \emptyset$
**while** $f(\psi) < Q$ **do**
    $e^* = \text{argmax}_{e \in E \setminus A} \frac{\Delta(e|\psi)}{c_e}$
    add $e^*$ to the selected items, i.e., $A \leftarrow A \cup \{e^*\}$
    select $e^*$ and observe $\Phi_{e^*}$
    update $\psi \leftarrow \psi \cup \{(e^*, \Phi_{e^*})\}$

**Remark:** Note that the policy $\pi$ remains the same if we replace the greedy choice by

$$e^* = \operatorname*{argmax}_{e \in E \setminus A} \frac{\Delta(e|\psi)}{c_e \cdot (Q - f(\psi))}. \tag{5.1}$$

This is because the additional term $Q - f(\psi)$ is the same for each item $e \in E \setminus A$ (note that at any particular step, $\psi$ is a fixed partial realization). We will make use of this alternative greedy criterion in our analysis.

### 5.1.3 Results and techniques

Our first main result is on the expected cost of the greedy policy.

**Theorem 12.** *Consider any instance of minimum cost adaptive-submodular cover, where the utility function $f : 2^{E \times \Omega} \to \mathbb{R}_{\geq 0}$ is monotone, coverable and adaptive-submodular w.r.t. the probability distribution $p(\cdot)$. Suppose that there is some value $\eta > 0$ such that $f(\psi) > Q - \eta$ implies $f(\psi) = Q$ for all partial realizations $\psi \subseteq E \times \Omega$. Then, the expected cost of the greedy policy is*

$$c_{exp}(\pi) \leq 4 \cdot (1 + \ln(Q/\eta)) \cdot c_{exp}(\sigma),$$

*where $\sigma$ denotes the optimal policy.*

This is an asymptotic improvement over the $(1 + \ln(Q/\eta))^2$-approximation bound from [GK17] and the $(1 + \ln(\frac{nQc_{max}}{\eta}))$-approximation bound from [EKM21]; the maximum item cost $c_{max}$ can even be exponentially larger than $Q$. Our bound is the best possible, up to the constant factor of 4, because the set cover problem is a special case of ASC (where $Q$ is the number of elements to cover and $\eta = 1$). [DS14] showed that, assuming $P \neq NP$, for any $\epsilon > 0$, one cannot obtain a better than $(1 - \epsilon) \ln(Q)$ approximation ratio for set cover.

As a consequence, we obtain the first $O(\ln Q)$-approximation algorithm for the viral marketing application mentioned earlier. We also obtain an improved bound for the optimal decision tree problem with uniform priors. See Section 5.3 for details.

In fact, we obtain Theorem 12 as a special case of the following general result on minimizing the $p^{th}$ moment of the coverage cost.

**Theorem 13.** *Consider any instance of minimum cost adaptive-submodular cover, where the utility function $f$ is monotone, coverable and adaptive-submodular w.r.t. the probability distribution $q(\cdot)$. Suppose that there is some value $\eta > 0$ such that $f(\psi) > Q - \eta$ implies $f(\psi) = Q$ for all partial realizations $\psi \subseteq E \times \Omega$. Then, for every $p \geq 1$, the $p^{th}$ moment cost*

*of the greedy policy is*

$$c_p(\pi) \leq (p+1)^{p+1} \cdot (1 + \ln(Q/\eta))^p \cdot c_p(\sigma_p),$$

*where $\sigma_p$ denotes the optimal policy for the $p^{th}$ moment objective.*

This result is also best possible for any $p \geq 1$, up to the leading constant of $(p+1)^{p+1}$. We emphasize that the greedy policy $\pi$ is oblivious to the choice of $p$: so Theorem 13 provides a universal algorithm that achieves the stated approximation ratio for *every* value of $p$. We are not aware of any previous result for minimizing higher moments, even for special cases such as (independent) stochastic submodular cover or optimal decision tree.

Our proof technique is very different from prior works on adaptive submodular cover, namely [GK17, EKM21, HKP21]. The approaches in these papers were tailored to bound the expected cost, and seem difficult to extend to higher moments $p > 1$. We start by expressing the $p^{th}$ moment objective of any policy as an appropriate integral of "non-completion probabilities" over all times. Then, we relate the non-completion probabilities in the greedy policy (at any time $t$) to that in the optimal policy (at a scaled time $\frac{t}{\alpha}$). In order to establish this relation, we consider the integral of the greedy criterion value over each suffix $(t, \infty)$ of time, and prove lower and upper bounds on this quantity. Finally, we relate the $p^{th}$ moment objectives of greedy and the optimal policy by analyzing a double integral over all suffixes. The high-level approach of using non-completion probabilities has been used earlier for minimizing the expected cost in a number of stochastic covering problems, including the independent special case of ASC by [INvdZ16]. We refine and improve this approach by (i) utilizing non-completion probabilities at all times (not just power-of-two times) and (ii) using a stronger upper bound on the greedy criterion value. Moreover, we extend this approach to $p^{th}$ moment objectives (prior work only looked at expected cost).

Additionally, our algorithm and analysis extend in a straightforward manner, to the setting with multiple adaptive-submodular functions, where the objective is the sum of $p^{th}$ moments of the "cover times" of all the functions. We obtain the same approximation ratio even for this more general problem. In fact, the multiple ASC problem with $Q = \eta = 1$ and an expected cost objective (i.e., $p = 1$) generalizes the min-sum set cover problem ([FLT04]), which is NP-hard to approximate better than factor 4. The constant factor $(p+1)^{p+1}$ in our approximation ratio for this problem is also 4, which implies that our bound is tight in this case.

Finally, we provide computational results of our algorithm on real-world instances of optimal decision tree. We compare the performance of our algorithm on $p^{th}$ moment objectives for $p = 1, 2, 3$ to lower-bounds that we obtain (via Huffman coding). Our algorithm performs

very well on the instances tested.

### 5.1.4 Related work

Adaptive submodularity was introduced by [GK11], where they considered both the maximum-coverage and the minimum-cost-cover problems. They showed that the greedy policy is a $(1 - \frac{1}{e})$ approximation for maximum coverage, where the goal is to maximize the expected value of an adaptive-submodular function subject to a cardinality constraint. They also claimed that the greedy policy is a $(1+\ln(Q/\eta))$ approximation for min-cost cover of an adaptive-submodular function. However, this result had an error which was found in [NS17], and a corrected proof ([GK17]) only provides a double-logarithmic $(1 + \ln(Q/\eta))^2$ approximation. Recently, [EKM21] obtained a single-logarithmic approximation bound of $(1 + \ln(\frac{nQc_{max}}{\eta}))$. However, this bound depends additionally on the number of items $n$ and their maximum cost $c_{max}$. Our result shows that the greedy policy is indeed an $O(\ln(Q/\eta))$ approximation. As noted earlier, our definition of ASC is simpler and slightly more restrictive than the original one in [GK11], although most applications of adaptive-submodularity do satisfy our definition.

The special case of adaptive-submodularity where the random variables are independent across items, has also been studied extensively. For the maximum-coverage version, [AN16] obtained a $(1 - \frac{1}{e})$-approximation algorithm via a "non adaptive" policy (that fixes a subset of items to select upfront). Subsequent work by [GNS17, BSZ19, ASW16] obtained constant factor approximation algorithms for a variety of constraints (beyond just cardinality). The minimum-cost cover problem (called *stochastic submodular cover*) was studied in [INvdZ16, AAK19, HKP21, HK18, GGN21]. In particular, an $O(\ln(Q/\eta))$ approximation algorithm follows from [INvdZ16], and recently [HKP21] proved that the greedy policy has a $(1 + \ln(Q/\eta))$ approximation guarantee. The latter guarantee is the best possible, even up to the constant factor: this also matches the best approximation ratio for the *deterministic* submodular cover problem due to [Wol82] and its special case of set cover ([DS14]). [AAK19] and [GGN21] studied stochastic submodular cover under limited "rounds of adaptivity", and obtained smooth tradeoffs between the approximation ratio and the number of rounds.

The (deterministic) submodular cover problem with multiple functions was introduced in [AG11], where they obtained an $O(\ln(Q/\eta))$ approximation algorithm for minimizing the sum of cover times. Subsequently, [INvdZ16] studied the *stochastic* submodular cover problem with multiple functions (which involved independent items), and obtained an $O(\ln(Q/\eta))$ approximation algorithm. The analysis in this chapter is similar, at a high level, to the analysis in [INvdZ16], which also relied on the non-completion probabilities. However, we

handle the more general adaptive-submodular setting (where items may be correlated), and we obtain a much better constant factor and extend the techniques to minimizing higher moments of the cost.

Minimizing higher moments of the covering cost has been studied previously in the *deterministic* setting by [GGKT08]. Specifically, they considered the $L_p$ *set cover* problem, where given a collection of sets (items in our setting) and elements, the goal is to find a sequence of sets so as to minimize the total $p^{th}$ moment of cover times over all elements. [GGKT08] showed that the greedy algorithm for $L_p$ set cover achieves an approximation ratio of $(p+1)^{p+1}$ for each $p \geq 1$ simultaneously. We note that $L_p$ set cover is a special case of the multiple ASC problem studied in this chapter, where $Q = \eta = 1$ and each function corresponds to an element in set cover. So, our approximation ratio for multiple ASC in this special case matches the best known bound for $L_p$ set cover. Moreover, [GGKT08] showed that for any fixed value of $p$, there is no approximation ratio better than $\Omega(p)^p$ for $L_p$ set cover, unless $NP \subseteq DTIME(n^{\log\log n})$. So, the leading factor $(p+1)^{p+1}$ in our bound for multiple ASC is nearly the best possible, for each $p$. As noted before, we are the first to consider higher moment objectives in the *stochastic* setting, even for special cases of ASC such as stochastic submodular cover and optimal decision tree. Our proof technique is also very different from that in [GGKT08].

A different (scenario based) model for correlations in adaptive submodular cover was studied in [NKN20, GHKL16]. Here, the utility function $f$ is just required to be submodular (not adaptive-submodular), but the algorithm requires an explicit description of the probability distribution $p(\cdot)$. In particular, [NKN20] obtained a greedy-style policy with approximation ratio $O(\ln(mQ/\eta))$ where $m$ is the support-size of distribution $p(\cdot)$, and $Q$ and $\eta$ are as before. We believe that our proof technique may be useful in extending the expected-cost results in [NKN20] to higher moments and in improving the constant factor in their approximation ratio.

## 5.2 Analyzing the Greedy Policy

In this section, we prove our main result (Theorem 13), which bounds the $p^{th}$ moment of the greedy policy cost. Note that setting $p = 1$ in Theorem 13 implies the bound on expected cost (Theorem 12). We first show how to re-write the $p^{th}$ moment objective in terms of "non-completion" probabilities. Then, we relate the non-completion probabilities in the greedy and optimal policies by analyzing the integral of the "greedy criterion value" (5.1) over time.

### 5.2.1 Non-completion probabilities and the moment objective

Our analysis is based on relating the "non-completion" probabilities at different times in the greedy policy $\pi$ and the optimal policy $\sigma$. We first define these quantities formally.

**Definition 6** (Non-completion probabilities)**.** *For any time $t \geq 0$, let*

$$o(t) := \Pr[\sigma \textit{ does not terminate by time } t] = \Pr[C(\sigma, \Phi) > t] = \Pr[f(\Psi(\sigma, t)) < Q].$$

*Similarly, for any $t \geq 0$, let*

$$a(t) := \Pr[\pi \textit{ does not terminate by time } t] = \Pr[C(\pi, \Phi) > t] = \Pr[f(\Psi(\pi, t)) < Q].$$

See Figure 5.1 for an example of the non-completion probabilities $o(t)$. Clearly, $o(t)$ and $a(t)$ are non-increasing functions of $t$. Moreover, $o(0) = a(0) = 1$ and we have $o(t) = a(t) = 0$ for all $t \geq \sum_{e \in E} c_e$ (this is because any policy would have selected all items by this time).

It is easy to see that the expected cost of any policy is exactly the integral of the non-completion probabilities over time. That is,

$$c_{exp}(\sigma) = \int_0^\infty \Pr[\sigma \text{ does not terminate by time } t]dt = \int_0^\infty o(t)dt.$$

$$c_{exp}(\pi) = \int_0^\infty \Pr[\pi \text{ does not terminate by time } t]dt = \int_0^\infty a(t)dt.$$

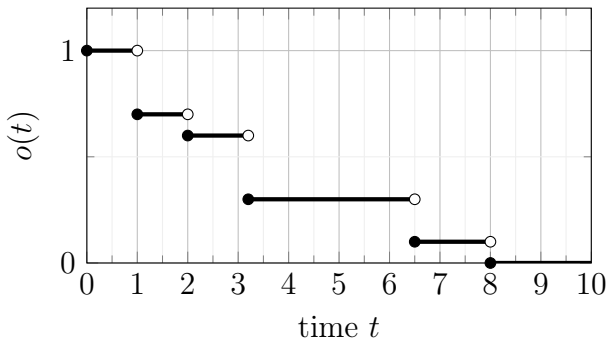



Figure 5.1: Graph of a simple $o(\cdot)$ function.

It turns out that we can also express the $p^{th}$ moment objective of any policy as a suitable integral of the non-completion probabilities. This relies on the following result.

**Lemma 14.** *Suppose that $X$ is a non-negative, discrete random variable with finite support. For any $p \geq 1$, the $p^{th}$ moment of $X$ is*

$$\mathbb{E}[X^p] = p \int_{t=0}^{\infty} t^{p-1} \cdot \Pr[X > t] dt.$$

*Proof.* Let $V$ denote the (finite) support of r.v. $X$. Note that $x^p = p \cdot \int_0^x t^{p-1} dt$ for all $x \geq 0$. So, we obtain

$$\begin{aligned}\mathbb{E}[X^p] &= \sum_{x \in V} \Pr[X = x] \cdot x^p = p \sum_{x \in V} \Pr[X = x] \int_{t=0}^{x} t^{p-1} dt \\ &= p \int_0^{\infty} t^{p-1} \sum_{x \in V : x > t} \Pr[X = x] dt = p \int_0^{\infty} t^{p-1} \cdot \Pr[X > t] dt.\end{aligned}$$

The third equality above switches the order of the integral and summation using Fubini's theorem. □

Now, we apply Lemma 14 with random variable $X = C(\sigma, \Phi)$ where $\sigma$ is the optimal policy and $\Phi$ denotes the r.v.s in the $\mathsf{ASC}$ instance. Note that the cost $C(\sigma, \Phi)$ is non-negative, discrete and bounded. Using the fact that $o(t) = \Pr[C(\sigma, \Phi) > t]$, we obtain:

$$c_p(\sigma) \;=\; \mathbb{E}_{\Phi}\left[C(\sigma, \Phi)^p\right] \;=\; p \int_0^{\infty} t^{p-1} \cdot o(t) dt. \tag{5.2}$$

Similarly, applying Lemma 14 to the r.v. $C(\pi, \Phi)$ for the greedy policy $\pi$,

$$c_p(\pi) \;=\; \mathbb{E}_{\Phi}\left[C(\pi, \Phi)^p\right] \;=\; p \int_0^{\infty} t^{p-1} \cdot a(t) dt. \tag{5.3}$$

### 5.2.2 Using the greedy criterion

Our analysis of $\pi$ relies on tracking the "greedy criterion value" defined in (5.1). The following definition formalizes this.

**Definition 7** (Greedy score)**.** *For $t \geq 0$ and any partial realization $\psi$ observed at time $t$, define*

$$score(t, \psi) := \begin{cases} \frac{\Delta(e|\psi)}{c_e[Q - f(\psi)]}, & \text{where } e \text{ is the item being selected in } \pi \text{ at time } t \\ & \text{and } \psi \text{ was observed just before selecting } e. \\ 0, & \text{if no item is being selected in } \pi \text{ at time } t \text{ when } \psi \text{ was observed.} \end{cases}$$

*Note that conditioned on $\psi$, the item $e$ being selected in $\pi$ at time $t$ is deterministic.*

The expression for *score* above is exactly the greedy criterion in (5.1). Moreover, the score may increase and decrease over time: see Figure 5.2 for an example, where $e_1, e_2, \ldots$ are greedy selections and $\psi_i$ is the partial realization just before selecting $e_i$.

In order to reduce notation, for any time $t$, we use $\Psi_t := \Psi(\pi, t)$ to denote the (random) partial realization observed by the greedy policy $\pi$ at time $t$; recall that this only includes items that have been completely selected by time $t$.

**Definition 8.** *(Potential gain) For any time $i \geq 0$, its potential* **gain** *is the expected total score accumulated after time $i$,*

$$G_i := \int_{t=i}^{\infty} \mathbb{E}\left[score(t, \Psi_t)\right] dt.$$

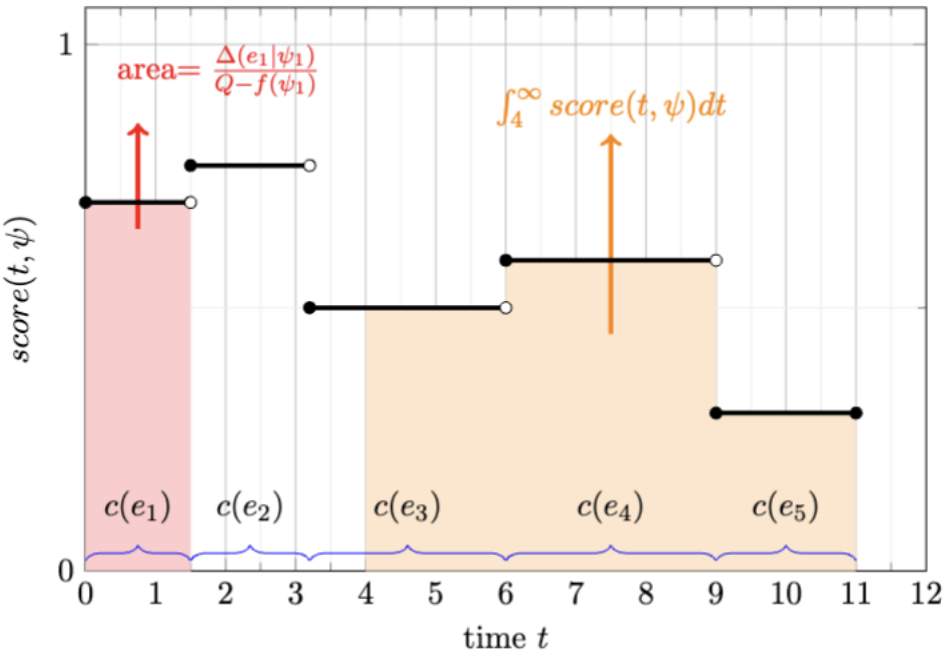



Figure 5.2: Graph of a simple $score(t, \psi)$ for illustration.

The key part of the analysis lies in upper and lower bounding the potential gain starting from all time points. The upper-bound relates to the non-completion probabilities $a(\cdot)$ in $\pi$ and the lower-bound relates to the non-completion probabilities $o(\cdot)$ in $\sigma$. Putting the upper and lower bound together allows us to relate $a(\cdot)$ and $o(\cdot)$, which in turn will be used to upper-bound $c_p(\pi)$ in terms of $c_p(\sigma)$. For the upper-bound (Lemma 15), we view gains as the expected values over full-realizations, which allows us to write the total (conditional) gain as a harmonic series. For the lower-bound (Lemma 17), we view the gain as an integral

of expected contributions over time and prove a lower bound for each time step using the adaptive-submodularity property.

Below, let $L := 1 + \ln(Q/\eta)$ and $\beta > 1$ be some constant value (that will be fixed later).

Below, we upper bound the total score for a single realization $\phi$. A similar fact and proof was used previously in [HKP21, INvdZ16, AG11]. We use this to upper bound potential gain from time $i$.

**Lemma 15.** *For any $i \geq 0$, the potential gain at time $i$ is*

$$G_i = \int_i^\infty \mathbb{E}\left[score(t, \Psi_t)\right] dt \leq L \cdot a(i).$$

*Proof.* We start by re-expressing the score and gain in terms of the full realization. For any time $t \geq 0$ and full realization $\phi$, let

$$S(t,\phi) := \begin{cases} \frac{f(\psi \cup (e,\phi_e)) - f(\psi)}{c_e[Q - f(\psi)]}, & \text{where } e \text{ is the item being selected in } \pi \text{ at time } t \text{ under } \phi, \\ & \text{and } \psi \preccurlyeq \phi \text{ is the partial realization just before selecting } e. \\ 0, & \text{if no item is being selected in } \pi \text{ at time } t \text{ under } \phi. \end{cases}$$

Then, for any $t \geq 0$ and any partial realization $\psi$ observed at time $t$,

$$score(t, \psi) = \mathbb{E}_\Phi[S(t, \Phi) | \psi \preccurlyeq \Phi].$$

This uses the definition of $\Delta(e|\psi)$ and the fact that conditioned on $\psi$, the item $e$ (being selected at time $t$) is fixed. Hence, for any time-point $k \geq 0$, its potential gain

$$G_k = \int_k^\infty \mathbb{E}\left[score(t, \Psi_t)\right] dt = \int_k^\infty \mathbb{E}_{\Psi_t}\left[\mathbb{E}_\Phi[S(t, \Phi) | \Psi_t \preccurlyeq \Phi]\right] dt = \int_k^\infty \mathbb{E}\left[S(t, \Phi)\right] dt.$$

Now, fix time $i \geq 0$ and condition on any (full) realization $\phi$.

Case 1: suppose that $\pi$ under $\phi$ terminates before ($<$) time $i$. Then, $S(t, \phi) = 0$ for all $t \geq i$, and so:

$$G_i(\phi) = \int_i^\infty S(t, \phi) dt = 0$$

Case 2: suppose that $\pi$ under $\phi$ terminates after ($\geq$) time $i$.

$$G_i(\phi) = \int_i^\infty S(t, \phi) dt = \int_i^\infty S(t, \phi) dt \leq \int_0^\infty S(t, \phi) dt \leq L,$$

where the last inequality is by Lemma 16.

Note that case 2 above happens exactly with probability $a(i)$. So,

$$G_i = \mathbb{E}_\Phi \left[ G_i(\Phi) \,|\, \text{case 2 occurs under } \Phi \right] \cdot \Pr[\text{case 2 occurs}] \leq L \cdot a(i),$$

which completes the proof. □

**Lemma 16.** *For any (full) realization $\phi$,*

$$\int_0^\infty S(t, \phi) dt \leq L = \ln(Q/\eta) + 1$$

*Proof.* Under $\phi$, let $e_1, e_2, ..., e_k$ be the sequence of items selected by $\pi$, let $\psi_i$ be the partial realization just before selecting $e_i$, and define $f_i := f(\psi_i)$. Note that $0 \leq f_1 \leq f_2 \leq ... \leq f_{k+1} = Q$ by monotonicity and the assumption that $f$ is always covered by $\pi$. Moreover, we have $f_k \leq Q - \eta$ by the definition of $\eta$: otherwise we would have $f_k = Q$ and $\pi$ would terminate before selecting $e_k$.

Define a function $g : [0, \infty) \to \mathbb{R}$ by

$$g(x) := \begin{cases} \frac{1}{Q - f_i}, & \text{for } x \in [f_i, f_{i+1}) \text{ and any } i = 1, 2, ..., k-1, \\ 0, & \text{otherwise.} \end{cases} \tag{5.4}$$

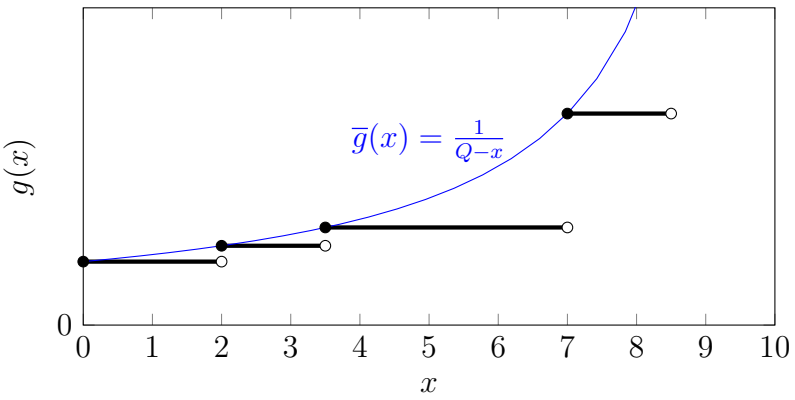



Figure 5.3: Example of $g(x)$

Note that $g(x) \leq \frac{1}{Q-x}$ for all $0 \leq x \leq Q - \eta$ (see Figure 5.3 for an example where $k = 5, Q = 10, \eta = 1$ and $f_1 = 0, f_2 = 2, f_3 = 3.5, f_4 = 7, f_5 = 8.5$). So,

$$\int_0^\infty S(t, \phi) dt = \sum_{i=1}^{k} \frac{f_{i+1} - f_i}{Q - f_i} = \sum_{i=1}^{k-1} \frac{f_{i+1} - f_i}{Q - f_i} + 1 = \int_0^\infty g(x) dx + 1 \leq \int_0^{Q-\eta} \frac{1}{Q - x} dx + 1 = L$$

The first equality uses the definition of $S(t, \phi)$ and the second equality uses $f_{k+1} = Q$. □

**Lemma 17.** *For any time $t \geq 0$,*

$$\mathbb{E}[score(t, \Psi_t)] \geq \frac{a(t) - o(t/L\beta)}{t/(L\beta)}.$$

*Hence, $G_i \geq \int_i^\infty \frac{a(t)-o(t/(L\beta))}{t/(L\beta)} dt$ for each time $i \geq 0$.*

*Proof.* Note that the first statement in the lemma immediately implies the second statement. Indeed,

$$G_i = \int_i^\infty \mathbb{E}[score(t, \Psi_t)]dt \geq \int_i^\infty \frac{a(t) - o(t/(L\beta))}{t/(L\beta)} dt.$$

We now prove the first statement. Henceforth, fix time $t \geq 0$.

**Truncated optimal policy.** Let $\overline{\sigma}$ denote the cost $t/(L\beta)$ truncation of policy $\sigma$. Note that the total cost of selected items in $\overline{\sigma}$ is always at most $t/(L\beta)$. However, $\overline{\sigma}$ may not fully cover the utility function $f$ (so it is not a feasible policy for min-cost adaptive submodular cover). We define the following random quantities associated with policy $\overline{\sigma}$:

$I_k :=$ set of first $k$ items selected by $\overline{\sigma}$, for $k = 0, 1, \cdots$.
$I_\infty :=$ set of *all* items selected by the end of $\overline{\sigma}$.
$P_k := \{(e, \Phi_e) : e \in I_k\}$, i.e. partial realization of the first $k$ items selected by $\overline{\sigma}$, for $k = 0, 1, \cdots$.
$P_\infty := \{(e, \Phi_e) : e \in I_\infty\}$, i.e. partial realization observed by the end of $\overline{\sigma}$.

Note that $\overline{\sigma}$ covers $f$ exactly when $f(P_\infty) = Q$. Moreover, by definition of the function $o(\cdot)$, we have $\Pr[\overline{\sigma} \text{ covers } f] = 1 - o(t/(L\beta))$.

**Conditioning on partial realizations in greedy.** Let $\psi$ be any partial realization corresponding to $\Psi_t$ with $f(\psi) < Q$. In other words, (i) $\psi$ is the partial realization observed at time $t$ in some execution of the greedy policy $\pi$, and (ii) the policy has not terminated (under realization $\psi$) by time $t$. Let $R(\pi, t)$ denote the collection of such partial realizations. Note that the partial realizations in $R(\pi, t)$ are mutually disjoint, and the total probability of these partial realizations equals the probability that $\pi$ does not terminate by time $t$. We will show that:

$$\Pr[\psi \preccurlyeq \Phi] \cdot score(t, \psi) \geq \frac{L\beta}{t} \cdot \Pr[(\psi \preccurlyeq \Phi) \wedge (\overline{\sigma} \text{ covers } f)], \qquad \forall \psi \in R(\pi, i). \tag{5.5}$$

We first complete the proof of the lemma assuming (5.5).

$$\begin{aligned}
\mathbb{E}[score(t,\Psi_t)] &\geq \sum_{\psi\in R(\pi,t)} p(\psi)\cdot score(t,\psi) = \sum_{\psi\in R(\pi,t)} \Pr[\psi \preccurlyeq \Phi]\cdot score(t,\psi) \\
&\geq \frac{L\beta}{t}\sum_{\psi\in R(\pi,t)} \Pr[(\psi\preccurlyeq\Phi)\wedge(\overline{\sigma}\text{ covers }f)] && (5.6)\\
&= \frac{L\beta}{t}\Pr\left[(\pi\text{ doesn't terminate by time }t)\wedge(\overline{\sigma}\text{ covers }f)\right] && (5.7)\\
&\geq \frac{L\beta}{t}\left(\Pr[\pi\text{ doesn't terminate by time }t] - \Pr[\overline{\sigma}\text{ does not cover }f]\right) && (5.8)\\
&= \frac{L\beta}{t}\cdot(a(t)-o(t/(\beta L))) && (5.9)
\end{aligned}$$

Inequality (5.6) is by (5.5). The equality in (5.7) uses the definition of $R(\pi,t)$. Inequality (5.8) is by a union bound. Equation (5.9) is by definition of the functions $a(\cdot)$ and $o(\cdot)$.

**Proof of** (5.5) Henceforth, fix any partial realization $\psi\in R(\pi,t)$. Our proof relies on the following quantity:

$$Z := \mathbb{E}_\Phi\left[\mathbf{1}(\psi\preccurlyeq\Phi)\cdot\frac{f(\psi\cup P_\infty)-f(\psi)}{Q-f(\psi)}\right] \tag{5.10}$$

This is the expected increase in policy $\overline{\sigma}$'s function value (relative to the "remaining" target $Q-f(\psi)$) when restricted to full realizations $\Phi$ that agree with partial realization $\psi$.

For any partial realization $\psi'$ such that $\psi\preccurlyeq\psi'$ and item $e\notin\mathsf{dom}(\psi')$, let $X_{e,\psi,\psi'}$ denote the indicator r.v. that policy $\overline{\sigma}$ selects item $e$ at some point when its observed realizations are precisely $(\psi'\setminus\psi)\cup\chi$ where $\chi\subseteq\psi$. That is, $X_{e,\psi,\psi'}=1$ if policy $\overline{\sigma}$ selects $e$ at a point where (i) all items in $\mathsf{dom}(\psi'\setminus\psi)$ have been selected and their realization is $\psi'\setminus\psi$, (ii) no item in $E\setminus\mathsf{dom}(\psi')$ has been selected, and (iii) if any item in $\mathsf{dom}(\psi)$ has been selected then its realization agrees with $\psi$. Note that conditioned on $\psi'\preccurlyeq\Phi$, $X_{e,\psi,\psi'}$ is a deterministic value: the realizations of items $\mathsf{dom}(\psi')$ are fixed by $\psi'$ and if any item in $E\setminus\mathsf{dom}(\psi')$ is selected (before $e$) then $X_{e,\psi,\psi'}=0$ irrespective of its realization.

We can write $Z$ as a sum of increments as follows:

$$
\begin{aligned}
Z &= \frac{1}{Q - f(\psi)} \mathbb{E}_\Phi \left[ \mathbf{1}(\psi \preccurlyeq \Phi) \cdot \sum_{k \geq 1} [f(\psi \cup P_k) - f(\psi \cup P_{k-1})] \right] \\
&= \frac{1}{Q - f(\psi)} \sum_{\psi' : \psi \preccurlyeq \psi'} \sum_{e \notin \mathsf{dom}(\psi')} \mathbb{E}_\Phi \left[ \mathbf{1}(\psi' \preccurlyeq \Phi) \cdot X_{e,\psi,\psi'} \cdot [f(\psi' \cup (e, \Phi_e)) - f(\psi')] \right] \\
&= \frac{1}{Q - f(\psi)} \sum_{\psi' : \psi \preccurlyeq \psi'} \sum_{e \notin \mathsf{dom}(\psi')} \Pr[\psi' \preccurlyeq \Phi \wedge X_{e,\psi,\psi'} = 1] \\
&\cdot \mathbb{E}_\Phi [f(\psi' \cup (e, \Phi_e)) - f(\psi') \,|\, (\psi' \preccurlyeq \Phi) \wedge (X_{e,\psi,\psi'} = 1)] \\
&= \frac{1}{Q - f(\psi)} \sum_{\psi' : \psi \preccurlyeq \psi'} \sum_{e \notin \mathsf{dom}(\psi')} \Pr[\psi' \preccurlyeq \Phi \wedge X_{e,\psi,\psi'} = 1] \cdot \mathbb{E}_\Phi [f(\psi' \cup (e, \Phi_e)) - f(\psi') \,|\, \psi' \preccurlyeq \Phi] \qquad (5.11) \\
&= \frac{1}{Q - f(\psi)} \sum_{\psi' : \psi \preccurlyeq \psi'} \sum_{e \notin \mathsf{dom}(\psi')} \Pr[\psi' \preccurlyeq \Phi \wedge X_{e,\psi,\psi'} = 1] \cdot \Delta(e|\psi') \\
&\leq \frac{1}{Q - f(\psi)} \sum_{\psi' : \psi \preccurlyeq \psi'} \sum_{e \notin \mathsf{dom}(\psi')} \Pr[\psi' \preccurlyeq \Phi \wedge X_{e,\psi,\psi'} = 1] \cdot \Delta(e|\psi) \qquad (5.12) \\
&= \sum_{\psi' : \psi \preccurlyeq \psi'} \sum_{e \notin \mathsf{dom}(\psi')} \Pr[\psi' \preccurlyeq \Phi \wedge X_{e,\psi,\psi'} = 1] \cdot c_e \cdot \frac{\Delta(e|\psi)}{c_e(Q - f(\psi))} \\
&\leq \sum_{\psi' : \psi \preccurlyeq \psi'} \sum_{e \notin \mathsf{dom}(\psi')} \Pr[\psi' \preccurlyeq \Phi \wedge X_{e,\psi,\psi'} = 1] \cdot c_e \cdot score(t, \psi) \qquad (5.13) \\
&= score(t, \psi) \sum_{\psi' : \psi \preccurlyeq \psi'} \sum_{e \notin \mathsf{dom}(\psi')} c_e \cdot \Pr[\psi' \preccurlyeq \Phi \wedge X_{e,\psi,\psi'} = 1] \\
&= score(t, \psi) \cdot \sum_{e \in E \setminus \mathsf{dom}(\psi)} c_e \cdot \sum_{\substack{\psi' : \psi \preccurlyeq \psi' \\ \mathsf{dom}(\psi') \not\ni e}} \Pr[\psi' \preccurlyeq \Phi \wedge X_{e,\psi,\psi'} = 1] \\
&= score(t, \psi) \cdot \sum_{e \in E \setminus \mathsf{dom}(\psi)} c_e \cdot \Pr[\psi \preccurlyeq \Phi \wedge e \in I_\infty] \qquad (5.14) \\
&\leq score(t, \psi) \cdot \mathbb{E}_\Phi \left[ \mathbf{1}(\psi \preccurlyeq \Phi) \cdot \sum_{e \in I_\infty} c_e \right] \\
&\leq score(t, \psi) \cdot (t/(L\beta)) \cdot \mathbb{E}_\Phi[\mathbf{1}(\psi \preccurlyeq \Phi)] \;=\; score(t, \psi) \cdot (t/(L\beta)) \cdot \Pr[\psi \preccurlyeq \Phi]. \qquad (5.15)
\end{aligned}
$$

The equality (5.11) uses the fact that $X_{e,\psi,\psi'}$ is deterministic when conditioned on $\psi' \preccurlyeq \Phi$. Inequality (5.12) is by adaptive submodularity. (5.13) is by the greedy selection criterion. The inequality in (5.15) uses the fact that the total cost of $\overline{\sigma}$'s selections is always bounded above by $t/(L\beta)$. Equation (5.14) uses the definition of $I_\infty$ (all selected items in $\overline{\sigma}$) and the

following identity:

$$\sum_{\substack{\psi':\psi\preccurlyeq\psi' \\ \mathsf{dom}(\psi')\not\ni e}} \mathbf{1}(\psi' \preccurlyeq \Phi)\cdot X_{e,\psi,\psi'} = \mathbf{1}(\psi \preccurlyeq \Phi \wedge e \in I_\infty), \qquad \forall e \in E \setminus \mathsf{dom}(\psi).$$

To see this, condition on any full realization $\phi$. If $\psi \not\preccurlyeq \phi$ then both the left-hand-side ($LHS$) and right-hand-side ($RHS$) are 0. If $\psi \preccurlyeq \phi$ and $e$ is not selected by $\overline{\sigma}$ under $\phi$, then again $LHS = RHS = 0$. If $\psi \preccurlyeq \phi$ and $e$ is selected by $\overline{\sigma}$ under $\phi$, then $RHS = 1$ and $LHS$ is the sum of $X_{e,\psi,\psi'}$ over $\psi'$ such that $\psi \preccurlyeq \psi' \preccurlyeq \phi$ and $e \notin \mathsf{dom}(\psi')$. In this case, $X_{e,\psi,\psi'} = 1$ for exactly one such partial realization $\psi'$, namely $\psi' = \psi \cup \kappa$ where $\kappa \preccurlyeq \phi$ is the partial realization immediately before $e$ is selected. So, $LHS = RHS$ in all cases.

Note that whenever $\overline{\sigma}$ covers $f$, we have $f(P_\infty) = Q$. Combined with the monotone property of $f$, we have $f(\psi \cup P_\infty) = Q$ whenever $\overline{\sigma}$ covers $f$. So, we have:

$$Z \geq \Pr[(\psi \preccurlyeq \Phi) \wedge (\overline{\sigma} \text{ covers } f)].$$

Combining the above inequality with (5.15) finishes the proof of (5.5).

### 5.2.3 Wrapping up

We are now ready to complete the proof of Theorem 13. Using Lemmas 15 and 17, we get:

$$a(i)L \;\geq\; G_i \;\geq\; L\beta \int_i^\infty \frac{a(t) - o(t/(L\beta))}{t} dt, \qquad \forall i \geq 0.$$

Multiplying this inequality by $i^{p-1}$ and integrating over all $i \geq 0$, we obtain:

$$\int_0^\infty i^{p-1} a(i) di \;\geq\; \beta \cdot \int_{i=0}^\infty i^{p-1} \int_{t=i}^\infty \frac{a(t) - o(t/(L\beta))}{t}\, dt\, di. \tag{5.16}$$

Now, using (5.3), the $p^{th}$ moment of the greedy cost is

$$c_p(\pi) = p \cdot \int_0^\infty i^{p-1} a(i) di \;\geq\; \beta p \cdot \int_{i=0}^\infty i^{p-1} \int_{t=i}^\infty \frac{a(t) - o(t/(L\beta))}{t} \, dt \, di \tag{5.17}$$

$$= \beta p \cdot \int_{t=0}^\infty \frac{a(t) - o(t/(L\beta))}{t} \int_{i=0}^t i^{p-1} \, di \, dt \;=\; \beta \cdot \int_0^\infty \frac{a(t) - o(t/L\beta)}{t} \cdot t^p dt \tag{5.18}$$

$$= \beta \cdot \int_0^\infty t^{p-1} \cdot a(t) dt \;-\; \beta \cdot \int_0^\infty t^{p-1} \cdot o(t/L\beta) dt$$

$$= \frac{\beta}{p} \cdot c_p(\pi) \;-\; \beta \cdot \int_0^\infty t^{p-1} \cdot o(t/L\beta) dt \tag{5.19}$$

$$= \frac{\beta}{p} \cdot c_p(\pi) \;-\; \beta (L\beta)^p \cdot \int_{y=0}^\infty y^{p-1} o(y) dy \;=\; \frac{\beta}{p} \cdot c_p(\pi) \;-\; \frac{\beta}{p} \cdot (L\beta)^p \cdot c_p(\sigma). \tag{5.20}$$

The inequality in (5.17) is by (5.16). The first equality in (5.18) is by interchanging the order of the integrals (by Fubini's theorem). Equality (5.19) uses (5.3) for $c_p(\pi)$. The first equality in (5.20) is by a change of variable $y = \frac{t}{L\beta}$ in the integral, and the last equality uses (5.2) for $c_p(\sigma)$.

It now follows that

$$\left(\frac{\beta}{p} - 1\right) \cdot c_p(\pi) \quad \leq \quad \frac{\beta}{p} \cdot (L\beta)^p \cdot c_p(\sigma).$$

In order to minimize the approximation ratio, we choose $\beta = p + 1$ (note that this is only used in the analysis), which implies:

$$c_p(\pi) \quad \leq \quad (p+1)^{p+1} \cdot L^p \cdot c_p(\sigma).$$

This completes the proof of Theorem 13. Setting $p = 1$, we also obtain Theorem 12.

## 5.3 Applications

Here, we provide some concrete applications of our framework. These applications were already discussed in [GK17], but as noted in Section 5.1.1, the function definition in ASC is slightly more restrictive than the framework in [GK17].

**Stochastic Submodular Cover.** In this problem, there are $n$ stochastic items (for example, corresponding to sensors). Each item $e$ can be in one of many "states", and this state is observed only after selecting item $e$. E.g., the state of a sensor indicates the extent to which it is working. The states of different items are independent. There is a utility function

$\hat{f} : 2^{E\times\Omega} \to \mathbb{R}_{\geq 0}$, where $E$ is the set of items and $\Omega$ the set of states. It is assumed that $\hat{f}$ is monotone and submodular. For example, $\hat{f}$ quantifies the information gained from a set of sensors having arbitrary states. Each item $e$ is also associated with a cost $c_e$. Given a quota $Q$, the goal is to select items sequentially to achieve utility at least $Q$, at the minimum expected cost. We assume that the quota $Q$ can always be achieved by selecting adequately many items, i.e., $\hat{f}(\{(e,\phi_e) : e \in E\}) \geq Q$ for all possible states $\{\phi_e \in \Omega\}_{e\in E}$ for the items. This is a special case of ASC, where the items $E$ and states (outcomes) $\Omega$ remain the same. We define a new utility function $f(\psi) = \min\left\{\hat{f}(\psi), Q\right\}$ for all $\psi \subseteq E \times \Omega$. Note that $Q$ is the maximal value of function $f$ and this value is achieved under every possible (full) realization. Moreover, $f$ is also monotone and submodular. Clearly, the monotonicity property (Definition 2) holds. The adaptive-submodularity property also holds because the items are independent. Indeed, for any partial realizations $\psi \preccurlyeq \psi'$ and $e \in E \setminus \mathsf{dom}(\psi')$,

$$\begin{aligned}
\Delta(e|\psi) &= \sum_{\omega\in\Omega} \Pr[\Phi_e = \omega | \psi \preccurlyeq \Phi] \cdot (f(\psi \cup (e,\omega)) - f(\psi)) \\
&= \sum_{\omega\in\Omega} \Pr[\Phi_e = \omega] \cdot (f(\psi \cup (e,\omega)) - f(\psi)) \\
&\geq \sum_{\omega\in\Omega} \Pr[\Phi_e = \omega] \cdot (f(\psi' \cup (e,\omega)) - f(\psi')) \\
&= \sum_{\omega\in\Omega} \Pr[\Phi_e = \omega | \psi' \preccurlyeq \Phi] \cdot (f(\psi' \cup (e,\omega)) - f(\psi')) = \Delta(e|\psi').
\end{aligned}$$

In particular, when $\hat{f}$ is integer-valued, Theorem 12 implies a $4(1+\ln Q)$-approximation algorithm. We note that [HKP21] obtained a $(1+\ln Q)$-approximation ratio, using a different analysis. The latter bound is the best possible, including the constant factor, as the problem generalizes set cover. However, our approach is more versatile and also provides a $(p+1)^{p+1} \cdot (1+\ln Q)^p$ approximation bound for the $p^{th}$ moment objective. Our result is the first approximation algorithm for higher moments ($p > 1$).

**Adaptive Viral Marketing.** This problem is defined on a directed graph $G = (V, A)$ representing a social network [KKT15]. Each node $v \in V$ represents a user. Each arc $(u, v) \in A$ is associated with a random variable $X_{uv} \in \{0, 1\}$. The r.v. $X_{uv} = 1$ if $u$ will influence $v$ (assuming $u$ itself is influenced); we also say that arc $(u, v)$ is *active* in this case. The r.v.s $X_{uv}$ are independent, and we are given the means $\mathbb{E}[X_{uv}] = p_{uv}$ for all $(u, v) \in A$. When a node $u$ is activated/influenced, all arcs $(u, v)$ out of $u$ are observed and if $X_{uv} = 1$ then $v$ is also activated. This process then continues on $u$'s neighbors to their neighbors and so on, until no new node is activated. We consider the "full feedback" model, where after

activating a node $w$, we observe the $X_{uv}$ r.v.s on all arcs $(u, v)$ such that $u$ is reachable from $w$ via a path of active arcs. Further, each node $v$ has a cost $c_v$ corresponding to activating node $v$ *directly*, e.g. by providing some promotional offer. Note that there is no cost incurred on $v$ if it is activated (indirectly) due to a neighbor $u$ with $X_{uv} = 1$. Given a quota $Q$, the goal is to activate at least $Q$ nodes at the minimum expected cost.

To model this as ASC, the items $E = V$ are all nodes in $G$. We add self-loops $A_o = \{(v, v) : v \in V\}$ that represent whether a node is activated directly. So, the new set of arcs is $A' = A \cup A_o$. The outcome $\Phi_w$ of any node $w \in V$ is represented by a function $\phi_w : A' \to \{0, 1, ?\}$ where $\phi_w((w, w)) = 1$, $\phi_w((v, v)) = 0$ for all $v \in V \setminus w$, and for any $(u, v) \in A$:

- $\phi_w(u, v) = 1$ if there is a $w - u$ path of active arcs and $X_{uv} = 1$ (i.e., $(u, v)$ is active).
- $\phi_w(u, v) = 0$ if there is a $w - u$ path of active arcs and $X_{uv} = 0$ (i.e., $(u, v)$ is not active).
- $\phi_w(u, v) =?$ if there is no $w-u$ path of active arcs (so, the status of $(u, v)$ is unknown).

Let $\Omega$ denote the collection of all such functions: this represents the outcome space. Note that $\Phi_w$ depends on the entire network (and not just node $w$). So, the r.v.s $\{\Phi_w\}_{w \in V}$ may be highly correlated. Observe that $\Phi_w$ is exactly the feedback obtained when node $w$ is activated directly (by incurring cost $c_w$) at any point in a policy. Define function $\bar{f} : 2^{E \times \Omega} \to \mathbb{R}_{\geq 0}$ as:

$$\bar{f}(\psi) = \sum_{v \in V} \min \left\{ \sum_{u:(u,v) \in A'} |\{w \in \mathsf{dom}(\psi) : \psi_w(u, v) = 1\}| \, , \, 1 \right\}. \tag{5.21}$$

$\bar{f}$ is a sum of set-coverage functions, which is monotone and submodular. Then, utility function $f : 2^{E \times \Omega} \to \mathbb{R}_{\geq 0}$ is $f(\psi) = \min\{\bar{f}(\psi), Q\}$. Function $f$ is clearly monotone (Definition 2). The adaptive-submodularity property also holds: see Theorem 19 in [GK17].

Hence, Theorem 12 implies a $4(1 + \ln Q)$-approximation algorithm for adaptive viral marketing. This is an improvement over previous approximation ratios of $(1+\ln Q)^2$ [GK17] and $(1 + \ln(nQc_{max}))$ [EKM21], where $n = |V|$ and $c_{max}$ is the maximum cost. We also obtain the first approximation algorithm for the $p^{th}$ moment objective.

**Optimal Decision Tree (uniform prior).** In this problem, there are $m$ hypotheses $H$ and $n$ binary tests $E$. Each test $e \in E$ costs $c_e$, and has a positive outcome on some subset $T_e \subseteq H$ of hypotheses and a negative outcome on the other hypotheses. (Our results also extend to the case of multiway tests with non-binary outcomes.) An unknown hypothesis $h^*$

is drawn from $H$ uniformly at random. The goal is to identify $h^*$ by sequentially performing tests, at minimum expected cost. This is a special case of ASC, where the items correspond to tests $E$ and the outcome space $\Omega = \{+, -\}$. The outcome $\Phi_e$ for any item $e$ is the test outcome under the (unknown) hypothesis $h^*$. For any test $e \in E$, define subsets $S_{e,+} = H \setminus T_e$ and $S_{e,-} = T_e$, corresponding to the hypotheses that can be eliminated when we observe a positive or negative outcome on $e$. The utility function is

$$f(\psi) = \frac{1}{|H|} \cdot \Big| \bigcup_{e \in \mathsf{dom}(\psi)} S_{e,\psi_e} \Big|.$$

The quota $Q = 1 - \frac{1}{|H|}$. Achieving value $Q$ means that $|H| - 1$ hypotheses have been eliminated, which implies that $h^*$ is identified. The function $f$ is again monotone and submodular. The monotonicity property (Definition 2) clearly holds. Moreover, using the fact that $h^*$ has a *uniform distribution*, $f$ is adaptive-submodular: see Lemma 23 in [GK17].

So, Theorem 12 implies a $4(1+\ln(|H|-1))$-approximation algorithm for this problem; we use $Q$ as above and $\eta = \frac{1}{|H|}$. The previous-best bounds for this problem were $(1 + \ln(|H| - 1))^2$ [GK17], $12 \cdot \ln |H|$ [GB09] and $(1 + \ln(n|H|c_{max}))$ [EKM21]. Again, our result is the first approximation algorithm for this problem under $p^{th}$ moment objectives.

We note that [GK17] also obtained a $\left(\ln \frac{1}{p_{min}}\right)^2$-approximation for the optimal decision tree problem with arbitrary priors (where the distribution of $h^*$ is not uniform); here $p_{min} \leq \frac{1}{|H|}$ is the minimum probability of any hypothesis. This uses a different utility function that falls outside our ASC framework (as our definition of function $f$ is more restrictive). Moreover, there are other approaches [GNR17, NKN20] that provide a better $O(\ln |H|)$-approximation bound even for the problem with arbitrary priors.

## 5.4 Adaptive Submodular Cover with Multiple Functions

Here, we extend ASC to the setting of covering multiple adaptive-submodular functions. In the multiple adaptive-submodular cover (MASC) problem, there is a set $E$ of items and outcome space $\Omega$ as before. Each item $e \in E$ has a cost $c_e$; we will view this cost as the item's *processing time*. Now, there are $k$ different utility functions $f_r : 2^{E \times \Omega} \to \mathbb{R}_{\geq 0}$ for $r \in [k]$. We assume that each of these functions satisfies the monotonicity, coverability and adaptive-submodularity properties. We also assume, without loss of generality (by scaling), that the maximal value of each function $\{f_r\}_{r=1}^k$ is $Q$. As for the basic ASC problem, a solution to MASC corresponds to a policy $\pi : 2^{E \times \Omega} \to E$, that maps partial realizations to

the next item to select. Given any policy $\pi$, the *cover time* of function $f_r$ is defined as:

$$C_r(\pi) := \text{the earliest time } t \text{ such that } f_r(\Psi(\pi, t)) = Q.$$

Recall that $\Psi(\pi, t) \subseteq E \times \Omega$ is the partial realization that has been observed by time $t$ in policy $\pi$. Note that the cover time is a random quantity. The expected cost objective in MASC is the expected total cover time of all functions, i.e., $\sum_{r=1}^{k} \mathbb{E}[C_r(\pi)]$. More generally, for any $p \geq 1$, the $p^{th}$ moment objective of policy $\pi$ is $\widetilde{c}_p(\pi) := \sum_{r=1}^{k} \mathbb{E}\left[C_r(\pi)^p\right]$. When we have just $k = 1$ function, the MASC problem reduces to ASC.

**Remark:** One might also consider an alternative multiple-function formulation where we are interested in the expected *maximum* cover time of the functions. This formulation can be directly solved as an instance of ASC where we use the single adaptive-submodular function $g = \sum_{r=1}^{k} f_r$ with maximal value $Q' = kQ$.

We extend the greedy policy for ASC to MASC, as described in Algorithm 6. For each $r \in [k]$ and item $e \in E$, we use $\Delta_r(e|\psi)$ to denote the marginal benefit of $e$ under function $f_r$. Notice that the greedy selection criterion here involves a sum of terms corresponding to each un-covered function. A similar greedy rule was used earlier in the (deterministic) submodular function ranking problem [AG11] and in the independent special case of MASC in [INvdZ16].

**Algorithm 6** Adaptive Greedy Policy $\pi$ for MASC.

selected items $A \leftarrow \emptyset$, observed realizations $\psi \leftarrow \emptyset$
**while** there exists function $f_r$ with $f_r(\psi) < Q$ **do**
    select item

$$e^* = \operatorname*{argmax}_{e \in E \setminus A} \frac{1}{c_e} \cdot \sum_{r \in [k]: f_r(\psi) < Q} \frac{\Delta_r(e|\psi)}{Q - f_r(\psi)},$$

    and observe $\Phi_{e^*}$
    add $e^*$ to the selected items, i.e., $A \leftarrow A \cup \{e^*\}$
    update $\psi \leftarrow \psi \cup \{(e^*, \Phi_{e^*})\}$

**Theorem 18.** *Consider any instance of adaptive-submodular cover with $k$ utility functions, where each function $f_r : 2^{E \times \Omega} \to \mathbb{R}_{\geq 0}$ is monotone, coverable and adaptive-submodular w.r.t. the same probability distribution $q(\cdot)$. Suppose that there is some value $\eta > 0$ such that $f_r(\psi) > Q - \eta$ implies $f_r(\psi) = Q$ for all partial realizations $\psi \subseteq E \times \Omega$ and $r \in [k]$. Then, for every $p \geq 1$, the $p^{th}$ moment of the cost of the greedy policy is*

$$\widetilde{c}_p(\pi) \leq (p+1)^{p+1} \cdot (1 + \ln(Q/\eta)) \cdot \widetilde{c}_p(\sigma_p),$$

*where $\sigma_p$ is the optimal* MASC *policy for the $p^{th}$ moment objective.*

In particular, setting $p = 1$, we obtain a $4(1 + \ln(Q/\eta))$ approximation algorithm for MASC with the expected cost objective. The proof of Theorem 18 is a natural extension of Theorem 13 for ASC. We use the same notations and definitions if not mentioned explicitly.

**Definition 9** (MASC non-completion probabilities)**.** *For any time $t \geq 0$ and $r \in [k]$, let*

$$\tilde{o}_r(t) := \Pr[\sigma \text{ does not cover } f_r \text{ by time } t] = \Pr[C_r(\sigma) > t] = \Pr\left[f_r(\Psi(\sigma, t)) < Q\right].$$
$$\tilde{a}_r(t) := \Pr[\pi \text{ does not cover } f_r \text{ by time } t] = \Pr[C_r(\pi) > t] = \Pr\left[f_r(\Psi(\pi, t)) < Q\right].$$

*Also, define for any $t \geq 0$, $\tilde{o}(t) := \sum_{r=1}^{k} \tilde{o}_r(t)$ and $\tilde{a}(t) := \sum_{r=1}^{k} \tilde{a}_r(t)$.*

For each $r \in [k]$, the functions $\tilde{o}_r(t)$ and $\tilde{a}_r(t)$ are non-increasing functions of $t$; moreover, $\tilde{o}_r(0) = \tilde{a}_r(0) = 1$ and $\tilde{o}_r(t) = \tilde{a}_r(t) = 0$ for all $t \geq \sum_{e \in E} c_e$ (this is because any policy would have selected all items by this time). So, the functions $\tilde{o}_r(t)$ and $\tilde{a}_r(t)$ share the same properties as $o(t)$ and $a(t)$ for ASC. As in (5.3) and (5.2), we can express the $p^{th}$ moment objective in MASC as:

$$\widetilde{c}_p(\sigma) = p \int_0^\infty t^{p-1} \cdot \tilde{o}(t) dt, \quad \text{and} \quad \widetilde{c}_p(\pi) = p \int_0^\infty t^{p-1} \cdot \tilde{a}(t) dt.$$

**Definition 10** (MASC greedy score)**.** *For any $t \geq 0$ and partial realization $\psi$ observed at time $t$, we define*

$$\widetilde{score}(t, \psi) := \frac{1}{c_e} \sum_{r \in [k]: f_r(\psi) < Q} \frac{\Delta_r(e|\psi)}{Q - f_r(\psi)} \quad \text{where } e \text{ is the item being selected in } \pi \text{ at time } t \text{ and } \psi \text{ was observed just before selecting } e,$$

*and $\widetilde{score}(t, \psi) := 0$ if no item is being selected in $\pi$ at time $t$ when $\psi$ was observed.*

**Definition 11** (MASC potential gain)**.** *For any time $i \geq 0$, its potential gain is the expected total score accumulated after time $i$,*

$$\tilde{G}_i := \int_i^\infty \mathbb{E}\left[\widetilde{score}(t, \Psi_t)\right] dt$$

The next two lemmas lower and upper bound the potential gain.

**Lemma 19.** *For any $i \geq 0$*

$$\tilde{G}_i \leq \sum_{r \in [k]} L \cdot \tilde{a}_r(t) = L \cdot \tilde{a}(t).$$

Lemma 19 follows by applying Lemma 15 to each $f_r$ and adding over $r \in [k]$.

**Lemma 20.** *For any $t \geq 0$ ,*

$$\mathbb{E}[\widetilde{score}(t, \Psi_t)] \geq \sum_{r\in[k]} \frac{\tilde{a}_r(t) - \tilde{o}_r(t/(L\beta))}{t/(L\beta)} = \frac{\tilde{a}(t) - \tilde{o}(t/(L\beta))}{t/(L\beta)}.$$

*Hence, $\tilde{G}_i \geq L\beta \int_i^\infty \left(\frac{\tilde{a}(t)-\tilde{o}(t/(L\beta)))}{t}\right) dt$ for each time $i \geq 0$.*

*Proof.* Proof Outline. We replicate all the steps in Lemma 17, except that we redefine $Z$ to be

$$\tilde{Z} := \mathbb{E}_\Phi \left[ \mathbf{1}(\psi \preccurlyeq \Phi) \cdot \sum_{r\in[k]:f_r(\psi)<Q} \frac{f_r(\psi \cup P_\infty) - f_r(\psi)}{Q - f_r(\psi)} \right],$$

which takes all $k$ functions into account. Here, $\psi$ is any partial realization observed at time $t$. We then obtain:

$$\widetilde{score}(t, \psi) \cdot \frac{t}{L\beta} \cdot \Pr[\psi \preccurlyeq \Phi] \geq \tilde{Z} \geq \sum_{r\in[k]:f_r(\psi)<Q} \Pr[(\psi \preccurlyeq \Phi) \wedge (\overline{\sigma} \text{ covers } f_r)].$$

Let $R(\pi, t)$ denote *all* the possible partial realizations observed at time $t$ in policy $\pi$. Then,

$$\begin{aligned}
\mathbb{E}[\widetilde{score}(t, \Psi_t)] &= \sum_{\psi\in R(\pi,t)} \Pr[\psi \preccurlyeq \Phi] \cdot \widetilde{score}(t, \psi) \\
&\geq \frac{L\beta}{t} \sum_{\psi\in R(\pi,t)} \sum_{r\in[k]:f_r(\psi)<Q} \Pr[(\psi \preccurlyeq \Phi) \wedge (\overline{\sigma} \text{ covers } f_r)] \\
&= \frac{L\beta}{t} \sum_{r\in[k]} \Pr\left[(\pi \text{ doesn't cover } f_r \text{ by time } t) \wedge (\overline{\sigma} \text{ covers } f_r)\right] \\
&\geq \frac{L\beta}{t} \sum_{r\in[k]} \left(\Pr[\pi \text{ doesn't cover } f_r \text{ by time } t] - \Pr[\overline{\sigma} \text{ doesn't cover } f_r]\right) \\
&= \frac{L\beta}{t} \sum_{r\in[k]} \left(\tilde{a}_r\left(t\right) - o_r(t/(L\beta))\right) \geq L\beta \frac{\tilde{a}(t) - \tilde{o}(t/(L\beta))}{t}.
\end{aligned}$$

□

Finally, we combine Lemma 19 and Lemma 20, as in Theorem 13, to obtain:

$$\widetilde{c}_p(\pi) \leq \frac{L^p \beta^{p+1}}{\beta - p} \cdot \widetilde{c}_p(\sigma).$$

Setting $\beta = p + 1$ to minimize the approximation ratio, we obtain Theorem 18.

We now list some applications of the MASC result.

- When items are deterministic, MASC reduces to the deterministic submodular ranking problem, for which an $O(\ln(Q/\eta))$ was obtained in [AG11]. We note that the result in [AG11] was only for unit costs, whereas our result holds for arbitrary costs. This problem generalizes the min-sum set cover problem, which is NP-hard to approximate better than factor 4 ([FLT04]). For min-sum set cover, the paramters $Q = \eta = 1$: so Theorem 18 implies a *tight* 4-approximation algorithm for it.

- When the outcomes are independent across items, MASC reduces to stochastic submodular cover with multiple functions, which was studied in [INvdZ16]. We obtain an $4 \cdot (1 + \ln(Q/\eta))$ approximation ratio that improves the bound of $56 \cdot (1 + \ln(Q/\eta))$ in [INvdZ16] by a constant factor. Although [INvdZ16] did not try to optimize the constant factor, their approach seems unlikely to provide such a small constant factor.

- Consider the following generalization of adaptive viral marketing. Instead of a single quota on the number of influenced nodes, there are $k$ different quotas $Q_1 \leq Q_2 \leq \cdots Q_k$. Now, we want a policy such that the *average* expected cost for achieving these quotas is minimized. Recall the function $\bar{f}$ defined in (5.21) for the single-quota problem. Then, corresponding to the different quotas, define functions $f_r(\psi) = \frac{1}{Q_r} \cdot \min\{\bar{f}(\psi), Q_r\}$ for each $r \in [k]$. Each of these functions is monotone, adaptive-submodular and has maximal value $Q = 1$. The parameter $\eta = 1/Q_k$, so we obtain a $4(1 + \ln Q_k)$-approximation algorithm.

## 5.5 Computational Results

In this section, we empirically evaluate the performance of the greedy policy on a real-world dataset of Optimal Decision Tree (ODT), which is one of the applications of ASC (see Section 5.3). This dataset has been used in a number of previous papers on ODT, see e.g., [BBS12, BAD+07, NKN20]. We also provide a lower-bound for the $p^{th}$ moment objective in ODT, and compare the greedy policy's performance to this bound.

Recall that in ODT, there are $m$ hypotheses (with uniform probabilities) and $n$ binary tests. We assume that all tests have unit costs. The goal is to identify the realized hypothesis by performing sequential tests. The objective is to minimize the $p^{th}$ moment of the testing cost. The case $p = 1$ is the usual expected cost objective, which has been analyzed before. We also consider higher moments with $p = 2, 3$, for which our result provides the first approximation ratio.

**Information theoretic lower bound for ODT.** It is well-known that the expected testing cost for any instance of ODT (with uniform probability and cost) is at least $\log_2(m)$. This follows from the entropy lower bound for binary coding. In fact, we can get a stronger lower bound using Huffman coding ([Huf52]), which corresponds to the ODT instance with $m$ hypotheses and all possible binary tests. Viewed this way, any policy for this ODT instance is a binary tree $T$ with $m$ leaves $L(T)$: the $p^{th}$ moment objective is then $\frac{1}{m}\sum_{i\in L(T)} d(i)^p$ where $d(i)$ is the depth of leaf $i$. Below, we show that the Huffman coding construction also minimizes the $p^{th}$ moment: so it provides a lower bound for ODT with $p^{th}$ moment objectives.

We define the *Huffman tree* with $m$ leaves to be a binary tree with all non-leaf nodes having two children, and exactly $2^{\lceil \log_2(m)\rceil} - m$ leaves at depth $\lfloor \log_2(m)\rfloor$ and $2m - 2^{\lceil \log_2(m)\rceil}$ leaves at depth $\lceil \log_2(m)\rceil$. When $m$ is a power of two, the Huffman tree corresponds to the "complete balanced binary tree" with all $m$ leaves at depth $\log_2(m)$. The following lemma summarizes this lower bound.

**Lemma 21.** *Among all binary trees with $m$ leaves, the Huffman tree minimizes the sum of $p^{th}$ powers of the leaf-depths, for all $p \geq 1$. Hence, the optimal $p^{th}$ moment for any ODT instance with $m$ hypotheses and unit-cost tests is at least:*

$$\frac{1}{m}\cdot\left(2^{\lceil \log_2(m)\rceil} - m\right)\cdot \lfloor \log(m)\rfloor^p \;+\; \frac{2}{m}\cdot\left(2m - 2^{\lceil \log_2(m)\rceil}\right)\cdot \lceil \log(m)\rceil^p.$$

*Proof.* Let $T$ denote the binary tree with $m$ leaves that minimizes the sum of $p^{th}$ powers of the leaf-depths, for any $p \geq 1$. We will show that $T$ must be the Huffman tree. Let $D_p(T) := \sum_{i\in L(T)} d(i)^p$ be the objective of tree $T$.

Note that every non-leaf node in $T$ must have 2 children. Indeed, if node $v \in T$ has exactly one child $u$ then we can remove $v$ and hang the subtree rooted at $u$ below $v$'s parent: this reduces the objective $D_p(T)$, contrary to the optimality of $T$.

We now claim that the difference between the maximum/minimum depth of leaves in $T$ is at most one. Let $b$ denote a deepest leaf in $T$, and $b'$ its sibling. Note that $b'$ is also a leaf in $T$ because $b$'s parent (which is non-leaf) must have 2 children. Let $a \in T$ be a leaf of minimum depth. Suppose, for a contradiction, that $d(a) \leq d(b) - 2$. Let tree $T'$ be obtained from $T$ by adding two children of $a$ as leaves and removing the leaves $\{b, b'\}$ (which makes

their parent a leaf in $T'$). Now,

$$\begin{aligned} D_p(T') - D_p(T) &= 2 \cdot (d(a)+1)^p + (d(b)-1)^p - (d(a)^p + 2 \cdot d(b)^p) \\ &= 2 \cdot ((d(a)+1)^p - d(b)^p) \; + \; (d(b)-1)^p - d(a)^p \\ &< (d(a)+1)^p - d(b)^p \; + \; (d(b)-1)^p - d(a)^p \\ &= (d(a)+1)^p - d(a)^p \; + \; (d(b)-1)^p - d(b)^p \;\; \leq \;\; 0. \end{aligned}$$

The (strict) inequality uses $d(a)+1 < d(b)$, and the last inequality is by convexity of $x^p$ and $d(a)+1 \leq b(b)-1$. This is a contradiction to the optimality of tree $T$.

Let $\ell$ be the maximum leaf-depth in $T$: so every leaf has depth $\ell-1$ or $\ell$. Let $r$ be the number of leaves at depth $\ell-1$. As there is no leaf at depth less than $\ell-1$ and each non-leaf node has 2 children, tree $T$ has $2^{\ell-1}-r$ non-leaf nodes at depth $\ell-1$. This also implies that there are exactly $2 \cdot (2^{\ell-1}-r)$ leaf nodes at depth $\ell$. As $T$ has $m$ leaves in total, we must have $r + 2 \cdot (2^{\ell-1}-r) = m$, which gives $r = 2^\ell - m$. Finally, using the fact that $0 \leq r \leq 2^{\ell-1}$, we get $2^{\ell-1} \leq m \leq 2^\ell$. So, $\ell = \lceil \log_2 m \rceil$, which means $T$ is the Huffman tree.

□

**Instances.** We use the WISER data (http://wiser.nlm.nih.gov/), which lists 415 toxins and 79 symptoms. Each toxin is associated with a set of observed symptoms that are reported in the data set. This corresponds to an ODT instance where the goal is to identify the toxin by testing for symptoms. For some toxin/symptom combinations it is unknown whether exposure to a toxin definitively exhibits a certain symptom. For such cases, we fill in randomly either a positive or negative outcome for the symptom, and generate 5 such variations, labelled as datasets $A_1$ to $A_5$. Given that we fill in the unknown symptoms randomly, some hypotheses are equivalent (i.e., have identical set of symptoms); so we remove all such duplicates, which causes the number of hypotheses $m$ to vary across instances.

Our results show that the greedy policy is optimal (or extremely close to the optimum) on all the datasets $A_1$ to $A_5$. This indicates that the WISER data set is highly structured with many balanced tests, and that the greedy policy is able to exploit this structure. In order to test the performance on less-structured instances, we also created 5 more instances (labeled $B_1$ to $B_5$) by restricting to a random subset of 15 (out of 79) tests. Again, we remove all duplicate hypotheses.

**Results.** In Table 5.1, we report the performance of the greedy policy on each of the 10 instances described above. For each instance, we report the number $m$ of hypotheses and the empirical approximation ratio (greedy objective divided by the lower-bound from Lemma 21)

| Instance | $m$ | Entropy bound | Revised bound | Sum of costs | Approx. factor $p = 1$ | Approx. factor $p = 2$ | Approx. factor $p = 3$ |
|---|---|---|---|---|---|---|---|
| $A_1$ | 405 | 3508.02 | 3538 | 3538 | 1.0000 | 1.0000 | 1.0000 |
| $A_2$ | 395 | 3407.16 | 3438 | 3438 | 1.0000 | 1.0000 | 1.0000 |
| $A_3$ | 399 | 3447.46 | 3478 | 3479 | 1.0000 | 1.0001 | 1.0001 |
| $A_4$ | 399 | 3447.46 | 3478 | 3478 | 1.0000 | 1.0000 | 1.0000 |
| $A_5$ | 395 | 3407.16 | 3438 | 3439 | 1.0003 | 1.0007 | 1.0012 |
| $B_1$ | 207 | 1592.55 | 1607 | 1666 | 1.0367 | 1.0941 | 1.1784 |
| $B_2$ | 248 | 1972.64 | 1976 | 2019 | 1.0218 | 1.0525 | 1.0931 |
| $B_3$ | 249 | 1982.04 | 1985 | 2025 | 1.0202 | 1.0488 | 1.0868 |
| $B_4$ | 266 | 2142.71 | 2148 | 2211 | 1.0293 | 1.0715 | 1.1284 |
| $B_5$ | 274 | 2218.86 | 2228 | 2269 | 1.0184 | 1.0440 | 1.0775 |

Table 5.1: Computational results on WISER dataset.

for $p = 1, 2, 3$. For the expectation objective ($p = 1$), we also report the objective value of the greedy policy and the two lower bounds: entropy-based (which was also used in prior work) and the Huffman bound (Lemma 21). We note that the Huffman bound is indeed better than the entropy-based lower-bound. For instances $A_1$ to $A_5$, the greedy policy's cost matches the lower bound, resulting in an empirical approximation ratio of 1. For the modified instances $B_1$ to $B_5$, our algorithm's performance is still very good: the empirical approximation for $p = 1, 2, 3$ is at most $1.04, 1.09$ and $1.18$ (respectively).

**Acknowledgments.** The result in this chapter is based on the paper [ATCN22].

# CHAPTER 6

# Conclusion and Future Directions

In this section we will discuss future directions for each of the topics studied in this thesis.

For the maximum-entropy sampling problem, the hardness result for the arrowhead sparsity structure where the support graph is a star [Ohs24] indicates that support graphs with high-degree $\Omega(n)$ vertices are NP-Hard. However, this does not rule out the possibility of exact algorithms if the support graph has low degree. As we saw in the solution for spider graphs, we may find an exact algorithm runs in time $O(n^{\Delta})$ where $\Delta$ is the maximum degree in the support graph. To start, it is worth understanding the complexity of the MESP when $C$ is a pentadiagonal matrix that corresponds to a support graph that is 4-regular.

In Chapter 3 we created valid instances of the MESP using data from the NADP/NTN. Recall, that we use a time-series model to fit the univariate data, this is done to ensure we have Gaussian streams of univariate data which is one necessary condition for the multivariate dataset to pass the Gaussianity tests. In general the method we used to fit the data does not work for creating instances of size $n \geq 100$. This is because a covariance matrix of this size $(n \times n)$ requires larger time span of data. Our model 3.1 could not fit the sensor data for time spans of larger than 100 months. This motivates the search for a spatio-temporal model that uses the spatial structure of the sensors along with the temporal component to fit the sensor data.

For the SMQ it would be interesting to see if we could find an instance that gives us a higher adaptivity gap or prove a better upper bound. In [WGW22] the authors show an adaptivity gap of $\Omega(n)$ for the budgeted minimum-element problem. While we know this is not the case for our problem due to the constant factor approximations we get for non-adaptive policies. It would still be desirable to show a more substantial lower bound to the adaptivity gap. It would also be interesting to explore whether there is a hardness result for non-adaptive setting of the SMQ where the permutation of intervals is chosen before any observations are made. [GGM10] show that the non-adaptive setting for the minimum-element problem is NP-Hard, through a reduction from the Covering Integer Program (CIP).

The ideas in this reduction do not directly carry over to our problem. Furthermore, it would be interesting to see if our algorithms can be generalized to the stochastic set selection problem considered in [MS23].

To create more comprehensive experiments for the SMQ, one could explore whether there is an efficiently calculated lower bound for the problem so that we can upper bound the performance ratio on instances with more than 15 intervals. Additionally, we could more effectively represent the performance of our algorithms by using real data from relevant applications, consider, for example, the query processing over replicated databases problem considered in [OW00, FMP+00].

One future direction for ASC is whether one can close the complexity gap, either by using a different approach in the analysis, changing the algorithm to get a smaller constant factor in the approximation, or showing no algorithm can achieve a better constant factor. Furthermore, it would be interesting to explore variants of the problem that capture a different set of applications by employing a different assumption than adaptive-submodularity.

# BIBLIOGRAPHY


[AAK19] Arpit Agarwal, Sepehr Assadi, and Sanjeev Khanna. Stochastic submodular cover with limited adaptivity. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*, page 323–342, 2019.

[AFLW96] Kurt M. Anstreicher, Marcia Fampa, Jon Lee, and Joy Williams. Continuous relaxations for constrained maximum-entropy sampling. In *Integer Programming and Combinatorial Optimization (Vancouver, BC, 1996)*, volume 1084 of *Lecture Notes in Computer Science*, pages 234–248. Springer, Berlin, 1996.

[AFLW99] Kurt M. Anstreicher, Marcia Fampa, Jon Lee, and Joy Williams. Using continuous nonlinear relaxations to solve constrained maximum-entropy sampling problems. *Mathematical Programming, Series A*, 85(2):221–240, 1999.

[AG01] Farid Alizadeh and Donald Goldfarb. Second-order cone programming. *Mathematical Programming*, 95:3–51, 2001.

[AG11] Yossi Azar and Iftah Gamzu. Ranking with submodular valuations. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1070–1079. SIAM, 2011.

[AH12] M. Adler and B. Heeringa. Approximating optimal binary decision trees. *Algorithmica*, 62(3-4):1112–1121, 2012.

[AL04] Kurt M. Anstreicher and Jon Lee. A masked spectral bound for maximum-entropy sampling. In *mODa 7 – Advances in Model-Oriented Design and Analysis*, Contributions to Statistics, pages 1–12. Physica, Heidelberg, 2004.

[AMO93] Ravindra K Ahujia, Thomas L Magnanti, and James B Orlin. Network flows: Theory, algorithms and applications. *New Jersey: Rentice-Hall*, 1993.

[AN16] Arash Asadpour and Hamid Nazerzadeh. Maximizing stochastic monotone submodular functions. *Manag. Sci.*, 62(8):2374–2391, 2016.

[Ans18a] Kurt M. Anstreicher. Efficient solution of maximum-entropy sampling problems. Preprint available at: https://www.biz.uiowa.edu/faculty/anstreicher/papers/linx.pdf, July 2018.

[Ans18b] Kurt M. Anstreicher. Maximum-entropy sampling and the Boolean quadric polytope. *Journal of Global Optimization*, 72(4):603–618, 2018.

[Ans18c] Kurt M. Anstreicher. Maximum-entropy sampling and the boolean quadric polytope. *Journal of Global Optimization*, 72(4):603–618, 2018.

[Ans20] Kurt M. Anstreicher. Efficient solution of maximum-entropy sampling problems. *Operations Research*, 68:1826–1835, 2020.

[App06] David L Applegate. *The traveling salesman problem: a computational study*, volume 17. Princeton university press, 2006.

[ASW16] Marek Adamczyk, Maxim Sviridenko, and Justin Ward. Submodular stochastic probing on matroids. *Math. Oper. Res.*, 41(3):1022–1038, 2016.

[ATCN22] Hessa Al-Thani, Yubing Cui, and Viswanath Nagarajan. Minimum cost adaptive submodular cover. *arXiv e-prints*, pages arXiv–2208, 2022.

[ATL20a] Hessa Al-Thani and Jon Lee. `MESgenCov`, 2020. https://github.com/hessakh/MESgenCov.

[ATL20b] Hessa Al-Thani and Jon Lee. An R package for generating covariance matrices for maximum-entropy sampling from precipitation chemistry data. *SN Operations Research Forum*, 1(3), Article 17 (21 pages), September 2020.

[ATL23] Hessa Al-Thani and Jon Lee. Tridiagonal maximum-entropy sampling and tridiagonal masks. *Discrete Applied Mathematics*, 337:120–138, 2023.

[BAD+07] Suresh K Bhavnani, Annie Abraham, Christopher Demeniuk, Messeret Gebrekristos, Abe Gong, Satyendra Nainwal, Gautam K Vallabha, and Rudy 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.

[BBS12] 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.

[BDE+21] Evripidis Bampis, Christoph Dürr, Thomas Erlebach, Murilo Santos de Lima, Nicole Megow, and Jens Schlöter. Orienting (hyper)graphs under explorable stochastic uncertainty. In *29th Annual European Symposium on Algorithms (ESA)*, volume 204 of *LIPIcs*, pages 10:1–10:18, 2021.

[BGL+12] 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.

[BL07] Samuel Burer and Jon Lee. Solving maximum-entropy sampling problems using factored masks. *Mathematical Programming, Series B*, 109(2-3):263–281, 2007.

[BLZ94] Philip J. Brown, Nhu D. Le, and James V. Zidek. Multivariate spatial interpolation and exposure to air pollutants. *Canad. J. Statist.*, 22(4):489–509, 1994.

[BSZ19] Domagoj Bradac, Sahil Singla, and Goran Zuzic. (Near) Optimal Adaptivity Gaps for Stochastic Multi-Value Probing. In *Approximation, Randomization, and Combinatorial Optimization*, volume 145, pages 49:1–49:21, 2019.

[CFL21] Zhongzhu Chen, Marcia Fampa, and Jon Lee. On computing with some convex relaxations for the maximum-entropy sampling problem, 2021. To appear in: *INFORMS Journal on Computing*.

[CFLL21] Zhongzhu Chen, Marcia Fampa, Amélie Lambert, and Jon Lee. Mixing convex-optimization bounds for maximum-entropy sampling. *Mathematical Programming, Series B*, 188:539–568, 2021.

[CHdT21] Steven Chaplick, Magnús M. Halldórsson, Murilo S. de Lima, and Tigran Tonoyan. Query minimization under stochastic uncertainty. *Theoretical Computer Science*, 895:75–95, 2021.

[Coo98] William J. Cook. *Combinatorial Optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Inc., 1998.

[Das04] S. Dasgupta. Analysis of a greedy active learning strategy. In *Advances in Neural Information Processing Systems*, pages 337–344, 2004.

[Dem97] James W Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.

[DGV08] B. C. Dean, M. X. Goemans, and J. Vondrák. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Math. Oper. Res.*, 33(4):945–964, 2008.

[DS14] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *ACM Symposium on Theory of Computing*, pages 624–633, 2014.

[dY15] Carlos M. da Fonseca and Fatih Yglmaz. Some comments on k-tridiagonal matrices: Determinant, spectra, and inversion. *Applied Mathematics and Computation*, 270:644–647, 2015.

[EHK+08] Thomas Erlebach, Michael Hoffmann, Danny Krizanc, Matús Mihal'Ák, and Rajeev Raman. Computing minimum spanning trees with uncertainty. In *STACS 2008*, pages 277–288. IBFI Schloss Dagstuhl, 2008.

[EHK16] Thomas Erlebach, Michael Hoffmann, and Frank Kammer. Query-competitive algorithms for cheapest set problems under uncertainty. *Theoretical Computer Science*, 613:51–64, 2016.

[EK72] Jack Edmonds and Richard M Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2):248–264, 1972.

[EKM21] Hossein Esfandiari, Amin Karbasi, and Vahab Mirrokni. Adaptivity in adaptive submodularity. In *Proceedings of 34th Conference on Learning Theory*, volume 134, pages 1823–1846. PMLR, 2021.

[FL00] Valerii Fedorov and Jon Lee. Design of experiments in statistics. In *Handbook of semidefinite programming*, volume 27 of *Internat. Ser. Oper. Res. Management Sci.*, pages 511–532. Kluwer Acad. Publ., Boston, MA, 2000.

[FL22] Marcia Fampa and Jon Lee. *Maximum-entropy sampling: algorithms and application*. Springer Nature, 2022.

[FLT04] U. Feige, L. Lovász, and P. Tetali. Approximating min sum set cover. *Algorithmica*, 40(4):219–234, 2004.

[FLX18] Hao Fu, Jian Li, and Pan Xu. A PTAS for a class of stochastic dynamic programs. In *45th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 107 of *LIPIcs*, pages 56:1–56:14, 2018.

[FMP+00] Tomás Feder, Rajeev Motwani, Rina Panigrahy, Chris Olston, and Jennifer Widom. Computing the median with uncertainty. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 602–607, 2000.

[GB09] Andrew Guillory and Jeff A. Bilmes. Average-case active learning with costs. In *Algorithmic Learning Theory*, volume 5809 of *Lecture Notes in Computer Science*, pages 141–155. Springer, 2009.

[GB17] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1, build 1123. http://cvxr.com/cvx, 2017.

[GG74] M.R. Garey and R.L. Graham. Performance bounds on the splitting algorithm for binary testing. *Acta Informatica*, 3:347–355, 1974.

[GGHK18] Dimitrios Gkenosis, Nathaniel Grammel, Lisa Hellerstein, and Devorah Kletenik. The Stochastic Score Classification Problem. In *26th Annual European Symposium on Algorithms (ESA)*, pages 36:1–36:14, 2018.

[GGKT08] Daniel Golovin, Anupam Gupta, Amit Kumar, and Kanat Tangwongsan. All-norms and all-l_p-norms approximation algorithms. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 2, pages 199–210, 2008.

[GGM10] Ashish Goel, Sudipto Guha, and Kamesh Munagala. How to probe for an extreme value. *ACM Transactions on Algorithms (TALG)*, 7(1):1–20, 2010.

[GGN21] Rohan Ghuge, Anupam Gupta, and Viswanath Nagarajan. The power of adaptivity for stochastic submodular cover. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 3702–3712, 2021.

[GHKL16] Nathaniel Grammel, Lisa Hellerstein, Devorah Kletenik, and Patrick Lin. Scenario submodular cover. In *International Workshop on Approximation and Online Algorithms*, pages 116–128. Springer, 2016.

[GK11] D. Golovin and A. Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *J. Artif. Intell. Res. (JAIR)*, 42:427–486, 2011.

[GK17] Daniel Golovin and Andreas Krause. Adaptive submodularity: A new approach to active learning and stochastic optimization. *CoRR*, abs/1003.3967, 2017.

[GKNR15] Anupam Gupta, Ravishankar Krishnaswamy, Viswanath Nagarajan, and R. Ravi. Running errands in time: Approximation algorithms for stochastic orienteering. *Math. Oper. Res.*, 40(1):56–79, 2015.

[GKS05] Carlos Guestrin, Andreas Krause, and Ajit Paul Singh. Near-optimal sensor placements in gaussian processes. In *Proceedings of the 22nd International Conference on Machine Learning*, page 265–272, 2005.

[GLSZ93] Peter Guttorp, Nhu D. Le, Paul D. Sampson, and James V. Zidek. Using entropy in the redesign of an environmental monitoring network. In *Multivariate environmental statistics*, volume 6 of *North-Holland Ser. Statist. Probab.*, pages 175–202. North-Holland, Amsterdam, 1993.

[GM07] Sudipto Guha and Kamesh Munagala. Approximation algorithms for budgeted learning problems. In *39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 104–113. ACM, 2007.

[GNR17] Anupam Gupta, Viswanath Nagarajan, and R. Ravi. Approximation algorithms for optimal decision trees and adaptive TSP problems. *Math. Oper. Res.*, 42(3):876–896, 2017.

[GNS17] Anupam Gupta, Viswanath Nagarajan, and Sahil Singla. Adaptivity gaps for stochastic probing: Submodular and XOS functions. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1688–1702, 2017.

[Goe11] Georg M. Goerg. Lambert W random variables - a new family of generalized skewed distributions with applications to risk estimation. *Annals of Applied Statistics*, 5(3):2197–2230, 2011.

[Goe16] Georg M. Goerg. *LambertW: Probabilistic Models to Analyze and Gaussianize Heavy-Tailed, Skewed Data. Version 0.6.4*, 2016. https://cran.r-project.org/web/packages/LambertW/.

[GV06] Michel Goemans and Jan Vondrák. Stochastic covering and adaptivity. In *LATIN 2006: Theoretical Informatics*, pages 532–543. Springer Berlin Heidelberg, 2006.

[HJ85] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, First edition, 1985.

[HK73] John E Hopcroft and Richard M Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.

[HK18] Lisa Hellerstein and Devorah Kletenik. Revisiting the approximation bound for stochastic submodular cover. *J. Artif. Intell. Res.*, 63:265–279, 2018.

[HKP21] Lisa Hellerstein, Devorah Kletenik, and Srinivasan Parthasarathy. A tight bound for stochastic submodular cover. *J. Artif. Intell. Res.*, 71:347–370, 2021.

[HLS24] Lisa Hellerstein, Naifeng Liu, and Kevin Schewior. Quickly determining who won an election. In *15th Innovations in Theoretical Computer Science Conference (ITCS)*, LIPIcs, pages 61:1–61:14, 2024.

[HLW01] Alan Hoffman, Jon Lee, and Joy Williams. New upper bounds for maximum-entropy sampling. In *mODa 6 – Advances in Model-Oriented Design and Analysis (Puchberg/Schneeberg, 2001)*, Contributions to Statistics, pages 143–153. Physica, Heidelberg, 2001.

[HR76] L. Hyafil and R. L. Rivest. Constructing optimal binary decision trees is $NP$-complete. *Information Processing Lett.*, 5(1):15–17, 1976.

[Huf52] David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[INvdZ16] Sungjin Im, Viswanath Nagarajan, and Ruben van der Zwaan. Minimum latency submodular cover. *ACM Trans. Algorithms*, 13(1):13:1–13:28, 2016.

[JLLS20] Haotian Jiang, Jian Li, Daogao Liu, and Sahil Singla. Algorithms and adaptivity gaps for stochastic k-tsp. In *11th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 151 of *LIPIcs*, pages 45:1–45:25, 2020.

[Kah91] Simon Kahan. A model for data in motion. In *Proceedings of the Twenty-third Annual ACM Symposium on Theory of computing*, pages 265–277, 1991.

[KGZ14] Selcuk Korkmaz, Dincer Goksuluk, and Gokmen Zararsiz. MVN: An R Package for Assessing Multivariate Normality. *The R Journal*, 6(2):151–162, 2014.

[KKT15] David Kempe, Jon M. Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. *Theory Comput.*, 11:105–147, 2015.

[KLQ95] Chun-Wa Ko, Jon Lee, and Maurice Queyranne. An exact algorithm for maximum entropy sampling. *Operations Research*, 43(4):684–691, 1995.

[KPB99] S. R. Kosaraju, T. M. Przytycka, and R. S. Borgstrom. On an Optimal Split Tree Problem. In *Proceedings of the 6th International Workshop on Algorithms and Data Structures*, pages 157–168, 1999.

[Lee98] Jon Lee. Constrained maximum-entropy sampling. *Operations Research*, 46(5):655–664, 1998.

[Lee12] Jon Lee. Maximum entropy sampling. In A.H. El-Shaarawi and W.W. Piegorsch, editors, *Encyclopedia of Environmetrics, 2nd ed.*, pages 1570–1574. Wiley, 2012.

[LW03a] Jon Lee and Joy Williams. A linear integer programming bound for maximum-entropy sampling. *Mathematical Programming, Series B*, 94(2–3):247–256, 2003.

[LW03b] Jon Lee and Joy Williams. A linear integer programming bound for maximum-entropy sampling. *Mathematical Programming*, 94(2-3, Ser. B):247–256, 2003.

[LX24] Yongchun Li and Weijun Xie. Best principal submatrix selection for the maximum entropy sampling problem: scalable algorithms and performance guarantees. *Operations Research*, 72(2):493–513, 2024.

[LZ06] Nhu D. Le and James V. Zidek. *Statistical analysis of environmental space-time processes*. Springer Series in Statistics. Springer, New York, 2006.

[LZWC19] Nhu Le, Jim Zidek, Rick White, and Davor Cubranic. EnviroStat: Statistical Analysis of Environmental Space-Time Processes. Version 0.4-2. https://rdrr.io/cran/EnviroStat/, 2019.

[Mar90] Silvano Martello. Knapsack problems: Algorithms and computer implementations, 1990.

[Mil13] Steven P. Millard. *EnvStats: An R Package for Environmental Statistics*. Springer, New York, 2013.

[MMS17] Nicole Megow, Julie Meißner, and Martin Skutella. Randomization helps computing a minimum spanning tree under uncertainty. *SIAM Journal on Computing*, 46(4):1217–1240, 2017.

[MMW08] Jeremy L. Martin, Matthew Morin, and Jennifer D. Wagner. On distinguishing trees by their chromatic symmetric functions. *Journal of Combinatorial Theory, Series A*, 115(2):237–253, 2008.

[MS23] Nicole Megow and Jens Schlöter. Set selection under explorable stochastic uncertainty via covering techniques. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 319–333. Springer, 2023.

[NAD18] NADP. National Acidic Deposition Program, National Trends Network. https://nadp.slh.wisc.edu/ntn/, 2018.

[Nik15] Aleksandar Nikolov. Randomized rounding for the largest simplex problem. In *Proc. of the 47th Annual ACM Symposium on Theory of Computing*, STOC '15, pages 861—-870, New York, 2015. ACM.

[NKN20] Fatemeh Navidi, Prabhanjan Kambadur, and Viswanath Nagarajan. Adaptive submodular ranking and routing. *Oper. Res.*, 68(3):856–877, 2020.

[NS17] Feng Nan and Venkatesh 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.

[Ohs24] Naoto Ohsaka. On the parameterized intractability of determinant maximization. *Algorithmica*, pages 1–33, 2024.

[OS90] Dianne P. O'Leary and Gilbert W. Stewart. Computing the eigenvalues and eigenvectors of arrowhead matrices. *Journal of Computational Physics*, 90:497–505, 1990.

[OW00] Chris Olston and Jennifer Widom. Offering a precision-performance tradeoff for aggregation queries over replicated data. Technical report, Stanford, 2000.

[RC12] A.C. Rencher and W.F. Christensen. *Methods of Multivariate Analysis*. Wiley Series in Probability and Statistics. Wiley, 2012.

[RV09] Yves Robert and Frédéric Vivien. *Introduction to scheduling*. CRC Press, 2009.

[SS21] Danny Segev and Sahil Singla. Efficient approximation schemes for stochastic probing and prophet problems. In *22nd ACM Conference on Economics and Computation (EC)*, pages 793–794. ACM, 2021.

[SW87a] Michael C. Shewry and Henry P. Wynn. Maximum entropy sampling. *Journal of Applied Statistics*, 14(2):165–170, 1987.

[SW87b] Michael C. Shewry and Henry P. Wynn. Maximum entropy sampling. *Journal of Applied Statistics*, 46:165–170, 1987.

[SW00] Paola Sebastiani and Henry P. Wynn. Maximum entropy sampling and optimal bayesian experimental design. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(1):145–157, 2000.

[TTT99] Kim-Chuan Toh, Michael J. Todd, and Reha H. Tütüncü. SDPT3: A Matlab software package for semidefinite programming, version 1.3. *Optimization Methods and Software*, 11(1-4):545–581, 1999.

[TTT12] Kim-Chuan Toh, Michael J. Todd, and Reha H. Tütüncü. On the implementation and usage of SDPT3 – a Matlab software package for semidefinite-quadratic-linear programming, version 4.0. In Miguel F. Anjos and Jean B. Lasserre, editors, *Handbook on Semidefinite, Conic and Polynomial Optimization*, pages 715–754. Springer US, Boston, MA, 2012.

[TV22] Vera Traub and Jens Vygen. An improved approximation algorithm for the asymmetric traveling salesman problem. *SIAM Journal on Computing*, 51(1):139–173, 2022.

[TWTD17] Guangmo Tong, Weili Wu, Shaojie Tang, and Ding-Zhu Du. Adaptive influence maximization in dynamic social networks. *IEEE/ACM Transactions on Networking*, 25(1):112–125, 2017.

[WGW22] Weina Wang, Anupam Gupta, and Jalani K Williams. Probing to minimize. In *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, volume 215, page 120. Schloss Dagstuhl–Leibniz-Zentrum f {\" u} r Informatik, 2022.

[Wol82] L.A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.

[WS11] David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.

[ZSL00] James V. Zidek, Weimin Sun, and Nhu D. Le. Designing and integrating composite networks for monitoring multivariate Gaussian pollution fields. *Journal of the Royal Statistical Society. Series C. Applied Statistics*, 49(1):63–79, 2000.