

**RESOURCE ALLOCATION AND SUBSET SELECTION: NEW APPROACHES  
AT THE INTERFACE BETWEEN DISCRETE AND CONTINUOUS  
OPTIMIZATION**

A Dissertation  
Presented to  
The Academic Faculty

By

Alfredo Torrico Palacios

In Partial Fulfillment  
of the Requirements for the Degree  
Doctor of Philosophy in the  
School of Industrial and Systems Engineering

Georgia Institute of Technology

August 2019

Copyright © Alfredo Torrico Palacios 2019

**RESOURCE ALLOCATION AND SUBSET SELECTION: NEW APPROACHES  
AT THE INTERFACE BETWEEN DISCRETE AND CONTINUOUS  
OPTIMIZATION**

Approved by:

Professor Mohit Singh  
Co-Advisor  
School of Industrial and Systems  
Engineering  
*Georgia Institute of Technology*

Professor Shabbir Ahmed  
School of Industrial and Systems  
Engineering  
*Georgia Institute of Technology*

Professor Sebastian Pokutta  
Co-Advisor  
School of Industrial and Systems  
Engineering  
*Georgia Institute of Technology*

Professor Santosh Vempala  
School of Computer Science  
*Georgia Institute of Technology*

Professor Alejandro Toriello  
Co-Advisor  
School of Industrial and Systems  
Engineering  
*Georgia Institute of Technology*

Date Approved: April 26, 2019

## ACKNOWLEDGEMENTS

Getting a Ph.D. is not easy. But not because you need to be extremely smart or because you require tons of abilities. This is the easiest part: you ask, you learn, you work hard, repeat. The most difficult part reduces to only one word: patience. You may be working in a lab all night, you may be trying to prove a theorem, or you may be looking for that annoying bug in your code. It is always about patience. Until you get that precious moment: results. This great moment corresponds to a tiny fraction of your Ph.D. life. The rest of the time is again, patience. It is in “that rest of the time” in which you have to acknowledge the people who support you. Without them, Ph.D. would be considerably hard.

First and foremost, I would like to thank my mother Blanca who has been an essential support during my life. Even though we are physically far from each other, she always finds the way to cheer me up, to give me an advice, to encourage me and to listen to all complaints. I love you Ma. Also, special thanks to my cousin Paola for all your help.

I would like to thank my advisors Mohit Singh, Sebastian Pokutta and Alejandro Toriello. I really appreciate your patience and intellectual guidance during all these years. I am grateful for all the knowledge and experience that all of you pass on to me. Also, I would like to thank the members of my thesis committee for your time and help: Shabbir Ahmed and Santosh Vempala.

Last, but not least: friends. Thank you very much to all of you. To you, Popa. To my old chilean friends: Kiko, Maruri, Javi, Caro, Pamish, Camila, Mamein, Daniel, Tania, Andrea. To my ISyE chilean friends: Siebert, Ramon, Guido, Cristobal, Sebastian, Berni, Felipe and to the entire chilean gang. To my foreign ISyE friends: Idil, Aster, Tony, Adrian, Jana and to the entire ISyE family. To my roommates: Ethan and Adolfo. To my previous roommates: Tim and Daniel. And the list could continue...

Muchísimas gracias! With all of you, the Ph.D. was one of the greatest moments of my life. And patience was easier to digest.

## TABLE OF CONTENTS

<b>Acknowledgments</b> . . . . .	iii
<b>List of Tables</b> . . . . .	viii
<b>List of Figures</b> . . . . .	ix
<b>Chapter 1: Introduction</b> . . . . .	1
1.1 Preliminary Definitions . . . . .	3
1.1.1 Approximation Algorithms . . . . .	3
1.1.2 Online Algorithms: Competitive Analysis . . . . .	4
1.1.3 Online Learning: Regret Analysis . . . . .	6
1.2 Online Bipartite Matching . . . . .	9
1.2.1 Model Description . . . . .	13
1.2.2 Overview of Our Contributions . . . . .	14
1.3 Constrained Submodular Function Maximization . . . . .	15
1.3.1 Background and Related Literature . . . . .	16
1.3.2 Overview of Our Contributions . . . . .	21
<b>Chapter 2: Static Relaxations for Online Bipartite Matching</b> . . . . .	25
2.1 Introduction . . . . .	25

2.1.1	Main Contributions . . . . .	26
2.2	A Novel Polyhedral Approach . . . . .	27
2.3	Projected Static Relaxations . . . . .	29
2.4	Policies Derived from Bounds . . . . .	36
2.5	Computational Study . . . . .	49
2.6	Concluding Remarks . . . . .	52
<b>Chapter 3: Dynamic Relaxations for Online Bipartite Matching . . . . .</b>		<b>54</b>
3.1	Introduction . . . . .	54
3.1.1	Main Contributions . . . . .	54
3.2	A Dynamic Polyhedral Approach . . . . .	55
3.3	Projected Dynamic Relaxations . . . . .	58
3.3.1	Policy Design . . . . .	64
3.3.2	Further Polyhedral Study . . . . .	65
3.4	Computational Study . . . . .	70
3.5	Concluding Remarks . . . . .	75
<b>Chapter 4: New Advances in Robust Submodular Maximization . . . . .</b>		<b>77</b>
4.1	Introduction . . . . .	77
4.1.1	Problem Formulation . . . . .	78
4.1.2	Main Contributions . . . . .	80
4.1.3	Related Work . . . . .	83
4.2	The Offline Robust Submodular Maximization Problem . . . . .	85
4.2.1	Offline Bi-criteria Algorithm and Analysis . . . . .	85

4.2.2	Computational Study for the Offline Problem . . . . .	92
4.2.3	Continuous Offline Bi-criteria Algorithm . . . . .	103
4.3	The Online Robust Submodular Maximization Problem . . . . .	108
4.3.1	Preliminaries for the Online Bi-criteria Algorithm . . . . .	109
4.3.2	Online Bi-criteria Algorithm and Analysis . . . . .	113
4.4	Extensions and Other Results . . . . .	118
4.4.1	Necessity of Monotonicity . . . . .	118
4.4.2	Knapsack Constraints . . . . .	119
4.4.3	Multiple Matroid Constraints . . . . .	120
4.4.4	Distributionally Robust over Polyhedral Sets . . . . .	121
4.5	Concluding Remarks . . . . .	122
<b>Chapter 5: The Sharpness Criteria in Submodular Maximization . . . . .</b>		<b>123</b>
5.1	Introduction . . . . .	123
5.1.1	Problem Formulation . . . . .	124
5.1.2	Main Contributions . . . . .	127
5.1.3	Related Work . . . . .	128
5.2	Submodular Sharpness: Cardinality Constraints . . . . .	131
5.2.1	Dynamic Sharpness . . . . .	134
5.2.2	Approximate Sharpness . . . . .	137
5.3	Computational Study . . . . .	140
5.3.1	Movie Recommendation . . . . .	143
5.3.2	Nonparametric learning . . . . .	149

5.3.3	Sensor location . . . . .	152
5.3.4	Exemplar-based clustering . . . . .	154
5.4	Concluding Remarks . . . . .	155
<b>Appendix A: Extra Material for Chapter 3 . . . . .</b>		<b>159</b>
A.1	Remaining Proofs . . . . .	159
A.2	Detailed Experiment Results . . . . .	171
<b>Appendix B: Extra Material for Chapter 4 . . . . .</b>		<b>174</b>
B.1	Remaining Proofs . . . . .	174
B.2	Extended Stochastic Greedy for Partition Matroid . . . . .	175
B.3	Pseudo-Code for the Offline Bi-criteria Algorithm . . . . .	175
<b>References . . . . .</b>		<b>187</b>

## LIST OF TABLES

2.1	Summary of experiment results for static relaxations and their corresponding policy. . . . .	50
2.2	Ratios for $k$ -regular bipartite graphs. . . . .	52
3.1	Summary of experiment results for dynamic relaxations and their corresponding policy. . . . .	73
3.2	Experiment results for regular graphs. . . . .	75
A.1	Experiment results for small instances. . . . .	171
A.2	Experiment results for large, dense instances. . . . .	172
A.3	Experiment results for large, sparse instances. . . . .	173

## LIST OF FIGURES

4.1	<i>Non-parametric learning</i> : performance profiles for running time (logarithmic scale in the $x$ -axis) . . . . .	96
4.2	<i>Non-parametric learning</i> : performance profiles for function calls. . . . .	96
4.3	<i>Non-parametric learning</i> : objective value versus the violation ratio in a single run of each method. . . . .	96
4.4	<i>Non-parametric learning</i> : box-plot for the function calls. . . . .	97
4.5	<i>Clustering</i> : (small) performance profiles for the running time. . . . .	98
4.6	<i>Clustering</i> : (small) performance profiles for the function calls. . . . .	99
4.7	<i>Clustering</i> : (Large) box-plots for the running times. . . . .	99
4.8	<i>Clustering</i> : (Large) box-plots for the function calls. . . . .	99
4.9	<i>Sensor Placement</i> : Box-plots for the function calls. . . . .	101
4.10	<i>Sensor Placement</i> : Box-plots for the running time. . . . .	101
4.11	<i>Sensor Placement</i> : objective value versus the violation ratio in a single run of each method. . . . .	101
4.12	<i>Sensor Placement</i> : Performance profiles for the running time. . . . .	102
4.13	<i>Sensor Placement</i> : Performance profiles for the function calls. . . . .	102
5.1	<i>How sharpness looks</i> : $\alpha = 0.1$ , $c = 1.01$ and $\theta = 0.0157$ . . . . .	144
5.2	<i>How sharpness looks</i> : $\alpha = 0.5$ , $c = 1.21$ and $\theta = 0.6394$ . . . . .	145
5.3	<i>How sharpness looks</i> : $\alpha = 0.9$ , $c = 1.06$ and $\theta = 0.8605$ . . . . .	145

5.4	<i>Movie Recommendation</i> $\alpha = 0.5$ : for different instances of size $n = 2K$ , we plot the empirical ratio of the greedy algorithm and the approximation factors obtained from the curvature and sharpness analysis. . . . .	146
5.5	<i>Movie Recommendation</i> $\alpha = 0.5$ : for different instances of size $n = 2K$ , we plot the approximation factors obtained from the curvature analysis, the sharpness analysis with no switching point and the sharpness analysis with one single switching point. . . . .	146
5.6	<i>Movie Recommendation</i> $\alpha = 0.5$ : for different instances of size $n = 2K$ , we plot the approximation factors obtained from the dynamic sharpness analysis with respect to the different candidates of switching point $t_0$ . . . . .	147
5.7	<i>Movie Recommendation - Facility Location</i> : for different instances of size $n = 2K$ , we plot the empirical ratio of the greedy algorithm and the approximation factors obtained from the curvature and sharpness analysis. . . . .	148
5.8	<i>Movie Recommendation - Facility Location</i> : for different instances of size $n = 2K$ , we plot the approximation factors obtained from the curvature analysis, the sharpness analysis with no switching point and the sharpness analysis with one single switching point. . . . .	148
5.9	<i>Movie Recommendation - Facility Location</i> : for different instances of size $n = 2K$ , we plot the approximation factors obtained from the dynamic sharpness analysis with respect to the different candidates of switching point $t_0$ . . . . .	149
5.10	<i>Non-parametric Learning</i> : for different instances of size $n = 2K$ , we plot the empirical ratio of the greedy algorithm and the approximation factors obtained from the curvature and sharpness analysis. . . . .	150
5.11	<i>Non-parametric Learning</i> : for different instances of size $n = 2K$ , we plot the approximation factors obtained from the curvature analysis, the sharpness analysis with no switching point and the sharpness analysis with one single switching point. . . . .	151
5.12	<i>Non-parametric Learning</i> : for different instances of size $n = 2K$ , we plot the approximation factors obtained from the dynamic sharpness analysis with respect to the different candidates of switching point $t_0$ . . . . .	151
5.13	<i>Sensor Location</i> : for different instances of size $n = 2K$ , we plot the empirical ratio of the greedy algorithm and the approximation factors obtained from the curvature and sharpness analysis. . . . .	153

5.14	<i>Sensor Location</i> : for different instances of size $n = 2K$ , we plot the approximation factors obtained from the curvature analysis, the sharpness analysis with no switching point and the sharpness analysis with one single switching point. . . . .	153
5.15	<i>Sensor Location</i> : for different instances of size $n = 2K$ , we plot the approximation factors obtained from the dynamic sharpness analysis with respect to the different candidates of switching point $t_0$ . . . . .	154
5.16	<i>Image Clustering</i> : for different instances of size $n = 2K$ , we plot the empirical ratio of the greedy algorithm and the approximation factors obtained from the curvature and sharpness analysis. . . . .	155
5.17	<i>Image Clustering</i> : for different instances of size $n = 2K$ , we plot the approximation factors obtained from the curvature analysis, the sharpness analysis with no switching point and the sharpness analysis with one single switching point. . . . .	156
5.18	<i>Image Clustering</i> : for different instances of size $n = 2K$ , we plot the approximation factors obtained from the dynamic sharpness analysis with respect to the different candidates of switching point $t_0$ . . . . .	156

## SUMMARY

Resource allocation and subset selection are two relevant classes of problems in the core of combinatorial optimization. Over the past decade, there has been an increased interest in these areas due to their significant impact in real-world applications. Online advertising, sharing-economy systems, and kidney exchange are a few examples of the applicability of resource allocation models. On the other hand, data summarization, influence modeling in social networks, and sensor location have extensively motivated the study of subset selection. In this thesis, we propose new approaches to two classical problems of these areas: the *online bipartite matching* problem and the *constrained submodular function maximization* problem. Our main objectives are: (1) to develop new models and algorithms that provide theoretical guarantees on their solutions, and (2) to present computational studies that empirically support our theoretical results.

In the first chapter of this thesis, we present the theoretical background that is needed to understand our results in both problems. We describe the notion of approximation algorithms, competitive analysis and regret analysis. Later we motivate the online bipartite matching and introduce the model considered in this thesis. Finally, we motivate the constrained submodular maximization problem, we present some of the previous approaches in the related literature and their corresponding guarantees.

In the second chapter of this thesis, we present a novel polyhedral approach to the online variant of the classical bipartite matching problem. We consider the i.i.d. model defined as follows: one side of the bipartition is fixed and known in advance, while nodes from the other side appear one at a time as i.i.d. realizations of a uniform distribution, and must immediately be matched or discarded. We consider various *static* relaxations of the polyhedral set of achievable probabilities, introduce valid inequalities, and discuss when they are facet-defining. We also show how several of these relaxations correspond to ranking policies and their time-dependent generalizations. We finally provide a computational

study of these relaxations and policies to determine their empirical performance.

In the third chapter of this thesis, we focus on *dynamic* polyhedral relaxations of the i.i.d. online bipartite matching problem. We show how they theoretically dominate the *static* relaxations from the previous part, and perform a polyhedral study to theoretically examine the strength of the new relaxations. We also discuss how to derive heuristic policies from the dual prices of the relaxations, in a similar fashion to dynamic resource prices used in network revenue management. We demonstrate the empirical quality of these new relaxations and policies via computational experiments.

In the fourth chapter of this thesis, we consider the *robust* submodular maximization problem with structured combinatorial constraints. Our approach is applicable to constraints defined by single or multiple matroids, knapsack as well as distributionally robust criteria. We consider both the offline setting where the data defining the problem is known in advance as well as the online setting where the input data is revealed over time. For the offline setting, we give a nearly optimal bi-criteria approximation algorithm that relies on new extensions of the classical greedy algorithm. Later, we provide an efficient version of this algorithm that theoretically performs less function calls than the previous one, at cost of adding a few more elements to the final solution. We also assess the performance of our offline algorithms in three real-world applications. Finally, in a similar manner than the offline setting, for the online variant of the problem we present an algorithm that returns a bi-criteria solution with sub-linear regret.

In the last chapter of this thesis, we explore the concept of *sharpness* in submodular function maximization. Empirical studies have shown that the performance of the greedy algorithm is substantially better in practice, even though its  $1 - 1/e$  worst-case guarantee. This raises a natural question of explaining this improved performance of the greedy algorithm. Our goal is to define *sharpness* for submodular functions as a candidate explanation for this phenomenon. The sharpness criterion is inspired by the concept of strong convexity in convex optimization and its objective is to measure the behavior of the objective function

around the set of optimal solutions. We show that the greedy algorithm provably performs better as the sharpness of the submodular function increases. This improvement ties closely to the faster convergence of the first order methods for strongly convex functions. Lastly, we present an exhaustive computational study to support this theoretical improvement.

# CHAPTER 1

## INTRODUCTION

Combinatorial optimization has seen significant progress over the past decades in the design of new models and heuristics. The development of new technologies, the endless number of necessities of the industry, and the progressive changes in societal thinking are some of the numerous reasons why the interest in this area of optimization has rapidly increased in the last years. To exemplify, factors such as the frenetic growth of social networks, the birth of sharing-economy systems, and the necessity of designing a fair school admission system, have driven and pushed the research community to develop alternative, efficient and viable solutions to the aforementioned challenges.

Many of these motivations fit in two of the most prominent classes of problems in combinatorial optimization: *resource allocation* and *subset selection*. Broadly speaking, a resource allocation problem corresponds to the class of models in which a decision-maker has a set of *items*, and the objective is to *optimally* assign (under certain criterion) each of those objects in a specific *location*. Classic combinatorial problems in this class are job scheduling, facility location and bipartite matching. Particularly, the last one of these examples has a variety of applications in kidney exchange [95], house allocation [1], combinatorial auctions [34], ride-sharing [93], school choice [43] and online advertising [81].

On the other hand, subset selection problems correspond to a different class of models in which a decision-maker faces a *universe* of elements, and the objective is to optimally select (under certain criterion) a subset of them. Moreover, these subsets may face combinatorial constraints such as cardinality or partition. This area of research is constantly growing given its significant number of applications in sensor location [71], nonparametric learning [68], computer vision [66], influence modeling in social networks [64], exemplar-

based clustering [48], migration [47], etc.

Both classes of problems previously mentioned can be modeled in two different manners: *offline* and *online*. A model is said to be offline if the decision-maker has access to all data beforehand. In other words, an offline algorithm receives as an input the objective function and the corresponding constraints of the optimization problem. Even though this type of models is attractive at first, in several real-world applications the information is revealed dynamically over time. On the one hand, revealing the data in an online manner makes the model closer to reality, on the other hand, the problem becomes theoretically more challenging.

Most of the relevant offline problems are *computationally hard*, namely there is no *efficient* algorithm that guarantees an optimal output solution. Given this, there is no hope for the online version of the same problem. Even if the offline problem is *computationally easy*, its online variant may be intractable. An example of this is the bipartite matching and the online bipartite matching problem. Therefore, it is imperative for the research community to develop offline and online algorithms that are efficient and provide *good guarantees* (see Section 1.1 for basic definitions of some performance guarantees). As we will see in this thesis, we need to design non-trivial techniques from continuous and discrete Optimization in order to attack both classes of problems.

The first part of this thesis, Chapters 2 and 3, is focused on the online variant of the classical resource allocation problem called bipartite matching problem. The second part of this thesis, Chapters 4 and 5, concentrates in one of the most used subset selection models, the constrained submodular function maximization problem.

In the remainder of this chapter we present basic definitions related to approximation algorithms, competitive analysis, and regret analysis in Section 1.1. In Section 1.2 we introduce the online bipartite matching problem, we detail the model considered in this thesis and we finally gave an overview of our contributions on this problem. Finally, in Section 1.3 we introduce the constrained submodular function maximization problem, we

detail the problem formulation and give a brief literature review on previous approaches, and we finish with an overview of our contributions on this problem.

## 1.1 Preliminary Definitions

In this section we will provide some basic definitions for the problems considered in this thesis.

### 1.1.1 Approximation Algorithms

Most of the interesting problems in discrete optimization that actually have a real-world application are known to be NP-hard. This means that there is no efficient procedure that finds an optimal solution, unless  $P = NP$ . Usually, *efficiency* is measured in terms of the running time, and an algorithm is said to be efficient if it has a running time that is bounded by a polynomial in the input size.

Given the *hardness* constraint above, one approach for this kind of optimization problems is to design algorithms that: (1) run in polynomial-time and (2) output a solution that is *good enough* in terms of its objective value when is compared to the true optimum. In other words, procedures that give us a guarantee on the quality of the output solution for any instance of the problem that is being considered. For this, it is commonly assumed [112] that the optimization problem, either maximization or minimization, is *feasible* and has an *objective function* that maps every feasible solution to a non-negative value. Therefore, we say an algorithm outputs a solution that is good enough in terms of its function value with respect to the optimal objective value. These algorithms are called *approximation algorithms*. Throughout this thesis, particularly in Chapters 4 and 5, we will follow the definition of approximation algorithm given in [112].

**Definition 1** ( $\lambda$ -approximation algorithm, [112]). *A  $\lambda$ -approximation algorithm for an optimization problem is a polynomial-time algorithm that for all instances of the problem produces a solution whose value is within a factor of  $\lambda$  of the value of an optimal solution.*

The parameter  $\lambda$  is sometimes called *performance guarantee*, *approximation factor* or simply *approximation ratio*. For maximization problems the convention is  $\lambda < 1$  and for minimization problems is  $\lambda > 1$ . For instance, when we say that the greedy algorithm achieves a  $1 - 1/e$  approximation factor for a maximization problem, it means that the value of the output solution of the greedy algorithm is at least a  $1 - 1/e$  fraction of the optimal value.

There are many standard algorithms and common techniques being used in the literature to obtain approximation guarantees such as greedy algorithms, local search, LP rounding, randomized rounding, the primal-dual method, etc. For a detailed explanation of each these methods and its application to classical problems in combinatorial optimization, we refer the interested reader to [112] and the references therein.

### 1.1.2 Online Algorithms: Competitive Analysis

As we mentioned before, for offline optimization problems it is assumed that all the information is given to the decision-maker in advance. In reality this may not be the case, since data could arrive dynamically over time. Therefore, the decision-maker has to make a new decision every time a piece of information is revealed. A classic example of this is the *ski rental problem*: Hugo decided to go skiing for the first time. For this he has to decide if he should buy the complete equipment for \$200 or rent it for \$40 each time he goes skiing. The decision in this problem is whether he should buy a whole new equipment or simply rent it. Clearly, his decision is going to depend on the total number of times he goes skiing in the future, which is unknown. Observe that if he goes at most 4 times in his life, then it is better to rent, but if he knows that he will go for sure at least 6 times or more, then it is better to buy now. Since future is unknown, which *online algorithm* should he use to decide every time he goes skiing? The only input for this procedure in stage  $t$  is that this is the  $t$ -th time he goes skiing, and that he has been renting so far (whenever he decides to buy, no future decisions have to be made). As in Section 1.1.1 the main goal in

this area of research is to design *online algorithms* that run in polynomial-time and output a final solution that is *good enough*. Competitive analysis (originally presented in [99]) is a method to quantify the quality of the solution given by the online algorithm. Other examples of online problems in the competitive analysis community are the paging problem, the secretary problem, online bipartite matching, online routing,  $k$ -server problem, online load balancing, etc. (see [4]).

In the following, we consider the framework given in [4]. Formally, several online problem can be defined as follows: An online algorithm  $\mathcal{A}$  is presented with a *request sequence*  $\sigma = \sigma(1), \sigma(2), \dots, \sigma(T)$ , where  $T$  is the last stage (known or unknown, depending on the model). In the ski rental problem, requests correspond to go skiing. The requests  $\sigma(t)$ , with  $t \in \{1, \dots, T\}$  must be served in the order of occurrence. When serving request  $\sigma(t)$ , algorithm  $\mathcal{A}$  does not know any request  $\sigma(t')$  with  $t' > t$ . Serving requests incurs cost (utility) and the goal is to minimize (maximize) the total cost (utility) obtained on the entire request sequence. There are many models for the arrival of requests such as adversarial, random, known i.i.d. For the purpose of this section let us consider the adversarial model: an adversary (or nature) chooses the worst-case instance of requests. The online algorithm  $\mathcal{A}$  will be compared to an *optimal offline algorithm* which knows the entire sequence of requests in advance and give an optimal solution. On a given sequence  $\sigma$ , we denote by  $\mathcal{A}(\sigma)$  and  $\text{OPT}(\sigma)$ , the cost (utility) obtained by the online algorithm and the offline optimal algorithm, respectively. We are ready to define the performance guarantee of an online algorithm  $\mathcal{A}$ .

**Definition 2** ( $\lambda$ -competitive, [99, 4]). *Given a minimization problem, an online deterministic algorithm  $\mathcal{A}$  is  $\lambda$ -competitive if  $\mathcal{A}(\sigma) \leq \lambda \cdot \text{OPT}(\sigma) + O(1)$  for all request sequences  $\sigma$*

For a maximization problem the inequality is reversed, or equivalently,  $\lambda$  is the minimum over all requests  $\sigma$ . In other words, an online algorithm is  $\lambda$ -competitive if it achieves an objective value at least  $\lambda$  times the best possible value achieved by an offline algorithm

with all information in advance, in every instance of the problem. The parameter  $\lambda$  is often called *competitive ratio*, or sometimes with an abuse of notation just *approximation guarantee*. For instance, when we say that the greedy algorithm achieves  $1/2$  competitive ratio for the adversarial model of the online bipartite matching, it means that on every instance of this problem the greedy algorithm ensures a  $1/2$  fraction of the optimal value by the offline optimal algorithm.

The definition of competitive ratio may vary depending on the problem, on the arrival model, or if the online algorithm is randomized [9] or deterministic. For instance, for the online bipartite matching problem studied in Chapters 2 and 3, we consider the competitive ratio as the minimum ratio, over all realizations, between the algorithm's expected value (if randomized) and the expected value of the *offline* optimum, where this last expectation is taken over the distribution on the arriving requests [81]. For other online problems and different definitions of competitive ratio we refer the interested reader to [4, 9].

### 1.1.3 Online Learning: Regret Analysis

A different variant of online problems is the one studied by the online learning community. In this case the decision-maker and adversary change the order in which they play. Specifically, during  $T$  steps we have the following game: in each stage  $t$  the online-player makes a decision, and then the adversary chooses a loss function (possible different than the one in previous epochs). The decision-maker suffers a loss but receives a feedback from the adversary. These losses were unknown for the decision maker beforehand. There are different models depending on the decision space, the class of loss functions, the type of adversary and the form of feedback (for more details see e.g. [51]). To exemplify this class of online problems let us consider the online shortest path problem: in this case the decision maker is given a network, an origin and a destination. On each day the decision maker chooses a path from the origin to the destination. After this, an adversary (or nature) chooses the time on each edge of the network. The player suffers a loss which corresponds to her travel time,

but at the same time she has access to the travel time of all arcs in the network in that stage, namely she receives a feedback after making her decision. Her objective is to minimize the total travel time during  $T$  days. The question in this case is how to design an *online algorithm* that incorporates the feedback and it minimizes the total travel time. The challenge in this problem is that the decision maker does not have access to the adversarial choices. For instance, the decision maker could choose a route that unexpectedly had an accident, which translates in a higher loss. As in the previous Section 1.1.2 the main goal in this area is to provide online algorithms that run in polynomial-time and guarantee a sequence of decisions that are *good enough*. Other examples of online problems in the regret analysis community are: prediction from experts, online spam filtering, portfolio selection, matrix completion and recommendation systems, online regression, online ranking, etc. For more details see e.g. [51, 98, 24].

For simplicity, we consider the convex framework given in [51]. Formally, there is a convex<sup>1</sup> decision set  $\mathcal{K} \subseteq \mathbb{R}^n$  and the losses are assumed to be convex<sup>2</sup> functions over  $\mathcal{K}$ . Several online learning problems can be stated as follows: at iteration  $t$  the online player chooses  $x_t \in \mathcal{K}$ . After the player has committed to this choice, a convex cost function  $f_t \in \mathcal{F} : \mathcal{K} \rightarrow \mathbb{R}$  is revealed. Here  $\mathcal{F}$  is the bounded family of cost functions available to the adversary. The cost incurred by the online player is  $f_t(x_t)$ . This game is repeated for a total of  $T$  epochs. This model considers *full information* feedback since the loss function is revealed completely to the online player after she makes a decision. *Partial information* models have also been considered, which are most commonly called *bandit* models. Using the previous example, the player's partial feedback corresponds to the total travel time of her decision in the specific stage, not the travel time of each arc in the network. Formally, she does not have access to the function  $f_t$ , she only observes the quantity  $f_t(x_t)$ . As one may expect, these models are far more technically challenging

---

<sup>1</sup>A set  $\mathcal{K}$  is convex if for any pair of points  $x, y \in \mathcal{K}$  and any scalar  $\rho \in [0, 1]$ , we have  $\rho x + (1 - \rho)y \in \mathcal{K}$ .

<sup>2</sup>A function  $f : \mathcal{K} \rightarrow \mathbb{R}$  is said to be convex if for any pair of points  $x, y \in \mathcal{K}$  and any scalar  $\rho \in [0, 1]$ , we have  $f(\rho x + (1 - \rho)y) \leq \rho f(x) + (1 - \rho)f(y)$ .

than assuming full information feedback. For more details on bandit models we refer the interested reader to [51, 98, 18].

How do we measure the performance of online algorithm in this context? As explained in [51], the suitable metric is the *regret*: the difference between the total cost that the decision-maker has incurred and the total cost of the best fixed decision in hindsight. Formally, the regret is defined as follows

**Definition 3** (Regret, see e.g. [51]). *Let  $\mathcal{A}$  be an algorithm for the online learning problem which maps a certain game history to decisions in  $\mathcal{K}$ , namely  $\mathcal{A}$  gives a sequence  $x_1, \dots, x_T$ . Given a sequence of functions  $\{f_1, \dots, f_T\} \subseteq \mathcal{F}$ , the regret of  $\mathcal{A}$  after  $T$  iterations is:*

$$\mathbf{Regret}(T) = \sum_{t=1}^T f_t(x_t) - \min_{x \in \mathcal{K}} \sum_{t=1}^T f_t(x)$$

In this context, we look for online algorithms that provide an upper bound on the worst-case regret, namely the upper bound does not depend on the sequence of functions  $\{f_1, \dots, f_T\} \subseteq \mathcal{F}$ . The previous definition can be appropriately adapted for a maximization problem. The objective is to design an online algorithm with sublinear regret in  $T$ , namely  $\mathbf{Regret}(T) = o(T)$ . Having sublinear regret translates in an algorithm that on average performs as well as the best fixed strategy in hindsight. We also look for online algorithms that run in polynomial-time in the dimension of  $\mathcal{K}$ ,  $T$ , and the parameters which describe the loss functions and  $\mathcal{K}$ . For other models and a detailed description of the online algorithms which are commonly used in this context, along with their regret guarantees, we refer the interested reader to [51, 98, 18].

We will use Definition 3 as a base to define the performance guarantee of our online model in Chapter 4, specifically in Section 4.1.1.

## 1.2 Online Bipartite Matching

The first part of this thesis, Chapter 2 and Chapter 3, is concerned in one of the most well-known online models used in resource allocation: the *online bipartite matching problem*. Many important emerging applications in e-commerce, and in the internet more generally, can be modeled as online two-sided markets, with buyers and sellers dynamically appearing and conducting transactions. When a platform or other entity controls one side of this market (usually the supply) and can choose what product to offer to dynamically appearing buyers, the system in question can be modeled as one of the most classic online resource allocation problems called *online bipartite matching* (OBM) problem. As more services move to online platforms in the coming years, the ubiquity and importance of OBM models will only increase.

An important application of OBM and its generalizations is in the rapidly growing sector of digital advertisement; in their *US Ad Spending Estimates and Forecast for 2017*, eMarketer reports that digital ad spending reached \$83 billion last year, an almost 16% year-over-year increase. Within digital marketing, search engine advertisement yields one application of OBM and similar models [81, 82]: Users input search terms, and the engine displays ad links in addition to the actual search results. The engine chooses the ad(s) to display (i.e. matches an ad to a user) based on the search term, user information, and advertisers' preferences and budget, with one typical objective being to maximize the expected revenue collected from advertisers. Similarly, OBM models can be applied to website banner and pop-up ads; here, each time a user loads a website, the site manager can choose ad(s) to display based on the user's information and browsing history as well as the advertisers' budget and target market.

OBM also finds applications in ride-hailing [93], another rapidly growing sector – one study by Goldman Sachs in 2017 projected that global revenues in the industry could reach \$285 billion by 2030<sup>3</sup>. Within these systems, when a user requests a ride, the ride-hailing

---

<sup>3</sup>C. Huston. "Ride-hailing industry expected to grow eightfold to \$285 billion by 2030." *MarketWatch*,

platform must match them to an available driver, with the overall goal of maximizing some measure of customer satisfaction or utility (for example, by minimizing users' average waiting time before a pickup).

As in classical deterministic bipartite matching, OBM involves matching nodes on opposite sides of a bipartite graph, with the objective of maximizing the cardinality of the matching or a more general weight function. In online versions of the problem, nodes on one or both sides of the bipartition may appear and/or disappear dynamically, matches are often irrevocable, and decisions must usually be made with only partial information about the underlying graph. In the version of OBM we study in this thesis, one side of the bipartition is fixed and known, representing the goods or resources the platform can offer; the nodes from the other side, representing customers, arrive sequentially, one at a time. Upon each arrival, the platform must immediately and irrevocably match the arriving node to a remaining compatible node from the other side or discard it. Given this model, most of the literature has focused in obtaining policies with provable guarantees in terms of competitive ratio (see Section 1.1.2 for a detailed definition). In simple words, the general goal is to design policies that achieve an objective value at least as good as a fraction of the offline optimum. In the following we provide a brief description of the different arrival models in OBM and the most relevant results in the literature.

**The Adversarial Model.** The OBM model was introduced by Karp, Vazirani and Vazirani [61], who studied the adversarial version in which there is an adversary (or nature) who chooses the sequence of arriving nodes and the decision maker does not have any information in advance about them. In this case, the adversary's objective is to maximize the difference between the cardinality of the decision maker's matching and the offline optimum. In other words, we may think in an adversary who wants our algorithm to perform as bad as possible, so he creates the worst possible instance: graph and input order. Karp, Vazirani and Vazirani [61] show that no deterministic algorithm can achieved a com-

---

published May 27, 2017.

petitive ratio better than a  $1/2$ . Then, they propose a randomized *ranking* algorithm that chooses a random permutation of the resource nodes and matches the highest-ranked available and compatible node according to the permutation yields an optimal competitive ratio of  $1 - 1/e$ ; see e.g. [13] for a simplified and corrected proof. Much subsequent work in the area has focused on less conservative or restrictive frameworks than the adversarial one, where the decision maker may have some advance information about the arriving nodes, such as an underlying distribution. Some examples are the *random model* and the *i.i.d. model*.

**The Random Model.** In this model, the adversary is only allowed to choose the graph but not the arrival order. The sequence of arriving nodes is assumed to be random, i.e., a permutation of the set of impressions is chosen uniformly at random before the process starts. Goel and Mehta [44] show that a simple *greedy* algorithm, which matches an arriving vertex to any available neighbor, achieves a competitive ratio of  $1 - 1/e$ . They show that this algorithm is a particular case of the *ranking* algorithm previously mentioned. Later, [59, 79] prove that under this model the *ranking* algorithm achieves a factor strictly better than  $1 - 1/e$ . In [59] the authors obtain a factor of 0.653 by solving a family of strongly factor revealing LP, while in [79] the authors obtain a competitive ratio of 0.696 by refining the original proof of the *ranking* algorithm.

**The I.I.D. Model.** The i.i.d. version of OBM (sometimes also called the *known i.i.d. model*) is in some sense the least conservative OBM variant, compared to the most conservative adversarial version. In this case, arriving nodes are i.i.d. draws from an underlying uniform distribution over possible node “types”, representing customer classes that may or may not be compatible with different resources or goods. For example, in search engine advertisement, advertisers indicate which search terms they wish their ads displayed with, and the search engine can only match ads with terms in these classes. The i.i.d. assumption implies this model is applicable in situations where the platform can forecast customer

behavior, e.g. based on past arrival data, and where this behavior is relatively stable over time. The model may not be as applicable in data-poor situations where the platform cannot confidently forecast customer behavior; the literature includes more conservative models for such cases [81], culminating with the adversarial model studied in [61]. Conversely, if customer behavior can be forecast but is not necessarily stable over time, the assumption of identical distributions may be problematic. While we are not aware of OBM models for this case, the revenue management literature includes many works in this vein, particularly in network revenue management, e.g. [104].

Perhaps because of its applicability in search engine advertisement, the algorithms community has extensively studied i.i.d. OBM for the past decade, starting with [39]. Feldman et al. [39] propose two different heuristics: first, a heuristic called *one suggested matching*, which follows the edges from a specific matching computed via a max-flow relaxation. This procedure achieves a  $1 - 1/e$  competitive ratio, equal to the best-possible factor in the adversarial case. The second heuristic called *two suggested matchings* uses two different matchings instead of one and achieves a factor of  $\frac{1 - \frac{2}{3}}{\frac{4}{3} - \frac{2}{3e}} \approx 0.67$ , improving the previous ratio. Later, in [6] the ratio is improved to 0.699 by using a modified version of the *two suggested matchings* algorithm. An adaptive version of the *two suggested matchings* heuristic is proposed in [80] which gives a competitive ratio of 0.702. In the same work, the authors prove that no online algorithm can achieve a factor better than 0.823, being the best upper bound known up to date. Haeupler et al. [49] propose a heuristic that uses a third *pseudo-matching*, improving the lower bound to 0.703 when the expected number of arrivals for each node is assumed to be integral. With the same integrality assumption, the ratio is improved to 0.705 in [17], and later Jaillet and Lu [57] present a randomized algorithm that achieves a competitive ratio of  $1 - \frac{2}{e^2} \approx 0.726$ ; to our knowledge, the current best factor. Also, Jaillet and Lu [57] prove that without the integrality assumption their algorithm gives a factor of 0.706.

Other models, variants and extensions have appeared in the literature; we refer the

interested reader to [81] for a detailed survey on this problem.

### 1.2.1 Model Description

In this thesis, we study the i.i.d. variant of the *online bipartite matching* (OBM) between two node sets,  $N$  and  $V$ , with edge set  $E \subseteq N \times V$ . This process occurs dynamically in the following way. The *right-hand* set  $V$ , with  $|V| = m$ , is known and given ahead of time. The *left-hand* set  $N$  with  $|N| = n$  represents node types that may appear, but we do not know which ones will appear and how often. We only know that  $T$  elements in total will appear sequentially, each one drawn independently from the (stationary) uniform distribution over  $N$ . In other words, at each stage a new node arrives with probability  $1/n$  of belonging to any of the types in  $N$ . A left-hand node's type indicates which elements of  $V$  it is connected to, and a new node of the same type is treated as a separate copy. Without loss of generality, we assume a uniform distribution over type of nodes in  $N$ . If the original distribution has different rational probabilities, then we can duplicate nodes to analyze the problem with a uniform distribution. Several past works on this OBM model (e.g. [39]) require  $T = n$  so that the expected number of appearances of any left-hand type is one, but for the model we do not need this assumption. Each time a left-hand node appears, it must be immediately (and irrevocably) matched to an available compatible node in  $V$  or discarded. The objective is to maximize the expected number of matches. Following convention from previous literature and the motivating application of search engine advertisement, we sometimes call  $i \in N$  an *impression*, and each  $j \in V$  an *ad*. In Chapter 2 we focus on maximizing the expected cardinality of the matching, even though our results still hold for the general model with weights on edges. In Chapter 3 we consider the dynamic weighted model in which each edge has a reward  $w_{ij}^t$ . The objective in this case is maximizing the expected total weight of the matching. As we will see, this provides us a more flexible framework, since we do not have to deal with the structure of the graph.

### 1.2.2 Overview of Our Contributions

The first part of this thesis, Chapters 2 and 3, focuses on providing a novel polyhedral approach to the well-known online bipartite matching problem (OBM). Specifically,

1. In Chapter 2, we focus on a *static* polyhedral approach. Most of the previous heuristic policies and their worst-case guarantees [39, 6, 80, 57] rely on simple linear programming (LP) relaxations, often with a network flow structure. However, there is comparably far less work focusing directly on the derivation of strong relaxations for i.i.d. OBM. These relaxations provide dual upper bounds useful for benchmarking any new heuristic policies and often can be employed in policy design as well.

Our main contribution in this chapter is to derive *static relaxations* from the polyhedral representation of the problem, in which the decision variables are not time-indexed. To our knowledge, this is the first polyhedral study in the context of OBM. In concrete, we obtain several valid inequalities for the polyhedron of achievable probabilities and we study their facial dimension. By constructing relaxations, we can obtain upper bounds for the true optimal value of the problem. On the other hand, we devise several policies via dual multipliers of the aforementioned inequalities. By designing new policies, we can obtain lower bounds for the optimal value. We show that some of these policies correspond to ranking policies that achieve a  $1 - 1/e$  competitive ratio. Finally, we present an extensive computational study in which we use the previous upper and lower bounds to obtain an empirical gap where we know the true optimum lies.

2. In Chapter 3, we focus on *dynamic relaxations* of the problem. While the notion of dynamic relaxations appears to be new in the OBM context, there is a stream of related literature in network revenue management, beginning with [2], who introduced *dynamic bid relaxations* and their corresponding policies. In this literature, the goal is often to show that a particular relaxation can be computed efficiently, e.g. [73,

106, 110], as a naive formulation involves a separation problem solved via an integer program. These dynamic relaxations have also been extended to customer choice models, e.g. [110, 114].

Our main contribution in this chapter is to derive *dynamic relaxations* from the polyhedral representation of the process. In this case the decision variables take into account the dynamic nature of the process since they are time-indexed. As it is done in Chapter 2, we obtain several valid inequalities for the polyhedron of achievable probabilities and we study their facial dimension. More importantly, we show that the new family of time-indexed inequalities theoretically imply the best performing upper bound obtained with static relaxations. This gives us a guarantee that the dynamic relaxations theoretically provide upper bounds at least as good as the static relaxations previously studied. We empirically show via several computational experiments that this improvement is significant, which translates in a tighter empirical gap where the optimal value lies. We also devise a new dynamic policy that resembles the dynamic bid policies from network revenue management [2].

### 1.3 Constrained Submodular Function Maximization

The second part of this thesis, Chapter 4 and Chapter 5, is focused on one of the most classic optimization problems related to subset selection: the *constrained submodular function maximization* problem.

Constrained submodular function maximization has seen significant progress in recent years in the design and analysis of new algorithms with guarantees [23, 36, 19, 102], as well as numerous applications - especially in constrained subset selection problems and more broadly machine learning. Common examples of these applications are influence modeling in social networks [64], sensor placement [71], document summarization [76], or in general constrained feature selection [69, 32, 70, 72, 92]. A typical example is the problem of picking a subset of candidate sensor locations for spatial monitoring of certain phenomena

such as temperature, ph values, humidity, etc. (see [71]). Here the goal is typically to find sensor locations that achieve the most coverage or give the most information about the observed phenomena. Submodularity naturally captures the decreasing marginal gain in the coverage, or the information acquired about relevant phenomena by using more sensors, [32]. In the following section, we will provide the necessary background and related literature about monotone submodular function maximization.

### 1.3.1 Background and Related Literature

Consider a ground set of  $n$  elements  $V = \{1, \dots, n\}$  and a non-negative set function  $f : 2^V \rightarrow \mathbb{R}_+$ . We denote the marginal value for any subset  $A \subseteq V$  and  $e \in V$  by  $f_A(e) := f(A + e) - f(A)$ , where  $A + e := A \cup \{e\}$ . A set function  $f$  is *submodular* if and only if it satisfies the *diminishing returns property*. Namely, for any  $e \in V$  and  $A \subseteq B \subseteq V \setminus \{e\}$ ,  $f_A(e) \geq f_B(e)$ . We say that  $f$  is *monotone* if for any  $A \subseteq B \subseteq V$ , we have  $f(A) \leq f(B)$ . To ease the notation, we will write the value of singletons as  $f(e) := f(\{e\})$ . For simplicity, it is generally assumed that  $f$  is *normalized*, i.e.,  $f(\emptyset) = 0$ . Otherwise, we could define  $\tilde{f}(S) := f(S) - f(\emptyset)$ , and properties such as monotonicity and submodularity are preserved.

#### *Cardinality Constraints*

The most common constraint used in submodular optimization is the cardinality constraint. In this setting, we are given a non-negative integer  $K$ , and the goal is to optimally select a subset  $S$  that contains at most  $K$  elements of  $V$ . Formally, given a non-negative monotone submodular function  $f$ , the optimization problem we are interested is the following

$$\max\{f(S) : |S| \leq K\}. \tag{1.1}$$

We assume the *value oracle model*, in which the decision-maker has access to the function  $f$  via a black box, i.e., for each query  $S \subseteq V$  the black box returns the value  $f(S)$ . Problem (1.1) is known to be NP-hard, namely it cannot be solved exactly in polynomial time. Moreover, it is also hard in the value oracle model, namely in the worst case one requires an exponential number of queries to obtain an optimal solution. Therefore, one natural approach is to design approximation algorithms that make a polynomial number of queries to the oracle and at the same time provide provable guarantees for its output solution, as defined in Section 1.1.1. The classical work of Nemhauser, Fisher and Wolsey [87] study the standard *greedy* algorithm, which constructs a set by adding in each iteration the element with the best marginal while maintaining feasibility. For completeness, we outline this procedure in Algorithm 1.

---

**Algorithm 1** Standard Greedy for Cardinality, [87]

---

**Input:** ground set  $V = \{1, \dots, n\}$ , monotone submodular function  $f : 2^V \rightarrow \mathbb{R}_+$ , and  $K \in \mathbb{Z}_+$ .

**Output:** feasible set  $S$  with  $|S| \leq K$ .

- 1: Initialize  $S = 0$ .
  - 2: **while**  $|S| < K$  **do**
  - 3:      $S \leftarrow S + \operatorname{argmax}_{e \in V \setminus S} f_S(e)$
- 

Nemhauser, Fisher and Wolsey [87] show that this algorithm achieves a  $1 - 1/e$  approximation factor in the worst-case and it requires  $O(n \cdot K)$  number of queries. More importantly, this ratio is the best possible, namely one requires an exponential number of evaluations in order to improve beyond  $1 - 1/e$  [86]. Even when the decision-maker has explicit access to the objective function, this factor is still the best possible, unless  $P = NP$  [38].

In real-world applications the standard greedy algorithm presented in [87] is computationally expensive for large data-sets, since requires  $O(n \cdot K)$  function calls. Since then, there has been significant progress on reducing the number of evaluations. The method called *lazy evaluations* for the standard greedy algorithm is presented in [83]. Even though, it is not possible to provide guarantees on the number of function calls when using this

method, empirically this tool considerably speeds up the vanilla version of the greedy algorithm. Later, Badanidiyuru and Vondrak [5] presented the *threshold greedy* algorithm that uses  $O\left(\frac{n}{\delta} \cdot \log \frac{n}{\delta}\right)$  number of function calls, where  $\delta > 0$  is the threshold parameter. As we can see the number of queries does not longer depend on the budget  $K$ . For completeness, we outline the threshold greedy algorithm in Algorithm 2.

---

**Algorithm 2** Threshold-Greedy for Cardinality, [5]

---

**Input:** ground set  $V = \{1, \dots, n\}$ , monotone submodular function  $f : 2^V \rightarrow \mathbb{R}_+$ ,  $K \in \mathbb{Z}_+$ , and  $\delta > 0$ .

**Output:** feasible set  $S$  with  $|S| \leq K$ .

- 1:  $S \leftarrow \emptyset$
  - 2:  $d \leftarrow \max_{e \in V} f(e)$
  - 3: **for** ( $\omega = d; \omega \geq \frac{\delta}{n}d; \omega \leftarrow (1 - \delta)\omega$ ) **do**
  - 4:     **for**  $e \in V$  **do**
  - 5:         **if**  $|S + e| \leq K$  and  $g_S(e) \geq \omega$  **then**
  - 6:              $S \leftarrow S + e$
- 

In simple words, Algorithm 2 allows to add more than one element in a single iteration, as long as its marginal value is above a certain threshold determined by  $\delta > 0$ . Badanidiyuru and Vondrak [5] proved that this algorithm achieves a  $(1 - 1/e - \delta)$ -factor approximation for Problem (1.1). Later, Mirzasoleiman et al. [84] introduced the *stochastic greedy* algorithm. Roughly speaking, this procedure samples a smaller ground set  $\tilde{V}$  in each iteration and chooses the element with the best marginal value among the elements in  $\tilde{V}$ . This algorithm performs  $O(n \cdot \log \frac{1}{\tilde{\epsilon}})$  function evaluations, where  $\tilde{\epsilon}$  is the parameter to control the number of samples. For completeness, we outline the stochastic greedy algorithm in Algorithm 3.

---

**Algorithm 3** Stochastic Greedy for Cardinality, [84]

---

**Input:** ground set  $V = \{1, \dots, n\}$ , monotone submodular function  $f : 2^V \rightarrow \mathbb{R}_+$ ,  $K \in \mathbb{Z}_+$  and  $\tilde{\epsilon} > 0$ .

**Output:** feasible set  $S$  with  $|S| \leq K$ .

- 1: Initialize  $S = \emptyset$ .
  - 2: **while**  $|S| < K$  **do**
  - 3:      $\tilde{V} \leftarrow$  a random subset obtained by sampling  $\frac{n}{K} \cdot \ln \frac{1}{\tilde{\epsilon}}$  random elements from  $V \setminus S$ .
  - 4:      $S \leftarrow S + \operatorname{argmax}_{e \in \tilde{V}} f_S(e)$
-

Mirzasoleiman et al. [84] proved that Algorithm 3 achieves in expectation a  $(1 - 1/e - \tilde{\epsilon})$ -factor approximation for Problem (1.1). For other efficient algorithms see e.g., [75, 20].

### *Matroid Constraints*

A natural generalization of the cardinality constraint is the class of *matroid* constraints. For a family of subsets  $\mathcal{I} \subseteq 2^V$ ,  $\mathcal{M} = (V, \mathcal{I})$  is a matroid if: (1) for all  $A \subset B \subseteq V$ , if  $B \in \mathcal{I}$  then  $A \in \mathcal{I}$ ; and (2) for all  $A, B \in \mathcal{I}$  with  $|A| < |B|$ , there is an element  $e \in B \setminus A$  such that  $A + e \in \mathcal{I}$ . Sets in such a family  $\mathcal{I}$  are called *independent* sets, or to avoid any confusion we simply call them *feasible* sets. Maximal independent sets are called *basis*. A single cardinality constraint corresponds to the *uniform matroid*,  $\mathcal{I} = \{S \subseteq V : |S| \leq K\}$ . Other important matroids are the *partition* matroid, *gammoids* and *graphical matroids*.

Given a non-negative monotone submodular function  $f$  and matroid  $\mathcal{M} = (V, \mathcal{I})$ , the generalization of problem (1.1) to a single matroid constraint can be stated as follows

$$\max\{f(S) : S \in \mathcal{I}\}. \tag{1.2}$$

In this case, besides assuming the value oracle for the objective function  $f$ , we also assume the existence of a membership oracle for  $\mathcal{I}$ . Since Problem (1.2) is a generalization of Problem (1.1), then (1.2) is also NP-hard to solve. As before, the objective is to design approximation algorithms that require a polynomial number of queries to the value and membership oracle. More importantly, we look for algorithms that provide probable guarantees of its output solution. The follow-up work of Fisher, Nemhauser and Wolsey [42] show that the standard greedy algorithm achieves  $1/2$  approximation factor for Problem (1.2). For completeness, we outline the greedy algorithm for matroid constraints in Algorithm 4.

Later, the  $1/2$  approximation factor was notably improved by Vondrák [108] using a continuous generalization of the greedy algorithm to obtain the best possible  $1 - 1/e$ . Be-

---

**Algorithm 4** Standard Greedy for Matroid, [87]

---

**Input:** ground set  $V = \{1, \dots, n\}$ , monotone submodular function  $f : 2^V \rightarrow \mathbb{R}_+$ , and matroid  $\mathcal{M} = (V, \mathcal{I})$ .

**Output:** feasible set  $S$  with  $|S| \leq r$ .

- 1: Initialize  $S = \emptyset$ .
  - 2: **while**  $V \neq \emptyset$  **do**
  - 3:      $e^* \leftarrow \operatorname{argmax}_{e \in V} f_S(e)$
  - 4:     **if**  $S + e^* \in \mathcal{I}$  **then**
  - 5:          $S \leftarrow S + e^*$
  - 6:      $V \leftarrow V - e^*$
- 

fore stating the continuous greedy algorithm, we need some preliminary definitions. We denote the indicator vector of a set  $S \subseteq V$  by  $\mathbb{1}_S \in \{0, 1\}^V$ , where  $\mathbb{1}_S(e) = 1$  if  $e \in S$  and zero otherwise; the matroid polytope by  $\mathcal{P}(\mathcal{M}) = \operatorname{conv}\{\mathbb{1}_S \mid S \in \mathcal{I}\}$  and for any  $\tau > 0$  let  $\tau \cdot \mathcal{P}(\mathcal{M}) = \operatorname{conv}\{\tau \cdot \mathbb{1}_S \mid S \in \mathcal{I}\}$  be the  $\tau$ -scaling of the matroid polytope. For any non-negative set function  $f : 2^V \rightarrow \mathbb{R}_+$ , its *multilinear extension*  $F : [0, 1]^V \rightarrow \mathbb{R}_+$  is defined for any  $y \in [0, 1]^V$  as the expected value of  $f(S_y)$ , where  $S_y$  is the random set generated by drawing independently each element  $e \in V$  with probability  $y_e$ . Formally,

$$F(y) = \mathbb{E}_{S \sim y}[f(S)] = \sum_{S \subseteq V} f(S) \prod_{e \in S} y_e \prod_{e \notin S} (1 - y_e). \quad (1.3)$$

Observe, this is in fact an extension of  $f$ , since for any subset  $S \subseteq V$ , we have  $f(S) = F(\mathbb{1}_S)$ . For any  $x, y \in [0, 1]^V$ , we will denote  $x \vee y$  the vector whose components are  $[x \vee y]_e = \max\{x_e, y_e\}$ .

**Fact 1.** [23] *Let  $f$  be a monotone submodular function and  $F$  its multilinear extension:*

1. *By monotonicity of  $f$ , we have  $\frac{\partial F}{\partial y_e} \geq 0$  for any  $e \in V$ . This implies that for any  $x \leq y$  coordinate-wise,  $F(x) \leq F(y)$ . On the other hand, by submodularity of  $f$ ,  $F$  is concave in any positive direction, i.e., for any  $e_1, e_2 \in V$  we have  $\frac{\partial^2 F}{\partial y_{e_1} \partial y_{e_2}} \leq 0$ .*
2. *Throughout this thesis we will denote by  $\nabla_e F(y) := \frac{\partial F(y)}{\partial y_e}$ , and*

$$\Delta_e F(y) := \mathbb{E}_{S \sim y}[f_S(e)]. \quad (1.4)$$

It is easy to see that  $\Delta_e F(y) = (1 - y_e) \nabla_e F(y)$ . Moreover, for any  $x, y \in [0, 1]^V$  it is easy to prove that

$$F(x \vee y) \leq F(x) + \Delta F(x) \cdot y \leq F(x) + \nabla F(x) \cdot y. \quad (1.5)$$

For more details on the properties of  $F$  see e.g., [23]. We are ready to outline the continuous greedy algorithm introduced by Vondrák [108] (for other versions of the algorithm we refer to [23, 41]). Broadly speaking, this algorithm starts with the empty solution, and in every step finds a feasible direction according to the weights given by the gradient of the multilinear extension. Then, it uses this feasible direction to obtain the next point. At the end, it finds a fractional feasible point in  $P(\mathcal{I})$  which can be rounded via *pipage rounding* [3] to obtain a feasible set. For completeness, we outline this procedure in Algorithm 5.

---

**Algorithm 5** Continuous Greedy [108, 23]

---

**Input:** multilinear extension  $F : [0, 1]^V \rightarrow \mathbb{R}_+$ , matroid polytope  $\mathcal{P}(\mathcal{I})$ .

**Output:** feasible set  $S \in \mathcal{I}$ .

- 1: Initialize  $x_0 = 0$ .
  - 2: **for**  $t \in [0, 1]$  **do**
  - 3:      $y_t \leftarrow \operatorname{argmax}_{y \in P(\mathcal{I})} \nabla F(x_t) \cdot y$
  - 4:      $\frac{dx_t}{dt} = y_t$
  - 5: Pipage rounding on  $x_1$  to obtain  $S$ .
- 

For other results in constrained submodular maximization with non-monotone objectives or different combinatorial constraints see e.g., [102, 74, 41, 22, 21, 36, 20]).

### 1.3.2 Overview of Our Contributions

The second part of this thesis, Chapters 4 and 5, focuses on providing new models and insights in submodular optimization. Specifically,

1. In Chapter 4, we focus on the *robust* version of Problem (1.2), in which we no longer consider a single monotone submodular function, but a collection of  $k > 1$  different

monotone submodular functions. The objective in this case is to obtain a single feasible set that maximizes the minimum of the  $k$  objective functions. It is not difficult to construct an example to show that the minimum of  $k$  submodular functions is no longer submodular, see e.g. [71]. Moreover, this problem is known to be NP-hard to approximate within any polynomial factor [71]. Therefore, one natural approach is to design bi-criteria algorithms that output a solution which may not be feasible but whose objective value is guaranteed to be nearly optimal. For the cardinality case, this was already studied in [71].

Our main contribution in Chapter 4 is to expand the previous literature by studying the offline and online variants of the robust submodular maximization problem under different classes of combinatorial constraints such as matroids, knapsack constraint and multiple matroids. For both models, we provide bi-criteria algorithms with provable guarantees. One of the main benefits of our bi-criteria approach is that the decision-maker can determine the tradeoff between the size of the solution versus the quality of the solution in terms of objective value. For the offline version of the problem, we give a nearly optimal approximation algorithm that uses an *extended* version of the standard greedy algorithm. This procedure outputs a *small* family of feasible solutions whose union achieves a nearly optimal objective value. The size of this family will depend logarithmically on  $k$  and the parameter that controls the quality of the solution. We also study efficient bi-criteria algorithms: for this we introduce an extended version of the threshold greedy algorithm adapted to matroid constraints along with other implementation improvements. We show via three computational experiments in real-world data-sets that we drastically improve the performance of the bi-criteria algorithms by introducing the aforementioned tools. For the online model, we provide an online bi-criteria algorithm that outputs in each stage a small family of feasible sets whose union will contribute to obtain a sub-linear regret. Again, the size of the family is controlled by the quality of the solution that

the decision-maker wants to achieve. Our online algorithm is built on the discretized version of the continuous greedy algorithm.

2. In Chapter 5, we focus on the concept of *sharpness* in submodular optimization. Even though, the worst-case guarantee for the standard greedy algorithm 1 is  $1 - 1/e$ , empirical studies have shown that this algorithm performs considerably better in practice. Previous literature has tried to explain this behavior via the concepts of *curvature* [30] and *submodular stability* [25]. The former measures how far the submodular function is from being linear. The latter defines instances in which a unique optimal solution remains unique under multiplicative perturbations. We show that both concepts are somehow unsatisfactory since there are real instances in which the standard greedy algorithm still finds an optimal solution, but the curvature analysis or stability analysis does not provide a fair explanation to this. Our main goal in Chapter 5 is to provide a candidate explanation based on the concept of sharpness that we borrow from continuous optimization. Broadly speaking, sharpness characterizes the behavior of the objective function around the set of optimal solutions. A natural example of this in continuous optimization is the class of strongly convex functions, in which the sharpness property provides faster convergence rates. In our case, *submodular sharpness* translates in an optimal solution that stands out over the rest of feasible solution.

Our main contribution in Chapter 5 is to introduce the concept of submodular sharpness; to our knowledge, the first attempt in submodular optimization. We show that classical algorithms such as the standard greedy and the continuous greedy provide better approximation factors as the sharpness of the submodular objective increases. We emphasize that these algorithms automatically adapt to the sharpness of the function, without requiring any extra parameter as part of the input. We also provide theoretical guarantees for two generalizations of the sharpness criterion: *dynamic sharpness* and *approximate sharpness*. Finally, we show in an exhaustive computa-

tional study with real-world data that the sharpness criterion stands as a better candidate to explain the behavior of the greedy algorithms, compared to the curvature and stability analysis.

## CHAPTER 2

### STATIC RELAXATIONS FOR ONLINE BIPARTITE MATCHING

#### 2.1 Introduction

The model presented in Section 1.2.1 has applications in a variety of resource allocation and revenue management areas, particularly in online search advertisement, where right-hand nodes represent ads and left-hand nodes are search terms that the search engine wants to show compatible ads to. This line of research starts with the classical work of Karp, Vazirani, and Vazirani [61] which showed that for a maximum cardinality objective, a randomized ranking policy achieves the best possible competitive ratio  $1 - 1/e$ , assuming an adversary chooses which left-hand nodes appear.

The adversarial model of node arrival is relatively pessimistic, and more recent research has focused on models where arrivals are at least partly governed by a distribution. In the simplest case, each arriving node is an i.i.d. sample from a known distribution. Work on this model and its variants began with Feldman et al. [39], and several improvements in terms of competitive analysis have been achieved since then, see e.g. [6, 80]. The best currently known analysis is by Jaillet and Lu [57], which uses a simple max-flow relaxation to generate a randomized adaptive algorithm. For more details on the guarantees of each of these works, we refer to Section 1.2. While these works sometimes employ simple relaxations to design policies, to our knowledge no researchers have specifically looked at the generation of good upper bounds for the problem. Our objective in this chapter is to provide upper and lower bounds for the optimal solution via a polyhedral approach.

### 2.1.1 Main Contributions

In this chapter we present a novel polyhedral approach to the i.i.d. variant of the online bipartite matching problem. Specifically, our main contributions are

1. First, to study the set of matching probabilities achievable by some feasible policy, which is implicitly encoded as the projection of a doubly exponential polyhedron, and to derive relaxations by identifying various classes of valid inequalities. More importantly, each of these relaxations provide an upper bound for the true optimal solution. This focus on the *achievable region* is used in applied probability, for example to study models in queueing and multi-armed bandits (e.g. [11, 29]), but to our knowledge it has not been applied to online matching.
2. Second, to show that optimal dual solutions of our relaxations imply specific policies, including simple ranking or time-dependent ranking policies in certain cases. This connection is established by enforcing intuitive value function approximations on the linear programming formulation of the problem's dynamic program. Moreover, each of these policies will provide a lower bound for the true optimal solution.
3. Finally, by constructing upper bounds from relaxations and lower bounds from policies, we provide an empirical gap in which we know the true optimum lies. For this, we present an extensive computational study to show the strength of our bounds in several classes of graphs including sparse, dense and regular instances.

In the remainder of this chapter, Section 2.2 presents our polyhedral approach to the problem. Section 2.3 discusses our proposed inequalities. Section 2.4 then shows how the relaxations imply policies, and Section 2.5 summarizes the results of our computational study. Section 2.6 then concludes and outlines future research avenues.

## 2.2 A Novel Polyhedral Approach

In this section, we will provide a short description of how the model presented in Section 1.2.1 can be approached in a polyhedral manner. For simplicity, our objective is to maximize the expected cardinality of the matching, even though our results hold in the general weighted case. For any impression  $i \in N$ , let  $\Gamma(i) \subseteq V$  denote  $i$ 's neighbors, and define  $\Gamma(j)$  for ad  $j \in V$  analogously; also, let  $\eta$  be the random variable with uniform distribution over  $N$ . We can now give a dynamic programming (DP) formulation for this OBM model. Let  $v_t^*(i, S)$  denote the optimal expected value given that  $i \in N$  appears when the set of ads  $S \subseteq V$  is available and  $t - 1$  draws from  $N$  still remain. Then, for all  $t = 1, \dots, T$ ,  $i \in N$  and  $S \subseteq V$ ,

$$v_t^*(i, S) = \max \begin{cases} \max_{j \in S \cap \Gamma(i)} \{1 + \mathbb{E}_\eta[v_{t-1}^*(\eta, S \setminus j)]\} \\ \mathbb{E}_\eta[v_{t-1}^*(\eta, S)], \end{cases} \quad (2.1)$$

where  $v_0^*(\cdot, \cdot)$  is identically zero, and the model's optimal expected value is given by  $\mathbb{E}_\eta[v_T^*(\eta, V)]$ . The first term in this recursion corresponds to matching  $i$  with one of its remaining neighbors  $j \in S \cap \Gamma(i)$ ; the second corresponds to discarding  $i$ . As with any DP, the optimal value function  $v^*$  induces an optimal policy: At any state  $(t, i, S)$ , we choose an action that attains the maximum in (2.1). It is easy to see that an optimal policy always matches an impression whenever possible, so that the discarding action is only taken when no compatible ad remains.

The recursion (2.1) can be equivalently captured with the following linear program that we denote  $\mathcal{D}$ :

$$\min_v \mathbb{E}_\eta[v_T(\eta, V)] \quad (2.2a)$$

$$\text{s.t. } v_t(i, S \cup j) - \mathbb{E}_\eta[v_{t-1}(\eta, S)] \geq 1, \quad t \in [T], i \in N, j \in \Gamma(i), S \subseteq V \setminus j \quad (2.2b)$$

$$v_t(i, S) - \mathbb{E}_\eta[v_{t-1}(\eta, S)] \geq 0, \quad t \in [T], i \in N, S \subseteq V \quad (2.2c)$$

$$v \geq 0. \quad (2.2d)$$

The value function  $v^*$  defined in (2.1) is optimal for (2.2). Moreover, this LP is a strong dual for OBM, in the sense that any feasible  $v$  has an objective greater than or equal to  $\mathbb{E}_\eta[v_T^*(\eta, V)]$ .

The dual of (2.2) is a primal formulation; any feasible solution encodes a feasible policy and its probability of reaching any state in the DP. That formulation is the following

$$\max_{x,y} \sum_{i \in N} \sum_{t \in [T]} \sum_{j \in \Gamma(i)} \sum_{S \subseteq V \setminus j} x_{i,j}^{t,S} \quad (2.3a)$$

$$\text{s.t.} \quad \sum_{j \in \Gamma(i)} x_{i,j}^{T,V \setminus j} + y_i^{T,V} \leq \frac{1}{n}, \quad i \in N, \quad (2.3b)$$

$$\sum_{j \in \Gamma(i) \cap S} x_{i,j}^{t,S \setminus j} + y_i^{t,S} - \frac{1}{n} \sum_{k \in N} y_k^{t+1,S} - \frac{1}{n} \sum_{k \in N} \sum_{j \in \Gamma(k) \cap S} x_{k,j}^{t+1,S} \leq 0, \quad (2.3c)$$

$$t \in [1, T-1], \quad i \in N, \quad \emptyset \neq S \subseteq V,$$

$$\sum_{j \in \Gamma(i) \cap S} x_{i,j}^{T,S \setminus j} + y_i^{T,S} \leq 0, \quad i \in N, \quad S \subsetneq V, \quad (2.3d)$$

$$x, y \geq 0. \quad (2.3e)$$

Here,  $x_{i,j}^{t,S}$ , the variable corresponding to dual constraint (2.2b), represents the probability that the policy chooses to match impression  $i$  to ad  $j$  in state  $(t, i, S \cup j)$ , and  $y_i^{t,S}$ , which corresponds to (2.2c), similarly represents a discarding action. As with its dual, the LP (2.3) has exponentially many variables and constraints, and is therefore difficult to work with directly. However, we can equivalently consider optimizing over the matching probabilities achieved by a feasible policy; this corresponds to optimizing over a projection of the feasible region of (2.3),

$$\max \left\{ \sum_{i \in N} \sum_{j \in \Gamma(i)} z_{ij} : \exists (x, y) \in (2.3b)-(2.3e) \text{ with } z_{ij} = \sum_{t \in [T]} \sum_{S \subseteq V \setminus j} x_{i,j}^{t,S} \right\}, \quad (2.4)$$

where  $z_{ij}$  is the probability that impression  $i$  is *ever* matched to ad  $j$ . Any such  $z$  is a vector of matching probabilities that is *achievable* by at least one feasible policy. Let  $Q_s$  denote this projected polyhedron in the space of  $z$  variables. Note that  $Q_s$  is full-dimensional in  $\mathbb{R}^{|E|}$ . We refer to this polytope as *static*, since its variables  $z_{ij}$  describe an overall probability of the entire process and do not depend on each stage  $t$ . In Chapter 3, we will study *dynamic* relaxations.

### 2.3 Projected Static Relaxations

The polyhedron  $Q_s$  captures the matching probabilities achievable by any feasible policy and optimizing over it would yield the expected value of an optimal policy. Although this optimization is computationally intractable, optimizing over any relaxation of  $Q_s$  yields a valid dual upper bound. In this section, we study various relaxations by presenting several classes of inequalities that are valid for  $Q_s$ . We begin by presenting the simplest relaxation.

Recall that each ad  $j$  can be matched at most once, so this constrains all probabilities involving  $j$  to not exceed one in total. Similarly, each impression type  $i$  appears in each epoch with probability  $1/n$ , and there are  $T$  stages, so the expected number of matches for  $i$  cannot exceed  $T/n$ . This gives us the LP

$$\max \sum_{(i,j) \in E} z_{ij} \tag{2.5a}$$

$$\text{s.t.} \quad \sum_{j \in \Gamma(i)} z_{ij} \leq \frac{T}{n}, \quad i \in N \tag{2.5b}$$

$$\sum_{i \in \Gamma(j)} z_{ij} \leq 1, \quad j \in V \tag{2.5c}$$

$$z \geq 0. \tag{2.5d}$$

In particular, when  $T = n$ , (2.5) gives the deterministic bipartite matching formulation over  $(N \cup V, E)$ , and more generally it encodes a simple max-flow model; see e.g. [39].

We can use similar probabilistic ideas to strengthen the relaxation. An impression  $i \in N$  will not appear at all with probability  $(1 - 1/n)^T$ , and thus

$$z_{ij} \leq 1 - (1 - 1/n)^T, \quad i \in N, j \in \Gamma(i) \quad (2.6)$$

is valid for  $Q_s$ ; see e.g. [49]. Though these inequalities were already known, the following result is new to our knowledge.

**Proposition 1.** *Constraints (2.6) are facet-defining for the polyhedron of achievable probabilities  $Q_s$ .*

*Proof.* Let  $\alpha := 1 - (1 - 1/n)^T$  denote the inequality's right-hand side, and define also the numbers  $\beta := 1 - (1 - 1/n)^T - \frac{T}{n}(1 - 1/n)^{T-1}$ ,  $\gamma := \frac{1}{n}(1 - 1/n)^{T-1}$ . Here  $\alpha$  represents the probability that  $i \in N$  appears at least once,  $\beta$  corresponds to the probability that  $i \in N$  appears at least twice, while  $\gamma$  is the probability that  $i \in N$  does not appear during the first  $T - 1$  epochs times the probability it appears in the last epoch. We denote the canonical vector by  $e_{ij} \in \mathbb{R}^{|E|}$ , i.e. a vector with a one in the coordinate  $(i, j)$  and zero elsewhere. We can construct the following  $|E|$  affinely independent points corresponding to policies that satisfy (2.6) with equality:

- The policy that simply matches  $(i, j)$  when possible and ignores other edges corresponds to  $\alpha e_{ij}$ .
- For any edge  $(i', j')$  that does not share an endpoint with  $(i, j)$ , the policy that matches either edge when possible corresponds to  $\alpha(e_{ij} + e_{i'j'})$ .
- For any  $j' \in \Gamma(i) \setminus j$ , the policy that matches  $(i, j)$  the first time  $i$  appears and then matches  $(i, j')$  the second time corresponds to  $\alpha e_{ij} + \beta e_{ij'}$ .
- For any  $i' \in \Gamma(j) \setminus i$ , the policy that matches  $(i, j)$  when possible but in the last epoch matches  $(i', j)$  if  $i'$  appears and  $i$  hasn't appeared corresponds to  $\alpha e_{ij} + \gamma e_{i'j}$ .  $\square$

We can generalize the previous concept to any set of impressions incident to an ad  $j \in V$ . Let  $I \subseteq \Gamma(j)$ ; no nodes from this set will appear at all with probability  $(1 - |I|/n)^T$ , and hence the set of *right-star* inequalities

$$\sum_{i \in I} z_{ij} \leq 1 - (1 - |I|/n)^T, \quad j \in V, I \subseteq \Gamma(j) \quad (2.7)$$

is valid. Moreover, their separation can be achieved in polynomial time by sorting the  $z_{ij}$  in non-increasing order of  $i$ , and testing each successive sum against the corresponding right-hand side. Inequality (2.7) also shows that (2.5c) can never be tight for a feasible policy unless  $|\Gamma(j)| = n$ ; in this case (2.5c) coincides with (2.7) for  $I = \Gamma(j) = N$ , and we get the following result.

**Proposition 2.** *If  $j \in V$  and  $\Gamma(j) = N$ , constraint (2.5c) is facet-defining for the polyhedron of achievable probabilities  $Q_s$ .*

*Proof.* Let  $j \in V$  be any ad. We can construct the following  $|E|$  affinely independent points corresponding to policies that satisfy (2.5c) with equality:

- For any  $i \in \Gamma(j)$ ,  $\alpha$  is the probability that this impression appears at least once. Take the policy that matches  $(i, j)$  whenever possible (with probability  $\alpha$ ), and if  $i$  never appears matches the impression  $k \in \Gamma(j) \setminus i$  that appears in the last draw, with probability  $\frac{1}{n-1}(1 - \alpha)$  respectively for each  $k \neq i$ . Therefore, we have  $n$  points of the form  $\alpha e_{ij} + \frac{1}{n-1}(1 - \alpha) \sum_{k \in \Gamma(j) \setminus i} e_{kj}$  for each edge  $(i, j)$ .
- The remaining policies are constructed similarly to the proof of Proposition 1.

Let us denote  $\epsilon = \frac{1}{n-1}(1 - \alpha)$ . So, we have a block matrix of size  $|E| \times |E|$ , in which the

first  $n \times n$  block corresponds to the first group of policies, and has the following form

$$C = \begin{pmatrix} \alpha & \epsilon & \epsilon & \cdots & \epsilon \\ \epsilon & \alpha & \epsilon & \cdots & \epsilon \\ \vdots & & \ddots & & \vdots \\ \epsilon & \epsilon & \epsilon & \cdots & \alpha \end{pmatrix}.$$

Let us compute the determinant of  $C$ . Add rows 2 through  $n$  to row 1 to get

$$\det(C) = \det \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ \epsilon & \alpha & \epsilon & \cdots & \epsilon \\ \vdots & & \ddots & & \vdots \\ \epsilon & \epsilon & \epsilon & \cdots & \alpha \end{pmatrix}.$$

Multiply the first row by  $-\epsilon$  and add it to every other row. Then,

$$\det(C) = \det \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & \alpha - \epsilon & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & \alpha - \epsilon \end{pmatrix} = (\alpha - \epsilon)^{n-1},$$

which is greater than zero for all  $T > 0$ . Therefore, this block matrix is invertible, which implies the points corresponding to the first group of policies are affinely independent. The remaining points are also affinely independent, as they are the same as in the previous proof.  $\square$

In general, inequalities of the form (2.7) are not facet-defining, except for those cases presented in Proposition 1 and 2.

**Proposition 3.** *If  $I \neq N$ , the face of  $Q_s$  induced by constraint (2.7) has dimension  $|E| - |I|$ .*

*Proof.* Let  $j \in V$  and  $I \subseteq \Gamma(j)$  with  $I \neq N$ . Using the same notation as in previous

propositions, construct the following points that satisfy (2.7) with equality:

- The policy that matches the first  $i \in I$  that appears to  $j$  produces the point  $\frac{1}{|I|}(1 - (1 - |I|/n)^T) \sum_{i \in I} e_{ij}$ .
- For any other edge we can construct points as we did in Proposition 1.

This gives us  $|E| - |I| + 1$  affinely independent points. Furthermore, the only way to achieve the probability  $1 - (1 - |I|/n)^T$  from the edges between  $I$  and  $j$  is to follow the outlined policy: The first time any node from  $I$  appears, it must be matched to  $j$ . Conditioning on an arrival from  $I$ , any of its members are equally likely to appear; so any point on this face must have  $z_{ij} = \frac{1}{|I|}(1 - (1 - |I|/n)^T)$  for every  $i \in I$ . This implies we cannot produce more affinely independent points.  $\square$

Let us examine the analogous situation on the other side. For an impression  $i \in N$ , consider a set  $J \subseteq \Gamma(i)$  of adjacent ads. Since  $i$  may appear more than once, the previous argument does not apply. However, we can still upper bound the corresponding probabilities by considering the expected number of matches we can hope to make with  $i$  in  $J$ . As before,  $i$  will never appear with probability  $(1 - 1/n)^T$ . Similarly,  $i$  will appear exactly once (and can thus only be matched once) with probability  $\frac{T}{n}(1 - 1/n)^{T-1}$ . This continues until we consider the event that  $i$  appears  $|J|$  or more times, because we cannot match  $i$  more than these many times in  $J$ . Let  $B(T, 1/n)$  denote a binomial random variable with  $T$  trials and probability of success  $1/n$ . The preceding argument shows that the *left-star* inequalities

$$\sum_{j \in J} z_{ij} \leq \mathbb{E}[\min\{|J|, B(T, 1/n)\}], \quad i \in N, J \subseteq \Gamma(i) \quad (2.8)$$

are valid. In addition, the same greedy algorithm used for (2.7) applies to separate them, this time sorting in non-decreasing order of  $j$ .

**Theorem 1.** *Constraints (2.8) are facet-defining for  $Q_s$  when  $|J| < T$ .*

*Proof.* Consider  $i \in N$  and  $J = \{j_1, \dots, j_r\} \subseteq \Gamma(i)$ , where  $r := |J| < T$ . Denote by  $\alpha_k$  the probability that  $i$  appears at least  $k$  times, i.e.

$$\alpha_1 = 1 - \left(1 - \frac{1}{n}\right)^T, \quad \alpha_2 = 1 - \left(1 - \frac{1}{n}\right)^T - \frac{T}{n} \left(1 - \frac{1}{n}\right)^{T-1}, \text{ etc.}$$

So, it is clear that

$$\mathbb{E}[\min\{r, B(T, 1/n)\}] = \sum_{k=1}^r \alpha_k.$$

Therefore, we can consider the following  $|E|$  affinely independent points:

- Take the  $|J|$  policies that match just node  $i$ , each time it appears, with nodes  $j \in J$  following each of the permutations given by the ordering; that is, the first permutation is  $(j_1, \dots, j_r)$ , the second is  $(j_2, \dots, j_r, j_1)$ , and so forth until the  $r$ -th permutation,  $(j_r, j_1, \dots, j_{r-1})$ . So, point  $\sum_{k=1}^r \alpha_k e_{ij_k}$  corresponds to the first policy,  $\alpha_r e_{ij_1} + \sum_{k=1}^{r-1} \alpha_k e_{ij_{k+1}}$  to the second, and so on. Thus, we need to prove that the matrix

$$C = \begin{pmatrix} \alpha_1 & \alpha_r & \alpha_{r-1} & \dots & \alpha_2 \\ \alpha_2 & \alpha_1 & \alpha_r & \dots & \alpha_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha_r & \alpha_{r-1} & \alpha_{r-2} & \dots & \alpha_1 \end{pmatrix}$$

is invertible, where  $\alpha_1 > \alpha_2 > \dots > \alpha_r$ . This type of matrix is called a *circulant*, and non-singularity follows from Proposition 24 in [67].

- For all  $j' \in \Gamma(i) \setminus J$ , we simply match  $(i, j')$  after matching all nodes in  $J$ ; this corresponds for instance to  $\sum_{k=1}^r \alpha_k e_{ij_k} + \alpha_{r+1} e_{ij'}$ . Note that  $|J| < T$ , so  $\alpha_{r+1} > 0$ .
- The points for the remaining edges follow an analogous construction to Proposition 1. □

When  $|J| \geq T$ , we have  $\mathbb{E}[\min\{|J|, B(T, 1/n)\}] = T/n$ , and therefore (2.8) for

$J \subsetneq \Gamma(i)$  is dominated by (2.5b). We can, however, use a similar proof to determine when this inequality is also facet-defining.

**Corollary 1.** *Constraints (2.8) for  $J = \Gamma(i)$  are facet-defining for  $Q_s$  regardless of  $T$ , and thus (2.5b) is facet-defining when  $|\Gamma(i)| \geq T$ .*

The proof for this corollary does not need an affinely independent point for  $(i, j)$  with  $j \notin J$  (since  $J = \Gamma(i)$ ), and hence does not require  $|J| < T$ .

Finally, consider two sets  $I \subseteq N$  and  $J \subseteq V$ . In the best case, they induce a complete bipartite subgraph, and we can proceed as before. No edges within the two sets will be matched at all with probability  $(1 - |I|/n)^T$ , exactly one will be matched with probability  $\frac{|I|}{n} (1 - |I|/n)^{T-1}$ , and so forth. Generalizing, let  $B(T, |I|/n)$  denote a binomial random variable with  $T$  trials and probability  $|I|/n$  of success. Then

$$\sum_{(i,j) \in E \cap (I \times J)} z_{ij} \leq \mathbb{E} [\min \{ |J|, B(T, |I|/n) \}], \quad I \subseteq N, J \subseteq V, \quad (2.9)$$

are valid. This general set of inequalities contains both (2.7) and (2.8) as special cases, by respectively taking  $J = \{j\}$  and  $I = \{i\}$ . Moreover, for any fixed  $I$  or  $J$  they can be separated using the same greedy algorithm, now applied to sums of the  $z$  variables; more generally, they can be separated with an integer program that maximizes the left-hand side for every fixed value of  $|I|$  and  $|J|$ . These inequalities are not necessarily facet-defining, except for the cases we have already pointed out.

All the inequalities described so far have 0-1 coefficients; a natural question is whether all of  $Q_s$ 's facets can be written with 0-1 coefficients. However, this is not true even for very small instances. We have constructed  $Q_s$  using PORTA for an instance with  $T = 3$ ,  $N = \{1, 2, 3\}$ ,  $V = \{a, b\}$  and  $E = \{1a, 1b, 2b\}$ . Not counting the non-negativity constraints,  $Q_s$  has 13 facets, but only the four identified in (2.8) have 0-1 coefficients.

## 2.4 Policies Derived from Bounds

In this section, we will study how policies can be constructed from the previous relaxations via dual multipliers. For some of those policies, we will provide their competitive ratio, i.e., the worst-case guarantee when it is compared to the optimal policy.

Any value function approximation  $v$  implies a policy by substituting it into (2.1): At any encountered state  $(t, i, S)$ , choose an ad that maximizes the right-hand side,

$$\operatorname{argmax}_{j \in S \cap \Gamma(i)} \{1 + \mathbb{E}_\eta[v_{t-1}(\eta, S \setminus \eta)]\} = \operatorname{argmax}_{j \in S \cap \Gamma(i)} \mathbb{E}_\eta[v_{t-1}(\eta, S \setminus \eta)],$$

or discard the impression if  $S \cap \Gamma(i) = \emptyset$ ; recall that an optimal policy only discards an impression if no match is possible. We next focus on policies based on the relaxations in the previous section, by generating approximations of the true value function that are feasible in the dynamic programming LP (2.2) but efficient to compute. We also show that several of these approximations lead to ranking policies (cf. [61]) and their generalizations. A *ranking policy* is specified by an ordering or permutation of  $V$ : Assuming we label ads in the permutation's order as  $V = \{1, \dots, m\}$ , at any decision epoch  $(t, i, S)$  we match the appearing impression  $i$  to  $\min\{j : j \in S \cap \Gamma(i)\}$ , the lowest-indexed compatible ad that is available. Such policies are appealing from a practical perspective, as they are completely specified by a permutation and can be implemented efficiently.

Our first two policies are ranking policies, and it then follows from [44] that their competitive ratio is  $1 - 1/e$  when  $m = n = T$ . As we explained in Section 1.2, the *competitive ratio* of a policy is the infimum, over all instances, of the policy's expected value divided by the expected value of the *offline* optimum, i.e. the maximum matching on the realized graph. The offline optimum provides an upper bound on the optimal policy's expected value, so a competitive ratio guarantee for a policy also guarantees the same multiplicative performance with respect to the optimal policy.

To begin, suppose we approximate the value at any state by considering the impression

that has just appeared and the remaining available ads. Specifically, suppose  $q_i \geq 0$  represents the value of having  $i \in N$  appear and be available to match. Similarly, let  $r_j \geq 0$  be the value we assign to each available ad  $j \in V$ . This leads to an approximation of the expected value of state  $(t, i, S)$  as

$$v_i(i, S) \approx q_i + (t - 1)\mathbb{E}_\eta[q_\eta] + \sum_{j \in S} r_j. \quad (2.10)$$

In the approximation,  $i$  has just appeared, and thus the state has value  $q_i$ . In addition, there are  $t - 1$  more draws remaining, so we will get the expected value of  $q_\eta$  that many more times. Finally, each  $j \in S$  still available to match contributes its value  $r_j$ .

**Proposition 4.** *Suppose we restrict the feasible region of  $\mathcal{D}$  by forcing solutions to have the form (2.10), where the decision variables are now  $q_i$ ,  $i \in N$  and  $r_j$ ,  $j \in V$ . The resulting LP is equivalent to*

$$\min_{q, r \geq 0} \left\{ \frac{T}{n} \sum_{i \in N} q_i + \sum_{j \in V} r_j : q_i + r_j \geq 1, (i, j) \in E \right\}, \quad (\mathcal{D}_1)$$

the dual of (2.5).

*Proof.* To prove the proposition, we must establish two facts. First, the objective function (2.2a) of  $\mathcal{D}$  reduces to  $(\mathcal{D}_1)$ 's objective under restriction (2.10); and second, the feasible region of  $\mathcal{D}$  collapses to the feasible region of  $(\mathcal{D}_1)$  under restriction (2.10).

First, for the objective function (2.2a) we have

$$\mathbb{E}_\eta \left[ q_\eta + (T - 1)\mathbb{E}_\eta[q_\eta] + \sum_{j \in V} r_j \right] = \frac{T}{n} \sum_{i \in N} q_i + \sum_{j \in V} r_j.$$

For the feasible region, we take each constraint class from  $\mathcal{D}$  separately. For any  $t \in [T]$ ,

$i \in N$ ,  $j \in \Gamma(i)$ , and  $S \subseteq V \setminus j$  in the matching constraint (2.2b), we get

$$q_i + (t-1)\mathbb{E}_\eta[q_\eta] + \sum_{\ell \in S} r_\ell + r_j - \mathbb{E}_\eta \left[ (t-1)q_\eta + \sum_{\ell \in S} r_\ell \right] = q_i + r_j \geq 1.$$

Similarly, the discarding constraint (2.2c) has

$$q_i + (t-1)\mathbb{E}_\eta[q_\eta] + \sum_{\ell \in S} r_\ell - \mathbb{E}_\eta \left[ (t-1)q_\eta + \sum_{\ell \in S} r_\ell \right] = q_i \geq 0.$$

We also require  $r_j \geq 0$ , since the value function must be non-negative. Altogether, this yields  $\mathcal{D}_1$ , the dual of (2.5).  $\square$

Let  $(q^{(2.10)}, r^{(2.10)})$  be an optimal extreme point solution of  $(\mathcal{D}_1)$ . This feasible region is the convex hull of node covers of  $(N \times V, E)$ ; thus  $(q^{(2.10)}, r^{(2.10)})$  is the incidence vector of a cover, and when  $T = n$  it is a minimum cardinality cover. Suppose we are at state  $(t, i, S)$  and employ the value function approximation (2.10) given by this solution in the dynamic programming recursion (2.1) to choose an action. Assuming  $S \cap \Gamma(i) \neq \emptyset$ , this yields

$$\operatorname{argmax}_{j \in S \cap \Gamma(i)} \left\{ 1 + \mathbb{E}_\eta \left[ (t-1)q_\eta^{(2.10)} + \sum_{\ell \in S \setminus j} r_\ell^{(2.10)} \right] \right\} = \operatorname{argmin}_{j \in S \cap \Gamma(i)} r_j^{(2.10)},$$

where the equivalence follows simply by removing terms that do not depend on  $j$ . This corresponds to a *cover ranking* policy: Given an optimal node cover, we match an arriving impression if possible to a non-cover ad, and only match it to an ad in the cover when no remaining non-cover ad is compatible. Note that any ranking that orders ads so that non-cover ads appear before ads in the cover induces a cover ranking policy.

**Corollary 2.** *If  $m = n = T$ , a cover ranking policy has a competitive ratio of  $1 - 1/e$ .*

*Proof.* This follows from [44], who show that any ranking algorithm (which they term “greedy”) has this competitive ratio.  $\square$

This first approximation (2.10) does not capture the interaction between impressions and ads. Suppose we add a value  $p_{ij} \geq 0$  to a state whenever  $i \in N$  appears and  $j \in S$  is one of the remaining ads. The new value function approximation is

$$v_t(i, S) \approx q_i + (t-1)\mathbb{E}_\eta[q_\eta] + \sum_{j \in S} \left( r_j + p_{ij} + (1 - (1 - 1/n)^{t-1}) \sum_{k \in N \setminus i} p_{kj} \right). \quad (2.11)$$

Since  $i \in N$  is the current impression, the approximation includes a value  $p_{ij}$  for all remaining ads  $j$ . This value will be zero if  $(i, j) \notin E$ , but we include it to simplify the expressions. Furthermore, each other impression  $k \in N \setminus i$  will appear at least once in the remaining epochs with probability  $1 - (1 - 1/n)^{t-1}$ , so we include these values as well, discounted by that probability; we only count these values once, because an ad can only be matched once.

**Proposition 5.** *Suppose we restrict the feasible region of  $\mathcal{D}$  by forcing solutions to have the form (2.11), where the decision variables are now  $q_i$ ,  $i \in N$ ,  $r_j$ ,  $j \in V$  and  $p_{ij}$ ,  $(i, j) \in E$ . The resulting LP is equivalent to*

$$\begin{aligned} \min \quad & \frac{T}{n} \sum_{i \in N} q_i + \sum_{j \in V} r_j + \left( 1 - \left( 1 - \frac{1}{n} \right)^T \right) \sum_{(i,j) \in E} p_{ij} \\ \text{s.t.} \quad & q_i + r_j + p_{ij} \geq 1, \quad (i, j) \in E, \\ & q, r, p \geq 0, \end{aligned} \quad (\mathcal{D}_2)$$

the dual of the LP obtained by adding constraints (2.6) to (2.5).

*Proof.* The proof follows in a similar fashion to the proof of Proposition 4. First, we have the following expectation

$$\mathbb{E}_\eta \left[ p_{\eta j} + (1 - (1 - 1/n)^{t-1}) \sum_{k \in N \setminus \eta} p_{kj} \right] = (1 - (1 - 1/n)^t) \sum_{i \in N} p_{ij},$$

which immediately shows that for  $t = T$  the term

$$\mathbb{E}_\eta[v_T(\eta, V)] = \frac{T}{n} \sum_{i \in N} q_i + \sum_{j \in V} r_j + \left(1 - \left(1 - \frac{1}{n}\right)^T\right) \sum_{(i,j) \in E} p_{ij}$$

is the objective (2.2a). Furthermore, for any  $t \in [T]$ ,  $i \in N$ ,  $j \in \Gamma(i)$ , and  $S \subseteq V \setminus j$ , we have

$$\begin{aligned} & \sum_{l \in S \cup j} \left( p_{il} + \left(1 - \left(1 - \frac{1}{n}\right)^{t-1}\right) \sum_{k \in N \setminus i} p_{kl} \right) \\ & \quad - \mathbb{E}_\eta \left[ \sum_{l \in S} \left( p_{\eta l} + \left(1 - \left(1 - \frac{1}{n}\right)^{t-2}\right) \sum_{k \in N \setminus \eta} p_{kj} \right) \right] \\ & = p_{ij} + \left(1 - \left(1 - \frac{1}{n}\right)^{t-1}\right) \sum_{k \in N \setminus i} p_{kj} \\ & \quad + \sum_{l \in S} \left( p_{il} + \left(1 - \left(1 - \frac{1}{n}\right)^{t-1}\right) \sum_{k \in N \setminus i} p_{kj} \right) - \left(1 - \left(1 - \frac{1}{n}\right)^{t-1}\right) \sum_{k \in N} \sum_{l \in S} p_{kl} \\ & = p_{ij} + \left(1 - \left(1 - \frac{1}{n}\right)^{t-1}\right) \sum_{k \in N \setminus i} p_{kj} + \left(1 - \frac{1}{n}\right)^{t-1} \sum_{l \in S} p_{ij} \geq p_{ij}, \end{aligned}$$

with equality holding when  $t = 1$  and  $S = \emptyset$ . It follows that the restriction of the feasible region of (2.2) with this approximation yields the model  $(\mathcal{D}_2)$ , the dual of (2.5) with the additional constraints (2.6).  $\square$

Let  $(q^{(2.11)}, r^{(2.11)}, p^{(2.11)})$  be an extreme point optimal solution for  $(\mathcal{D}_2)$ , and suppose we use the approximation given by this optimal solution in (2.1); we call this a *probability bound policy*. At state  $(t, i, S)$  with  $S \cap \Gamma(i) \neq \emptyset$ , after removing terms that do not depend on  $j$ , this results in the optimization problem

$$\begin{aligned} & \operatorname{argmax}_{j \in S \cap \Gamma(i)} \sum_{\ell \in S \setminus j} \left( r_\ell^{(2.11)} + \left(1 - \left(1 - \frac{1}{n}\right)^{t-1}\right) \sum_{k \in N} p_{k\ell}^{(2.11)} \right) = \\ & \operatorname{argmin}_{j \in S \cap \Gamma(i)} \left\{ r_j^{(2.11)} + \left(1 - \left(1 - \frac{1}{n}\right)^{t-1}\right) \sum_{k \in N} p_{kj}^{(2.11)} \right\}. \end{aligned}$$

Though it is not immediately clear, this policy is also a ranking policy.

**Theorem 2.** *The probability bound policy is a ranking policy. Therefore, when  $m = n = T$  it has a competitive ratio of  $1 - 1/e$ .*

*Proof.* Let  $(q^{(2.11)}, r^{(2.11)}, p^{(2.11)})$  be an extreme point optimal solution for  $(\mathcal{D}_2)$ ; then it is binary because  $(\mathcal{D}_2)$  is the dual of a network flow. Moreover, by optimality it follows that if  $p_{ij}^{(2.11)} = 1$ , then  $q_i^{(2.11)} = r_j^{(2.11)} = 0$ . In particular, for any  $j \in V$ , at most one of  $r_j^{(2.11)}$  and  $\sum_k p_{kj}^{(2.11)}$  can be positive, and

$$(1 - (1 - 1/n)^{T-1}) \sum_k p_{kj}^{(2.11)} < 1,$$

because otherwise we can set  $r_j^{(2.11)} = 1$  and  $p_{ij}^{(2.11)} = 0$  for all  $i \in N$  and obtain a new solution with a better objective.

Define a ranking of ads that orders them in non-decreasing order of

$$\epsilon_j(T-1) := r_j^{(2.11)} + (1 - (1 - 1/n)^{T-1}) \sum_k p_{kj}^{(2.11)}.$$

We claim the probability bound policy chooses nodes to match based on this ranking. To see this, observe that because the solution is binary,  $\epsilon_j(T-1)$  can take at most  $n+2$  values: It can be zero or one, or  $(1 - (1 - 1/n)^{T-1}) \sum_k p_{kj}^{(2.11)}$ , where  $\sum_k p_{kj}^{(2.11)} \in \{0, \dots, n\}$ . The proof then follows by noting that the ordering of the  $\epsilon_j(t)$  values is the same for all  $t = 2, \dots, T-1$ .  $\square$

Note that a probability bound policy does not necessarily perform strictly better than a cover ranking policy. For instance, consider  $m = n = T$  and the graph is an even cycle. In this case, it is easy to see that  $p_{ij}^{(2.11)} = 0$  for all edges  $(i, j)$ , and thus the two solutions coincide.

We next generalize this approach to include all right-star inequalities.

**Theorem 3.** Consider the value function approximation

$$v_t(i, S) \approx q_i + (t-1)\mathbb{E}_\eta[q_\eta] + \sum_{j \in S} \left( r_j + \sum_{\substack{I \subseteq \Gamma(j) \\ I \ni i}} p_{Ij} + \sum_{\substack{I \subseteq \Gamma(j) \\ I \not\ni i}} (1 - (1 - |I|/n)^{t-1}) p_{Ij} \right), \quad (2.12)$$

where  $q \in \mathbb{R}_+^N$ ,  $r \in \mathbb{R}_+^V$ , and  $p_{Ij} \in \mathbb{R}_+$  for  $j \in V$  and  $I \subseteq \Gamma(j)$ . Restricting the feasible region of  $\mathcal{D}$  with this approximation yields

$$\begin{aligned} \min \quad & \frac{T}{n} \sum_{i \in N} q_i + \sum_{j \in V} r_j + \sum_{j \in V} \sum_{I \subseteq \Gamma(j)} (1 - (1 - |I|/n)^T) p_{Ij} \\ \text{s.t.} \quad & q_i + r_j + \sum_{\substack{I \subseteq \Gamma(j) \\ I \ni i}} p_{Ij} \geq 1, \quad (i, j) \in E, \\ & q, r, p \geq 0, \end{aligned} \quad (\mathcal{D}_3)$$

the dual of the LP obtained by adding constraints (2.7) to (2.5).

*Proof.* The proof follows the same structure as the proofs of Propositions 4 and 5. First, we have the following expectation

$$\mathbb{E}_\eta \left[ \sum_{\substack{I \subseteq \Gamma(j) \\ I \ni \eta}} p_{Ij} + \sum_{\substack{I \subseteq \Gamma(j) \\ I \not\ni \eta}} (1 - (1 - |I|/n)^{t-1}) p_{Ij} \right] = \sum_{I \subseteq \Gamma(j)} (1 - (1 - |I|/n)^t) p_{Ij}$$

which immediately shows that for  $t = T$  the term

$$\mathbb{E}_\eta[v_T(\eta, V)] = \frac{T}{n} \sum_{i \in N} q_i + \sum_{j \in V} r_j + \sum_{j \in V} \sum_{I \subseteq \Gamma(j)} (1 - (1 - |I|/n)^T) p_{Ij},$$

is the objective (2.2a). Furthermore, for any  $t \in [T]$ ,  $i \in N$ ,  $j \in \Gamma(i)$ , and  $S \subseteq V \setminus j$ , we

have

$$\begin{aligned}
& \sum_{l \in S \cup j} \left( \sum_{\substack{I \subseteq \Gamma(l), \\ I \ni i}} p_{Il} + \sum_{\substack{I \subseteq \Gamma(l), \\ I \not\ni i}} (1 - (1 - |I|/n)^{t-1}) p_{Il} \right) \\
& \quad - \mathbb{E}_\eta \left[ \sum_{l \in S} \left( \sum_{\substack{I \subseteq \Gamma(l), \\ I \ni \eta}} p_{Il} + \sum_{\substack{I \subseteq \Gamma(l), \\ I \not\ni \eta}} (1 - (1 - |I|/n)^{t-2}) p_{Il} \right) \right] \\
& = \sum_{\substack{I \subseteq \Gamma(j), \\ I \ni i}} p_{Ij} + \sum_{\substack{I \subseteq \Gamma(l), \\ I \not\ni i}} (1 - (1 - |I|/n)^{t-1}) p_{Ij} + \sum_{l \in S} \sum_{\substack{I \subseteq \Gamma(l), \\ I \ni i}} (1 - |I|/n)^{t-1} p_{Il} \\
& \geq \sum_{\substack{I \subseteq \Gamma(j), \\ I \ni i}} p_{Ij},
\end{aligned}$$

with equality holding when  $t = 1$  and  $S = \emptyset$ . It follows that the restriction of the feasible region of (2.2) with this approximation yields the model  $(\mathcal{D}_3)$ , the dual of (2.5) with the additional constraints (2.7).  $\square$

Let  $(q^{(2.12)}, r^{(2.12)}, p^{(2.12)})$  be optimal for  $(\mathcal{D}_3)$ . At state  $(t, i, S)$  with  $S \cap \Gamma(i) \neq \emptyset$ , using the value function approximation (2.12) we obtain the optimization problem

$$\begin{aligned}
& \operatorname{argmax}_{j \in S \cap \Gamma(i)} \sum_{\ell \in S \setminus j} \left( r_\ell^{(2.12)} + \sum_{I \subseteq \Gamma(\ell)} (1 - (1 - |I|/n)^{t-1}) p_{I\ell}^{(2.12)} \right) \\
& = \operatorname{argmin}_{j \in S \cap \Gamma(i)} \left\{ r_j^{(2.12)} + \sum_{I \subseteq \Gamma(j)} (1 - (1 - |I|/n)^{t-1}) p_{Ij}^{(2.12)} \right\}.
\end{aligned}$$

The proof of Theorem 2 does not apply here, because the coefficients multiplying the  $p_{Ij}$  values may decay at different rates with respect to  $t$ , depending on the cardinality of  $|I|$ . Nevertheless, the policy is a *time-dependent* ranking policy: At any epoch  $t$ , the policy's ranking is given by a linear combination of the  $r_j^{(2.12)}$  and  $p_{Ij}^{(2.12)}$  values; the influence of the  $p$  values in the ranking is highest in the first epoch, and decays until vanishing in the last one. As with a (static) ranking policy, we can pre-compute the  $T$  rankings and implement the policy efficiently.

We can state a similar correspondence between constraints (2.8) and a value function approximation.

**Theorem 4.** *Consider the value function approximation*

$$\begin{aligned}
v_t(i, S) &\approx q_i + (t - 1)\mathbb{E}_\eta[q_\eta] + \sum_{j \in S} r_j \\
&+ \sum_{J \subseteq \Gamma(i)} p_{iJ} \mathbb{E}[\min\{|J \cap S|, B(t - 1, 1/n) + 1\}] \\
&+ \sum_{k \in N \setminus i} \sum_{J \subseteq \Gamma(k)} p_{kJ} \mathbb{E}[\min\{|J \cap S|, B(t - 1, 1/n)\}], \tag{2.13}
\end{aligned}$$

where  $q \in \mathbb{R}_+^N$ ,  $r \in \mathbb{R}_+^V$ ,  $p_{iJ} \in \mathbb{R}_+$  for  $i \in N$  and  $J \subseteq \Gamma(i)$ . Restricting the feasible region of  $\mathcal{D}$  with this approximation yields

$$\begin{aligned}
\min & \frac{T}{n} \sum_{i \in N} q_i + \sum_{j \in V} r_j + \sum_{i \in N} \sum_{J \subseteq \Gamma(i)} p_{iJ} \mathbb{E}[\min\{|J|, B(T, 1/n)\}] \\
\text{s.t.} & q_i + r_j + \sum_{\substack{J \subseteq \Gamma(i), \\ J \ni j}} p_{iJ} \geq 1, \quad (i, j) \in E, \tag{D_4} \\
& q, r, p \geq 0,
\end{aligned}$$

the dual of the LP obtained by adding constraints (2.8) to (2.5).

*Proof.* First, we have the following expectation

$$\begin{aligned}
\mathbb{E}_\eta & \left[ \sum_{J \subseteq \Gamma(\eta)} p_{\eta J} \mathbb{E}[\min\{|J \cap S|, B(t - 1, 1/n) + 1\}] \right. \\
& \left. + \sum_{k \in N \setminus \eta} \sum_{J \subseteq \Gamma(k)} p_{kJ} \mathbb{E}[\min\{|J \cap S|, B(t - 1, 1/n)\}] \right] \\
& = \sum_{i \in N} \sum_{J \subseteq \Gamma(i)} p_{iJ} \mathbb{E}[\min\{|J \cap S|, B(t, 1/n)\}],
\end{aligned}$$

which immediately shows that for  $t = T$  the term

$$\mathbb{E}_\eta[v_T(\eta, V)] = \frac{T}{n} \sum_{i \in N} q_i + \sum_{j \in V} r_j + \sum_{i \in N} \sum_{J \subseteq \Gamma(i)} p_{iJ} \mathbb{E}[\min\{|J|, B(T, 1/n)\}]$$

is the objective (2.2a) by taking  $t = T$  and  $S = V$ . Furthermore, for any  $t \in [T]$ ,  $i \in N$ ,  $j \in \Gamma(i)$ , and  $S \subseteq V \setminus j$ , we have

$$\begin{aligned} v_t(i, S \cup j) &= \dots + \sum_{J \subseteq \Gamma(i)} p_{iJ} \mathbb{E}[\min\{|J \cap (S \cup j)|, B(t-1, 1/n) + 1\}] \\ &\quad + \sum_{k \in N \setminus i} \sum_{J \subseteq \Gamma(k)} p_{kJ} \mathbb{E}[\min\{|J \cap (S \cup j)|, B(t-1, 1/n)\}]. \end{aligned}$$

First, notice that if  $J \ni j$  then  $|J \cap (S \cup j)| = 1 + |J \cap S|$ , otherwise if  $J \not\ni j$  then  $|J \cap (S \cup j)| = |J \cap S|$ , so

$$\begin{aligned} \sum_{J \subseteq \Gamma(i)} p_{iJ} \mathbb{E}[\min\{|J \cap (S \cup j)|, B(t-1, 1/n) + 1\}] \\ &= \sum_{\substack{J \subseteq \Gamma(i), \\ J \ni j}} p_{iJ} \mathbb{E}[\min\{1 + |J \cap S|, B(t-1, 1/n) + 1\}] \\ &\quad + \sum_{\substack{J \subseteq \Gamma(i), \\ J \not\ni j}} p_{iJ} \mathbb{E}[\min\{|J \cap S|, B(t-1, 1/n) + 1\}]. \end{aligned}$$

In the same way we have

$$\begin{aligned} \sum_{k \in N \setminus i} \sum_{J \subseteq \Gamma(k)} p_{kJ} \mathbb{E}[\min\{|J \cap (S \cup j)|, B(t-1, 1/n)\}] \\ &= \sum_{\substack{k \in N \setminus i, \\ J \subseteq \Gamma(k), \\ J \ni j}} p_{kJ} \mathbb{E}[\min\{1 + |J \cap S|, B(t-1, 1/n)\}] \\ &\quad + \sum_{\substack{k \in N \setminus i, \\ J \subseteq \Gamma(k), \\ J \not\ni j}} p_{kJ} \mathbb{E}[\min\{|J \cap S|, B(t-1, 1/n)\}]. \end{aligned}$$

Hence, for  $v_t(i, S \cup j)$  we obtain

$$v_t(i, S \cup j) = \dots + \sum_{\substack{J \subseteq \Gamma(i), \\ J \ni j}} p_{iJ} \mathbb{E}[\min\{1 + |J \cap S|, B(t-1, 1/n) + 1\}] \quad (\text{A})$$

$$+ \sum_{\substack{J \subseteq \Gamma(i), \\ J \not\ni j}} p_{iJ} \mathbb{E}[\min\{|J \cap S|, B(t-1, 1/n) + 1\}] \quad (\text{B})$$

$$+ \sum_{k \in N \setminus i} \sum_{\substack{J \subseteq \Gamma(k), \\ J \ni j}} p_{kJ} \mathbb{E}[\min\{1 + |J \cap S|, B(t-1, 1/n)\}] \quad (\text{C})$$

$$+ \sum_{k \in N \setminus i} \sum_{\substack{J \subseteq \Gamma(k), \\ J \not\ni j}} p_{kJ} \mathbb{E}[\min\{|J \cap S|, B(t-1, 1/n)\}], \quad (\text{D})$$

and from previous observations we know

$$\begin{aligned} \mathbb{E}_\eta [v_{t-1}(\eta, S)] &= \sum_{i \in N} \sum_{J \subseteq \Gamma(i)} p_{iJ} \mathbb{E}[\min\{|J \cap S|, B(t-1, 1/n)\}] \\ &= \sum_{\substack{J \subseteq \Gamma(i), \\ J \ni j}} p_{iJ} \mathbb{E}[\min\{|J \cap S|, B(t-1, 1/n)\}] \end{aligned} \quad (\text{A}')$$

$$+ \sum_{\substack{J \subseteq \Gamma(i), \\ J \not\ni j}} p_{iJ} \mathbb{E}[\min\{|J \cap S|, B(t-1, 1/n)\}] \quad (\text{B}')$$

$$+ \sum_{k \in N \setminus i} \sum_{\substack{J \subseteq \Gamma(k), \\ J \ni j}} p_{kJ} \mathbb{E}[\min\{|J \cap S|, B(t-1, 1/n)\}] \quad (\text{C}')$$

$$+ \sum_{k \in N \setminus i} \sum_{\substack{J \subseteq \Gamma(k), \\ J \not\ni j}} p_{kJ} \mathbb{E}[\min\{|J \cap S|, B(t-1, 1/n)\}]. \quad (\text{D}')$$

First, we clearly have  $(D) - (D') = 0$ , and  $(B) - (B'), (C) - (C') \geq 0$ . Also, we obtain

$$(A) - (A') = \sum_{\substack{J \subseteq \Gamma(i), \\ J \ni j}} p_{iJ}.$$

In other words we have

$$\begin{aligned}
& \sum_{J \subseteq \Gamma(i)} p_{iJ} \mathbb{E}[\min\{|J \cap (S \cup j)|, B(t-1, 1/n) + 1\}] \\
& + \sum_{k \in N \setminus i} \sum_{J \subseteq \Gamma(k)} p_{kJ} \mathbb{E}[\min\{|J \cap (S \cup j)|, B(t-1, 1/n)\}] \\
& - \sum_{i \in N} \sum_{J \subseteq \Gamma(i)} p_{iJ} \mathbb{E}[\min\{|J \cap S|, B(t-1, 1/n)\}] \\
& \geq \sum_{\substack{J \subseteq \Gamma(i), \\ J \ni j}} p_{iJ},
\end{aligned}$$

with equality holding when  $t = 1$  and  $S = \emptyset$ . So, it follows that

$$v_t(i, S \cup j) - \mathbb{E}_\eta[v_{t-1}(\eta, S)] \geq q_i + r_j + \sum_{\substack{J \subseteq \Gamma(i), \\ J \ni j}} p_{iJ} \geq 1.$$

Therefore, the restriction of the feasible region of (2.2) with this approximation yields  $(\mathcal{D}_4)$ , the dual of (2.5) with the additional constraints (2.8).  $\square$

This approximation generalizes the intuition behind approximation (2.11) to subsets of ads. Suppose we model the value  $p_{iJ}$  of impression  $i$  interacting with a set of compatible ads  $J \subseteq \Gamma(i)$ . When  $i$  appears in epoch  $t$ , we expect no more than  $\mathbb{E}[\min\{|J \cap S|, B(t-1, 1/n) + 1\}]$  matches between  $i$  and  $J$  from that point forward: The number of matches cannot exceed the number of remaining ads in the set,  $|J \cap S|$ , but it also cannot exceed the number of times we expect  $i$  to appear, the current appearance plus  $B(t-1, 1/n)$  more. A similar argument applies to any impression that did not appear in this epoch.

This value function approximation (2.13) seems not to define a dynamic ranking policy; let  $(q^{(2.13)}, r^{(2.13)}, p^{(2.13)})$  be optimal for  $(\mathcal{D}_4)$ . At state  $(t, i, S)$  with  $S \cap \Gamma(i) \neq \emptyset$ , employing

the approximation (2.13) in (2.1) results in

$$\operatorname{argmax}_{j \in S \cap \Gamma(i)} \left\{ \sum_{\ell \in S \setminus j} r_\ell^{(2.13)} + \sum_{i \in N} \sum_{J \subseteq \Gamma(i)} p_{iJ}^{(2.13)} \mathbb{E}[\min\{|J \cap (S \setminus j)|, B(t-1, 1/n)\}] \right\}.$$

Because the coefficients multiplying the  $p$  variables depend explicitly on the set  $S$  of remaining ads, it is impossible to compute these expressions a priori to obtain a ranking. We have nevertheless also implemented this policy in our computational experiments, outlined in Section 2.5.

Finally, we can combine and generalize our previous approximations to derive a correspondence to (2.9).

**Theorem 5.** *Consider the value function approximation*

$$\begin{aligned} v_t(i, S) \approx & q_i + (t-1)\mathbb{E}_\eta[q_\eta] + \sum_{j \in S} r_j \\ & + \sum_{\substack{I \subseteq N \\ I \ni i}} \sum_{J \subseteq V} p_{IJ} \mathbb{E}[\min\{|J \cap S|, B(t-1, |I|/n) + 1\}] \\ & + \sum_{\substack{I \subseteq N \\ I \not\ni i}} \sum_{J \subseteq V} p_{IJ} \mathbb{E}[\min\{|J \cap S|, B(t-1, |I|/n)\}], \end{aligned} \quad (2.14)$$

where  $q \in \mathbb{R}_+^N$ ,  $r \in \mathbb{R}_+^V$ ,  $p_{IJ} \in \mathbb{R}_+$  for  $I \subseteq N$  and  $J \subseteq V$ . Restricting the feasible region of  $\mathcal{D}$  with this approximation yields

$$\begin{aligned} \min & \frac{T}{n} \sum_{i \in N} q_i + \sum_{j \in V} r_j + \sum_{I \subseteq N} \sum_{J \subseteq V} p_{IJ} \mathbb{E}[\min\{|J|, B(T, |I|/n)\}] \\ \text{s.t.} & q_i + r_j + \sum_{\substack{I \subseteq N, J \subseteq V \\ I \ni i, J \ni j}} p_{IJ} \geq 1, (i, j) \in E, \\ & q, r, p \geq 0, \end{aligned} \quad (\mathcal{D}_5)$$

the dual of (2.5) with the additional constraints (2.9).

The proof of Theorem 5 is analogous to Theorem 4, but also uses parts of the proof of Theorem 3. Finally, this approximation defines a policy similar to the one given by (2.13).

## 2.5 Computational Study

In this section we outline the experiments we conducted to test the proposed bounds and policies. All of the test instances we constructed have  $T = n = m$ , and consist of the following:

1. A cycle of size 20 ( $n = 10$ ).
2. A cycle of size 200 ( $n = 100$ ).
3. 20 small instances with  $n = 10$ , each one randomly generated by having a possible edge in  $N \times V$  be present independently with a probability of 10%.
4. 20 large dense instances with  $n = 100$ , each one randomly generated by having a possible edge in  $N \times V$  be present independently with a probability of 10%.
5. 20 large sparse instances with  $n = 100$ , each one randomly generated by having a possible edge in  $N \times V$  be present independently with probability of 2.5%.

We use “small” instances with  $n = 10$  because the exact DP recursion (2.1) is still computationally tractable, yet it becomes intractable for even slightly bigger instances. Furthermore, the dimension of the polyhedron  $Q$  grows as  $O(n^2)$ , and all of our inequality classes are separable in polynomial time, except for (2.9). We can solve the LP’s for “large” instances with  $n = 100$  in a few seconds, but as  $n$  grows the dimension of the problem itself becomes the bottleneck. For example, for our large dense instances, the expected number of variables is 1,000; a similarly constructed instance with  $n = 500$  would already have 25,000 variables in expectation.

We tested various bounds on the instances by solving the initial relaxation (2.5) and then adding the inequalities we introduced in Section 2.3. For the policies, we simulated

20,000 realizations of the small instances and 200 realizations of the large instances, and we report the sample average of each policy. To benchmark our results, for the small instances we computed the optimal expected value given by the DP (2.1), and for the larger instances we calculated the sample mean of the maximum expected off-line matching, by computing the maximum cardinality matching of each simulated realization; this yields an upper bound on any policy as it affords the decision maker early access to information. As policy comparisons, we implemented two heuristics: The single-matching policy computes a maximum cardinality matching in  $(N \times V, E)$ , and matches only these edges, ignoring all others; this heuristic has an approximation ratio of  $1 - (1 - 1/n)^T$  (approximately  $1 - 1/e$  when  $T = n$ ) [39]. The two-matching policy is a heuristic modification of the algorithm from [39] that uses a maximum cardinality 2-matching in  $(N \times V, E)$ .

Table 2.1 summarizes the results. For each instance class, in each row we present the geometric mean of each bound or policy’s ratio to the best available benchmark (the DP value for small instances and the expected maximum matching for large ones). We also report the sample standard deviation of the ratios in parenthesis.

Table 2.1: Summary of experiment results for static relaxations and their corresponding policy.

Bound/Policy	20-Cycle	200-Cycle	Small	Large Dense	Large Sparse
(2.5)	1.2681	1.2659	1.3151 (0.081)	1.0018 (0.0011)	1.2167 (0.010)
(2.5) + (2.6)	1.2681	1.2659	1.0886 (0.042)	1.0018 (0.0011)	1.1227 (0.011)
(2.5) + (2.7)	1.1319	1.0980	1.0536 (0.038)	1.0006 (0.0006)	1.0794 (0.009)
(2.5) + (2.8)	1.1606	1.1370	1.0570 (0.031)	1.0013 (0.0009)	1.0961 (0.011)
(2.5) + (2.7) + (2.8)	1.1319	1.0980	1.0536 (0.038)	1.0004 (0.0005)	1.0717 (0.009)
(2.5) + (2.9)	1.1319	1.0980	1.0288 (0.023)	-	-
Exp. Matching	1.0514	1	1.0030 (0.005)	1	1
(2.1)	1	-	1	-	-
Matching	0.8259	0.8025	0.8566 (0.053)	0.6351 (0.0007)	0.7768 (0.028)
2-Matching	0.9859	0.9496	0.9616 (0.037)	0.7500 (0.0023)	0.8793 (0.030)
(2.10)	0.9861	0.9474	0.9883 (0.020)	0.9232 (0.0048)	0.9306 (0.008)
(2.11)	0.9861	0.9474	0.9963 (0.005)	0.9232 (0.0048)	0.9358 (0.008)
(2.12)	0.9861	0.9474	0.9974 (0.005)	0.9471 (0.0091)	0.9560 (0.006)
(2.13)	0.9980	0.9539	0.9732 (0.032)	0.9409 (0.0036)	0.8635 (0.070)

With respect to the bounds, the right-star inequalities (2.7) improve the basic bound

(2.5) more than the left-star ones (2.8), even though the latter are facet-defining. For small instances, the complete subgraph inequalities (2.9) can further cut the gap to under 3%; however, we weren't able to compute this bound in a reasonable time for larger instances because of the significant additional computational burden. For the dense large instances, our bounds are all quite close to the expected maximum matching benchmark, which is unsurprising since in most realizations of these instances there is a perfect or near-perfect matching.

In terms of policies, the best performer overall is the time-dependent ranking policy corresponding to the right-star inequalities and approximation (2.12). However, the non-ranking policy corresponding to the left-star inequalities and approximation (2.13) does perform better on the two cycle instances. In contrast, the single-matching heuristic policy does not perform well, and even the 2-matching heuristic's performance significantly worsens for the large instances; this may indicate the benefit of having more than two choices per impression in larger graphs.

The next set of experiments generalizes the previous cycle instances to  $k$ -regular graphs constructed in the following way: We labeled impressions and ads from 0 to  $n - 1$ , and for each  $i \in \{0, \dots, n - 1\}$  we set  $\Gamma(i) = \{i, i + 1, \dots, i + k - 1\} \pmod{n}$ . We present the results in Table 2.2 for bipartite graphs with  $n = m = 100$  and  $k = 3, 4, 5, 6$ . We compare bounds and policies with respect to the sample mean of the maximum expected off-line matching. As with the previous large instances, we again did not compute the exact optimal solution or the last bound corresponding to (2.5) + (2.9).

In terms of bounds, we again observe that inequality (2.7) improves the basic bound (2.5) more than the other classes of inequalities, including (2.8). Nonetheless, as  $k$  increases, both bounds (right-star and left-star) are much tighter. Also, we observe that (2.6) does not improve the bound from (2.5). This is expected; when  $n = T$ , in  $(\mathcal{D}_2)$  we will have  $p_{ij}^{(2.11)} = 0$  for all edges incident to nodes of degree 2 or greater (all nodes in these instances), since their objective coefficient satisfies  $1 - (1 - 1/n)^n \geq 1 - 1/e$ .

Table 2.2: Ratios for  $k$ -regular bipartite graphs.

Bound/Policy	3-regular	4-regular	5-regular	6-regular
(2.5)	1.1687	1.1202	1.0951	1.0696
(2.5) + (2.6)	1.1687	1.1202	1.0951	1.0696
(2.5) + (2.7)	1.1131	1.1013	1.0886	1.0674
(2.5) + (2.8)	1.1424	1.1157	1.0944	1.0695
(2.5) + (2.7) + (2.8)	1.1131	1.1013	1.0886	1.0674
Exp. Matching	1	1	1	1
Matching	0.7409	0.7102	0.6943	0.6781
2-Matching	0.8630	0.8343	0.8129	0.8030
(2.10)	0.9347	0.9303	0.9295	0.9254
(2.11)	0.9347	0.9303	0.9295	0.9254
(2.12)	0.9347	0.9303	0.9295	0.9254
(2.13)	0.9429	0.9396	0.9426	0.9454

Regarding the policies, (2.10), (2.11) and (2.12) perform exactly in the same way. This is unsurprising, because the graphs' symmetry means none of the policies can differentiate among the ads. On the other hand, the left-star policy (2.13) does better than the rest of the policies, as we observed also in the cycle instances. Once again, the single-matching and 2-matching policies perform much worse than the ranking policies.

## 2.6 Concluding Remarks

In this chapter, we have studied *static* relaxations for the i.i.d. online bipartite matching problem, by deriving several classes of valid inequalities for the polyhedron of attainable probabilities. We have also determined which of these inequalities are facet-defining, and used them to design heuristic policies, many of which turn out to be of ranking type.

Our results motivate a variety of questions for future work. In particular, we mention at the end of Section 2.3 that, even in very small instances, all facets not included in our study exhibit a more complicated structure. For example, they cannot be written as 0-1 inequalities. More polyhedral results are still needed, and they may require the study of sub-structures in the underlying graph or may need to include the timing of policies' choices; our preliminary results in this vein reveal quite complex inequalities. Another

research direction is to study other type of relaxations, as we will see in Chapter 3, we can construct *dynamic* relaxations in which the decision variables depend on the current stage.

A more general question is to apply methods like the ones in this chapter to other online matching and resource allocation problems. One prominent example is the AdWords problem [44, 82], but there are several other models in advertising and revenue management that may benefit from such an approach.

## CHAPTER 3

### DYNAMIC RELAXATIONS FOR ONLINE BIPARTITE MATCHING

#### 3.1 Introduction

In Chapter 2 we considered static relaxations which use as their primary variables the probability that an arriving customer node of some type is ever matched to a fixed resource node. Though this reduces the number of variables to consider, it also means the corresponding relaxations are coarser and looser, as they cannot easily capture the model’s dynamics. Furthermore, with few exceptions, the policies derived from such relaxations are also mostly static in nature; that is, though a decision may depend on the arriving node type and the remaining available resource nodes, it usually does not depend on the decision epoch itself and how close or far it might be from the end of the horizon.

##### 3.1.1 Main Contributions

In this chapter, we present a dynamic polyhedral approach to the i.i.d. variant of the online bipartite matching problem. Specifically, our main contributions are:

1. To explicitly account for the problem’s sequential nature and consider dynamic relaxations. Specifically, these relaxations use as decision variables the probability that a particular match occurs *in a particular stage*. Using these time-indexed probabilities affords several modeling advantages, such as allowing us to include edge weights that vary by time and thus simplify the analysis by capturing all compatibility information in the objective.
2. The primary appeal of dynamic relaxations is the possibility of providing tighter dual bounds for the model. As one of our main results, we establish that our simplest dynamic relaxation is provably at least as tight as the right-star relaxation (2.7);

furthermore, the latter relaxation includes exponentially many inequalities and relies on a separation algorithm, whereas our new relaxation has polynomially many variables and constraints. To further understand our new relaxation, we also perform a polyhedral study, demonstrating that all of its inequalities are facet-defining for the underlying polytope of achievable probabilities. We then extend this polyhedral study and introduce more complex inequalities, all facet-defining as well. Our empirical study verifies the strength of the new relaxation; it improves the previous best gaps by 4% to 5% in absolute terms on average.

3. To show how our new relaxation can be leveraged to construct a dynamic heuristic policy. Although this kind of policy is new in OBM to our knowledge, our policy can be viewed as the OBM analogue to dynamic bid price policies, introduced in [2] for network revenue management. To design the policy, we establish a connection between our relaxation and a value function approximation of the model’s dynamic programming (DP) formulation. Our empirical results also verify the new policy’s quality in comparison to the best empirically performing policy (2.12) in the previous chapter.

In the remainder of the chapter, Section 3.2 presents our dynamic polyhedral approach. Section 3.3 introduces our relaxations and gives our theoretical results, while Section 3.4 outlines our computational study. Section 3.5 concludes and discusses possible future work.

### **3.2 A Dynamic Polyhedral Approach**

In this section, we will show how to adapt the polyhedral approach in Section 2.2 when we consider dynamic weights on each edge of the graph. By considering time-indexed weights  $w_{ij}^t$ , we generalize much of the existing literature and can avoid dealing with specific graph structure. In particular, we may assume that the process occurs in a complete bipartite

graph, i.e. every node type in  $N$  is connected or compatible with every node in  $V$ ; non-existent edges simply get weight zero.

Moreover, we can assume  $m = n = T$  without loss of generality. Indeed, if  $m < T$  we add dummy nodes to  $V$  and assign zero weight to all corresponding edges. Similarly, if  $m > T$  we increase the number of stages and give zero weight to all edges in the new stages. If  $n > m = T$ , we again add dummy nodes and stages. Finally, if  $n < m = T$  we make  $\kappa$  copies of every node type in  $N$  (and the corresponding edges) for the smallest  $\kappa$  with  $\kappa n \geq m$ , then proceed as before. To ease notation, in the remainder of the chapter we write  $n$  for  $m$  and  $T$ , but we use the indices  $i$  for impressions,  $j$  for ads, and  $t$  for stages. We also use the shorthand  $[n] := \{1, \dots, n\}$  and we count stages down from  $n$ , meaning stage  $t$  occurs when  $t$  decision epochs (including the current one) remain in the process

Similarly than (2.1) in Section 2.2, we can give a DP formulation for this OBM model. Denote by  $\eta$  the uniform random variable over  $N$ . Let  $v_t^*(i, S)$  denote the optimal expected value given that  $i \in N$  appears in stage  $t$  when the set of ads  $S \subseteq V$  is available. Then, for all  $t = 1, \dots, n, i \in N$  and  $S \subseteq V$ ,

$$v_t^*(i, S) = \max \begin{cases} \max_{j \in S} \{w_{ij}^t + \mathbb{E}_\eta[v_{t-1}^*(\eta, S \setminus j)]\} \\ \mathbb{E}_\eta[v_{t-1}^*(\eta, S)], \end{cases} \quad (3.1)$$

where  $v_0^*(\cdot, \cdot)$  is identically zero, and the optimal expected value of the model is given by  $\mathbb{E}_\eta[v_n^*(\eta, V)] = \frac{1}{n} \cdot \sum_{i \in N} v_n^*(i, V)$ . Observe that we are considering dynamic weights  $w_{ij}^t$ , therefore we do not need to deal with the structure of the graph, meaning the set of neighbors of  $i$ . The first term in this recursion corresponds to matching  $i$  with one of the remaining ads  $j \in S$ ; the second corresponds to discarding  $i$ . As with any DP, the optimal value function  $v^*$  induces an optimal policy: At any state  $(t, i, S)$ , we choose an action that attains the maximum in (3.1).

As we did in (2.2), we can capture the recursion (3.1) with the linear program

$$\min_{v \geq 0} \mathbb{E}_\eta[v_n(\eta, V)] \quad (3.2a)$$

$$\text{s.t. } v_t(i, S \cup j) - \mathbb{E}_\eta[v_{t-1}(\eta, S)] \geq w_{ij}^t, \quad t \in [n], i \in N, j \in V, S \subseteq V \setminus j \quad (3.2b)$$

$$v_t(i, S) - \mathbb{E}_\eta[v_{t-1}(\eta, S)] \geq 0, \quad t \in [n], i \in N, S \subseteq V. \quad (3.2c)$$

The value function  $v^*$  defined in (3.1) is optimal for (3.2). Moreover, this LP is a strong dual for OBM; any feasible  $v$  has an objective greater than or equal to  $\mathbb{E}_\eta[v_n^*(\eta, V)]$ . The dual of (3.2) is a primal formulation where any feasible solution encodes a feasible policy and its probability of choosing any action from any state in the DP. That formulation is the LP

$$\max_{x, y \geq 0} \sum_{i \in N} \sum_{j \in V} \sum_{t \in [n]} \sum_{S \subseteq V \setminus j} w_{ij}^t x_{i,j}^{t,S} \quad (3.3a)$$

$$\text{s.t. } \sum_{j \in V} x_{i,j}^{n, V \setminus j} + y_i^{n, V} \leq \frac{1}{n}, \quad i \in N, \quad (3.3b)$$

$$\begin{aligned} & \sum_{j \in S} x_{i,j}^{t, S \setminus j} + y_i^{t, S} \cdot \mathbb{1}_{\{t \neq 1\}} - \frac{1}{n} \cdot \mathbb{1}_{\{|S| > t\}} \cdot \sum_{k \in N} y_k^{t+1, S} \\ & - \frac{1}{n} \sum_{k \in N} \sum_{j \in V \setminus S} x_{k,j}^{t+1, S} \leq 0, \quad t \in [n-1], i \in N, \emptyset \neq S \subset V, |S| \geq t, \end{aligned} \quad (3.3c)$$

$$\sum_{j \in V} x_{i,j}^{t, V \setminus j} + y_i^{t, V} \cdot \mathbb{1}_{\{t \neq 1\}} - \frac{1}{n} \sum_{k \in N} y_k^{t+1, V} \leq 0, \quad i \in N, t \in [n-1]. \quad (3.3d)$$

We denote by  $\mathbb{1}_{\mathcal{A}}$  the indicator function for a condition  $\mathcal{A}$ , which takes value one if condition  $\mathcal{A}$  is satisfied, zero otherwise. Decision variable  $x_{i,j}^{t,S}$  represents the probability that the policy chooses to match impression  $i$  to ad  $j$  in state  $(t, i, S \cup j)$ , and  $y_i^{t,S}$  similarly represents a discarding action.

As with its dual, (3.3) has exponentially many variables and constraints, and is therefore difficult to analyze directly. However, we can equivalently consider the probability that a

feasible policy makes a particular match between  $i$  and  $j$  in stage  $t$  without tracking the other remaining ads  $S \subseteq V \setminus j$ ; this corresponds to optimizing over a projection of the feasible region of (3.3),

$$\max \left\{ \sum_{i \in N} \sum_{j \in V} \sum_{t \in [n]} w_{ij}^t z_{ij}^t : \exists (x, y) \geq 0 \text{ satisfying (3.3b)–(3.3d) with } z_{ij}^t = \sum_{S \subseteq V \setminus j} x_{i,j}^{t,S} \right\},$$

where  $z_{ij}^t$  is the probability that impression  $i$  is matched to ad  $j$  in stage  $t$ . Any such  $z$  is a vector of matching probabilities that is *achievable* by at least one feasible policy. Let  $Q_d$  denote this projected polyhedron in the space of  $z_{ij}^t$  variables, and note that  $Q_d$  is full-dimensional in  $\mathbb{R}^{n^3}$ . Optimizing over  $Q_d$  is as difficult as solving the original DP formulation (3.1), but optimizing over any relaxation of  $Q_d$  yields a valid upper bound; this is our main goal in this chapter. We refer to this polytope as *dynamic*, since its variables  $z_{ij}^t$  change according to each stage  $t$ . Observe that  $Q_s$  defined in Section 2.2 is a projection of polyhedron  $Q_d$ .

### 3.3 Projected Dynamic Relaxations

In this section, we introduce various classes of valid inequalities for  $Q_d$  and study their facial dimension. These inequalities always include variables corresponding to complete bipartite subgraphs; therefore, to ease notation we define for any  $I \subseteq N, J \subseteq V$

$$Z_{I,J}^t := \sum_{i \in I} \sum_{j \in J} z_{ij}^t.$$

We begin by presenting a simple inequality class to motivate our approach. For an impression  $i \in N$ , the probability of matching  $i$  in each stage  $t \in [n]$  is at most  $1/n$ ; this corresponds to

$$Z_{i,V}^t \leq 1/n, \quad i \in N, t \in [n]. \quad (3.4)$$

Note that by summing these constraints over all  $t$  for a fixed  $i$ , we obtain (2.5b).

**Proposition 6.** *Constraints (3.4) are facet-defining for the polyhedron of achievable probabilities  $Q_d$ .*

*Proof.* Fix  $i \in N$  and  $t \in [n]$ . We use  $e_{k,j}^\tau \in \mathbb{R}^{n^3}$  to denote the canonical vector, i.e., a vector with a one in the coordinate  $(k, j, \tau)$  and zero elsewhere, indicating that we match impression  $k$  with ad  $j$  in stage  $\tau$ . We construct the following  $n^3$  affinely independent points corresponding to policies that satisfy (3.4) with equality:

- Policy for  $(i, j, t)$  with  $j \in V$ : If  $i$  appears in stage  $t$ , which happens with probability  $1/n$ , we match it with  $j$ . This corresponds to the point  $\frac{1}{n}e_{i,j}^t$ .
- Policy for  $(k, j, \tau)$  with  $j \in V$ ,  $\tau \neq t$ , and  $k \in N$ : If  $k$  appears in stage  $\tau$  (with probability  $1/n$ ), we match it to  $j$ . Then, if  $i$  appears in stage  $t$  with probability  $1/n$ , we match it to some  $\ell \in V$ ,  $\ell \neq j$ , so we have the point  $\frac{1}{n}e_{k,j}^\tau + \frac{1}{n}e_{i,\ell}^t$ .
- Policy for  $(k, j, t)$  with  $j \in V$ , and  $k \neq i$ : If  $k$  appears in stage  $t$  (with probability  $1/n$ ), we match it to  $j$ . On the other hand, if  $i$  appears in stage  $t$  with probability  $1/n$ , we match it to some  $\ell \in V$ ,  $\ell \neq j$ , so we have the point  $\frac{1}{n}e_{k,j}^t + \frac{1}{n}e_{i,\ell}^t$ .

These points are linearly independent, which implies they are affinely independent.  $\square$

We now introduce our general inequality family. Fix a set of ads  $J \subseteq V$  and a family of impression sets  $I_t \subseteq N$ ,  $t \in [n]$ . For any vector  $\alpha \in \mathbb{R}_+^n$ , we have a valid inequality for  $Q_d$  of the form

$$\sum_{t=1}^n \alpha_t Z_{I_t, J}^t \leq R(\alpha, (I_t), J), \quad (3.5)$$

where  $R(\alpha, (I_t), J)$  defines the maximum of the left-hand side over  $Q_d$ . As one example, inequalities (3.4) are a special case of (3.5) where  $J = V$ ,  $I_t = \{i\}$ ,  $\alpha_t = 1$ , and  $I_\tau = \emptyset$ ,  $\alpha_\tau = 0$  for  $\tau \neq t$ .

**Proposition 7.** *For  $\alpha \in \mathbb{R}_+^n$ , set family  $I_t \subseteq N$ ,  $t \in [n]$ , and  $J \subseteq V$ , constraints (3.5) are valid for the polyhedron of achievable probabilities  $Q_d$ , and  $R(\alpha, (I_t), J)$  can be computed in polynomial time via a DP.*

*Proof.* Define variables  $p_t \in \{0, 1\}$  to indicate whether a node from  $I_t$  appears in stage  $t$  or not, and denote by  $d \in \{0, \dots, |J|\}$  the number of remaining nodes from  $J$ . Given this, we can state a DP recursion using the value function  $R(t, d, p_t)$ , the expected value in stage  $t$  when  $d$  nodes from  $J$  are available and  $p_t$  has occurred. For example, if only one stage remains,  $d$  nodes from  $J$  are available, and no element of  $I_1$  appears,  $R(1, d, 0) = 0$  since we cannot match any node in  $I_1$ . Conversely,  $R(1, d, 1) = \alpha_1 \min\{1, d\}$ , since we can match a node and obtain value  $\alpha_1$  as long as at least one element of  $J$  remains.

In general, if  $d$  nodes are available in stage  $t$  and no node from  $I_t$  appears ( $p_t = 0$ ), then the expected value  $R(t, d, 0)$  can be computed recursively by conditioning on terms from stage  $t - 1$ :

$$R(t, d, 0) = \frac{n - |I_t|}{n} R(t - 1, d, 0) + \frac{|I_t|}{n} R(t - 1, d, 1).$$

On the other hand, to compute  $R(t, d, 1)$  we choose the maximum between discarding or matching, with value

$$R(t, d, 1) = \max\{R(t - 1, d, 0), \alpha_t + R(t - 1, d - 1, 0)\}$$

Finally, the value of the right-hand side is

$$R(\alpha, (I_t), J) = \frac{n - |I_n|}{n} R(n, |J|, 0) + \frac{|I_n|}{n} R(n, |J|, 1).$$

The number of states is  $n \times |J| \times 2 = O(n^2)$  and the number of operations to calculate a state's value is constant, so the entire recursion takes  $O(n^2)$  time.  $\square$

In the remainder of this section, we study particular cases of inequalities (3.5). We construct them intuitively using probabilistic arguments, but their right-hand sides can also be calculated directly using the DP from Proposition 7.

As a first example, let  $i \in N$ ,  $j \in V$  and  $t \in [n - 1]$ . Matching  $i$  to  $j$  in stage  $t$  implies the intersection of two independent events. First,  $j$  is not matched in any previous stage

$[t + 1, n]$ , and second,  $i$  appears in stage  $t$ . In terms of probability this means

$$\mathbb{P}(\text{match } i \text{ with } j \text{ in } t) \leq \frac{1}{n}(1 - \mathbb{P}(\text{match } j \text{ in } [t + 1, n])),$$

which is equivalent to

$$\mathbb{P}(\text{match } j \text{ in stages } [t + 1, n]) + n\mathbb{P}(\text{match } i \text{ with } j \text{ in } t) \leq 1.$$

The previous expression is equivalent to

$$\sum_{\tau=t+1}^n Z_{N,j}^{\tau} + nz_{i,j}^t \leq 1 \quad \forall i \in N, j \in V, t \in [n]. \quad (3.6)$$

Inequality family (3.6) corresponds to a particular case of (3.5), with  $|J| = 1$ ,  $I_{\tau} = N$  for  $\tau \in [t + 1, n]$ ,  $|I_t| = 1$ ,  $I_{\tau} = \emptyset$  for  $\tau \leq t - 1$ ,  $\alpha_{\tau} = 1$  for  $\tau \in [t + 1, n]$ ,  $\alpha_t = n$ , and  $\alpha_{\tau} = 0$  for  $\tau \leq t - 1$ . Furthermore, for a fixed  $j \in V$  and  $t = 1$ , by summing the inequalities over all  $i \in N$  we obtain (2.5c).

**Proposition 8.** *Constraints (3.6) are facet-defining for the polyhedron of achievable probabilities  $Q_d$  when  $t \leq n - 1$ .*

*Proof.* Fix  $i \in N$ ,  $j \in V$ ,  $t \in [n - 1]$ . We construct the following  $n^3$  affinely independent points corresponding to policies that satisfy (3.6) with equality:

1. Policy for  $(i, j, t)$ : if  $i$  appears in stage  $t$ , then match it to  $j$  with probability  $1/n$ .

This corresponds to the point  $\frac{1}{n}e_{i,j}^t$ .

2. Policy for  $(k, j, \tau)$  with any  $k \in N$ , and any  $\tau \in [t + 1, n]$ : If  $k$  appears in stage  $\tau$ , match it to  $j$  with probability  $1/n$ , but if  $k$  does not appear and  $i$  appears in stage  $t$ , we match  $i$  to  $j$  with probability  $\frac{1}{n}(1 - \frac{1}{n})$ . This corresponds to the point  $\frac{1}{n}e_{k,j}^{\tau} + \frac{1}{n}(1 - \frac{1}{n})e_{i,j}^t$ . As we chose any  $k$  and any  $\tau$ , we have  $n(n - t)$  points.

So far, we only have  $n(n-t)+1$  points. For the remaining points, we can use modifications of policy 1 above.

- Policy for  $(k, j, \tau)$  with any  $k \in N$  and  $\tau \leq t-1$ : If  $i$  appears in stage  $t$  with probability  $1/n$ , then match it with  $j$ ; if  $i$  does not appear (with probability  $1-1/n$ ), and if  $k$  appears in stage  $\tau$  (with probability  $1/n$ ), then match it with  $j$ . This corresponds to  $\frac{1}{n}e_{i,j}^t + \frac{1}{n}\left(1-\frac{1}{n}\right)e_{k,j}^\tau$ . As we chose any  $k$ , and any  $\tau \leq t-1$ , we have  $n(t-1)$  points.
- Policy for  $(k, \ell, \tau)$  with any  $k \in V$ ,  $\ell \in V$  such that  $\ell \neq j$ , and  $\tau \in [n]$ : if  $i$  appears in stage  $t$  with probability  $1/n$ , then match it with  $j$ ; if  $k$  appears in stage  $\tau$  (with probability  $1/n$ ), then match it with  $\ell$ . This corresponds to  $\frac{1}{n}e_{i,j}^t + \frac{1}{n}e_{k,\ell}^\tau$ . In total, this yields  $n(n-1)n$  points.
- Policy for  $(k, j, t)$  with  $k \in V$  such that  $k \neq i$ : if  $i$  appears in stage  $t$  with probability  $1/n$ , then match it with  $j$ ; if  $k$  appears in stage  $t$  (with probability  $1/n$ ), then match it with  $j$ . This corresponds to  $\frac{1}{n}e_{i,j}^t + \frac{1}{n}e_{k,j}^t$ . In this family, we have  $n-1$  points.

If we order these points in a suitable way, they form the columns of a block matrix

$$A = \begin{pmatrix} A_1 & A_2 \\ 0 & A_3 \end{pmatrix},$$

where  $A_1$  is upper triangular and  $A_3$  is a diagonal matrix.  $A_1$  is formed by the first  $n(n-t)+1$  points from policy 1 and the policies of item 2, while  $A_2$  and  $A_3$  are given by the remaining points. All diagonal entries of  $A_1$  and  $A_3$  are positive, implying that  $A$  has positive determinant. This shows that the points previously described are linearly independent, completing the proof.  $\square$

We next compare the inequalities we have introduced so far to the known results for the lower-dimensional polyhedron  $Q_s$  of achievable probabilities that are not time-indexed,

which we detailed in Section 2.3. Note that  $Q_s$  is a projection of  $Q_d$  obtained by aggregating variables  $z_{ij}^t$  over all stages,  $z_{ij} = \sum_{t \in [n]} z_{ij}^t$ . We already indicated how inequality families (2.5b) and (2.5c) are respectively implied by (3.4) and (3.6). We next discuss the right-star inequalities (2.7).

**Theorem 6.** *Inequalities (3.6) imply the right-star inequalities (2.7).*

*Proof.* Fix  $j \in V$  and  $I \subseteq N$ . First, for  $t = n$  (3.6) is simply  $n z_{ij}^n \leq 1$  (it is also a weakened version of (3.4)), and summing over  $I$  we get  $n \sum_{i \in I} z_{ij}^n \leq |I|$ . For  $t \leq n - 1$ , if we sum over  $i \in I$  in (3.6) we get

$$|I| \sum_{\tau \in [t+1, n]} \sum_{k \in N} z_{k,j}^\tau + n \sum_{i \in I} z_{i,j}^t \leq |I|, \quad \forall t \in [n - 1],$$

and since  $\sum_{i \in I} z_{i,j}^\tau \leq \sum_{k \in N} z_{k,j}^\tau$ , we have

$$|I| \sum_{\tau \in [t+1, n]} \sum_{i \in I} z_{i,j}^\tau + n \sum_{i \in I} z_{i,j}^t \leq |I|, \quad \forall t \in [n - 1].$$

Then, multiply each inequality for  $t \in [n - 1]$  by  $\frac{1}{n} \left(1 - \frac{|I|}{n}\right)^{t-1}$ , and add all of them (including the one for  $t = n$ ); the resulting coefficient for each  $z_{ij}^t$  is

$$\left(1 - \frac{|I|}{n}\right)^{t-1} + \sum_{\tau \leq t-1} \frac{|I|}{n} \left(1 - \frac{|I|}{n}\right)^{\tau-1} = 1.$$

We thus obtain  $\sum_{t \in [n]} \sum_{i \in I} z_{ij}^t = \sum_{i \in I} z_{ij}$  in the left-hand side. In the right-hand side, we get

$$\frac{|I|}{n} \sum_{t=1}^n \left(1 - \frac{|I|}{n}\right)^{t-1} = 1 - \left(1 - \frac{|I|}{n}\right)^n. \quad \square$$

This result shows that inequalities (3.4) and (3.6) yield an upper bound that theoretically dominates the bound given by LP (2.5) with additional inequalities (2.7), the best empirical bound previously known for OBM, see Section 2.5. In terms of dimension, the LP given by (3.4) and (3.6) with non-negativity constraints has  $O(n^3)$  inequalities in  $\mathbb{R}^{n^3}$ , while (2.5)

with (2.7) has exponentially many inequalities in  $\mathbb{R}^{n^2}$ .

### 3.3.1 Policy Design

Theorem 6 establishes that an LP in the space of  $z_{ij}^t$  variables with inequalities (3.4) and (3.6),

$$\max_{z \geq 0} \left\{ \sum_{i \in N} \sum_{j \in V} \sum_{t \in [n]} w_{ij}^t z_{ij}^t : (3.4), (3.6) \right\}, \quad (3.7)$$

is guaranteed to provide a bound at least as good as the state of the art. We can also devise a policy from the LP (3.7), in a similar fashion to dynamic bid policies from network revenue management [2]. Denote by  $\lambda_i^t \geq 0$  and  $\mu_{ij}^t \geq 0$  the dual multipliers corresponding to constraints (3.4) and (3.6) respectively. Along the lines of Section 2.4 and other approximate DP approaches [2], we construct an approximation of the true value function (3.1): Interpret each  $\lambda_i^t$  as the value of having an impression of type  $i$  appear in period  $t$ , and similarly interpret each  $\mu_{ij}^t$  as the value of having impression  $i$  appear in period  $t$  when ad  $j$  is available to match. For state  $(t, i, S)$  this yields the value function approximation

$$v_t(i, S) \approx \lambda_i^t + \sum_{\tau \in [t-1]} \mathbb{E}_\eta[\lambda_\eta^\tau] + \sum_{j \in S} \left( \mu_{ij}^t + \sum_{\tau \in [t-1]} \mathbb{E}_\eta[\mu_{\eta j}^\tau] \right). \quad (3.8)$$

By imposing the constraints from (3.2) on this approximation of the value function, we obtain the dual of (3.7):

$$\begin{aligned} \min_{v \geq 0} \mathbb{E}_\eta[v_n(\eta, V)] & \qquad \min_{\lambda, \mu \geq 0} \sum_{t \in [n]} \left( \mathbb{E}_\eta[\lambda_\eta^t] + \sum_{j \in V} \mathbb{E}_\eta[\mu_{\eta j}^t] \right) \\ \text{s.t. } v_t(i, S \cup j) - \mathbb{E}_\eta[v_{t-1}(\eta, S)] & \geq w_{ij}^t, \quad \xrightarrow{(3.8)} \quad \text{s.t. } \lambda_i^t + \mu_{ij}^t + \sum_{\tau \in [t-1]} \mathbb{E}_\eta[\mu_{\eta j}^\tau] \geq w_{ij}^t. \\ v_t(i, S) - \mathbb{E}_\eta[v_{t-1}(\eta, S)] & \geq 0, \end{aligned}$$

Furthermore, by replacing (3.8) in the DP recursion (3.1) for a state  $(t, i, S)$ , we get the heuristic policy

$$\begin{aligned}
& \operatorname{argmax} \left\{ \max_{j \in S} \{w_{ij}^t + \mathbb{E}_\eta[v_{t-1}(\eta, S \setminus j)]\}, \mathbb{E}_\eta[v_{t-1}(\eta, S)] \right\} \\
\stackrel{(3.8)}{\approx} & \operatorname{argmax} \left\{ \max_{j \in S} \left\{ w_{ij}^t + \sum_{\tau \in [t-1]} \left( \mathbb{E}_\eta[\lambda_\eta^\tau] + \sum_{\ell \in S \setminus j} \mathbb{E}_\eta[\mu_{\eta\ell}^\tau] \right) \right\}, \right. \\
& \left. \sum_{\tau \in [t-1]} \mathbb{E}_\eta[\lambda_\eta^\tau] + \sum_{\ell \in S} \sum_{\tau \in [t-1]} \mathbb{E}_\eta[\mu_{\eta\ell}^\tau] \right\} \\
= & \operatorname{argmax} \left\{ \max_{j \in S} \left\{ w_{ij}^t - \sum_{\tau \in [t-1]} \mathbb{E}_\eta[\mu_{\eta j}^\tau] \right\}, 0 \right\}. \tag{3.9}
\end{aligned}$$

Intuitively, this policy evaluates the net benefit of a potential match of impression  $i$  to ad  $j$  in period  $t$  as the match's weight minus the value we give up by losing ad  $j$  in the subsequent remaining periods. The policy chooses the match with the largest such benefit (if positive), and otherwise discards the impression.

### 3.3.2 Further Polyhedral Study

Inequalities (3.6) correspond to a particular case of (3.5), when the fixed set of ads  $J$  has one element. We can apply a similar idea to a subset of any size; take the next simplest case of (3.5), a set of size two, say  $J = \{j_1, j_2\}$ . Consider also two impressions  $i_1, i_2 \in N$ , where we may have  $i_1 = i_2$ . In terms of probability, the event of matching  $i_2$  with  $j_1$  or  $j_2$  in stage  $t$  implies  $i_2$  must appear in stage  $t$  with probability  $1/n$  and either of two events happens: First, neither  $j_1$  nor  $j_2$  are matched in stages  $[t+2, n]$ , and then  $i_1$  appears in stage  $t+1$  with probability  $1/n$  (and can be matched to one of the ads or not); and second,  $j_1$  or  $j_2$  (but not both) are matched in stages  $[t+2, n]$ , and  $i_1$  is not matched to  $j_1$  nor  $j_2$  in stage  $t+1$  (this includes the case of another impression being matched to one of them). Since

matching  $j_1$  or  $j_2$  in  $t$  are mutually exclusive events, we have the inequality

$$\mathbb{P}(\text{match } i_2 \text{ with } j_1 \text{ or } j_2 \text{ in } t) \leq \frac{1}{n} \left[ \frac{1}{n} (1 - \mathbb{P}(\text{match } j_1 \text{ or } j_2 \text{ in } [t+2, n])) + (1 - \mathbb{P}(\text{match } i_1 \text{ with } j_1 \text{ or } j_2 \text{ in } t+1)) \right].$$

In terms of variables  $z$ , this is equivalent to

$$\sum_{\tau \in [t+2, n]} Z_{N, J}^\tau + n Z_{i_{t+1}, J}^{t+1} + n^2 Z_{i_t, J}^t \leq 1 + n \quad (3.10)$$

$$\forall i_t, i_{t+1} \in N, J \subseteq V, |J| = 2, t \in [n-2].$$

This probabilistic argument can be generalized for any set  $J \subseteq V$  with  $|J| = h \in [n-1]$  and any  $t \leq n-h$ . Let  $(i_1, \dots, i_h)$  be a sequence of nodes in  $N$  allowing repeats; the general constraint corresponds to

$$\begin{aligned} & \mathbb{P}(\text{match } i_h \text{ to some } j \in J \text{ in } t) \\ & \leq \frac{1}{n} \left[ \frac{1}{n^{h-1}} (1 - \mathbb{P}(\text{match } j_1 \text{ or } j_2 \text{ or } \dots \text{ or } j_h \text{ in } [t+h, n])) \right. \\ & \quad + \frac{1}{n^{h-2}} (1 - \mathbb{P}(\text{match } i_1 \text{ to some } j \in J \text{ in } t+h-1)) \\ & \quad + \frac{1}{n^{h-3}} (1 - \mathbb{P}(\text{match } i_2 \text{ to some } j \in J \text{ in } t+h-2)) \\ & \quad \left. + \dots + (1 - \mathbb{P}(\text{match } i_{h-1} \text{ to some } j \in J \text{ in } t+1)) \right]. \end{aligned}$$

Therefore, we can give a general expression for this particular subclass of inequalities (3.5):

$$\sum_{\tau=t+h}^n Z_{N, J}^\tau + \sum_{\tau=t}^{t+h-1} n^{t+h-\tau} Z_{i_\tau, J}^\tau \leq 1 + \sum_{\tau=1}^{h-1} n^\tau, \quad (3.11)$$

$$\forall J \subseteq V, |J| = h \in [n-1], t \in [n-h], i_t, \dots, i_{t+h-1} \in N.$$

**Theorem 7.** *Constraints (3.11) are facet-defining for  $Q_d$ .*

The proof of this theorem can be found in the Appendix A.

So far we have only considered either  $I_\tau = N$  or  $|I_\tau| = 1$  within inequalities (3.5). We next propose a generalization for other sets  $I$ . Consider the case  $J = \{j\}$ , and any subset  $I \subseteq N$ ; suppose we naively apply the same argument behind inequality (3.6). Matching an element of  $I$  with  $j$  in stage  $t$  implies the intersection of two independent events: First,  $j$  is not matched in stages  $[t + 1, n]$ , and second, some element in  $I$  appears in stage  $t$ . In probabilistic terms,

$$\mathbb{P}(\text{match any element in } I \text{ with } j \text{ in } t) \leq \frac{|I|}{n} (1 - \mathbb{P}(\text{match } j \text{ in } [t + 1, n])),$$

which is equivalent to

$$|I| \sum_{\tau=t+1}^n Z_{N,j}^\tau + nZ_{I,j}^t \leq |I|.$$

However, this inequality is made redundant by (3.6), because we can sum over  $i \in I$  for the same fixed  $t$  to get it.

Consider instead  $J = \{j_1, j_2\}$ , any  $I_1 \subseteq N$  with  $|I_1| \geq 2$ , and another impression  $i_2 \in N$ ; we apply the same argument used for (3.10), but substituting  $I_1$  for the single impression  $i_1$ . Matching  $i_2$  with  $j_1$  or  $j_2$  in stage  $t$  implies  $i_2$  appears in stage  $t$  with probability  $1/n$ , and either of two previous events happens: First, neither  $j_1$  nor  $j_2$  are matched in stages  $[t+2, n]$ , and then any element in  $I_1$  appears in stage  $t+1$  with probability  $|I_1|/n$  (and is matched to one of the ads or not); and second, one of  $j_1$  or  $j_2$  is matched in stages  $[t + 2, n]$ , and no element from  $I_1$  is matched to  $j_1$  nor  $j_2$  in  $t + 1$  (this includes the case of another impression being matched to one of them). Since matching  $j_1$  or  $j_2$  in  $t$  are mutually exclusive, we have

$$\begin{aligned} \mathbb{P}(\text{match } i_2 \text{ with } j_1 \text{ or } j_2 \text{ in } t) &\leq \frac{1}{n} \left[ \frac{|I_1|}{n} (1 - \mathbb{P}(\text{match } j_1 \text{ or } j_2 \text{ in } [t + 2, n])) \right. \\ &\quad \left. + (1 - \mathbb{P}(\text{match some } i \in I_1 \text{ with } j_1 \text{ or } j_2 \text{ in } t + 1)) \right], \end{aligned}$$

which is equivalent to

$$|I_1| \sum_{\tau=t+2}^n Z_{N,J}^\tau + nZ_{I_1,J}^{t+1} + n^2 Z_{i_2,J}^t \leq |I_1| + n. \quad (3.12)$$

As with inequalities (3.6), if we attempt to naively extend this argument by considering a larger set  $I_2$  instead of the single impression  $i_2$ , we simply get redundant inequalities. However, we can generalize (3.12) using the same argument for (3.11): For any  $I \subseteq N$  with  $|I| = r \leq n - 1$  and any  $J \subseteq V$  with  $|J| = h \leq n - 1$ , we obtain the inequalities

$$r \sum_{\tau=t+h}^n Z_{N,J}^\tau + nZ_{I,J}^{t+h-1} + \sum_{\tau=t}^{t+h-2} n^{t+h-\tau} Z_{i_\tau,J}^\tau \leq r + \sum_{\tau=1}^{h-1} n^\tau, \\ \forall J \subseteq V, |J| = h \in [n - 1], I \subseteq N, |I| = r \in [n - 1], t \in [n - h], i_t, \dots, i_{t+h-2} \in N. \quad (3.13)$$

**Theorem 8.** *Constraints (3.13) are facet-defining for  $Q_d$ .*

For a proof of this theorem, see the Appendix A.

In inequalities (3.13), we do not consider  $I = N$ . Suppose we apply the same argument for (3.12) in this case; we then obtain

$$n \sum_{\tau=t+h}^n Z_{N,J}^\tau + nZ_{N,J}^{t+h-1} + \sum_{\tau=t}^{t+h-2} n^{t+h-\tau} Z_{i_\tau,J}^\tau \leq n + \sum_{\tau=1}^{h-1} n^\tau.$$

Dividing by  $n$ , we get

$$\sum_{\tau=t+h}^n Z_{N,J}^\tau + Z_{N,J}^{t+h-1} + \sum_{\tau=t}^{t+h-2} n^{t+h-\tau-1} Z_{i_\tau,J}^\tau \leq 1 + \sum_{\tau=1}^{h-1} n^{\tau-1},$$

which is equivalent to

$$\sum_{\tau=t+h}^n Z_{N,J}^\tau + Z_{N,J}^{t+h-1} + nZ_{i_{t+h-2},J}^{t+h-2} + \sum_{\tau=t}^{t+h-3} n^{t+h-\tau-1} Z_{i_\tau,J}^\tau \leq 2 + \sum_{\tau=1}^{h-2} n^\tau.$$

This idea also generates valid inequalities for  $Q_d$ , but we can generalize it even more. Thus far, we consider an arbitrary subset  $I$  in stage  $t + h - 1$ , but in this last inequality the arbitrary subset can “shift” to stage  $t + h - 2$ , so we can actually state a more general valid inequality

$$r \sum_{\tau=t+h-1}^n Z_{N,J}^\tau + nZ_{I,J}^{t+h-2} + \sum_{\tau=t}^{t+h-2} n^{t+h-\tau-1} Z_{i_\tau,J}^\tau \leq 2r + \sum_{\tau=1}^{h-2} n^\tau,$$

$$\forall t \in [n - h], J \subseteq V, l \in [n - 1], I \subseteq N, r \in [n - 1], i_t, \dots, i_{t+h-2} \in N.$$

In this last inequality we only consider  $r \in [n - 1]$ , but as before, we can actually again take  $I = N$  in stage  $t + h - 2$ . After dividing by  $n$ , we get

$$\sum_{\tau=t+h-1}^n Z_{N,J}^\tau + Z_{N,J}^{t+h-2} + \sum_{\tau=t}^{t+h-2} n^{t+h-\tau-2} Z_{i_\tau,J}^\tau \leq 2 + \sum_{\tau=1}^{h-2} n^{\tau-1},$$

equivalent to

$$\sum_{\tau=t+h-2}^n Z_{N,J}^\tau + nZ_{i_{t+h-3},J}^{t+h-3} + \sum_{\tau=t}^{t+h-4} n^{t+h-\tau-2} Z_{i_\tau,J}^\tau \leq 3 + \sum_{\tau=1}^{h-3} n^\tau.$$

This is also a valid inequality for  $Q_d$ , and we can continue doing this process as many as  $h - 2$  times until we get

$$r \sum_{\tau=t+2}^n Z_{N,J}^\tau + nZ_{i_{t+1},J}^{t+1} + n^2 Z_{i_t,J}^t \leq r(h - 1) + n,$$

which is a generalization of (3.10). Finally, if we do this process one more time we get

$$\sum_{\tau=t+1}^n Z_{N,J}^\tau + nZ_{i_t,J}^t \leq h,$$

which is clearly implied by summing over  $j \in J$  in (3.6).

Denote by  $q \in [0, h - 2]$  the number of times we apply this procedure. We now state the

most general family of valid inequalities we have obtained as a specific subclass of (3.5). We subdivide this class using a 4-tuple  $(h, r, t, q)$ , which respectively identifies the size of  $J$ , the size of  $I$ , the stage, and the number of times we apply the previous procedure. So, for any  $J \subseteq V$  with  $|J| = h \in [2, n-1]$ , any  $I \subseteq N$  with  $|I| = r \in [n-1]$ , any  $t \in [n-h]$ , and any  $q \in [0, h-2]$ , we have the following valid inequality

$$r \sum_{\tau=t+h-q}^n Z_{N,J}^\tau + n Z_{I,J}^{t+h-q-1} + \sum_{\tau=t}^{t+h-q-2} n^{t+h-q-\tau} Z_{i_\tau,J}^\tau \leq r(q+1) + \sum_{\tau=1}^{h-q-1} n^{h-q-\tau}. \quad (3.14)$$

We have already proved that the inequalities given by  $(h, r, t, 0)$  are facet-defining; here we give the general result.

**Theorem 9.** *Inequalities (3.14) identified by  $(h, r, t, q)$  are facet-defining for  $Q_d$  when  $h \in [2, n-1]$ ,  $r \in [n-1]$ ,  $t \in [n-h]$  and  $q \in [0, h-2]$ .*

Finally, we show the following complexity result for this general family of facet-defining inequalities.

**Proposition 9.** *It is NP-hard to separate inequalities (3.13), and thus also (3.14).*

*Proof.* Fix  $h = r$  and  $t$ . Suppose we have a solution  $z$  that is zero (or constant) in all values except for stage  $t+h-1$ . In this case, the separation problem for this  $h, r$  and  $t$  is equivalent to

$$\max\{Z_{I,J}^{t+h-1} : I \subseteq N, J \subseteq V, |I| = |J| = h = r\}.$$

This is a weighted version of the maximum balanced biclique problem, which is NP-hard [33]. For  $h \neq r$ , the problem can be transformed to make the two cardinalities equal.  $\square$

### 3.4 Computational Study

Our main experimental goal is testing the effectiveness of our new dynamic relaxations and comparing the new bounds given by these relaxations to several benchmarks. As a sec-

ondary goal, we also study the heuristic policy (3.9) implied by our relaxation and compare it with the best empirically performing policy from Section 2.5.

The best empirical bound previously known for OBM is the LP (2.5) with additional inequalities (2.7), see experimental results in Section 2.5. Our results in the previous section establish that (3.7) is guaranteed to be no worse. So we compare these two bounds to determine how much of an improvement the latter LP (3.7) offers over the former. In addition, we would like to examine if some of the other inequalities we introduce can further improve the bound. However, testing these additional inequality classes involves computational challenges. In particular, the LP’s dimension grows as  $n^3$ , implying a relatively large number of variables even for moderately sized instances. This practically limits both the number of inequalities we consider, and the actual number we can dynamically add to the LP. To this end, we test adding inequalities (3.10) to (3.7); these inequalities are still polynomially many,  $\Theta(n^5)$ , and relatively efficient to separate over. We also considered including the special case of inequalities (3.11) with  $h = n - 1$  and  $t = 1$ , as they are also simple to separate over despite numbering  $\Theta(n^n)$ . However, our preliminary experiments revealed numerical difficulties with these inequalities; the smallest non-zero coefficient is 1, while the largest is  $n^{n-1}$ , and although these numbers (and all of the coefficients and right-hand sides of our inequalities) require  $O(n \log n)$  space in binary representation and are thus of polynomial size, in practical terms these differences in scale make it difficult to even determine whether a particular inequality is violated, and thus to separate over the entire family. We therefore did not include these inequalities in our experiments.

As for lower bounds given by heuristic policies, we introduced a *time-dependent ranking* policy derived from (2.5) with additional inequalities (2.7) in Section 2.4, and results in Section 2.5 establish it as the best performing policy among several from the literature. We use it as a benchmark to test policy (3.9).

Finally, we include as additional benchmarks the optimal value given by the DP recursion (3.1) (for small instances where it can be computed), as well as the max-weight ex-

pected off-line matching, the expected value of the matching we would choose if we could observe the entire sequence of realized impressions before making a decision. This latter benchmark is also an upper bound on the optimal value, as it relaxes non-anticipativity.

All of the instances we tested have  $n = m = T$ , with binary edge weights constant over time,  $w_{ij}^t = w_{ij} \in \{0, 1\}$ . In other words, all the instances are max-cardinality OBM problems with static edges; the static weights are required because the benchmarks we use to compare against do not accommodate weights that vary over time. We generate instances with the following rubrics:

1. 20 *small* instances with  $n = 10$ , each one randomly generated by having a possible edge in  $N \times V$  be present independently with a probability of 25%, so the expected average degree is 2.5.
2. 20 *large, dense* instances with  $n = 100$ , each one randomly generated by having a possible edge in  $N \times V$  be present independently with a probability of 10%, so the expected average degree is 10.
3. 20 *large, sparse* instances with  $n = 100$ , each one randomly generated by having a possible edge in  $N \times V$  be present independently with probability of 2.5%, so the expected average degree is 2.5.
4. A set of *large,  $k$ -regular* graphs with  $n = 100$  and  $k \in \{3, 4, 5, 6\}$ , constructed in the following way: Indexing both impressions and ads from 0 to  $n - 1$ , each impression  $i$  is adjacent to ads  $\{i, i + 1, \dots, i + k - 1\} \bmod k$ . The motivation for this last set of experiments is that the relaxations and policies may behave differently on instances with a high degree of symmetry, as opposed to randomly generated instances.

For any experiment requiring simulation, including computing the expected value of the heuristic policies and the max-weight off-line matching, we used 20,000 simulations and report the sample mean and sample standard deviation.

For small instances, all the bound experiments took a few seconds on average. For the larger instances, we solved the benchmark LP's following the approach in Chapter 2. For the new bounds, we formulated (3.7) but eliminated all variables corresponding to missing edges; this results in models with an average of 25,000, 100,000 and  $10,000 \times k$  variables for sparse, dense and regular instances respectively. The solution times for these LP's were roughly one hour for dense instances, and under a minute for sparse instances, with regular instances varying as  $k$  grows. After solving this LP, we switched to constraint generation for inequalities (3.10); however, after preliminary experiments we did this only for small and large sparse instances, because of protracted solve times with minimal bound improvement in the other cases.

Table 3.1 summarizes the experiment results for all instances except regular ones, which are detailed individually below. For each instance class, in each row we present the geometric mean of each bound or policy's ratio to a fixed benchmark – the DP value for small instances and the max-weight expected off-line matching for large ones. We also report the sample standard deviation of these ratios in parenthesis.

Table 3.1: Summary of experiment results for dynamic relaxations and their corresponding policy.

Bound/Policy	Small	Large Dense	Large Sparse
(2.5) + (2.7)	1.0845 (0.0263)	1.0004 (0.0004)	1.0813 (0.0085)
(3.7)	1.0407 (0.0099)	0.9739 (0.0011)	1.0283 (0.0050)
(3.7) + (3.10)	1.0381 (0.0091)	-	1.0278 (0.0050)
Off-Line Exp. Matching	1.0253 (0.0134)	1	1
(3.1)	1	-	-
Policy (3.9)	0.9974 (0.0017)	0.9561 (0.0025)	0.9627 (0.0066)
Policy (2.12)	0.9901 (0.0080)	0.9539 (0.0025)	0.9529 (0.0073)

We know from our results in the previous section that the bound given by (3.7) is guaranteed to outperform the bound given by (2.5) with (2.7). However, our results show that the improvement is significant, with the new bound cutting the gap by about 4% on average for small instances and approximately 3% to 5% for large ones. Furthermore, the improve-

ment is consistent across all the tested instances; in particular, the two bounds never match.

The results for large, dense instances are particularly noteworthy; the new bound from (3.7) also beats the max-weight expected off-line matching, not only on average but in every instance. Our intuition for this result is the following. In dense instances, there is likely a perfect or near-perfect matching in every realization, and thus the off-line matching will be very close to  $n$  in expectation. Of course, even in a dense instance it may be that no online policy can guarantee a perfect or near-perfect matching, and explicitly accounting for temporal aspects of the problem, particularly as inequalities (3.6) do, captures this phenomenon and tightens the bound, unlike the off-line matching or the more static approach of the benchmark LP.

Interestingly, our results also reveal that the bound from (3.7) is not improved much with the addition of inequalities (3.10), especially considering the significant additional computing time. In light of these results, we also performed experiments to test the bound given by (3.4) and (3.10) only (without inequalities (3.6)). However, the resulting bounds were much looser, confirming that inequalities (3.6) are crucial to providing a tight bound.

In terms of policies, our new heuristic (3.9) is consistently better than the time-dependent ranking policy, the best performing policy studied in Section 2.5. This improvement occurs in almost every tested instance, though the magnitude of the improvement varies. The new policy is near-optimal for small instances, and cuts the gap for large instances, by about 0.7% to 1% on average in absolute terms. This improvement in policy quality mirrors results in other areas, such as revenue management, where heuristic policies derived from dynamic relaxations also outperform policies stemming from “static” LP’s; see e.g. [2, 114].

The results for regular graphs are in Table 3.2, shown here in absolute terms since we are not averaging multiple experiments. We observe similar improvements in terms of upper bounds, where our new bound significantly cuts the gap, by around 7%. On the other hand, we observe no improvement on the policy side. Intuitively, this last result is unsurprising,

since both heuristic policies depend on dual multipliers of LP’s that are symmetric for regular instances, in the sense that they both have dual optimal solutions in which every value at some stage is equal. Both policies are thus choosing a match uniformly at random.

Table 3.2: Experiment results for regular graphs.

Instance	(2.5) + (2.7)	(3.7)	Exp. Matching	(3.9)	Policy (2.12)
3-regular	95.2447	87.9224	85.5680	79.8960	79.8960
4-regular	98.3130	90.9901	89.2247	82.8647	82.8647
5-regular	99.4079	92.8303	91.5918	85.1153	85.1153
6-regular	99.7945	94.0548	93.2837	86.9821	86.9821

### 3.5 Concluding Remarks

This chapter proposes dynamic relaxations for the i.i.d. OBM problem and studies them from a polyhedral point of view. While several past results have used different LP relaxations, ours is the first to explicitly consider the time dimension. Among various benefits of the approach, this allows for the model to accommodate time-varying edge weights, and also allows us to elide the instance’s structure in the analysis, by capturing all of this information in the problem’s objective. Our study centers on the polyhedron of time-indexed achievable probabilities  $Q_d$ , and includes a large class of facet-defining inequalities for this polytope based on choosing complete bipartite subgraphs. Furthermore, our experiments confirm that the time-indexed approach offers significant benefits; the bound given by the simplest members of our proposed inequality family already significantly outperforms the best empirical bounds given by static LP’s, and a heuristic policy derived from this new bound also significantly outperforms the best policy based on a static relaxation.

Our results motivate a variety of questions for future work. For example, we would like to understand the structure of valid inequalities that are not based on complete bipartite subgraphs, to potentially further improve the dual bound. Using Fourier-Motzkin elimination and the software PORTA, we have derived the full description of  $Q_d$  for small cases,

such as  $n = m = T = 3$ . We observed many different inequalities, including some that are somewhat similar to our general family (3.5), so there may be a more general class to propose that still lends itself to analysis similar to ours.

Much of the literature on OBM studies the worst-case performance of heuristic policies based on relaxations. Although that was not our goal in this chapter, the positive empirical results we observed when implementing our new heuristic policy suggest a similar analysis for that policy, especially since it appears to differ in structural terms from many OBM heuristics. More generally, an interesting question is whether a polyhedral analysis similar to ours can be applied to derive new bounds and policies in related online matching and resource allocation contexts.

## CHAPTER 4

### NEW ADVANCES IN ROBUST SUBMODULAR MAXIMIZATION

#### 4.1 Introduction

Optimizing a single submodular function offers an attractive model for several scenarios such as sensor location. However, there are a few key shortcomings which motivated *robust submodular optimization* (see [71]) in the cardinality case, so as to optimize against several functions *simultaneously*:

1. The sensors are typically used to measure various parameters at the same time. Observations for these parameters need to be modeled via different submodular functions.
2. Many of the phenomena being observed are non-stationary and highly variable in certain locations. To obtain a good solution, a common approach is to use different submodular functions to model different spatial regions.
3. The submodular functions are typically defined using data obtained from observations, and imprecise information can lead to unstable optimization problems. Thus, there is a desire to compute *solutions that are robust* to perturbations of the submodular functions.

Our main contribution in this chapter is the development of new algorithms with provable guarantees for robust submodular optimization under a large class of *combinatorial constraints*. These include partition constraints, where local cardinality constraints are placed on disjoint parts of the ground set. More generally, we consider matroid and knapsack constraints. We provide bi-criteria approximations that trade-off the approximation factor with the “size” of the solution, measured by the number  $\ell$  of feasible sets  $\{S_i\}_{i \in [\ell]}$

whose union constitutes the final solution  $S$ . While this might be nonintuitive at first, it turns out that the union of feasible sets corresponds to an appropriate relaxation of the single cardinality constraint. Some special cases of interest are:

1. *Partition constraints.* Given a partition of the candidate sensor locations, the feasible sets correspond to subsets that satisfy a cardinality constraint on each part of the partition. The union of feasible sets here corresponds to relaxing the cardinality constraints separately for each part. This results in a stronger guarantee than relaxing the constraint globally as would be the case in the single cardinality constraint case.
2. *Gammoid.* Given a directed graph and a subset of nodes  $T$ , the feasible sets correspond to subsets  $S$  that can reach  $T$  via disjoint paths in the graph. Gammoids appear in flow based models, for example in reliable routing. The union of feasible sets now corresponds to sets  $S$  that can reach  $T$  via paths such that each vertex appears in few paths.

We consider both offline and online versions of the problem, where the data is either known a-priori or is revealed over time, respectively. We give a simple and efficient greedy-like algorithm for the offline version of the problem. The analysis relies on new insights on the performance of the classical greedy algorithm for submodular maximization, when extended to produce a solution comprising of a union of multiple feasible sets. For the online case, we introduce new technical ingredients that might be broadly applicable in online robust optimization. Our work significantly expands on previous works on robust submodular optimization that focused on a single cardinality constraint [71].

#### 4.1.1 Problem Formulation

In this chapter, we study offline and online variations of *robust submodular maximization under structured combinatorial constraints*. While our results hold for more general constraints, we focus our attention first on *matroid* constraints that generalize the partition as

well as the gammoid structural constraints mentioned above. We discuss extensions to other class of constraints in Section 4.4.

Recall the definitions of monotone submodular functions and matroids given in Section 1.3. We consider the robust variation of submodular optimization. That is, for a matroid  $\mathcal{M} = (V, \mathcal{I})$ , and a given collection of  $k$  monotone submodular functions  $f_i : 2^V \rightarrow \mathbb{R}_+$  for  $i \in [k]$ , our goal is to select a set  $S$  that maximizes  $\min_{i \in [k]} f_i(S)$ . We define a  $(1 - \epsilon)$ -approximately optimal solution  $S$  as

$$\min_{i \in [k]} f_i(S) \geq (1 - \epsilon) \max_{S \in \mathcal{I}} \min_{i \in [k]} f_i(S). \quad (4.1)$$

We also consider the online variation of the above optimization problem in presence of an adversary. In this setting, we are given a fixed matroid  $\mathcal{M} = (V, \mathcal{I})$ . At each time step  $t \in [T]$ , we choose a set  $S^t$ . An adversary then selects a collection of  $k$  monotone submodular functions  $\{f_i^t\}_{i \in [k]} : 2^V \rightarrow [0, 1]$ . We receive a reward of  $\min_{i \in [k]} \mathbb{E}[f_i^t(S^t)]$ , where the expectation is taken over any randomness in choosing  $S^t$ . We can then use the knowledge of the adversary's actions, i.e., oracle access to  $\{f_i^t\}_{i \in [k]}$ , in our future decisions. We consider non-adaptive adversaries whose choices  $\{f_i^t\}_{i \in [k]}$  are independent of  $S^\tau$  for  $\tau < t$ . In other words, an adversarial sequence of functions  $\{f_i^1\}_{i \in [k]}, \dots, \{f_i^T\}_{i \in [k]}$  is chosen upfront without being revealed to the optimization algorithm. Our goal is to design an algorithm that maximizes the total payoff  $\sum_{t \in [T]} \min_{i \in [k]} \mathbb{E}[f_i^t(S^t)]$ . Thus, we would like to obtain a cumulative reward that competes with that of the fixed set  $S \in \mathcal{I}$  we should have played had we known all the functions  $f_i^t$  in advance, i.e., compete with  $\max_{S \in \mathcal{I}} \sum_{t \in [T]} \min_{i \in [k]} f_i^t(S)$ . As in the offline optimization problem, we also consider competing with  $(1 - \epsilon)$  fraction of the above benchmark. In this case,  $\mathbf{Regret}_{1-\epsilon}(T)$  denotes how far we are from this goal. That is,

$$\mathbf{Regret}_{1-\epsilon}(T) = (1 - \epsilon) \cdot \max_{S \in \mathcal{I}} \sum_{t \in [T]} \min_{i \in [k]} f_i^t(S) - \sum_{t \in [T]} \min_{i \in [k]} \mathbb{E}[f_i^t(S^t)]. \quad (4.2)$$

We desire algorithms whose  $(1 - \epsilon)$ -regret is sublinear in  $T$ . That is, we get arbitrarily close to a  $(1 - \epsilon)$  fraction of the benchmark as  $T \rightarrow \infty$ .

The offline (Equation 4.1) or online (Equation 4.2) variations of robust monotone submodular functions, are known to be NP-hard to approximate to any polynomial factor when the algorithm’s choices are restricted to the family of independent sets  $\mathcal{I}$  [71]. Therefore, to obtain any reasonable approximation guarantee we need to relax the algorithm’s constraint set. Such an approximation approach is called a *bi-criteria* approximation scheme in which the algorithm outputs a set with a *nearly optimal objective value*, while ensuring that the set used is the *union of only a few independent sets in  $\mathcal{I}$* . More formally, to get a  $(1 - \epsilon)$ -approximate solutions, we may use a set  $S$  where  $S = S_1 \cup \dots \cup S_\ell$  such that  $S_1, \dots, S_\ell \in \mathcal{I}$  and  $\ell$  is a function of  $\frac{1}{\epsilon}$  and other parameters. This “solution” may initially seem counterintuitive for general matroids, however for many cases of interest, this is just a generalization of what is done when relaxing a single cardinality constraint. To exemplify this, consider partition constraints: here we are given a partition  $\{P_1, \dots, P_q\}$  of the ground set and the goal is to pick a subset that includes at most  $b_j$  elements from part  $P_j$  for each  $j$ . Then, the union of  $\ell$  feasible sets have at most  $\ell \cdot b_j$  elements in each part. Since the output set  $S$  is possibly infeasible, we define the *violation ratio*  $\nu$  as the minimum number of feasible sets whose union is  $S$ . In our example, this is equivalent to  $\nu = \max_{j \in [q]} \lceil |S \cap P_j| / b_j \rceil$ .

#### 4.1.2 Main Contributions

We present (nearly tight) bi-criteria approximation algorithms for the offline and online variations of robust monotone submodular optimization under matroid constraints. Throughout the paper, we assume that the matroid is accessible via an independence oracle and the submodular functions are accessible via a value oracle. Moreover, we use  $\log_a$  to denote logarithm with base  $a$  (when the subscript is not explicit we assume is base 2, i.e.,  $\log := \log_2$ ) and  $\ln$  to denote the natural logarithm.

Consider a polynomial time  $(1 - \beta)$ -approximation algorithm  $\mathcal{A}$  for maximizing a single non-negative monotone submodular function over a matroid constraint. A common example for  $\mathcal{A}$  is the standard greedy algorithm with  $\beta = 1/2$ . We denote by  $\text{time}(\mathcal{A})$  the running time of  $\mathcal{A}$ . Therefore, for the offline setting of the problem we obtain the following general result:

**Theorem 10.** *For the offline robust submodular optimization problem (4.1), for any  $0 < \epsilon < 1$ , there is a polynomial time algorithm that runs in*

$$O\left(\text{time}(\mathcal{A}) \log_{1/\beta}\left(\frac{k}{\epsilon}\right) \log(n) \min\left\{\frac{nk}{\epsilon}, \log_{1+\epsilon}(\max_{e,j} f_j(e))\right\}\right)$$

*time and returns a set  $S^{\text{ALG}}$ , such that*

$$\min_{i \in [k]} f_i(S^{\text{ALG}}) \geq (1 - \epsilon) \cdot \max_{S \in \mathcal{I}} \min_{j \in [k]} f_j(S),$$

*where  $S^{\text{ALG}} = S_1 \cup \dots \cup S_\ell$  with  $\ell = O(\log_{1/\beta} \frac{k}{\epsilon})$ , and  $S_1, \dots, S_\ell \in \mathcal{I}$ .*

The algorithm that achieves this result is an extended version of the algorithm  $\mathcal{A}$ . It reuses  $\mathcal{A}$  in an iterative scheme, so that it generates a *small family* of independent sets whose union achieves the  $(1 - \epsilon)$ -guarantee. The argument is reminiscent of a well-known fact for submodular function maximization under cardinality constraints: letting the standard greedy run longer results in better approximations at the expense of violating the cardinality constraint. Our extended algorithm version of  $\mathcal{A}$  works in a similar spirit, however it iteratively produces independent sets in the matroid. In particular when we consider the standard greedy algorithm as  $\mathcal{A}$ , we get the following corollary:

**Corollary 3.** *For the offline robust submodular optimization problem (4.1), for any  $0 < \epsilon < 1$ , there is a polynomial time algorithm that runs in*

$$O\left(nr \log\left(\frac{k}{\epsilon}\right) \log(n) \min\left\{\frac{nk}{\epsilon}, \log_{1+\epsilon}(\max_{e,j} f_j(e))\right\}\right)$$

time and returns a set  $S^{\text{ALG}}$ , such that

$$\min_{i \in [k]} f_i(S^{\text{ALG}}) \geq (1 - \epsilon) \cdot \max_{S \in \mathcal{I}} \min_{j \in [k]} f_j(S),$$

where  $S^{\text{ALG}} = S_1 \cup \dots \cup S_\ell$  with  $\ell = O(\log_2 \frac{k}{\epsilon})$ , and  $S_1, \dots, S_\ell \in \mathcal{I}$ .

We present the main results and the corresponding proofs in Section 4.2. It is well-known that due to the number of function calls the standard greedy algorithm is computationally inefficient when the ground set is large. In order to reduce the number of queries of our bi-criteria algorithm, we propose an extended version of the threshold greedy algorithm [5] in Section 4.2.1. Moreover, to support our theoretical guarantees we provide an exhaustive computational study in Section 4.2.2. We observe that the main computational bottleneck in the bi-criteria algorithm is to *certify* near-optimality of the obtained solution. To solve this, we present significant implementation improvements such as lazy evaluations and an early stopping criterion, which empirically show how the computational cost can be drastically improved.

Additionally, for the offline problem we propose a second, randomized algorithm relying on continuous extensions of submodular functions that achieves tight bounds in line with the hardness result in [71] (see Section 4.2.3). This algorithm will form the basis of the online algorithm that we present later in Section 4.3. Finally, one might hope that similar results for the offline model can be obtained even when functions are non-monotone (but still submodular). As we show in Section 4.4.1 this is not possible.

A natural question is whether our algorithm can be carried over into the online setting, where functions are revealed over time. For the online setting, we present the first results for robust submodular optimization.

**Theorem 11.** *For the online robust submodular optimization problem, for any  $0 < \epsilon < 1$ , there is a randomized polynomial time algorithm that returns a set  $S^t$  for each stage  $t \in [T]$ ,*

we get

$$\sum_{t \in [T]} \min_{i \in [k]} \mathbb{E} [f_i^t(S^t)] \geq (1 - \epsilon) \cdot \max_{S \in \mathcal{I}} \sum_{t \in [T]} \min_{i \in [k]} f_i^t(S) - O \left( n^{\frac{5}{4}} \sqrt{T} \ln \frac{1}{\epsilon} \right),$$

where  $S^t = S_1^t \cup \dots \cup S_\ell^t$  with  $\ell = O(\ln \frac{1}{\epsilon})$ , and  $S_1^t, \dots, S_\ell^t \in \mathcal{I}$ .

We remark that the guarantee of Theorem 11 holds with respect to the minimum of  $\mathbb{E}[f_i^t(S^t)]$ , as opposed to the guarantee of Theorem 10 that directly bounds the minimum of  $f_i(S)$ . Therefore, the solution for the online algorithm is a union of only  $O(\ln \frac{1}{\epsilon})$  independent sets, in contrast to the offline solution which is the union of  $O(\log_2 \frac{k}{\epsilon})$  independent sets.

The main challenge in the online algorithm is to deal with non-convexity and non-smoothness due to submodularity exacerbated by the robustness criteria. Our approach to coping with the robustness criteria is to use the *soft-min* function  $-\frac{1}{\alpha} \ln \sum_{i \in [k]} e^{-\alpha g_i}$ , defined for a collection of smooth functions  $\{g_i\}_{i \in [k]}$  and a suitable parameter  $\alpha > 0$ . While the choice of the specific soft-min function is seemingly arbitrary, one feature is crucial for us: its gradient is a convex combination of the gradients of the  $g_i$ 's. Using this observation, we use parallel instances of the Follow-the-Perturbed-Leader (FPL) algorithm, presented by Kalai and Vempala [58], one for each discretization step in the continuous greedy algorithm. We believe that the algorithm might be of independent interest to perform online learning over a minimum of many functions, a common feature in robust optimization. The main result and a summary of its proof appears in Section 4.3.

Our main results naturally extend to other types of combinatorial constraints, such as knapsack constraints or multiple matroids. We describe these extensions in Section 4.4.

#### 4.1.3 Related Work

Robust submodular maximization generalizes submodular function maximization under a matroid constraint for which a  $(1 - \frac{1}{e})$ -approximation is known [23] and is optimal. The

problem has been studied for constant  $k$  by [26] who give a  $(1 - \frac{1}{e} - \epsilon)$ -approximation algorithm with running time  $O\left(n^{\frac{k}{\epsilon}}\right)$ . Closely related to our problem is the submodular cover problem where we are given a submodular function  $f$ , a target  $b \in \mathbb{R}_+$ , and the goal is to find a set  $S$  of minimum cardinality such that  $f(S) \geq b$ . A simple reduction shows that robust submodular maximization under a cardinality constraint reduces to the submodular cover problem [71]. Wolsey [113] showed that the greedy algorithm gives an  $O(\ln \frac{n}{\epsilon})$ -approximation, where the output set  $S$  satisfies  $f(S) \geq (1 - \epsilon)b$ . Krause et al. [71] use this approximation to build a bi-criteria algorithm which achieves tight bounds. However, this approach falls short of achieving a tight bi-criteria approximation when the problem is defined over a matroid. Powers et al. [91] considers the same robust problem with matroid constraints. However, they take a different approach by presenting a bi-criteria algorithm that outputs a feasible set that is good only for a fraction of the  $k$  monotone submodular functions. A deletion-robust submodular optimization model is presented in [71], which is later studied in [89, 14, 63]. Influence maximization [64] in a network has been a successful application of submodular maximization and recently, He and Kempe [52] and Chen et al. [28] study the robust influence maximization problem. Robust optimization for non-convex objectives (including submodular functions) has been also considered in [27], however with weaker guarantees than ours due to the extended generality. Specifically, their algorithm outputs  $\frac{r \log k}{\epsilon^2 \text{OPT}}$  feasible sets whose union achieves a factor of  $(1 - 1/e - \epsilon)$ . Finally, Wilder [111] studies a similar problem in which the set of feasible solutions is the set of all distributions over independent sets of a matroid. In particular, for our setting Wilder [111] gives an algorithm that outputs  $O(\frac{\log k}{\epsilon^3})$  feasible sets whose union obtains  $(1 - 1/e)^2$  fraction of the optimal solution. Our results are stronger than the ones obtained in [27] and [111], since we provide the same guarantees using the union of fewer feasible sets. Other variants of the robust submodular maximization problem are studied in [85, 100].

In this chapter, we also study efficient algorithms for the offline version of the robust

submodular maximization problem. In real-world applications the standard greedy algorithm presented in [87] is inefficient, since requires  $nr$  function calls, where  $r$  is the rank of the matroid. This could be computationally expensive for large data-sets. After that, there has been significant progress on reducing the number of evaluations, see e.g. [83, 5, 84] for the vanilla version of submodular maximization (see Section 1.3 for more details).

There has been some prior work on online submodular function maximization that we briefly review here. Streeter and Golovin [101] study the *budgeted maximum submodular coverage* problem and consider several feedback cases (denote  $B$  a integral bound for the budget): in the full information case, a  $(1 - \frac{1}{e})$ -expected regret of  $O(\sqrt{BT \ln n})$  is achieved, but the algorithm uses  $B$  experts which may be very large. In a follow-up work, Golovin et al. [46] study the online submodular maximization problem under partition constraints, and then they generalize it to general matroid constraints. For the latter one, the authors present an online version of the continuous greedy algorithm, which relies on the Follow-the-Perturbed-Leader algorithm of Kalai and Vempala [58] and obtain a  $(1 - \frac{1}{e})$ -expected regret of  $O(\sqrt{T})$ . Similar to this approach, our bi-criteria online algorithm will also use the Follow-the-Perturbed-Leader algorithm as a subroutine.

## 4.2 The Offline Robust Submodular Maximization Problem

In this section, we consider offline robust optimization (Equation 4.1) under matroid constraints. We present a procedure that achieves a (nearly) tight bi-criteria approximation for the problem of interest and prove Theorem 10. We study efficient algorithms for the problem and provide an extensive computational study. We finally present a continuous offline algorithm that achieves the same (up to a constant) bi-criteria guarantees.

### 4.2.1 Offline Bi-criteria Algorithm and Analysis

Consider a single non-negative monotone submodular function  $g : 2^V \rightarrow \mathbb{R}_+$ , a matroid  $\mathcal{M} = (V, \mathcal{I})$ , and a polynomial time  $(1 - \beta)$ -approximation algorithm  $\mathcal{A}$  for the vanilla

submodular maximization problem under a single matroid constraint (1.2). Without loss of generality assume that  $g(\emptyset) = 0$ . Formally, algorithm  $\mathcal{A}$  receives as input a monotone submodular function and a matroid, and outputs a feasible set  $S^{\mathcal{A}} \in \mathcal{I}$  such that

$$g(S^{\mathcal{A}}) \geq (1 - \beta) \cdot \max_{S \in \mathcal{I}} g(S)$$

If  $g(\emptyset) \neq 0$ , then we define a new function  $g' : 2^V \rightarrow \mathbb{R}_+$  as  $g'(S) := g(S) - g(\emptyset)$ , which remains being monotone and submodular. However, the approximation guarantee carries over an extra term, specifically we have

$$g(S^{\mathcal{A}}) - g(\emptyset) \geq (1 - \beta) \cdot \max_{S \in \mathcal{I}} \{g(S) - g(\emptyset)\}.$$

We define an *extended version* of  $\mathcal{A}$  that runs iteratively  $\ell \geq 1$  times the algorithm  $\mathcal{A}$ , see Algorithm 6.

---

**Algorithm 6** Extended Algorithm  $\mathcal{A}$

---

**Require:**  $\ell \geq 1$ , monotone submodular function  $g : 2^V \rightarrow \mathbb{R}_+$ , Matroid  $\mathcal{M} = (V, \mathcal{I})$ .

**Ensure:** sets  $S_1, \dots, S_\ell \in \mathcal{I}$ .

- 1: **for**  $\tau = 1, \dots, \ell$  **do**
  - 2:     Define  $\tilde{g}(S) = g(\cup_{j=1}^{\tau-1} S_j \cup S)$ .
  - 3:      $S_\tau \leftarrow \mathcal{A}(\tilde{g}, \mathcal{M})$
- 

Observe that function  $\tilde{g}$  defined in Algorithm 6 remains being monotone and submodular. We recover algorithm  $\mathcal{A}$  when we consider  $\ell = 1$  in Algorithm 6. We are ready to prove the following guarantee for Algorithm 6.

**Theorem 12.** *For any  $\ell \geq 1$  and monotone submodular function  $g : 2^V \rightarrow \mathbb{R}_+$  with  $g(\emptyset) = 0$ , the extended Algorithm 6 returns sets  $S_1, \dots, S_\ell$  such that*

$$g(\cup_{\tau=1}^{\ell} S_\tau) \geq (1 - \beta^\ell) \max_{S \in \mathcal{I}} g(S).$$

*Proof.* From the first iteration of Algorithm 6 and using the guarantees of  $\mathcal{A}$  we conclude

that  $g(S_1) - g(\emptyset) \geq (1 - \beta) \cdot \max_{S \in \mathcal{I}} \{g(S) - g(\emptyset)\}$ . We use the above statement to prove our theorem by induction. For  $\tau = 1$ , the claim follows directly. Consider any  $\ell \geq 2$ . Observe that the algorithm in iteration  $\tau = \ell$ , is exactly the greedy algorithm run on submodular function  $\tilde{g} : 2^V \rightarrow \mathbb{R}_+$  where  $\tilde{g}(S) := g(S \cup \cup_{\tau=1}^{\ell-1} S_\tau)$ . This procedure returns  $S_\ell$  such that  $\tilde{g}(S_\ell) - \tilde{g}(\emptyset) \geq (1 - \beta) \cdot \max_{S \in \mathcal{I}} \{\tilde{g}(S) - \tilde{g}(\emptyset)\}$ , which implies that

$$g(\cup_{\tau=1}^{\ell} S_\tau) - g(\cup_{\tau=1}^{\ell-1} S_\tau) \geq (1 - \beta) \cdot \max_{S \in \mathcal{I}} \{g(S) - g(\cup_{\tau=1}^{\ell-1} S_\tau)\}.$$

By induction we know  $g(\cup_{\tau=1}^{\ell-1} S_\tau) \geq (1 - \beta^{\ell-1}) \cdot \max_{S \in \mathcal{I}} g(S)$ . Thus, we obtain

$$g(\cup_{\tau=1}^{\ell} S_\tau) \geq (1 - \beta) \cdot \max_{S \in \mathcal{I}} g(S) + \beta \cdot g(\cup_{\tau=1}^{\ell-1} S_\tau) \geq (1 - \beta^\ell) \cdot \max_{S \in \mathcal{I}} g(S).$$

□

We now apply Theorem 12 for the robust submodular problem, in which we are given monotone submodular functions  $f_i : 2^V \rightarrow \mathbb{R}_+$  for  $i \in [k]$ . First, given parameter  $\epsilon > 0$ , we obtain an estimate  $\gamma$  on the value of the optimal solution  $\text{OPT} := \max_{S \in \mathcal{I}} \min_{i \in [k]} f_i(S)$  via a binary search with a relative error of  $1 - \frac{\epsilon}{2}$ , i.e.,  $(1 - \frac{\epsilon}{2}) \text{OPT} \leq \gamma \leq \text{OPT}$ . As in [71], let  $g : 2^V \rightarrow \mathbb{R}_+$  be defined for any  $S \subseteq V$  as follows

$$g(S) := \frac{1}{k} \sum_{i \in [k]} \min\{f_i(S), \gamma\}. \quad (4.3)$$

Observe that  $\max_{S \in \mathcal{I}} g(S) = \gamma$  whenever  $\gamma \leq \text{OPT}$ . Moreover, note that  $g$  is also a monotone submodular function.

*Proof of Theorem 10.* Consider the family of monotone submodular functions  $\{f_i\}_{i \in [k]}$  and define  $g$  as in equation (4.3) using parameter  $\gamma$  with relative error of  $1 - \frac{\epsilon}{2}$ . If we run Algorithm 6 on  $g$  with  $\ell \geq \lceil \log_{1/\beta} \frac{2k}{\epsilon} \rceil$ , we get a set  $S^{\text{ALG}} = S_1 \cup \dots \cup S_\ell$ , where  $S_j \in \mathcal{I}$

for all  $j \in [\ell]$ . Moreover, Theorem 12 implies that

$$g(S^{\text{ALG}}) \geq (1 - \beta^\ell) \max_{S \in \mathcal{I}} g(S) \geq \left(1 - \frac{\epsilon}{2k}\right) \gamma.$$

Now, we will prove that  $f_i(S^{\text{ALG}}) \geq (1 - \frac{\epsilon}{2}) \gamma$ , for all  $i \in [k]$ . Assume by contradiction that there exists an index  $i^* \in [k]$  such that  $f_{i^*}(S^{\text{ALG}}) < (1 - \frac{\epsilon}{2}) \gamma$ . Since, we know that  $\min\{f_i(S^{\text{ALG}}), \gamma\} \leq \gamma$  for all  $i \in [k]$ , then

$$\begin{aligned} g(S^{\text{ALG}}) &\leq \frac{1}{k} f_{i^*}(S^{\text{ALG}}) + \frac{k-1}{k} \gamma \\ &< \frac{1 - \epsilon/2}{k} \gamma + \frac{k-1}{k} \gamma = \left(1 - \frac{\epsilon}{2k}\right) \gamma, \end{aligned}$$

contradicting  $g(S^{\text{ALG}}) \geq (1 - \frac{\epsilon}{2k}) \gamma$ . Therefore, we obtain  $f_i(S^{\text{ALG}}) \geq (1 - \frac{\epsilon}{2}) \gamma \geq (1 - \epsilon) \text{OPT}$ , for all  $i \in [k]$  as claimed.  $\square$

To prove Corollary 3 we simply consider the standard greedy algorithm presented in [42] as algorithm  $\mathcal{A}$ , which gives 1/2-approximation (in this case  $\beta = 1/2$ ). For completeness, we formalize the extended version of the standard greedy algorithm in Algorithm 7.

---

**Algorithm 7** Extended Greedy Algorithm for Submodular Optimization

---

**Require:**  $\ell \geq 1$ , monotone submodular function  $f : 2^V \rightarrow \mathbb{R}_+$ , Matroid  $\mathcal{M} = (V, \mathcal{I})$ .

**Ensure:** sets  $S_1, \dots, S_\ell \in \mathcal{I}$ .

- 1: **for**  $\tau = 1, \dots, \ell$  **do**
- 2:      $S_\tau \leftarrow \emptyset$
- 3:     **while**  $S_\tau$  is not a basis of  $\mathcal{M}$  **do**
- 4:         **Compute**

$$e^* = \operatorname{argmax}_{S_\tau + e \in \mathcal{I}} f(\cup_{j=1}^{\tau} S_j + e).$$

- 5:         **Update**  $S_\tau \leftarrow S_\tau + e^*$ .
- 

**Running time analysis of Algorithm 6.** In this section, we study the running time of the bi-criteria algorithm we just presented. To show that a set of polynomial size of values

for  $\gamma$  exists such that one of them satisfies  $(1 - \epsilon/2) \text{OPT} \leq \gamma \leq \text{OPT}$ , we simply try  $\gamma = n f_i(e) (1 - \epsilon/2)^j$  for all  $i \in [k]$ ,  $e \in V$ , and  $j = 0, \dots, \lceil \ln_{1-\epsilon/2}(1/n) \rceil$ . Note that there exists an index  $i^* \in [k]$  and a set  $S^* \in \mathcal{I}$  such that  $\text{OPT} = f_{i^*}(S^*)$ . Now let  $e^* = \operatorname{argmax}_{e \in S^*} f_{i^*}(e)$ . Because of submodularity and monotonicity we have  $\frac{1}{|S^*|} f_{i^*}(S^*) \leq f_{i^*}(e^*) \leq f_{i^*}(S^*)$ . So, we can conclude that  $1 \geq \text{OPT} / n f_{i^*}(e^*) \geq 1/n$ , which implies that  $j = \lceil \ln_{1-\epsilon/2}(\text{OPT} / n f_{i^*}(e^*)) \rceil$  is in the correct interval, obtaining

$$(1 - \epsilon/2) \text{OPT} \leq n f_{i^*}(e^*) (1 - \epsilon/2)^j \leq \text{OPT}.$$

We remark that the dependency of the running time on  $\epsilon$  can be made logarithmic by running a binary search on  $j$  as opposed to trying all  $j = 0, \dots, \lceil \ln_{1-\epsilon/2}(1/n) \rceil$ . This would take at most  $\frac{nk}{\epsilon} \cdot \log n$  iterations. We could also say that doing a binary search to get a value up to a relative error of  $1 - \epsilon/2$  of  $\text{OPT}$  would take  $\log_{1+\epsilon} \text{OPT}$ . So, we consider the minimum of those two quantities  $\min\{\frac{nk}{\epsilon} \cdot \log n, \log_{1+\epsilon} \text{OPT}\}$ . Given that Algorithm 6 runs in  $\ell \cdot \text{time}(\mathcal{A})$  where  $\ell = O(\log \frac{k}{\epsilon})$  is the number of rounds, we conclude that the bi-criteria algorithm runs in  $O(\text{time}(\mathcal{A}) \cdot \log \frac{k}{\epsilon} \cdot \min\{\frac{nk}{\epsilon} \cdot \log n, \log_{1+\epsilon} \text{OPT}\})$  time. In particular, if  $\mathcal{A}$  is the standard greedy algorithm, then we know that  $\text{time}(\mathcal{A}) = nr$  where  $r$  is the rank of the matroid, so we obtain a bi-criteria algorithm that runs in  $O(n \cdot r \cdot \log \frac{k}{\epsilon} \cdot \min\{\frac{nk}{\epsilon} \cdot \log n, \log_{1+\epsilon} \text{OPT}\})$  time.

From the previous running time analysis, we observe that the extended version of the greedy algorithm performs  $n \cdot r \cdot \ell$  function calls, which can be inefficient when  $n$  is sufficiently large. Therefore, we are interested in studying efficient bi-criteria algorithms for problem (4.1). In Section 4.2.2, we support our theoretical results with a computational study in several real-world applications.

As we mentioned in Section 4.1.3, there has been some progress on reducing the number of function calls of the standard greedy algorithm, see e.g. [83, 5, 84]. In the remainder of this section we study the *threshold greedy* algorithm introduced in [5]. We formalize its

extended version in Algorithm 8. The original version of Badanidiyuru and Vondrak [5] corresponds to considering  $\ell = 1$ .

---

**Algorithm 8** Extended Threshold-Greedy

---

**Input:**  $\ell \geq 1$ , ground set  $V$  with  $n := |V|$ , monotone submodular function  $g : 2^V \rightarrow \mathbb{R}_+$ , matroid  $\mathcal{M} = (V, \mathcal{I})$  and  $\delta > 0$ .

**Output:** feasible sets  $S_1, \dots, S_\ell \in \mathcal{I}$ .

```

1: for  $\tau = 1, \dots, \ell$  do
2:    $S_\tau \leftarrow \emptyset$ 
3:    $d \leftarrow \max_{e \in V} g(\cup_{j=1}^{\tau-1} S_j + e)$ 
4:   for ( $\omega = d; \omega \geq \frac{\delta}{n}d; \omega \leftarrow (1 - \delta)\omega$ ) do
5:     for  $e \in V \setminus S_\tau$  do
6:       if  $S_\tau + e \in \mathcal{I}$  and  $g_{\cup_{j=1}^{\tau-1} S_j}(e) \geq \omega$  then
7:          $S_\tau \leftarrow S_\tau + e$ 

```

---

For the problem of maximizing a single submodular function subject to a single cardinality constraint, Badanidiyuru and Vondrak [5] prove that the threshold greedy algorithm achieves a  $(1 - 1/e - \delta)$ -approximation factor, where  $\delta$  is the parameter used for decreasing the threshold. Moreover, it requires only  $O(\frac{n}{\delta} \cdot \log \frac{n}{\delta})$  function calls, a considerable decrease compared to the time complexity of the standard greedy algorithm. Along the same lines of the proof in [5], we can prove the following proposition for matroid constraints.

**Corollary 4.** *Given  $\delta > 0$ , Algorithm 8 with  $\ell = 1$  gives a  $(1 - \frac{1}{2-\delta})$ -approximation for the problem (1.2), using  $O(\frac{n}{\delta} \cdot \log \frac{n}{\delta})$  queries.*

*Proof.* Denote by  $r$  the rank of matroid  $\mathcal{M}$ . Let  $S^* = \{e_1^*, \dots, e_r^*\}$  and  $S = \{e_1, \dots, e_r\}$  be the optimal set and the set obtained with the algorithm, respectively. W.l.o.g, we can assume that  $S^*$  and  $S_G$  are both basis in  $\mathcal{M}$ , so there exists a bijection  $\phi$  such that  $\phi(e_i) = e_i^*$  for all  $i \in [r]$ . Denote by  $S_{i-1} = \{e_1, \dots, e_{i-1}\}$  the set of elements after iteration  $i - 1$ . Observe that if  $e_i$  is the next element chosen by the algorithm and the current threshold value is  $w$ , then we get the inequalities

$$g_{S_{i-1}}(x) = \begin{cases} \geq w & \text{if } x = e_i \\ \leq w/(1 - \delta) & \text{if } x \in V \text{ s.t. } S_{i-1} + x \in \mathcal{I} \end{cases}$$

This implies that  $g_{S_{i-1}}(e_i) \geq (1 - \delta)f_{S_{i-1}}(x)$  for all  $x \in V \setminus S_{i-1}$  such that  $S_{i-1} + x \in \mathcal{I}$ . In particular for  $x = e_i^*$  we have then

$$(1 - \delta)g_{S_{i-1}}(e_i^*) \leq g(S_i) - g(S_{i-1}).$$

On the other hand, if we apply submodularity twice we get

$$g(S^*) - g(S) \leq \sum_{i=1}^K g_S(e_i^*) \leq \sum_{i=1}^K g_{S_{i-1}}(e_i^*)$$

Using the previous two inequalities we get

$$(1 - \delta)[g(S^*) - g(S)] \leq \sum_{i=1}^K g(S_i) - g(S_{i-1}) = g(S).$$

So we finally obtain

$$g(S) \geq \left(1 - \frac{1}{2 - \delta}\right) \cdot g(S^*)$$

□

In other words, given  $\delta > 0$  the threshold greedy algorithm is  $(1 - \frac{1}{2 - \delta})$ -approximation for problem (1.2). Therefore, by using Theorem 12 we can obtain the following corollary.

**Corollary 5.** *For problem (4.1), there is a polynomial time algorithm that returns a set  $S^{\text{ALG}}$ , such that for given  $0 < \epsilon, \delta < 1$ , for all  $j \in [k]$  it holds*

$$f_j(S^{\text{ALG}}) \geq (1 - \epsilon) \cdot \max_{S \in \mathcal{I}} \min_{i \in [k]} f_i(S),$$

where  $S^{\text{ALG}} = S_1 \cup \dots \cup S_\ell$  with  $\ell = \lceil \log \frac{2k}{\epsilon} / \log(2 - \delta) \rceil$ , and  $S_1, \dots, S_\ell$  are feasible.

### 4.2.2 Computational Study for the Offline Problem

Our objective in this section is to empirically demonstrate the improvements of our efficient bi-criteria algorithm that uses the extended threshold greedy when is compared to the extended standard greedy. We consider other implementation improvements such as lazy evaluations and an early stopping rule that considerable speed up the original bi-criteria algorithm.

First, let us recall how the bi-criteria algorithm works in general terms. In an outer loop we obtain an estimate  $\gamma$  on the value of the optimal solution  $\text{OPT} := \max_{S \in \mathcal{I}} \min_{i \in [k]} f_i(S)$  via a binary search. Next, for each guess  $\gamma$  we define a new submodular set function as  $g^\gamma(S) := \frac{1}{k} \sum_{i \in [k]} \min\{f_i(S), \gamma\}$ . Finally, we run either Algorithm 7 or Algorithm 8 to obtain a candidate solution. Depending on this result, we update the binary search on  $\gamma$ , and we iterate. We stop the binary search whenever we get a relative error of  $1 - \epsilon/2$ , namely,  $(1 - \epsilon/2) \text{OPT} \leq \gamma \leq \text{OPT}$ . By considering Algorithm 8 we can get (nearly) optimal objective value using less function evaluations than Algorithm 7 at cost of producing a slightly bigger family of feasible sets.

Unfortunately, the main bottleneck of the bi-criteria algorithm remains obtaining a certificate of (near)-optimality or equivalently, a good upper bound on the optimum. We obtain that the optimum value is at most  $\gamma$  whenever running an extended approximation algorithm on function  $g^\gamma$  fails to return a solution of desired objective. Due to the desired accuracy in binary search and the number of steps in the extended algorithm, obtaining good upper bounds on the optimum is computationally prohibitive. We resolve this issue by implementing an early stopping rule in the bi-criteria algorithm. When running an extended approximation algorithm on function  $g^\gamma$  (as explained above) we use the stronger guarantee given in Proposition 12. When  $\gamma$  is much larger than  $\text{OPT}$  and we fail to realize the guarantee in Proposition 12: if in iteration  $\tau \in [\ell]$  we obtain a set  $S_\tau$  such that  $g(\cup_{t=1}^\tau S_t) < (1 - \beta^\tau) \cdot \gamma$ , then we stop and update the upper bound on the optimum to be  $\gamma$ . This allows us to stop the iteration much earlier since in many real instances  $\tau$  is typically

much smaller than  $\ell$  when  $\gamma$  is large. Note that  $\beta = 1/2$  for Algorithm 7 and  $\beta = 1/(2 - \delta)$  for Algorithm 8. This leads to a drastic improvement in the number of function calls as well as CPU time. Indeed, without this improvement, Algorithm 7 even with lazy evaluations has a poor performance with a CPU time of more than 4 hours in small instances of  $n = 5,000$  elements as compared to few minutes after addition of this rule.

In this section, we assess the performance of these implementation improvements in three applications, showing empirically that the tested algorithms using these small changes significantly outperform Algorithm 7 without improvements. Also, we present a simple heuristic adapted from the *stochastic greedy* algorithm introduced in [84] (see more details in Appendix B.2).

**Lazy evaluations** All algorithms and baselines are implemented with lazy evaluations [83]. This means, we keep a list of an upper bound  $\rho(e)$  on the marginal gain for each element (initially  $\infty$ ) in decreasing order, and at each iteration, it evaluates the element at the top of the list  $e'$ . If the marginal gain of this element satisfies  $g_S(e') \geq \rho(e)$  for all  $e \neq e'$ , then submodularity ensures  $g_S(e') \geq g_S(e)$ . In this way, greedy does not have to evaluate all marginal values to select the best element.

**Bounds initialization.** To compute the initial LB and UB for the binary search, we run the lazy greedy [83] for each function in a small sub-collection  $\{f_i\}_{i \in [k']}$ , where  $k' \ll k$ , leading to  $k'$  solutions  $A^1, \dots, A^{k'}$  with guarantees  $f_i(A^i) \geq (1/2) \cdot \max_{S \in \mathcal{I}} f_i(S)$ . Therefore, we set  $UB = 2 \cdot \min_{i \in [k']} f_i(A^i)$  and  $LB = \max_{j \in [k']} \min_{i \in [k]} f_i(A^j)$ . This two values correspond to upper and lower bounds for the true optimum OPT.

To facilitate the interpretation of our theoretical results, we will consider partition constraints in all experiments: the ground set  $V$  is partitioned in  $q$  sets  $\{P_1, \dots, P_q\}$  and the family of feasible sets is  $\mathcal{I} = \{S : |S \cap P_j| \leq b, \forall j \in [q]\}$ , same budget  $b$  for each part. We test four methods: (prevE-G) the extended greedy 7 with no improvements, and the rest

with improvements, (E-G) the extended greedy 7, (E-ThG) the extended threshold greedy 8, and (E-StochG) a heuristic we called extended stochastic greedy. The last procedure is an extended version of the *stochastic greedy* [84], and adapted to partition constraints (see Appendix B.2). Finally, we consider  $\ell = \lceil \log \frac{2k}{\epsilon} \rceil$  for E-G and E-StochG, and  $\ell = \lceil \log \frac{2k}{\epsilon} / \log(2 - \delta) \rceil$  for E-ThG. See Appendix B.3 for the final pseudo-code of the main algorithm.

After running the four algorithms, we save the solution  $S^{\text{ALG}}$  with the largest violation ratio  $\nu$ , and denote by  $\tau_{\max} := \lceil \nu \rceil$ . Observe that  $S^{\text{ALG}} = S_1 \cup \dots \cup S_{\tau_{\max}}$  where  $S_\tau \in \mathcal{I}$  for all  $\tau \in [\tau_{\max}]$ . We consider two additional baseline algorithms (without binary search): Random Selection (RS) which outputs a set  $\tilde{S} = \tilde{S}_1 \cup \dots \cup \tilde{S}_{\tau_{\max}}$  such that for each  $\tau \in [\tau_{\max}]$ :  $\tilde{S}_\tau$  is feasible, constructed by selecting elements uniformly at random, and  $|\tilde{S}_\tau \cap P_j| = |S_\tau \cap P_j|$  for each part  $j \in [q]$ . Secondly, (G-Avg) we run  $\tau_{\max}$  times the lazy greedy algorithm on the average function  $\frac{1}{k} \sum_{i \in [k]} f_i$  and considering constraints  $\mathcal{I}_\tau = \{S : |S \cap P_j| \leq |S_\tau \cap P_j|, \forall j \in [q]\}$  for each iteration  $\tau \in [\tau_{\max}]$ .

In all experiments we consider the following parameters: approximation  $1 - \epsilon = 0.99$ , threshold  $\delta = 0.1$ , and sampling in E-StochG with  $\epsilon' = 0.1$ . The composition of each part  $P_j$  is always uniformly at random from  $V$ .

**Non-parametric Learning.** We follow the setup in [84]. Let  $X_V$  be a set of random variables corresponding to bio-medical measurements, indexed by a ground set of patients  $V$ . We assume  $X_V$  to be a Gaussian Process (GP), i.e., for every subset  $S \subseteq V$ ,  $X_S$  is distributed according to a multivariate normal distribution  $\mathcal{N}(\mu_S, \Sigma_{S,S})$ , where  $\mu_S = (\mu_e)_{e \in S}$  and  $\Sigma_{S,S} = [\mathcal{K}_{e,e'}]_{e,e' \in S}$  are the prior mean vector and prior covariance matrix, respectively. The covariance matrix is given in terms of a positive definite kernel  $\mathcal{K}$ , e.g., a common choice in practice is the squared exponential kernel  $\mathcal{K}_{e,e'} = \exp(-\|x_e - x_{e'}\|_2^2/h)$ . Most efficient approaches for making predictions in GPs rely on choosing a small subset of data points. For instance, in the Informative Vector Machine (IVM) the goal is to obtain a sub-

set  $A$  such that maximizes the information gain,  $f(A) = \frac{1}{2} \log \det(\mathbf{I} + \sigma^{-2} \Sigma_{A,A})$ , which is known to be monotone and submodular [68]. In our experiment, we use the Parkinson Telemonitoring dataset [107] consisting of  $n = 5,875$  patients with early-stage Parkinson’s disease and the corresponding bio-medical voice measurements with 22 attributes (dimension of the observations). We normalize the vectors to zero mean and unit norm. With these measurements we computed the covariance matrix  $\Sigma$  considering the squared exponential kernel with parameter  $h = 0.75$ . For our robust criteria, we consider  $k = 20$  perturbed versions of the information gain defined with  $\sigma^2 = 1$ , i.e., problem (4.1) corresponds to  $\max_{A \in \mathcal{I}} \min_{i \in [20]} f(A) + \sum_{e \in A \cap \Lambda_i} \eta_e$ , where  $f(A) = \frac{1}{2} \log \det(\mathbf{I} + \Sigma_{AA})$ ,  $\Lambda_i$  is a random set of size 1,000 with different composition for each  $i \in [20]$ , and  $\eta \sim [0, 1]^V$  is a uniform error vector.

We made 20 random runs considering  $q = 3$  parts and budget  $b = 5$ . We report the results in Figures 4.1 - 4.4. In the performance profiles ( $p$  denotes the fraction of instances and  $\theta$  the threshold of performance) Figures 4.1 and 4.2, we observe that any of the three algorithms clearly outperform `prevE-G`, either in terms of running time or function calls. For example, in Figure 4.1 we note that all the algorithms with improvements outperform at least 10 times the running time of (`prevE-G`). With this, we show empirically that our implementation improvements help in the performance of the algorithm. We also note that `E-StochG` is likely to have the best performance. Box-plots for the function calls in Figure 4.4 confirm this fact, since `E-StochG` has the lowest median. In this figure, we do not present the results of `prevE-G` because of the difference in magnitude on the number of function calls. Finally, in Figure 4.3 we present the objective values obtained in a single run, and we observe that the stopping rule is useful since the three tested algorithms find a good solution earlier (using fewer elements) outperforming `prevE-G` and the benchmarks, and at much less cost as we mentioned before.

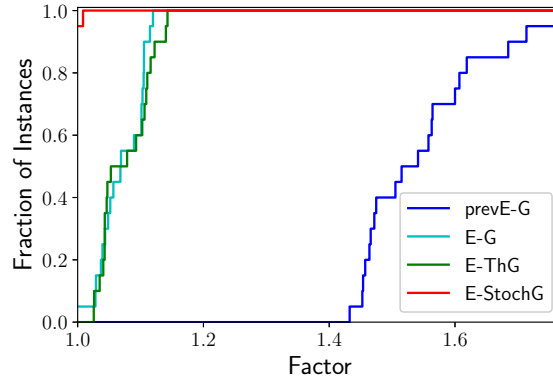


Figure 4.1: *Non-parametric learning*: performance profiles for running time (logarithmic scale in the  $x$ -axis)

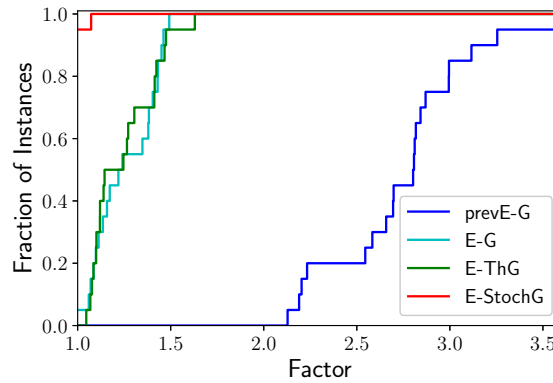


Figure 4.2: *Non-parametric learning*: performance profiles for function calls.

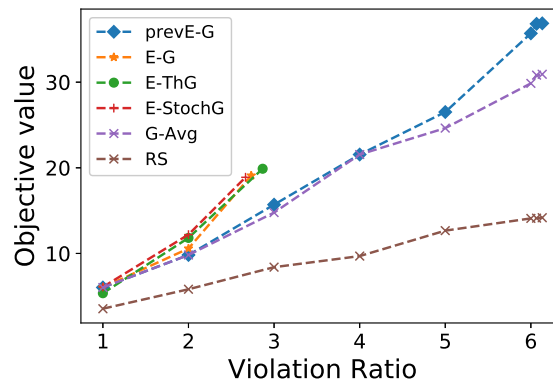


Figure 4.3: *Non-parametric learning*: objective value versus the violation ratio in a single run of each method.

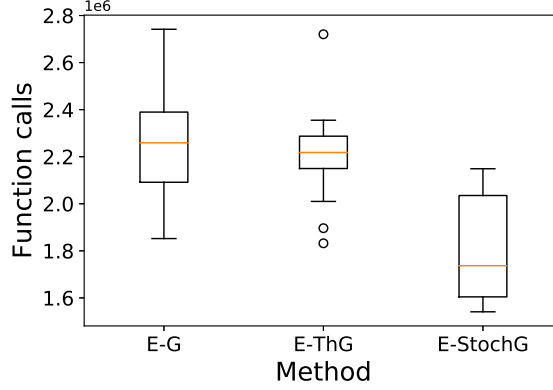


Figure 4.4: *Non-parametric learning*: box-plot for the function calls.

**Exemplar-based Clustering.** We follow the setup in [84]. Solving the  $k$ -medoid problem is a common way to select a subset of exemplars that represent a large dataset  $V$  [62]. This is done by minimizing the sum of pairwise dissimilarities between elements in  $A \subseteq V$  and  $V$ . Formally, define  $L(A) = \frac{1}{V} \sum_{e \in V} \min_{v \in A} d(e, v)$ , where  $d : V \times V \rightarrow \mathbb{R}_+$  is a distance function that represents the dissimilarity between a pair of elements. By introducing an appropriate auxiliary element  $e_0$ , it is possible to define a new objective  $f(A) := L(\{e_0\}) - L(A + e_0)$  that is monotone and submodular [48], thus maximizing  $f$  is equivalent to minimizing  $L$ . In our experiment, we use the VOC2012 dataset [37]. The ground set  $V$  corresponds to images, and we want to select a subset of the images that best represents the dataset. Each image has several (possible repeated) associated categories such as person, plane, etc. There are around 20 categories in total. Therefore, images are represented by feature vectors obtained by counting the number of elements that belong to each category, for example, if an image has 2 people and one plane, then its feature vector is  $(2, 1, 0, \dots, 0)$  (where zeros correspond to other elements). We choose the Euclidean distance  $d(e, e') = \|x_e - x_{e'}\|_2$  where  $x_e, x_{e'}$  are the feature vectors for images  $e, e'$ . We normalize the feature vectors to mean zero and unit norm, and we choose  $e_0$  as the origin. For our robust criteria, we consider  $k = 20$  perturbations of the function  $f$  defined above, i.e., problem (4.1) corresponds to  $\max_{A \in \mathcal{I}} \min_{i \in [20]} f(A) + \sum_{e \in A \cap \Lambda_i} \eta_e$ , where  $\Lambda_i$  is a ran-

dom set of fixed size with different composition for each  $i \in [20]$ , and finally,  $\eta \sim [0, 1]^V$  is a uniform error vector.

We consider two experiments: (*small*) with  $n = 3,000$  images, 20 random instances considering  $q = 6$  and  $b = 70$ ,  $|\Lambda_i| = 500$  and (*large*) with  $n = 17,125$  images, 20 random instances  $q \in \{10, \dots, 29\}$  parts and budget  $b = 5$ ,  $|\Lambda_i| = 3,000$  (we do not implement `prevE-G` because of the exorbitant running time). We report the results of the experiments in Figures 4.5 - 4.8. For *small*, Figures 4.5 and 4.6 confirm our theoretical results: `E-ThG` is the most likely to use less function calls (Figure 4.6) and running time (Figure 4.5) when the rank is relatively high (in this case  $q \cdot b = 420$ ) which contrasts with the performance of `E-G` that depends on the rank (Figure 4.6 reflects this). For *large*, we can see in Figures 4.7 and 4.8 that the results are similar, either in terms of running time or function evaluations, so when we face large ground sets, we could choose any algorithm, but we would still prefer `E-ThG` since it has no dependency on the rank.

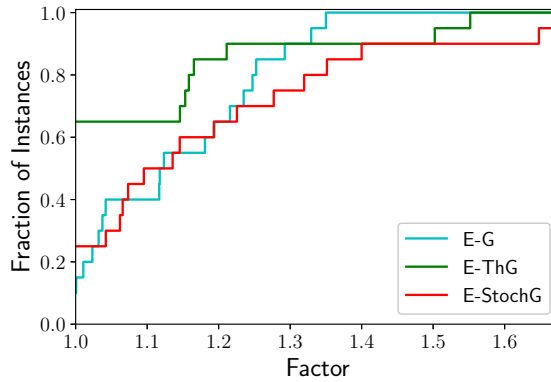


Figure 4.5: *Clustering*: (*small*) performance profiles for the running time.

**Sensor Placement** For this problem we follow the setup in [71]. Here, we are given a set of sensors  $V$  with fixed locations in a specific region. Each sensor  $s$  measures certain phenomena such as temperature, humidity and light, which define a random vector  $X_s$ . We assume that the set of random variables  $X_V$  is distributed according to a multivariate normal distribution, which corresponds to a Gaussian Process (GP). The predictive vari-

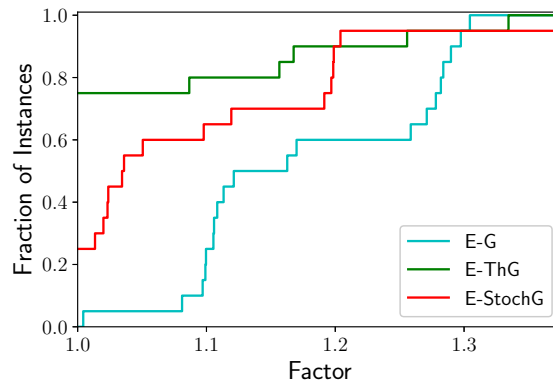


Figure 4.6: *Clustering: (small)* performance profiles for the function calls.

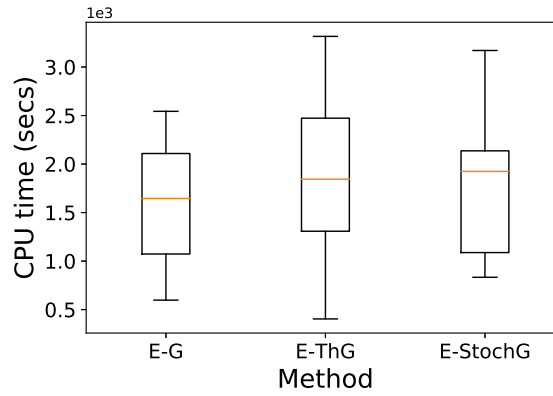


Figure 4.7: *Clustering: (Large)* box-plots for the running times.

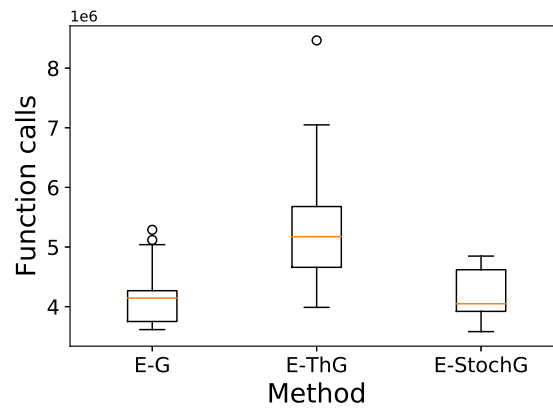


Figure 4.8: *Clustering: (Large)* box-plots for the function calls.

ance of sensor  $s$  after obtaining observations from a subset of sensors  $A \subseteq V$  is given by  $\sigma_{s|A}^2 = \sigma_s^2 - \Sigma_{sA} \Sigma_{AA}^{-1} \Sigma_{As}$ , where  $\Sigma_{AA}$  is the covariance matrix of the measurements at the chosen locations  $A$ ,  $\Sigma_{sA}$  is the row-vector in  $\Sigma$  with row  $s$  and columns  $A$ , and  $\sigma_s^2$  is the *a priori* variance of sensor  $s$ . Traditionally, the goal is to find a subset  $A$  of sensors that minimizes the predictive variance. However, let us assume that the *a priori* variance  $\sigma_s^2$  is constant for all locations  $s$  and define the variance reduction  $f_s(A) := \Sigma_{sA} \Sigma_{AA}^{-1} \Sigma_{As}$ . Das and Kempe [32] show that  $f_s$  is monotone and submodular for certain distributions. Therefore, minimizing  $\sigma_{s|A}^2$  is equivalent to maximizing  $f_s$  when  $\sigma_s^2$  is assumed to be constant.

We use the Intel Research Berkeley dataset of  $n = 44$  sensors, which contains measurements of temperature (T), humidity (L), and light (L). We consider data of three consecutive days, and we construct the corresponding covariance matrices  $\Sigma^T$ ,  $\Sigma^H$ , and  $\Sigma^L$ . For our robust criteria, we consider perturbed versions of the average variance reduction for each observation  $k = 3$ , i.e., problem (4.1) corresponds to  $\max_{A \in \mathcal{I}} \min_{k \in \{T, H, L\}} \{f_k(A) + \sum_{e \in A \cap \Lambda_k} \eta_e\}$ , where  $f_k(A) = \frac{1}{44} \sum_{s \in [44]} \Sigma_{sA}^k (\Sigma_{AA}^k)^{-1} \Sigma_{As}^k$ ,  $\Lambda_k$  is random set of size 15, different in composition for each  $k$ , and  $\eta \sim [0, 1]^V$  is an error vector.

We made 30 random runs considering the number of parts  $q = 3$  and budget  $b = 1$ . We report the results of the instances in Figures 4.9 - 4.13. We observe that the three tested algorithms clearly outperform `PREV-E-G`, either in running time (see box-plot in Figure 4.10 and performance profile in Figure 4.12) and in the number of function evaluations (see box-plot in Figure 4.9 and performance profile in Figure 4.13). When we only compared the three tested algorithm, the performance is very similar (see Figures 4.9 and 4.10), but `E-STochG` is the most likely to use less number of function calls, see Figure 4.13. In terms of running time `E-G` and `E-SToch` are the most likely to solve the problem faster, see Figure 4.12. Finally, in Figure 4.11 we observe that the early stopping rule help to find a good candidate solution earlier by using less elements and at much less computational cost.

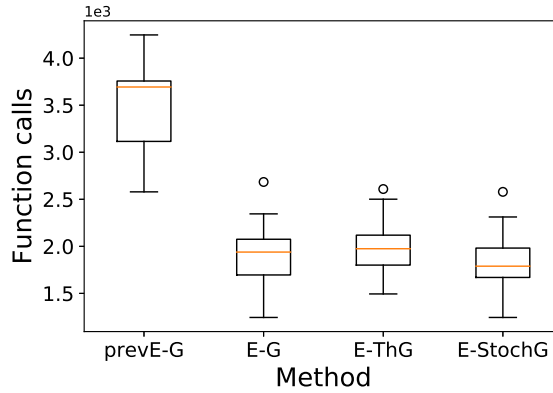


Figure 4.9: *Sensor Placement*: Box-plots for the function calls.

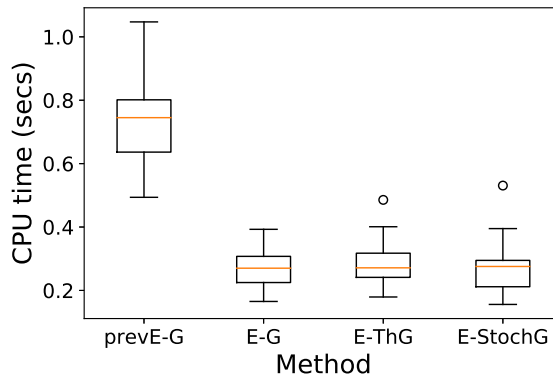


Figure 4.10: *Sensor Placement*: Box-plots for the running time.

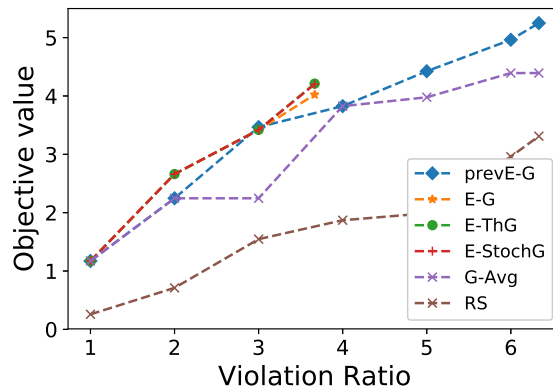


Figure 4.11: *Sensor Placement*: objective value versus the violation ratio in a single run of each method.

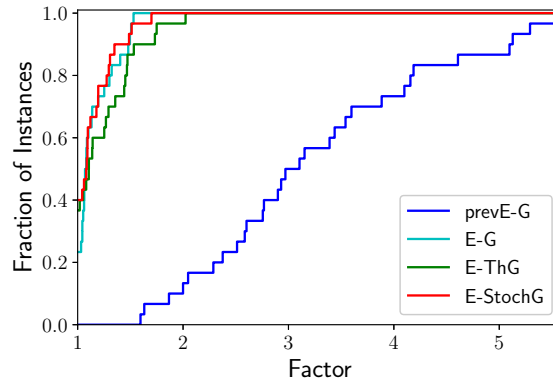


Figure 4.12: *Sensor Placement*: Performance profiles for the running time.

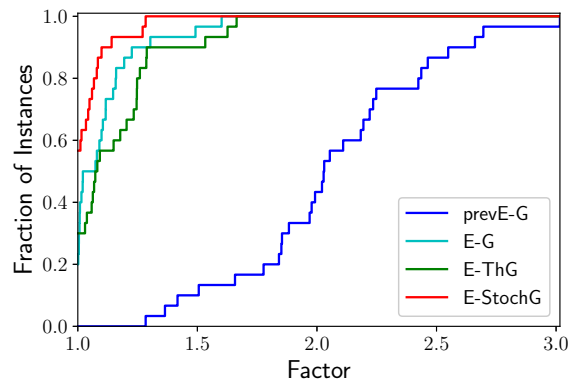


Figure 4.13: *Sensor Placement*: Performance profiles for the function calls.

### 4.2.3 Continuous Offline Bi-criteria Algorithm

In this section, we present a continuous algorithm that achieves a tight bi-criteria approximation for the robust submodular optimization problem (4.1) and prove Theorem 13. This algorithm outputs a random set  $S^{\text{ALG}}$  which is the union of  $O(\ln \frac{k}{\epsilon})$  independent sets and such that with constant probability has value close to the true optimum. The number of independent sets required for obtaining this result is smaller up to a constant than the number of sets obtained by the extended greedy, more importantly, the guarantees of this continuous algorithm match the hardness result in [71].

**Theorem 13.** *Let  $(V, \mathcal{I})$  be a matroid and let  $f_i : 2^V \rightarrow \mathbb{R}_+$  be a monotone submodular function for  $i \in [k]$ . Then, there is a randomized polynomial time algorithm that with constant probability returns a set  $S^{\text{ALG}}$ , such that for all  $i \in [k]$ , for a given  $0 < \epsilon < 1$ ,*

$$f_i(S^{\text{ALG}}) \geq (1 - \epsilon) \cdot \max_{S \in \mathcal{I}} \min_{j \in [k]} f_j(S),$$

and  $S^{\text{ALG}} = S_1 \cup \dots \cup S_\ell$  for  $\ell = O(\ln \frac{k}{\epsilon})$ , and  $S_1, \dots, S_\ell \in \mathcal{I}$ .

Our overall approach is to first find a fractional solution with a desirable approximation guarantee and then round it to an integral solution. We use a relaxation of a matroid to its convex hull to accommodate the search for a fractional solution.

For this algorithm, we need an estimate  $\gamma$  of the value of the optimal solution which we denote by OPT. We prove the following lemma which solves an approximate decision version of our optimization problem. The proof of Theorem 13 follows from the lemma and a search over an approximate value for OPT.

**Lemma 1.** *There is a randomized polynomial time algorithm that given  $\gamma \leq \text{OPT}$  and  $0 < \epsilon < 1$  returns with constant probability a set  $S^{\text{ALG}}$  such that for all  $i \in [k]$ ,*

$$f_i(S^{\text{ALG}}) \geq (1 - \epsilon) \cdot \gamma,$$

where  $S^{\text{ALG}} = \bigcup_{j \in [\ell]} S_j$  with  $\ell = O(\ln \frac{k}{\epsilon})$  and  $S_j \in \mathcal{I}$  for each  $j \in [\ell]$ .

First, we finish the proof of Theorem 13 assuming Lemma 1.

*Proof of Theorem 13.* We apply the algorithm from Lemma 1 with approximation loss  $\epsilon/2$  and with different values of  $\gamma$ , some of which may be larger than  $\text{OPT}$ , but at least one of them is guaranteed to satisfy  $(1 - \epsilon/2) \text{OPT} \leq \gamma \leq \text{OPT}$ . At the end we return the set  $S^{\text{ALG}}$  from our runs with the highest value of  $\min_{i \in [k]} f_i(S^{\text{ALG}})$ .

Before describing the set of candidate values of  $\gamma$  that we try, note that if the algorithm succeeds for the particular value of  $\gamma$  satisfying  $(1 - \epsilon/2) \text{OPT} \leq \gamma \leq \text{OPT}$ , then we get

$$\min_{i \in [k]} f_i(S^{\text{ALG}}) \geq (1 - \epsilon/2) \cdot \gamma \geq (1 - \epsilon) \text{OPT},$$

and since we return the set with the highest  $\min_{i \in [k]} f_i(S^{\text{ALG}})$ , the algorithm's output will have the desired approximation guarantee.

It remains to show that a set of polynomial size of values for  $\gamma$  exists such that one of them satisfies  $(1 - \epsilon/2) \text{OPT} \leq \gamma \leq \text{OPT}$ . To this end we simply try  $\gamma = n f_i(e) (1 - \epsilon/2)^j$  for all  $i \in [k]$ ,  $e \in V$ , and  $j = 0, \dots, \lceil \ln_{1-\epsilon/2}(1/n) \rceil$ . Note that there exists an index  $i^* \in [k]$  and a set  $S^* \in \mathcal{I}$  such that  $\text{OPT} = f_{i^*}(S^*)$ . Now let  $e^* = \text{argmax}_{e \in S^*} f_{i^*}(e)$ . Because of submodularity and monotonicity we have  $\frac{1}{|S^*|} f_{i^*}(S^*) \leq f_{i^*}(e^*) \leq f_{i^*}(S^*)$ . So, we can conclude that  $1 \geq \text{OPT} / n f_{i^*}(e^*) \geq 1/n$ , which implies that  $j = \lceil \ln_{1-\epsilon/2}(\text{OPT} / n f_{i^*}(e^*)) \rceil$  is in the correct interval, obtaining

$$(1 - \epsilon/2) \text{OPT} \leq n f_{i^*}(e^*) (1 - \epsilon/2)^j \leq \text{OPT}.$$

This finishes the proof. □

We remark that the dependency of the running time on  $\epsilon$  can be made logarithmic by running a binary search on  $j$  as opposed to trying all  $j = 0, \dots, \lceil \ln_{1-\epsilon/2}(1/n) \rceil$ . We just

need to run the algorithm from Lemma 1 for each  $\gamma$  polynomially many times to make the failure probability exponentially small whenever  $\gamma \leq \text{OPT}$ .

The rest of this section is devoted to prove Lemma 1. To achieve a strong concentration bound when rounding the fractional solution, we truncate  $f_i$  to  $\min\{\gamma, f_i\}$ . Hereafter, we use  $f_i^\gamma$  to refer to  $\min\{\gamma, f_i\}$ . Note that submodularity and monotonicity is preserved under this truncation. Also, we denote by  $F_i^\gamma$  the corresponding multilinear extension of  $f_i^\gamma$ . Recall the definition of the multilinear extension of a set function given in Section 1.3.

We describe the continuous process counterpart of the algorithm in this section. The discretization details follow using standard methods [108].

**Continuous Greedy.** We start a continuous gradient step process where  $x_\tau$  represents the point at time  $\tau$  we are at. We start at  $x_0 = 0$  and take continuous gradient steps in direction  $\frac{dx_\tau}{d\tau} = v_{\text{all}}(x)$ , such that  $v_{\text{all}}(x)$  satisfies the following conditions:

- (a)  $v_{\text{all}}(x) \cdot \nabla F_i^\gamma(x) \geq \gamma - F_i^\gamma(x)$  for all  $i \in [k]$ ,
- (b)  $v_{\text{all}}(x) \in \mathcal{P}(\mathcal{M})$ , and
- (c)  $v_{\text{all}}(x) + x \in [0, 1]^V$ .

First, we show that such  $v_{\text{all}}$  always exists. Take  $x^*$  to be the indicator vector corresponding to the optimal solution. For any  $x$ ,  $v^* = (x^* - x) \vee 0$  is a positive direction satisfying inequality (1.5), and for all  $i \in [k]$ :

$$v^* \cdot \nabla F_i^\gamma(x) \geq F_i^\gamma(x + v^*) - F_i^\gamma(x) = \gamma - F_i^\gamma(x), \quad (4.4)$$

where the last equality holds since  $F_i^\gamma(x) \leq \gamma$  for all  $x$ . It is easy to check that  $v^*$  satisfies the rest of the constraints (a)-(c), implying that there exists a feasible solution to the above system of linear inequalities. Therefore, we can solve a linear program defined by these inequalities to obtain a solution  $v_{\text{all}}(x)$ .

The above continuous process goes on until time  $\ell = O(\ln \frac{k}{\epsilon})$ . We intentionally set  $\ell > 1$  to obtain a (fractional) solution with a higher budget, which is useful for achieving a bi-criteria approximation. Next, we show the following claim.

**Claim 1.** For any  $\tau \geq 0$ ,  $x_\tau \in \tau \cdot \mathcal{P}(\mathcal{M}) \cap [0, 1]^V$  and for all  $i \in [k]$ ,

$$F_i^\gamma(x_\tau) \geq (1 - e^{-\tau})\gamma.$$

*Proof.* For any  $\tau \geq 0$ , we have

$$x_\tau = \int_0^\tau v_{\text{all}}(x_s) ds = \int_0^1 \tau \cdot v_{\text{all}}(x_{\tau s}) ds.$$

So,  $x_\tau$  is a convex combination of vectors in  $\tau \cdot \mathcal{P}(\mathcal{M})$ . Moreover,  $(v_{\text{all}}(x))_j = 0$  when  $x_j = 1$ , thus  $x_\tau \in [0, 1]^V$  proving the first part of the claim.

For the second part, observe that for all  $i \in [k]$  we have

$$\frac{dF_i^\gamma(x_\tau)}{d\tau} = \frac{dx_\tau}{d\tau} \cdot \nabla F_i^\gamma(x_\tau) = v_{\text{all}}(x_\tau) \cdot \nabla F_i^\gamma(x_\tau) \geq \gamma - F_i^\gamma(x_\tau).$$

Moreover,  $F_i^\gamma(0) = 0$ . Now we solve the above differential equation to obtain

$$F_i^\gamma(x_\tau) \geq (1 - e^{-\tau}) \cdot \gamma$$

for each  $i \in [k]$  as claimed. □

Thus, by setting  $\ell = \ln \frac{k}{\epsilon} + \ln \frac{1}{c}$ , we obtain  $F_i^\gamma(x_\ell) \geq (1 - \frac{\epsilon}{k} \cdot c) \cdot \gamma$  for all  $i \in [k]$  and a desired constant  $c < 1$ . We next show how to obtain an integral solution.

**Rounding.** The next lemma summarizes our rounding. We first show that the fractional solution at time  $\ell$  is contained in the matroid polytope of the  $\ell$ -fold union of matroid  $\mathcal{M}$ . We then do randomized swap rounding, introduced in [26], in this matroid polytope. The

truncation of the submodular functions, as well as properties of randomized swap rounding, play a crucial role in the proof.

**Lemma 2.** *Let  $\ell = \lceil \ln \frac{k}{\epsilon} + \ln \frac{1}{c} \rceil$  be an integer and  $x_\ell$  be the output of the continuous greedy algorithm at time  $\ell$  such that  $F_i^\gamma(x_\ell) \geq (1 - \frac{\epsilon}{k} \cdot c) \cdot \gamma$  for each  $i \in [k]$  and some constant  $c < 1$ . Then, there exists a polynomial time randomized algorithm that outputs a set  $S$  such that with probability  $\Omega(1)$ , for each  $i \in [k]$  we have*

$$f_i(S) \geq (1 - \epsilon) \cdot \gamma.$$

Moreover,  $S$  is a union of at most  $\ell$  independent sets in  $\mathcal{M}$ .

*Proof.* Let  $\mathcal{M}_\ell = \vee_\ell \mathcal{M}$  be the  $\ell$ -fold union of matroid  $\mathcal{M}$ , i.e.,  $S$  is an independent set in  $\mathcal{M}_\ell$  if and only if  $S$  is a union of  $\ell$  independent sets of  $\mathcal{M}$ . We denote by  $\mathcal{I}_\ell$  the set of independent sets of  $\mathcal{M}_\ell$ . The rank function of  $\mathcal{M}_\ell$  is given by  $r_{\mathcal{M}_\ell}(S) = \min_{A \subseteq S} |S \setminus A| + \ell \cdot r_{\mathcal{M}}(A)$  (see [97]). We first show that  $x = x_\ell$  is in the convex hull of independent sets of matroid  $\mathcal{M}_\ell$ , i.e.,  $\mathcal{P}(\mathcal{M}_\ell)$ . This polytope is given by  $\mathcal{P}(\mathcal{M}_\ell) = \{z \in \mathbb{R}_+^V : z(S) \leq r_{\mathcal{M}_\ell}(S), \forall S \subseteq V\}$ , where  $z(S) = \sum_{e \in S} z_e$ . We now prove that  $x \in \mathcal{P}(\mathcal{M}_\ell)$ . For any  $S \subseteq V$  and  $A \subseteq S$ , we have  $x(S) = \sum_{e \in S \setminus A} x_e + x(A) \leq |S \setminus A| + \ell \cdot r_{\mathcal{M}}(A)$ , where the last inequality is due to the fact that  $x_e \leq 1$  for all  $e$ , and  $x(A) \leq \ell \cdot r_{\mathcal{M}}(A)$  because  $x \in \ell \cdot \mathcal{P}(\mathcal{M})$  by Claim 1. Therefore,  $x \in \mathcal{P}(\mathcal{M}_\ell)$ .

Next, we apply a randomized swap rounding (see [26]) in matroid  $\mathcal{M}_\ell$  to obtain the solution. A feature of the randomized swap rounding is that it is oblivious to the specific function  $f_i$  used, and it is only a randomized function of the matroid space and the fractional solution.

**Theorem 14** (Theorem II.1, [26]). *Let  $f$  be a monotone submodular function and  $F$  be its multilinear extension. Let  $x \in \mathcal{P}(\mathcal{M}')$  be a point in the polytope of matroid  $\mathcal{M}'$  and  $S'$  a random independent set obtained from it by randomized swap rounding. Then,  $\mathbb{E}[f(S')] \geq F(x)$ .*

Applying Theorem 14 to fractional solution  $x_\ell$  and matroid  $\mathcal{M}_\ell$ , we obtain a random set  $S \in \mathcal{I}_\ell$  such that

$$\mathbb{E}[f_i^\gamma(S)] \geq F_i^\gamma(x_\ell) \geq \left(1 - \frac{\epsilon}{k} \cdot c\right) \cdot \gamma$$

for all  $i \in [k]$ .

Due to the initial truncation, we have that  $f_i^\gamma(S) \leq \gamma$  with probability one. Thus, using Markov's inequality for each  $i \in [k]$ , we obtain that with probability at least  $1 - \frac{\epsilon}{k}$ , we have  $f_i^\gamma(S) \geq (1 - \epsilon)\gamma$ . Therefore, taking a union bound over  $k$  functions, we obtain  $f_i^\gamma(S) \geq (1 - \epsilon)\gamma$  for all  $i \in [k]$  with probability at least  $1 - c$ , and since  $f_i(S) \geq f_i^\gamma(S)$  we get an integral solution  $S$  with max-min value at least  $(1 - \epsilon)\gamma$  as claimed.  $\square$

### 4.3 The Online Robust Submodular Maximization Problem

In this section, we consider the online robust optimization problem (Equation 4.2) under matroid constraints. We introduce an online bi-criteria algorithm that achieves a sublinear  $(1 - \epsilon)$ -regret (4.2) while using solution  $S^t$  at time  $t$  that is a union of  $O(\ln \frac{1}{\epsilon})$  independent sets from  $\mathcal{I}$ .

In Section 4.2.3 we provided a continuous randomized algorithm for the offline problem. Broadly speaking, at every step, this algorithm finds a feasible direction that improves all  $k$  functions and moves in that direction. We use an LP to find this direction similar to the approach in [108] for the case of  $k = 1$ . However, for the online robust optimization problem, we immediately face with two challenges. First, it is not clear how to find a feasible direction  $v_{\text{all}}$  (as was found via an LP for the offline problem) that is good for all  $k$  submodular functions. To resolve this issue, we use a *soft-min* function that converts robust optimization over  $k$  functions into optimizing of a single function. Secondly, robust optimization leads to non-convex and non-smooth optimization combined with online arrival of such submodular functions. To deal with this, we use the Follow-the-Perturbed-Leader (FPL) online algorithm introduced by Kalai and Vempala [58]. To start, let us first present

definitions and known results that play a key role in this online optimization problem.

### 4.3.1 Preliminaries for the Online Bi-criteria Algorithm

#### *Multilinear Extension*

In Section 1.3 we already defined the multilinear extension of a set function, and its corresponding properties when the set function is monotone and submodular. For the remainder of this chapter, recall the definition of  $\Delta F(y)$  for a given multilinear extension  $F$ : for all  $e \in V$  let

$$\Delta_e F(y) := \mathbb{E}_{S \sim y} [f(S + e) - f(S)] = (1 - y_e) \nabla_e F(y).$$

Multilinear extension plays a crucial role in designing approximation algorithms for various constrained submodular optimization problems. Notably, Vondrak [108] introduced the *discretized continuous greedy* algorithm that achieves a  $1 - 1/e$  approximate solution for maximizing a single submodular function under matroid constraints (see [41] for the variant of the continuous greedy that we use). At a high level, this algorithm discretizes interval  $[0, 1]$  into points  $\{0, \delta, 2\delta, \dots, 1\}$ . Starting at  $x_0 = 0$ , for each  $\tau \in \{\delta, 2\delta, \dots, 1\}$  the algorithm uses an LP to compute the direction  $y_\tau = \operatorname{argmax}_{y \in \mathcal{P}(\mathcal{M})} \Delta F(x_{\tau-\delta}) \cdot y$ . Then the algorithm takes a step in the direction of  $y_\tau$  by setting  $x_{\tau,e} \leftarrow x_{\tau-\delta,e} + \delta \cdot y_{\tau,e} \cdot [1 - x_{\tau-\delta,e}]$  for all  $e \in V$ . Finally, it outputs a set  $S$  by rounding the fractional solution  $x_1$ . We will use this discretized version of the continuous greedy to construct our online algorithm in the following section.

#### *The Soft-min Function*

Consider a set of  $k$  twice differentiable, real-valued functions  $g_1, \dots, g_k$ . Let  $g_{\min}$  be the minimum among these functions, i.e., for each point  $x$  in the domain, define  $g_{\min}(x) := \min_{i \in [k]} g_i(x)$ . This function can be approximated by using the so-called *soft-min* function

$H$  defined as

$$H(x) := -\frac{1}{\alpha} \ln \sum_{i \in [k]} e^{-\alpha g_i(x)},$$

where  $\alpha > 0$  is a fixed parameter. We now present some of the key properties of this function in the following lemma.

**Lemma 3.** *For any set of  $k$  twice differentiable, real-valued functions  $g_1, \dots, g_k$ , the soft-min function  $H$  satisfies the following properties:*

1. *Bounds:*

$$g_{\min}(x) - \frac{\ln k}{\alpha} \leq H(x) \leq g_{\min}(x). \quad (4.5)$$

2. *Gradient:*

$$\nabla H(x) = \sum_{i \in [k]} p_i(x) \nabla g_i(x), \quad (4.6)$$

where  $p_i(x) := e^{-\alpha g_i(x)} / \sum_{j \in [k]} e^{-\alpha g_j(x)}$ . Clearly, if  $\nabla g_i \geq 0$  for all  $i \in [k]$ , then  $\nabla H \geq 0$ .

3. *Hessian:*

$$\frac{\partial^2 H(x)}{\partial x_{e_1} \partial x_{e_2}} = \sum_{i \in [k]} p_i(x) \left( -\alpha \frac{\partial g_i(x)}{\partial x_{e_1}} \frac{\partial g_i(x)}{\partial x_{e_2}} + \frac{\partial^2 g_i(x)}{\partial x_{e_1} \partial x_{e_2}} \right) + \alpha \nabla_{e_1} H(x) \cdot \nabla_{e_2} H(x)$$

Moreover, if for all  $i \in [k]$  we have  $\left| \frac{\partial g_i}{\partial x_{e_1}} \right| \leq L_1$ , and  $\left| \frac{\partial^2 g_i}{\partial x_{e_1} \partial x_{e_2}} \right| \leq L_2$ , then  $\left| \frac{\partial^2 H}{\partial x_{e_1} \partial x_{e_2}} \right| \leq 2\alpha L_1^2 + L_2$ .

4. *Comparing the average of the  $g_i$  functions with  $H$ : given  $T > 0$  we have*

$$H(x) \leq \sum_{i \in [k]} p_i(x) g_i(x) \leq H(x) + \frac{n + \ln T}{\alpha} + \frac{\ln k}{\alpha} + \frac{ke^{-n}}{T}. \quad (4.7)$$

So, for  $\alpha > 0$  sufficiently large  $\sum_{i \in [k]} p_i(x) g_i(x)$  is a good approximation of  $H(x)$ .

*Proof.* We will just prove properties 1 and 4, since the rest is a straightforward calculation.

1. First, for all  $i \in [k]$  we have  $e^{-\alpha g_i(x)} \leq e^{-\alpha g_{\min}(x)}$ . Thus,

$$H(x) = -\frac{1}{\alpha} \ln \sum_{i \in [k]} e^{-\alpha g_i(x)} \geq -\frac{1}{\alpha} \ln (k e^{-\alpha g_{\min}(x)}) = g_{\min}(x) - \frac{\ln k}{\alpha}$$

On the other hand,  $\sum_{i \in [k]} e^{-\alpha g_i(x)} \geq e^{-\alpha g_{\min}(x)}$ . Hence,

$$H(x) \leq -\frac{1}{\alpha} \ln (e^{-\alpha g_{\min}(x)}) = g_{\min}(x).$$

4. Let us consider sets  $A_1 = \{i \in [k] : g_i(x) \leq g_{\min}(x) + (n + \ln T)/\alpha\}$  and  $A_2 = \{i \in [k] : g_i(x) > g_{\min}(x) + (n + \ln T)/\alpha\}$ . Our intuitive argument is the following: when  $\alpha$  is sufficiently large, those  $p_i(x)$ 's with  $i \in A_2$  are exponentially small, and  $p_i(x)$ 's with  $i \in A_1$  go to a uniform distribution over elements in  $A_1$ . First, observe that for each  $i \in A_2$  we have

$$p_i(x) = \frac{e^{-\alpha g_i(x)}}{\sum_{i \in [k]} e^{-\alpha g_i(x)}} < \frac{e^{-\alpha [g_{\min}(x) + (n + \ln T)/\alpha]}}{e^{-\alpha g_{\min}(x)}} = \frac{e^{-n}}{T},$$

so  $\sum_{i \in A_2} p_i(x) g_i(x) \leq \frac{k e^{-n}}{T}$ . On the other hand, for any  $i \in A_1$  we have

$$\sum_{i \in A_1} p_i(x) g_i(x) \leq \left( g_{\min}(x) + \frac{n + \ln T}{\alpha} \right) \sum_{i \in A_1} p_i(x) \leq H(x) + \frac{n + \ln T}{\alpha} + \frac{\ln k}{\alpha}$$

where in the last inequality we used the approximation property of the soft-min function. Therefore,

$$\sum_{i \in [k]} p_i(x) g_i(x) \leq H(x) + \frac{n + \ln T}{\alpha} + \frac{\ln k}{\alpha} + \frac{k e^{-n}}{T}.$$

Finally, the other inequality is clear since  $\sum_{i \in [k]} p_i(x) g_i(x) \geq g_{\min}(x) \geq H(x)$ . □

Finally, we state a lemma which is used to prove Theorem 11. This lemma can be

proven via a simple Taylor approximation, but we defer the details to the Appendix B.

**Lemma 4.** *Fix a parameter  $\delta > 0$ . Consider  $T$  collections of  $k$  twice-differentiable functions, namely  $\{g_i^1\}_{i \in [k]}, \dots, \{g_i^T\}_{i \in [k]}$ . Assume  $0 \leq g_i^t(x) \leq 1$  for any  $x$  in the domain, for all  $t \in [T]$  and  $i \in [k]$ . Define the corresponding sequence of soft-min functions  $H^1, \dots, H^T$ , with a common parameter  $\alpha > 0$ . Then, any two sequences of points  $\{x^t\}_{t \in [T]}, \{\tilde{x}^t\}_{t \in [T]} \subseteq [0, 1]^V$  with  $|x^t - \tilde{x}^t| \leq \delta$  satisfy*

$$\sum_{t \in [T]} H^t(\tilde{x}^t) - \sum_{t \in [T]} H^t(x^t) \geq \sum_{e \in V} \sum_{t \in [T]} \nabla_e H^t(x^t)(\tilde{x}_e^t - x_e^t) - O(Tn^2\delta^2\alpha).$$

#### *Follow-the-Perturbed-Leader algorithm*

In this section, we briefly recall the well-known Follow-the-Perturbed-Leader (FPL) algorithm introduced in [58] and used in many online optimization problems (see e.g., [94]). The classical online learning framework is as follows (see Section 1.1.3): Consider a dynamic process over  $T$  time steps. In each stage  $t \in [T]$ , a decision-maker has to choose a point  $d_t \in \mathcal{D}$  from a fixed (possibly infinite) set of actions  $\mathcal{D} \subseteq \mathbb{R}^n$ , then an adversary chooses a vector  $s_t$  from a set  $\mathcal{S}$ . Finally, the player observes vector  $s_t$  and receives reward  $s_t \cdot d_t$ , and the process continues. The goal of the player is to maximize the total reward  $\sum_{t \in [T]} s_t \cdot d_t$ , and we compare her performance with respect to the best single action picked in hindsight, i.e.,  $\max_{d \in \mathcal{D}} \sum_{t=1}^T s_t \cdot d$ . This performance with respect to the best single action in hindsight is called (expected) *regret*, formally:

$$\mathbf{Regret}^{(T)} = \max_{d \in \mathcal{D}} \sum_{t \in [T]} s_t \cdot d - \mathbb{E} \left[ \sum_{t \in [T]} s_t \cdot d_t \right].$$

Kalai and Vempala [58] showed that even if one has only access to a linear programming oracle for  $\mathcal{D}$ , i.e., we can efficiently solve  $\max_{d \in \mathcal{D}} s \cdot d$  for any  $s \in \mathcal{S}$ , then the FPL Algorithm 9 achieves sub-linear regret, specifically  $O(\sqrt{T})$ .

In order to state the main result in [58], we need the following. We assume that the decision set  $\mathcal{D}$  has diameter at most  $D$ , i.e., for all  $d, d' \in \mathcal{D}$ ,  $\|d - d'\|_1 \leq D$ . Further, for all  $d \in \mathcal{D}$  and  $s \in \mathcal{S}$  we assume that the absolute reward is bounded by  $L$ , i.e.,  $|d \cdot s| \leq L$  and that the  $\ell_1$ -norm of the reward vectors is bounded by  $R$ , i.e., for all  $s \in \mathcal{S}$ ,  $\|s\|_1 \leq R$ .

**Theorem 15** ([58]). *Let  $s_1, \dots, s_T \in \mathcal{S}$  be a sequence of rewards. Running the FPL algorithm 9 with parameter  $\eta \leq 1$  ensures regret*

$$\mathbf{Regret}(T) \leq \eta LRT + \frac{D}{\eta}.$$

Moreover, if we choose  $\eta = \sqrt{D/LRT}$ , then  $\mathbf{Regret}(T) \leq 2\sqrt{DLRT} = O(\sqrt{T})$ .

---

**Algorithm 9** Follow-the-Perturbed-Leader (FPL), [58]

---

**Require:** Parameter  $\eta > 0$

**Ensure:** Sequence of decisions  $d_1, \dots, d_T$

1: Sample  $q \sim [0, 1/\eta]^n$ .

2: **for**  $t = 1$  to  $T$  **do**

3:     **Play**  $d_t = \operatorname{argmax}_{d \in \mathcal{D}} \left( \sum_{j=1}^{t-1} s_j + q \right)^\top d$ .

---

### 4.3.2 Online Bi-criteria Algorithm and Analysis

For any collection of monotone submodular functions  $\{f_i^t\}_{i \in [k]}$  played by the adversary, we define the *soft-min* function with respect to the corresponding multilinear extensions

$\{F_i^t\}_{i \in [k]}$  as

$$H^t(y) := -\frac{1}{\alpha} \ln \sum_{i \in [k]} e^{-\alpha F_i^t(y)},$$

where  $\alpha > 0$  is a suitable parameter. Recall we assume functions  $f_i^t$  taking values in  $[0, 1]$ , then their multilinear extensions  $F_i^t$  also take values in  $[0, 1]$ . The following properties of the soft-min function as defined above are easy to verify and crucial for our result.

1. Approximation:

$$\min_{i \in [k]} F_i^t(y) - \frac{\ln k}{\alpha} \leq H^t(y) \leq \min_{i \in [k]} F_i^t(y). \quad (4.8)$$

2. Gradient:

$$\nabla H^t(y) = \sum_{i \in [k]} p_i^t(y) \nabla F_i^t(y),$$

where  $p_i^t(y) \propto e^{-\alpha F_i^t(y)}$  for all  $i \in [k]$ .

Note that as  $\alpha$  increases, the soft-min function  $H^t$  becomes a better approximation of  $\min_{i \in [k]} \{F_i^t\}$ , however, its smoothness degrades (see Property (4.7) in Section 4.3.1). On the other hand, the second property shows that the gradient of the soft-min function is a convex combination of the gradients of the multilinear extensions, which allows us to optimize all the functions at the same time. Indeed, define  $\Delta_e H^t(y) := \sum_{i \in [k]} p_i^t(y) \Delta_e F_i^t(y) = (1 - y_e) \nabla_e H^t(y)$ . At each stage  $t \in [T]$ , we use the information from the gradients previously observed, in particular,  $\{\Delta H^1, \dots, \Delta H^{t-1}\}$  to decide the set  $S^t$ . To deal with adversarial input functions, we use the FPL algorithm [58] presented in the previous section and Theorem 15.

Our online algorithm works as follows: first, given  $0 < \epsilon < 1$  we denote  $\ell := \lceil \ln \frac{1}{\epsilon} \rceil$ . We consider the following discretization indexed by  $\tau \in \{0, \delta, 2\delta, \dots, \ell\}$  and construct fractional solutions  $x_\tau^t$  for each iteration  $t$  and discretization index  $\tau$ . At each iteration  $t$ , ideally we would like to construct  $\{x_\tau^t\}_{\tau=0}^\ell$  by running the continuous greedy algorithm using the soft-min function  $H^t$  and then play  $S^t$  using these fractional solutions. But in the online model, function  $H^t$  is revealed only after playing set  $S^t$ . To remedy this, we aim to construct  $x_\tau^t$  using FPL algorithm based on gradients  $\{\nabla H^j\}_{j=1}^{t-1}$  obtained from previous iterations. Thus we have multiple FPL instances, one for each discretization parameter, being run by the algorithm. Finally, at the end of iteration  $t$ , we have a fractional vector  $x_\ell^t$  which belongs to  $\ell \cdot \mathcal{P}(\mathcal{M}) \cap [0, 1]^V$  and therefore can be written, fractionally, as a union of  $\ell$  independent sets using the matroid union theorem [97]. We round the fractional solution  $x_\ell^t$  using the randomized swap rounding proposed in [26] for matroid  $\mathcal{M}_\ell$  to obtain the set  $S^t$  to be played at time  $t$ . Specifically, Theorem 14 gives the necessary property of the randomized swap rounding that we use. Below in Algorithm 10, we formalize the details

of our online procedure (observe that  $\ell/\delta \in \mathbb{Z}_+$ ).

---

**Algorithm 10** OnlineSoftMin Algorithm

---

**Require:** learning parameter  $\eta > 0$ ,  $\epsilon > 0$ ,  $\alpha = n^2 T^2$ , discretization  $\delta = n^{-6} T^{-3}$ , and  $\ell = \lceil \ln \frac{1}{\epsilon} \rceil$ .

**Ensure:** sequence of sets  $S_1, \dots, S_T$ .

- 1: Sample  $q \sim [0, 1/\eta]^V$
- 2: **for**  $t = 1$  to  $T$  **do**
- 3:      $y_0^t = 0$
- 4:     **for**  $\tau \in \{\delta, 2\delta, \dots, \ell\}$  **do**
- 5:         **Compute**

$$y_\tau^t = \operatorname{argmax}_{y \in \mathcal{P}(\mathcal{M})} \left[ \sum_{j=1}^{t-1} \Delta H^j(x_{\tau-\delta}^j) + q \right] \cdot y.$$

- 6:         **Update** For each  $e \in V$ ,

$$x_{\tau,e}^t = x_{\tau-\delta,e}^t + \delta [1 - x_{\tau-\delta,e}^t] \cdot y_{\tau,e}^t.$$

- 7:         **Play**  $S^t \leftarrow \text{SwapRounding}(x_\ell^t)$ . **Receive** and **observe** new collection  $\{f_i^t\}_{i \in [k]}$ .
- 

**Observation 1.** *In order to get sub-linear regret for the FPL algorithm 9, Kalai and Vempala [58] assume a couple of conditions on the problem that we outlined in Section 4.3.1. Similarly, for our online model we need to consider the following for any  $t \in [T]$ :*

1. *bounded diameter of  $\mathcal{P}(\mathcal{M})$ , i.e., for all  $y, y' \in \mathcal{P}(\mathcal{M})$ ,  $\|y - y'\|_1 \leq D$ ;*
2. *for all  $x, y \in \mathcal{P}(\mathcal{M})$ , we require  $|y \cdot \Delta H^t(x)| \leq L$ ;*
3. *for all  $y \in \mathcal{P}(\mathcal{M})$ , we require  $\|\Delta H^t(y)\|_1 \leq R$ ,*

Now, we give a complete proof of Theorem 11 for any given learning parameter  $\eta > 0$ , but the final result follows with  $\eta = \sqrt{D/LRT}$  and assuming  $L \leq n$ ,  $R \leq n$  and  $D \leq \sqrt{n}$ , which gives a  $O(n^{5/4})$  dependency on the dimension in the regret.

*Proof.* Consider the sequence of multilinear extensions  $\{F_i^1\}_{i \in [k]}, \dots, \{F_i^T\}_{i \in [k]}$  derived from the monotone submodular functions  $f_i^t$  obtained during the dynamic process. Since  $f_i^t$ 's have value in  $[0, 1]$ , we have  $0 \leq F_i^t(y) \leq 1$  for any  $y \in [0, 1]^V$  and  $i \in [k]$ . Consider

the corresponding soft-min functions  $H^t$  for collection  $\{F_i^t\}_{i \in [k]}$  with  $\alpha = n^2 T^2$  for all  $t \in [T]$ . Denote  $\ell = \lceil \ln \frac{1}{\epsilon} \rceil$  and fix  $\tau \in \{\delta, 2\delta, \dots, \ell\}$  with  $\delta = n^{-6} T^{-3}$ . According to the update in Algorithm 10,  $\{x_\tau^t\}_{t \in [T]}$  and  $\{x_{\tau-\delta}^t\}_{t \in [T]}$  satisfy conditions of Lemma 4. Thus, we obtain

$$\sum_{t \in [T]} H^t(x_\tau^t) - H^t(x_{\tau-\delta}^t) \geq \sum_{t \in [T]} \nabla H^t(x_{\tau-\delta}^t) \cdot [x_\tau^t - x_{\tau-\delta}^t] - O(Tn^3 \delta^2 \alpha).$$

Then, since the update is  $x_{\tau,e}^t = x_{\tau-\delta,e}^t + \delta[1 - x_{\tau-\delta,e}^t] \cdot y_{\tau,e}^t$ , we get

$$\begin{aligned} \sum_{t \in [T]} H^t(x_\tau^t) - H^t(x_{\tau-\delta}^t) &\geq \delta \sum_{t \in [T]} \sum_{e \in V} \nabla_e H^t(x_{\tau-\delta}^t) \cdot [1 - x_{\tau-\delta,e}^t] \cdot y_{\tau,e}^t - O(Tn^3 \delta^2 \alpha) \\ &= \delta \sum_{t \in [T]} \Delta H^t(x_{\tau-\delta}^t) \cdot y_\tau^t - O(Tn^3 \delta^2 \alpha). \end{aligned} \quad (4.9)$$

Observe that an FPL algorithm is implemented for each  $\tau$ , so we can state a regret bound for each  $\tau$  by using Theorem 15. Specifically,

$$\mathbb{E} \left[ \sum_{t \in [T]} \Delta H^t(x_{\tau-\delta}^t) \cdot z_\tau^t \right] \geq \max_{z \in \mathcal{P}(\mathcal{M})} \mathbb{E} \left[ \sum_{t \in [T]} \Delta H^t(x_{\tau-\delta}^t) \cdot z \right] - \mathcal{R}_\eta,$$

where  $\mathcal{R}_\eta = \eta LRT + \frac{D}{\eta}$  is the regret guarantee for a given  $\eta > 0$ . By taking expectation in (4.9) and using the regret bound we just mentioned, we obtain

$$\begin{aligned} \mathbb{E} \left[ \sum_{t \in [T]} H^t(x_\tau^t) - H^t(x_{\tau-\delta}^t) \right] &\geq \delta \cdot \left( \max_{y \in \mathcal{P}(\mathcal{M})} \mathbb{E} \left[ \sum_{t \in [T]} \Delta H^t(x_{\tau-\delta}^t) \cdot y \right] \right) - \delta \mathcal{R}_\eta \\ &\quad - O(Tn^3 \delta^2 \alpha) \\ &\geq \delta \cdot \mathbb{E} \left( \sum_{t \in [T]} \left[ H^t(x^*) - \sum_{i \in [k]} p_i^t(x_{\tau-\delta}^t) F_i^t(x_{\tau-\delta}^t) \right] \right) - \delta \mathcal{R}_\eta \\ &\quad - O(Tn^3 \delta^2 \alpha), \end{aligned} \quad (4.10)$$

where  $x^*$  is the true optimum for  $\max_{x \in \mathcal{P}(\mathcal{M})} \sum_{t \in [T]} \min_{i \in [k]} F_i^t(x)$ . Observe that (4.10) follows from monotonicity and submodularity of each  $f_i^t$ , specifically we know that

$$\begin{aligned} \Delta H^t(x) \cdot y &= \sum_{i \in [k]} p_i^t(x) \Delta F_i^t(x) \cdot y \geq \sum_{i \in [k]} p_i^t(x) F_i^t(x^*) - \sum_{i \in [k]} p_i^t(x) F_i^t(x) \quad (\text{eq. (1.5)}) \\ &\geq F_{\min}^t(x^*) - \sum_{i \in [k]} p_i^t(x) F_i^t(x) \\ &\geq H^t(x^*) - \sum_{i \in [k]} p_i^t(x) F_i^t(x). \end{aligned}$$

By applying property (4.7) of the soft-min in expression (4.10) we get

$$\begin{aligned} \mathbb{E} \left[ \sum_{t \in [T]} H^t(x_\tau^t) - H^t(x_{\tau-\delta}^t) \right] &\geq \delta \cdot \mathbb{E} \left( \sum_{t \in [T]} H^t(x^*) - H^t(x_{\tau-\delta}^t) \right) - \delta \mathcal{R}_\eta - O(Tn^3\delta^2\alpha) \\ &\quad - \delta \cdot T \cdot \left( \frac{n + \ln T}{\alpha} - \frac{\ln k}{\alpha} - \frac{ke^{-n}}{T} \right), \quad (4.11) \end{aligned}$$

Given the choice of  $\alpha$  and  $\delta$ , the last two terms in the right-hand side of inequality (4.11) are small compared to  $\mathcal{R}_\eta$ , so by re-arranging terms we can state the following

$$\sum_{t \in [T]} H^t(x^*) - \mathbb{E} \left[ \sum_{t \in [T]} H^t(x_\tau^t) \right] \leq (1 - \delta) \cdot \left( \sum_{t \in [T]} H^t(x^*) - \mathbb{E} \left[ \sum_{t \in [T]} H^t(x_{\tau-\delta}^t) \right] \right) + 2\delta \cdot \mathcal{R}_\eta$$

By iterating  $\frac{\ell}{\delta}$  times in  $\tau$ , we get

$$\begin{aligned} \sum_{t \in [T]} H^t(x^*) - \mathbb{E} \left[ \sum_{t \in [T]} H^t(x_\ell^t) \right] &\leq (1 - \delta)^{\frac{\ell}{\delta}} \cdot \left( \sum_{t \in [T]} H^t(x^*) - \sum_{t \in [T]} H^t(x_0^t) \right) \\ &\quad + O \left( \mathcal{R}_\eta \ln \frac{1}{\epsilon} \right) \\ &\leq \epsilon \cdot \left[ \sum_{t \in [T]} H^t(x^*) + \frac{\ln k}{n^2 T} \right] + O \left( \mathcal{R}_\eta \ln \frac{1}{\epsilon} \right), \end{aligned}$$

where in the last inequality we used  $(1 - \delta) \leq e^{-\delta}$ . Given that the term  $\epsilon \cdot \frac{\ln(k)}{n^2 T}$  is small (for

$T$  and  $n$  sufficiently large) we can bound it by  $O(\mathcal{R}_\eta \ln \frac{1}{\epsilon})$ . Since  $\alpha$  is sufficiently large, we can apply the approximation property of soft-min function to obtain the following regret bound

$$(1 - \epsilon) \cdot \sum_{t \in [T]} \min_{i \in [k]} F_i^t(x^*) - \mathbb{E} \left[ \sum_{t \in [T]} \min_{i \in [k]} F_i^t(x_\ell^t) \right] \leq O \left( \mathcal{R}_\eta \ln \frac{1}{\epsilon} \right).$$

Since we are doing randomized swap rounding on each  $x_\ell^t$ , Theorem 14 shows that there is a random set  $S^t$  that is independent in  $\mathcal{M}_\ell$  (i.e.,  $S^t$  is the union of at most  $\ell$  independent sets in  $\mathcal{I}$ ) such that  $\mathbb{E}[f_i^t(S^t)] \geq F_i^t(y_\ell^t)$  for all  $t \in [T]$  and  $i \in [k]$ . Thus, we finally obtain

$$(1 - \epsilon) \cdot \max_{S \in \mathcal{I}} \sum_{t \in [T]} \min_{i \in [k]} f_i^t(S) - \sum_{t \in [T]} \min_{i \in [k]} \mathbb{E}[f_i^t(S^t)] \leq O \left( \mathcal{R}_\eta \ln \frac{1}{\epsilon} \right).$$

□

**Observation 2.** *Theorem 11 could be easily extended to an adaptive adversary by sampling in each stage  $t \in [T]$  a different perturbation  $q_t \sim [0, 1/\eta]^V$  in Algorithm 9 as shown in [58].*

## 4.4 Extensions and Other Results

In this section, we provide a hardness result for the offline problem (4.1) when the monotonicity of the functions is not assumed and provable guarantees for several different classes of constraints.

### 4.4.1 Necessity of Monotonicity

In light of the approximation algorithms for non-monotone submodular function maximization under matroid constraints (see, for example, [74]), one might hope that an analogous bi-criteria approximation algorithm could exist for robust non-monotone submodular function maximization. However, we show that even without any matroid constraints, getting

any approximation in the non-monotone case is NP-hard.

**Lemma 5.** *Unless  $P = NP$ , no polynomial time algorithm can output a set  $\tilde{S} \subseteq V$  given general submodular functions  $f_1, \dots, f_k$  such that  $\min_{i \in [k]} f_i(\tilde{S})$  is within a positive factor of  $\max_{S \subseteq V} \min_{i \in [k]} f_i(S)$ .*

*Proof.* We use a reduction from SAT. Suppose that we have a SAT instance with variables  $x_1, \dots, x_n$ . Consider  $V = \{1, \dots, n\}$ . For every clause in the SAT instance we introduce a nonnegative linear (and therefore submodular) function. For a clause  $\bigvee_{i \in A} x_i \vee \bigvee_{i \in B} \bar{x}_i$  define

$$f(S) := |S \cap A| + |B \setminus S|.$$

It is easy to see that  $f$  is linear and nonnegative. If we let  $S$  be the set of true variables in a truth assignment, then it is easy to see that  $f(S) > 0$  if and only if the corresponding clause is satisfied. Consequently, finding a set  $S$  such that all functions  $f$  corresponding to different clauses are positive is as hard as finding a satisfying assignment for the SAT instance.  $\square$

#### 4.4.2 Knapsack Constraints

Consider a knapsack constraint  $\mathcal{K} = \{S \subseteq [n] : \sum_{e \in S} c_e \leq 1\}$ , where  $c_e > 0$  for all  $e \in [n]$ . Our interest is to solve the following robust problem

$$\max_{S \in \mathcal{K}} \min_{i \in [k]} f_i(S) \tag{4.12}$$

**Corollary 6.** *For Problem (4.12), there is a polynomial time algorithm that returns a set  $S^{\text{ALG}}$ , such that for all  $i \in [k]$ , for a given  $0 < \epsilon < 1$ ,*

$$f_i(S^{\text{ALG}}) \geq (1 - \epsilon) \cdot \max_{S \in \mathcal{K}} \min_{j \in [k]} f_j(S),$$

*and  $\sum_{e \in S^{\text{ALG}}} c_e \leq \ell$  for  $\ell = O(\ln \frac{k}{\epsilon})$ . Moreover,  $S^{\text{ALG}}$  can be covered by at most  $\ell$  sets in*

$\mathcal{K}$ .

Instead of using the standard greedy for every  $\tau = \{1, \dots, \ell\}$ , we design an extended version of the “bang-per-buck” greedy algorithm. We formalize this procedure in Algorithm 11 below. Even though the standard “bang-per-buck” greedy algorithm does not provide any approximation factor, if we relax the knapsack constraint to be  $\sum_{e \in S} c_e \leq 2$ , then the algorithm gives a  $1 - 1/e$  factor. There are other approaches to avoid this relaxation, see e.g. [102].

---

**Algorithm 11** Extended “Bang-per-Buck” Algorithm for Knapsack Constraints

---

**Require:**  $\ell \geq 1$ , monotone submodular function  $g : 2^V \rightarrow \mathbb{R}_+$ , knapsack constraint  $\mathcal{K}$ .

**Ensure:** sets  $S_1, \dots, S_\ell \in \mathcal{K}$ .

- 1: **for**  $\tau = 1, \dots, \ell$  **do**
- 2:      $S_\tau \leftarrow \emptyset$
- 3:     **while**  $V \neq \emptyset$  **do**
- 4:         **Compute**

$$e^* = \operatorname{argmax}_{e \in V} \frac{g(\cup_{j=1}^{\tau} S_j + e) - g(\cup_{j=1}^{\tau} S_j)}{c_e}.$$

- 5:         **if**  $\sum_{e \in S_\tau} c_e + c_{e^*} \leq 2$  **then**
  - 6:              $S_\tau \leftarrow S_\tau + e^*$ .
  - 7:              $V \leftarrow V - e^*$
  - 8:         **Restart** ground set  $V$ .
- 

Given a monotone submodular function  $g : 2^V \rightarrow \mathbb{R}_+$ , Algorithm 11 produces a set  $S^{\text{ALG}} = S_1 \cup \dots \cup S_\ell$  such that  $g(S^{\text{ALG}}) \geq (1 - \frac{1}{e}) \cdot \max_{S \in \mathcal{K}} g(S)$ . Therefore, Corollary 6 can be easily proved by defining  $g$  in the same way as in Theorem 10, and running Algorithm 11 on  $g$  with  $\ell = O(\ln \frac{k}{\epsilon})$ .

#### 4.4.3 Multiple Matroid Constraints

Consider a family of  $r$  matroids  $\mathcal{M}_j = (V, \mathcal{I}_j)$  for  $j \in [r]$ . Our interest is to solve the following robust problem

$$\max_{S \in \bigcap_{j=1}^r \mathcal{I}_j} \min_{i \in [k]} f_i(S) \tag{4.13}$$

**Corollary 7.** *For Problem (4.13), there is a polynomial time algorithm that returns a set  $S^{\text{ALG}}$ , such that for all  $i \in [k]$ , for a given  $0 < \epsilon < 1$ ,*

$$f_i(S^{\text{ALG}}) \geq (1 - \epsilon) \cdot \max_{S \in \bigcap_{j=1}^r \mathcal{I}_j} \min_{i \in [k]} f_i(S),$$

where  $S^{\text{ALG}}$  is the union of  $O(\log \frac{k}{\epsilon} / \log \frac{r+1}{r})$  independent sets in  $\mathcal{I}$ .

The standard greedy algorithm gives a  $1/(1+r)$  approximation for problem (4.13) when  $k = 1$  [42]. Therefore, we can adapt Algorithm 7 to produce a set  $S^{\text{ALG}} = S_1 \cup \dots \cup S_\ell$  such that

$$f(S^{\text{ALG}}) \geq \left(1 - \left(\frac{r}{r+1}\right)^\ell\right) \cdot \max_{S \in \bigcap_{j=1}^r \mathcal{I}_j} f(S).$$

Then, Corollary 7 can be proved similarly to Theorem 10 by choosing  $\ell = O(\log \frac{k}{\epsilon} / \log \frac{r+1}{r})$

#### 4.4.4 Distributionally Robust over Polyhedral Sets

Let  $\mathcal{Q} \subseteq \Delta(k)$  be a polyhedral set, where  $\Delta(k)$  is the probability simplex on  $k$  elements. For  $q \in \mathcal{Q}$ , denote  $f_q := q_1 f_1 + \dots + q_k f_k$ , which is also monotone and submodular. Given a matroid  $\mathcal{M} = (V, \mathcal{I})$ , our interest is to solve the following distributionally robust problem

$$\max_{S \in \mathcal{I}} \min_{q \in \mathcal{Q}} f_q(S) \tag{4.14}$$

Denote by  $\text{Vert}(\mathcal{Q})$  the set of extreme points of  $\mathcal{Q}$ , which is finite since  $\mathcal{Q}$  is polyhedral. Then, problem (4.14) is equivalent to  $\max_{S \in \mathcal{I}} \min_{q \in \text{Vert}(\mathcal{Q})} f_q(S)$ . Then, we can easily derive Corollary 8 below by applying Theorem 10 in the equivalent problem. Note that when  $\mathcal{Q}$  is the simplex we get the original Theorem 10.

**Corollary 8.** *For Problem (4.14), there is a polynomial time algorithm that returns a set  $S^{\text{ALG}}$ , such that for all  $i \in [k]$ , for a given  $0 < \epsilon < 1$ ,*

$$f_i(S^{\text{ALG}}) \geq (1 - \epsilon) \cdot \max_{S \in \mathcal{I}} \min_{q \in \mathcal{Q}} f_q(S),$$

with  $S^{\text{ALG}} = S_1 \cup \dots \cup S_\ell$  for  $\ell = O(\log \frac{|\text{Vert}(\mathcal{Q})|}{\epsilon})$  and  $S_1, \dots, S_\ell \in \mathcal{I}$ .

## 4.5 Concluding Remarks

In this chapter, we studied the problem of robust submodular maximization with structured combinatorial constraints. In the robust variant of submodular optimization, rather than optimizing a single submodular function, one has to simultaneously optimize (the minimum of)  $k$  sub-modular functions. We studied the online and offline variants of this problem and provided bi-criteria algorithms that work in presence of commonly used combinatorial constraints set, such as single or multiple matroids and knapsack.

Our results include nearly tight bi-criteria approximation algorithms. In the offline case, we presented an algorithm that obtains a  $(1 - \epsilon)$  fraction of the solution value of the optimal independent set using a solution that is the union of  $O(\log(k/\epsilon))$  independent sets. In the online case, in expectation our algorithm obtains a  $(1 - \epsilon)$  fraction of the solution value of the optimal independent set in hindsight using solutions that are unions of  $O(\ln(1/\epsilon))$  independent sets. A benefit of such bi-criteria characterization of the guarantees of our algorithms is that a user can determine the tradeoff between the quality of the solution and the size of the solution for a given applications — based on the importance of the quality of the solution and the scarcity of resources in any given application.

We also presented efficient bi-criteria algorithms for the offline version of the problem. For this, we used an extended version of the threshold greedy algorithm which requires less functional calls at cost of a small error in the approximation guarantee. In the bi-criteria procedure this translates in a slightly larger output solution. We also showed that by using several implementation improvements, such as lazy evaluations and an early stopping criterion, we can drastically reduce the computational cost of our algorithms. We support our theoretical results with three applications in non-parametric learning, image clustering, and sensor location. Even though our efficient algorithms perform well in instances of 20,000 elements, we still have a gap to close in terms of efficiency with larger data-sets.

## CHAPTER 5

### THE SHARPNESS CRITERIA IN SUBMODULAR MAXIMIZATION

#### 5.1 Introduction

During the past decades, the interest in constrained submodular maximization has increased significantly, especially due to its numerous applications in real-world problems. To illustrate the submodular property, consider a simple example of selecting the most informative subset of patients in a group of people with certain illness. Submodularity measures the decreasing marginal gain on the information captured from medical tests when selecting more patients [68]. In this chapter, we study the problem of maximizing a monotone submodular function subject to a single cardinality constraint. It is well-known that  $1 - 1/e$  is the best possible approximation guarantee [86] for this problem. However, empirical observations have shown that standard algorithms such as the greedy algorithm performs considerably better in practice. The worst-case instances presented in [86] are rare, while real-world data instead provides considerably much *tractable* objective functions. We focus on giving a candidate explanation to those instances in which the optimal solution clearly stands out over the rest of feasible solutions. For this, we consider the concept of *sharpness* initially presented in continuous optimization [78] and we adapt it to the submodular case. Roughly speaking, this property measures the behavior of the objective function around the set of optimal solutions. Sharpness in continuous optimization translates in faster convergence rates. Equivalently, we will show that in submodular maximization the greedy algorithm performs better as the sharpness of the objective function increases.

### 5.1.1 Problem Formulation

Consider a ground set of elements  $V = \{1, \dots, n\}$ , a non-negative monotone submodular function  $f : 2^V \rightarrow \mathbb{R}_+$ , and a budget  $K \in \mathbb{Z}_+$ . We are interested in the following submodular maximization problem

$$\max\{f(S) : |S| \leq K\}. \quad (5.1)$$

Our objective is to show that the Greedy Algorithm 1 achieves better approximation guarantees when the sharpness of the function  $f$  increases. We will see that this algorithm automatically adapts to the sharpness of the function without requiring any extra input.

Given parameters  $c \geq 1$  and  $\theta \in [0, 1]$ , we define *submodular sharpness* as follows

**Definition 4** (Submodular Sharpness). *A non-negative monotone submodular function  $f : 2^V \rightarrow \mathbb{R}_+$  is said to be  $(c, \theta)$ -sharp, if there exists an optimal solution  $S^*$  for Problem (5.1) such that for any subset  $S \subseteq V$  with  $|S| \leq K$  the function satisfies*

$$\sum_{e \in S^* \setminus S} f_S(e) \geq \left[ \frac{|S^* \setminus S|}{K \cdot c} \right]^{\frac{1}{\theta}} \cdot f(S^*)$$

**Example (linear functions).** To clarify the concept of sharpness let us consider a simple example with a linear objective; even though it is well-known that in this case the greedy algorithm outputs an optimal solution. Due to submodularity, the left-hand side in Definition 4 is a monotone decreasing function on  $S$ , so we only need to check those subsets of size exactly  $K$ . Consider non-negative weights  $w_e$  for each element  $e \in V$  and  $f(S) = \sum_{e \in S} w_e$ . Let us show that  $f$  is at least  $(\frac{1}{Kf(e^*)}, 1)$ -sharp where  $f(e^*) = \min_{e \in S^*} f(e)$ . By scaling arguments, w.l.o.g. we can assume that  $f(S^*) = 1$  and  $f(e) > 0$  for all  $e \in S^*$ . Let us consider any  $e \in S^*, e' \notin S^*$  and  $S = S^* - e + e'$  in Definition 4, then we obtain the

following constraint:

$$w_e \geq \left(\frac{1}{Kc}\right)^{\frac{1}{\theta}}, \forall e \in S^* \Leftrightarrow \min_{e \in S^*} w_e \geq \left(\frac{1}{Kc}\right)^{1/\theta}$$

Observe that since  $f(S^*) = 1$ , then either: (1)  $w_e = 1/K$  for all  $e \in S^*$  or (2) there exists an element  $e^* \in S^*$  such that  $w_{e^*} < 1/K$ . In the first case, the inequality is trivially satisfied with  $c = 1$  and  $\theta = 1$ . In the second case,  $c = 1$  and  $\theta = 1$  does not work anymore. Let us fix  $\theta = 1$  and assume  $w_{e^*} = \min_{e \in S^*} w_e$ , so we need  $c \geq \frac{1/K}{f(e^*)} > 1$ . Now, let us make the same analysis for a pair of elements: consider any  $e_1, e_2 \in S^*$  and any  $e'_1, e'_2 \notin S^*$ . Then, by considering  $S = S^* - \{e_1, e_2\} + \{e'_1, e'_2\}$  in Definition 4, we obtain

$$\min_{e_1, e_2 \in S^*} \{w_{e_1} + w_{e_2}\} \geq \left(\frac{2}{Kc}\right)^{1/\theta}$$

Now, let us replace  $c = \frac{1/K}{w_{e^*}}$  and  $\theta = 1$  obtained in the singleton analysis. We get

$$\min_{e_1, e_2 \in S^*} \{w_{e_1} + w_{e_2}\} \geq 2w_{e^*},$$

which is true given the definition of  $e^*$ . This procedure can be done for any subset of  $S^*$ , showing the desired result. However, this only shows that the linear function is at least  $(\frac{1}{Kf(e^*)}, 1)$ -sharp, which are not necessarily the tightest parameters for the corresponding function.

Observe that any monotone submodular function is  $(c, \theta)$ -sharp for some set of parameters  $c$  and  $\theta$ . In particular we have the following results

**Lemma 6.** *Consider a monotone submodular function  $f$ . Then:*

1.  $f$  is  $(c, \theta)$ -sharp for parameters  $c$  and  $\theta$  such that  $(1/c)^{1/\theta} \rightarrow 0$ .
2. If  $f$  is  $(c, \theta)$ -sharp, then is also  $(c', \theta')$ -sharp for  $c' \geq c$  or  $\theta' \leq \theta$ .

*Proof.* 1. Note that  $\frac{|S^* \setminus S|}{K} \leq 1$ , so  $\left[\frac{|S^* \setminus S|}{K \cdot c}\right]^{\frac{1}{\theta}} \leq \left[\frac{1}{c}\right]^{\frac{1}{\theta}}$ , which shows that  $\left[\frac{|S^* \setminus S|}{K \cdot c}\right]^{\frac{1}{\theta}} \rightarrow 0$ .

Therefore, Definition 4 is simply  $\sum_{e \in S^* \setminus S} f_S(e) \geq 0$ , which  $f$  satisfies since from monotonicity we have  $f_S(e) \geq 0$ .

2. Define  $L(c, \theta) := \left[\frac{|S^* \setminus S|}{K \cdot c}\right]^{\frac{1}{\theta}}$  for  $c \geq 1$  and  $\theta \in [0, 1]$ . Observe that  $L$  is increasing in  $\theta$  and decreasing in  $c$ . Therefore,  $L(c, \theta) \geq L(c', \theta')$  for  $c' \geq c$  and  $\theta' \leq \theta$ .

□

Throughout this chapter, when we say a function is  $(c, \theta)$ -sharp, we refer to the tightest set of parameters  $c$  and  $\theta$ , which actually provides the best approximation factor for the given function. Observe that our definition only needs the existence of an optimal solution satisfying the inequality, which is weaker than requiring the inequality for all optimal solutions. Finally, Definition 4 is merely descriptive since parameters  $c$  and  $\theta$  are intrinsic to the function itself and cannot not be explicitly obtained when  $n$  is large. Even though, we will show that the standard greedy algorithm still adapts without requiring the sharpness parameters as part of the input.

Definition 4 can be considered as a *static* notion of sharpness, since parameters  $c$  and  $\theta$  do not change with respect to  $S$ . We generalize this definition by considering the notion of *dynamic sharpness*, in which the parameters  $c$  and  $\theta$  depend on the size of the feasible sets, i.e.,  $c_{|S|} \geq 1$  and  $\theta_{|S|} \in [0, 1]$ . Formally, we define dynamic sharpness as follows

**Definition 5** (Dynamic Submodular Sharpness). *A non-negative monotone submodular function  $f : 2^V \rightarrow \mathbb{R}_+$  is said to be dynamic  $(c, \theta)$ -sharp, where  $c \in [1, \infty)^K$  and  $\theta \in [0, 1]^K$ , if there exists an optimal solution  $S^*$  for Problem (5.1) such that for any subset  $S \subseteq V$  with  $|S| \leq K$  the function satisfies*

$$\sum_{e \in S^* \setminus S} f_S(e) \geq \left[\frac{|S^* \setminus S|}{K \cdot c_{|S|}}\right]^{\frac{1}{\theta_{|S|}}} \cdot f(S^*)$$

### 5.1.2 Main Contributions

To the best of our knowledge, this is the first study to define sharpness in submodular optimization. More importantly, we show that the classical greedy algorithm achieves better provable guarantees as the sharpness of the objective function increases. Even more, this algorithm automatically adapts to the sharpness of the objective function without requiring the sharpness parameters as part of the input. Specifically, we obtain the following result for the problem of maximizing a monotone submodular function subject to a single cardinality constraint.

**Theorem 16.** *Consider a non-negative monotone submodular function  $f : 2^V \rightarrow \mathbb{R}_+$  which is  $(c, \theta)$ -sharp. Then, for Problem (5.1) the greedy algorithm returns a feasible set  $S^g$  such that*

$$f(S^g) \geq \left[ 1 - \left( 1 - \frac{\theta}{c} \right)^{\frac{1}{\theta}} \right] \cdot f(S^*).$$

Observe that the greedy algorithm recovers an optimal solution when  $c = 1$  and  $\theta = 1$ .

**Corollary 9.** *Every non-negative monotone submodular function satisfies Definition 4 when  $c \rightarrow 1$  and  $\theta \rightarrow 0$ . In this case, we recover the standard approximation factor  $1 - 1/e$ .*

For the notion of dynamic sharpness we consider a set of parameters that has only one switching moment  $t_0 \in \{0, \dots, K - 1\}$ . For this case, we obtain the following result

**Theorem 17.** *Consider a non-negative monotone submodular function  $f : 2^V \rightarrow \mathbb{R}_+$  that is dynamic  $(c, \theta)$ -sharp with dynamic parameters  $c_{|S|} \geq 1$  and  $\theta_{|S|} \in [0, 1]$  such that:*

$$c_{|S|} := \begin{cases} c_0 & \text{if } |S| < t_0 \\ c_1 & \text{if } |S| \geq t_0 \end{cases}, \quad \theta_{|S|} := \begin{cases} \theta_0 & \text{if } |S| < t_0 \\ \theta_1 & \text{if } |S| \geq t_0 \end{cases},$$

where  $t_0 \in \{0, \dots, K - 1\}$  is the moment when the function  $f$  changes its sharpness. Then,

for Problem (5.1) the greedy algorithm outputs a set  $S^g$  such that

$$f(S^g) \geq \left[ 1 - \left( \left( 1 - \frac{\theta_0 t_0}{c_0 K} \right)^{\frac{\theta_1}{\theta_0}} + \frac{\theta_1 t_0}{c_1 K} - \frac{\theta_1}{c_1} \right)^{\frac{1}{\theta_1}} \right] \cdot f(S^*).$$

Observe that we recover Theorem 16 when  $t_0 = 0$ . This result can be easily generalized to several switching moments.

In Section 5.2.2, we define *approximate sharpness* as a generalization of Definition 4 in which the marginal values are slightly perturbed. Moreover, we show that the standard greedy algorithm achieves provable guarantees under this alternative condition.

Finally, in Section 5.3 we provide an exhaustive computational study in real-world applications such as non-parametric learning, clustering and sensor placement. In this section, our objective is to contrast our results with the existing literature, such as the concepts of *curvature* [30] and *stability* [25] (for a detailed definition we refer to Section 5.1.3). With these experiments, we show that the curvature and stability analysis are in some cases not satisfactory enough to explain the performance of the greedy algorithm, whereas the sharpness criterion stands as a better candidate. For example, consider the facility-location function  $f(S) = \sum_{i \in V} \max_{j \in S} w_{ij}$  with  $w_{ij} \geq 0$ , which is monotone and submodular. We will see that the curvature analysis does not improve beyond  $1 - 1/e$  and the instance is non-stable. In this case, the sharpness criterion gives a better approximation guarantees.

### 5.1.3 Related Work

It is well known that even for Problem (5.1), one requires an exponential number of evaluations in order to improve beyond  $1 - 1/e$  [86]. Even when the decision-maker has explicit access to the objective function, this factor is the best possible, unless  $P = NP$  [38].

**Curvature.** In order to get better approximation ratios the research has focused on subclasses of submodular functions. Conforti and Cornuéjols [30] introduced the concept of

*curvature* which measures how far the function is from being linear. Formally, a monotone submodular function  $f : 2^V \rightarrow \mathbb{R}_+$  has *total curvature*  $\gamma \in [0, 1]$  if

$$\gamma = 1 - \min_{e \in V^*} \frac{f_{V-e}(e)}{f(e)}, \quad (5.2)$$

where  $V^* = \{e \in V : f(e) > 0\}$ . First note that this property implies that for any  $S \subseteq V$  and  $e \notin S$ ,  $f_S(e) \geq (1 - \gamma)f(e)$ . Thus, a submodular function with a small  $\gamma$  ensures that marginal values do not decrease significantly. Recall that any submodular function satisfies  $f_S(e) \leq f(e)$ . Moreover, when  $\gamma = 0$  marginals are constant so the function is linear, and when  $\gamma \rightarrow 1$  then  $f$  is an arbitrary monotone submodular function. When a submodular function has total curvature  $\gamma \in [0, 1]$ , Conforti and Cornuéjols [30] showed that the greedy algorithm guarantees an approximation factor of  $(1 - e^{-\gamma})/\gamma$  for a single cardinality constraint and  $1/(1 + \gamma)$  for a single matroid constraint. Observe that results given in [87, 42] are recovered when  $\gamma \rightarrow 1$ . Later, Vondrák [109] proved that the continuous greedy algorithm achieves  $(1 - e^{-\gamma})/\gamma$  for any matroid. In the same work, the author defines the concept of *curvature with respect to a set* as follows: given a fixed set  $S \subseteq V$ , a submodular function  $f : 2^V \rightarrow \mathbb{R}_+$  has curvature  $\gamma_S$  with respect to  $S$ , if  $\gamma_S \in [0, 1]$  is the smallest value such that for any  $T \subseteq V$

$$f(S \cup T) - f(S) + \sum_{e \in S \cap T} f_{S \cup T - e}(e) \geq (1 - \gamma_S)f(T). \quad (5.3)$$

Note that this notion is weaker than the total curvature defined in [30], since a function with total curvature  $\gamma$  has curvature  $\gamma_S = \gamma$  with respect to any set  $S \subseteq V$ . The author finally shows that when a monotone submodular function has curvature  $\gamma$  with respect to the optimal solution then  $(1 - e^{-\gamma})/\gamma$  is the best possible for a single matroid constraint. This sequence of results finishes with the work of Sviridenko et al. [103]. The authors provide a *modified continuous greedy* algorithm to obtain approximate solutions for the problem of maximizing the sum of a monotone submodular function and a linear function.

This algorithm consists of two steps: guessing the value of the linear function in the optimal solution and then run the continuous greedy algorithm in a smaller feasible region determined by this guess. Sviridenko et al. [103] show that for a monotone submodular function with total curvature  $\gamma \in [0, 1]$ , this algorithm achieves an approximation factor of  $1 - \gamma/e$  for a single matroid. Moreover, in the same work they show that this result is the best possible under the definition given in [30]. One drawback of the *modified continuous greedy* is the guessing step since it may affect the running time, but recently Feldman [40] presented a guess-free algorithm which ensures the same guarantee. The notion of curvature has been also used when minimizing submodular functions [56], and the equivalent notion of *steepness* in supermodular function minimization [55], we refer the interested reader to the literature therein for more details.

**Stability.** Close to our setting is the concept of stability widely studied in discrete optimization. Broadly speaking, there are instances in which the unique optimal solution still remains unique even if the objective function is slightly perturbed. For instance, the concept of *clusterability* has been widely studied in order to show the existence of *easy instances* in clustering [7, 31]; we refer the interested reader to the survey [10] for more details. Stability has been also studied in other contexts such as influence maximization [53], Nash equilibria [8], and Max-Cut [12]. Building on the work of Bilu and Linial [12], Chatziafratis et al. [25] study the concept of stability under multiplicative perturbations in submodular maximization. Formally, given a non-negative monotone submodular function  $f$ ,  $\tilde{f}$  is a  $\gamma$ -perturbation if: (1)  $\tilde{f}$  is non-negative monotone submodular, (2)  $f \leq \tilde{f} \leq \gamma \cdot f$ , and (3) for any  $S \subseteq V$  and  $e \in V \setminus S$ ,  $0 \leq \tilde{f}_S(e) - f_S(e) \leq (\gamma - 1) \cdot f(e)$ . Now, assume we have an instance of Problem (1.2) with a unique optimal solution, then this instance is said to be  $\gamma$ -stable if for any  $\gamma$ -perturbation of the objective function, the original optimal solution remains being unique. For matroid constraints, Chatziafratis et al. [25] show that the greedy algorithm recovers the unique optimal solution for 2-stable instances. However,

it is not hard to show that 2-stability for Problem (1.2) is a strong assumption, since 2-stable instances can be easily solved by maximizing the sum of singleton values over the set of basis of the matroid.

**Sharpness in continuous optimization.** The concept of *sharpness* has been mainly studied in continuous optimization. It is also known as Hölderian error bound on the distance to the set of optimal solutions [54, 78, 90, 77, 15]. Broadly speaking, this property characterizes the behavior of a function around the set of optimal solutions. Formally, if  $X^* \subseteq X$  is the set of optimal solutions in a universe of feasible points  $X$  and  $d(\cdot, X^*) : X \rightarrow \mathbb{R}_+$  is some distance function, then a function  $f$  is said to be  $(c, \theta)$ -sharp if for any  $x \in X$

$$d(x, X^*) \leq c(f^* - f(x))^\theta, \quad (5.4)$$

where  $f^*$  is the optimal objective value of a maximization problem. Sharpness has been widely used to study convergence rates in convex and non-convex optimization, see e.g. [88, 60, 16, 96, 65]. For a detailed review on the sharpness condition in continuous optimization, we refer the interested reader to [96].

## 5.2 Submodular Sharpness: Cardinality Constraints

In this section, we focus on the analysis of the standard greedy algorithm for problem (1.1) when the objective function is  $(c, \theta)$ -sharp. We emphasize that the greedy algorithm does not require access to the sharpness parameters in order to obtain the desired guarantees.

We are ready to prove Theorem 16. Recall that given parameters  $c \geq 1$  and  $\theta \in [0, 1]$ , a function is  $(c, \theta)$ -sharp if there exists an optimal set  $S^*$  such that for any set  $S$  with at most  $K$  elements, then

$$\sum_{e \in S^* \setminus S} f_S(e) \geq \left[ \frac{|S^* \setminus S|}{K \cdot c} \right]^{\frac{1}{\theta}} \cdot f(S^*)$$

*Proof of Theorem 16.* Let us denote by  $S_i$  the set we obtained in the  $i$ -th iteration of Algo-

rithm 1. Note that  $S^g := S_K$ . By using the properties of the function  $f$ , we can obtain the following sequence of inequalities

$$\begin{aligned} f(S_i) - f(S_{i-1}) &= \frac{\sum_{e \in S^* \setminus S_{i-1}} [f(S_i) - f(S_{i-1})]}{|S^* \setminus S_{i-1}|} & (5.5) \\ &\geq \frac{\sum_{e \in S^* \setminus S_{i-1}} f_{S_{i-1}}(e)}{|S^* \setminus S_{i-1}|} & (\text{choice of greedy}) \end{aligned}$$

From the sharpness condition we know that

$$\frac{1}{|S^* \setminus S_{i-1}|} \geq \frac{f(S^*)^\theta}{Kc} \cdot \left( \sum_{e \in S^* \setminus S_{i-1}} f_{S_{i-1}}(e) \right)^{-\theta}$$

so we obtain the following bound

$$\begin{aligned} f(S_i) - f(S_{i-1}) &\geq \frac{f(S^*)^\theta}{Kc} \cdot \left( \sum_{e \in S^* \setminus S_{i-1}} f_{S_{i-1}}(e) \right)^{1-\theta} & (\text{sharpness}) \\ &\geq \frac{f(S^*)^\theta}{Kc} \cdot [f(S_{i-1} \cup S^*) - f(S_{i-1})]^{1-\theta} & (\text{submodularity}) \\ &\geq \frac{f(S^*)^\theta}{Kc} \cdot [f(S^*) - f(S_{i-1})]^{1-\theta}. & (\text{monotonicity}) \end{aligned}$$

Therefore, we need to solve the following recurrence

$$a_i \geq a_{i-1} + \frac{a^\theta}{kc} \cdot [a - a_{i-1}]^{1-\theta} \quad (5.6)$$

where  $a_i = f(S_i)$ ,  $a_0 = 0$  and  $a = f(S^*)$ . Define  $h(x) = x + \frac{a^\theta}{kc} \cdot [a - x]^{1-\theta}$ , where  $x \in [0, a]$ . Observe that  $h'(x) = 1 - \frac{a^\theta(1-\theta)}{kc} \cdot [a - x]^{-\theta}$ . Therefore,  $h$  is increasing inside the interval  $I := \left\{ x : 0 \leq x \leq a \cdot \left( 1 - \left( \frac{1-\theta}{kc} \right)^{1/\theta} \right) \right\}$ . Let us define

$$b_i := a \cdot \left[ 1 - \left( 1 - \frac{\theta}{c} \cdot \frac{i}{K} \right)^{\frac{1}{\theta}} \right].$$

First, let us check that  $b_i \in I$  for all  $i \in \{0, \dots, K\}$ . Namely, for any  $i$  we need to show that

$$a \cdot \left[ 1 - \left( 1 - \frac{\theta}{c} \cdot \frac{i}{K} \right)^{\frac{1}{\theta}} \right] \leq a \cdot \left( 1 - \left( \frac{1-\theta}{Kc} \right)^{1/\theta} \right) \Leftrightarrow (Kc - i\theta) \geq 1 - \theta$$

The expression  $Kc - i\theta$  is decreasing on  $i$ . Hence, we just need the inequality for  $i = k$ , namely  $K(c - \theta) \geq 1 - \theta$ , which is true since  $c \geq 1$  and  $k \geq 1$ .

Our goal is to prove that  $a_i \geq b_i$ , so by induction let us assume that  $a_{i-1} \geq b_{i-1}$  is true. By using monotonicity of  $h$  on the interval  $I$ , we get  $h(a_{i-1}) \geq h(b_{i-1})$ . Also, observe that recurrence (5.6) is equivalent to write  $a_i \geq h(a_{i-1})$  which implies that  $a_i \geq h(b_{i-1})$ . To finish the proof we will show that  $h(b_{i-1}) \geq b_i$ .

Assume for simplicity that  $a = 1$ . For  $x \in [0, K]$ , define

$$g(x) := \left( 1 - \frac{\theta}{Kc} \cdot x \right)^{1/\theta}.$$

Note that  $g'(x) = -\frac{1}{Kc} \cdot g(x)^{1-\theta}$  and  $g''(x) = \frac{1-\theta}{(Kc)^2} \cdot g(x)^{1-2\theta}$ . Observe that  $g$  is convex, so for any  $x_1, x_2 \in [0, K]$  we have  $g(x_2) \geq g(x_1) + g'(x_1) \cdot (x_2 - x_1)$ . By considering  $x_2 = i$  and  $x_1 = i - 1$ , we obtain

$$g(i) - g(i-1) - g'(i-1) \geq 0 \tag{5.7}$$

On the other hand,

$$\begin{aligned} h(b_{i-1}) - b_i &= 1 - \left( 1 - \frac{\theta}{c} \cdot \frac{i-1}{K} \right)^{\frac{1}{\theta}} + \frac{1}{Kc} \cdot \left( 1 - \frac{\theta}{c} \cdot \frac{i-1}{K} \right)^{\frac{1-\theta}{\theta}} - 1 + \left( 1 - \frac{\theta}{c} \cdot \frac{i}{K} \right)^{\frac{1}{\theta}} \\ &= \left( 1 - \frac{\theta}{c} \cdot \frac{i}{K} \right)^{\frac{1}{\theta}} - \left( 1 - \frac{\theta}{c} \cdot \frac{i-1}{K} \right)^{\frac{1}{\theta}} + \frac{1}{Kc} \cdot \left( 1 - \frac{\theta}{c} \cdot \frac{i-1}{K} \right)^{\frac{1-\theta}{\theta}} \end{aligned}$$

which is exactly the left-hand side of (5.7), proving  $h(b_{i-1}) - b_i \geq 0$ .

Finally,

$$f(S^g) = a_K \geq b_K = \left[ 1 - \left( 1 - \frac{\theta}{c} \right)^{\frac{1}{\theta}} \right] \cdot f(S^*),$$

proving the desired guarantee.  $\square$

Finally, we prove Corollary 9, which shows that our guarantee  $1 - \left( 1 - \frac{\theta}{c} \right)^{\frac{1}{\theta}}$  is at least as good as  $1 - 1/e$  in the worst case.

*Proof of Corollary 9.* We already proved in Lemma 6 that any monotone submodular function is  $(c, \theta)$ -sharp when  $(1/c)^{1/\theta} \rightarrow 0$ , in particular when  $\theta \rightarrow 0$  and  $c \rightarrow 1$ . On the other hand, we know that  $1 - \left( 1 - \frac{\theta}{c} \right)^{\frac{1}{\theta}}$  is increasing on  $\theta$ , so for all  $c \geq 1$  we have  $1 - \left( 1 - \frac{\theta}{c} \right)^{\frac{1}{\theta}} \geq 1 - e^{-1/c}$ . Moreover, when  $\theta \rightarrow 0$  then  $1 - \left( 1 - \frac{\theta}{c} \right)^{\frac{1}{\theta}} \rightarrow 1 - e^{-1/c}$ . Finally, since  $c \rightarrow 1$  we have that  $1 - e^{-1/c} \rightarrow 1 - 1/e$ . In other words,  $1 - \left( 1 - \frac{\theta}{c} \right)^{\frac{1}{\theta}} \rightarrow 1 - 1/e$  when  $\theta \rightarrow 0$  and  $c \rightarrow 1$ .  $\square$

### 5.2.1 Dynamic Sharpness

This section is devoted to prove Theorem 17, when one single switching moment is considered. We stress that the greedy algorithm automatically adapts to the dynamic sharpness of the function without requiring parameters  $c_0, \theta_0, c_1, \theta_1$ , and  $t_0$  as part of the input.

*Proof of Theorem 17.* Observe that in the  $i$ -th iteration of the greedy algorithm  $|S_i| = i$ , so the change of the sharpness parameters will occur for  $S_{t_0}$ . The proof is similar to Theorem 16, but the recursion needs to be splitted in moment  $t_0$ . Let us recall the recursion in the proof of Theorem 16. We have

$$a_i \geq a_{i-1} + \frac{a^\theta}{Kc_{i-1}} \cdot [a - a_{i-1}]^{1-\theta_{i-1}},$$

where  $a_i = f(S_i)$ ,  $a_0 = 0$ , and  $a = f(S^*)$ . The only difference is that now  $c$  and  $\theta$  are parameters that may vary in each step of the greedy algorithm, that is why we write  $c_{i-1}$  and  $\theta_{i-1}$ . We assume there exists a single switching moment  $t_0 \in \{0, \dots, K-1\}$ . First,

observe that for any step  $i \in \{0, \dots, t_0 - 1\}$  of the greedy algorithm we have  $c_i = c_0$  and  $\theta_i = \theta_0$ . This implies that for any step  $i \in \{1, \dots, t_0\}$  we have our first recurrence

$$a_i \geq a_{i-1} + \frac{a^{\theta_0}}{Kc_0} \cdot [a - a_{i-1}]^{1-\theta_0} \quad (5.8)$$

And from the proof of Theorem 16 we already know that

$$a_{t_0} \geq \left[ 1 - \left( 1 - \frac{\theta}{c_0} \cdot \frac{t_0}{K} \right)^{\frac{1}{\theta_0}} \right] \cdot a.$$

It remains to analyze the rest of the recurrence for steps  $i \in \{t_0 + 1, \dots, K\}$ . During these steps we have  $c_i = c_1$  and  $\theta_i = \theta_1$ . Therefore, we have the following recurrence for steps  $i \in \{t_0 + 1, \dots, K\}$

$$a_i \geq a_{i-1} + \frac{a^{\theta_1}}{Kc_1} \cdot [a - a_{i-1}]^{1-\theta_1} \quad (5.9)$$

We will show that in this case the solution for this recurrence satisfies

$$a_i \geq \left[ 1 - \left( \left( 1 - \frac{\theta_0 t_0}{c_0 K} \right)^{\frac{\theta_1}{\theta_0}} + \frac{\theta_1 t_0}{c_1 K} - \frac{\theta_1}{c_1} \cdot \frac{i}{K} \right)^{\frac{1}{\theta_1}} \right] \cdot a. \quad (5.10)$$

For simplicity assume that  $a = 1$ . Similarly to the proof of Theorem 16, we define  $h(x) = x + \frac{1}{Kc_1}[1 - x]^{1-\theta_1}$  for  $x \in [0, 1]$ , which is increasing in the interval  $I := \left\{ x : 0 \leq x \leq 1 - \left( \frac{1-\theta_1}{Kc_1} \right)^{1/\theta_1} \right\}$ . Now, define

$$b_i = 1 - \left( \left( 1 - \frac{\theta_0 t_0}{c_0 K} \right)^{\frac{\theta_1}{\theta_0}} + \frac{\theta_1 t_0}{c_1 K} - \frac{\theta_1}{Kc_1} \cdot i \right)^{\frac{1}{\theta_1}}.$$

Let us prove that  $b_i \in I$ . First, observe that  $b_i \geq 0$ . For the other inequality in  $I$  we have

$$b_i \leq 1 - \left( \frac{1 - \theta_1}{Kc_1} \right)^{1/\theta_1} \Leftrightarrow Kc_1 \left( 1 - \frac{\theta_0 t_0}{c_0 K} \right)^{\theta_1/\theta_0} - \theta_1(i - t_0) \geq 1 - \theta_1 \quad (5.11)$$

The left-hand side of inequality (5.11) is decreasing in  $i$  so it is sufficient to show it for  $i = K$ , which is

$$K \left[ c_1 \left( 1 - \frac{\theta_0 t_0}{c_0 K} \right)^{\theta_1 / \theta_0} - \theta_1 \right] + \theta_1 t_0 \geq 1 - \theta_1,$$

which is clearly satisfied for sufficiently large  $K$ .

Similarly than the proof of Theorem 16, for  $x \in [t_0 + 1, K]$  define

$$g(x) := \left( \left( 1 - \frac{\theta_0 t_0}{c_0 K} \right)^{\frac{\theta_1}{\theta_0}} + \frac{\theta_1 t_0}{c_1 K} - \frac{\theta_1}{c_1 K} \cdot x \right)^{\frac{1}{\theta_1}}.$$

Note that  $g'(x) = -\frac{1}{K c_1} \cdot g(x)^{1-\theta_1}$  and  $g''(x) = \frac{1-\theta_1}{(K c_1)^2} \cdot g(x)^{1-2\theta_1}$ . Observe that  $g$  is convex, so for any  $x_1, x_2 \in [t_0 + 1, K]$  we have  $g(x_2) \geq g(x_1) + g'(x_1) \cdot (x_2 - x_1)$ . By considering  $x_2 = i$  and  $x_1 = i - 1$ , we obtain

$$g(i) - g(i - 1) - g'(i - 1) \geq 0 \tag{5.12}$$

Inequality (5.12) is exactly  $h(b_{i-1}) - b_i \geq 0$ . To finish the proof, we will use induction.

First, we need to prove that  $a_{t_0} \geq b_{t_0}$ . Let us show the following

$$b_{t_0} \leq 1 - \left( 1 - \frac{\theta_0 t_0}{c_0 K} \right)^{\frac{1}{\theta_0}} \Leftrightarrow \left( 1 - \frac{\theta_0 t_0}{c_0 K} \right)^{\frac{\theta_1}{\theta_0}} + \frac{\theta_1 t_0}{c_1 K} - \frac{\theta_1}{c_1} \cdot \frac{t_0}{K} \geq \left( 1 - \frac{\theta_0 t_0}{c_0 K} \right)^{\frac{1}{\theta_0}} \tag{5.13}$$

which is true since  $\theta_1 \leq 1$ . Inequality (5.13) implies that  $a_{t_0} \geq b_{t_0}$  since  $a_{t_0} \geq 1 - \left( 1 - \frac{\theta_0 t_0}{c_0 K} \right)^{\frac{1}{\theta_0}}$ . Now by induction let us assume that  $a_{i-1} \geq b_{i-1}$  is true, then  $a_i \geq h(a_{i-1}) \geq h(b_{i-1}) \geq b_i$ . The first inequality is the definition of the recursion, the second inequality is due to the monotonicity of  $h$  in the interval  $I$ , and finally, the last inequality was proven in Inequality (5.12). Therefore,  $a_K \geq b_K$  which proves the desired guarantee since  $f(S^g) = a_K$ .  $\square$

### 5.2.2 Approximate Sharpness

For some functions the sharpness parameters are close to the extreme case of an arbitrary monotone submodular function, i.e.,  $c \approx 1$  and  $\theta \approx 0$ . However, from computational results we observe that the greedy algorithm still performs considerably well. This motivates us to define another generalization of sharpness in which the marginal values are slightly perturbed. Formally, consider an extra parameter  $\delta \in [0, 1 - \frac{1}{K}]$ . For a set function  $f : 2^V \rightarrow \mathbb{R}_+$  we denote the  $\delta$ -marginal value for any subset  $A \subseteq V$  and  $e \in V$  by  $f_A^\delta(e) := f(A + e) - (1 - \delta)f(A)$ . Now, for a given set of parameters  $\delta \in [0, 1 - \frac{1}{K}]$ ,  $\theta \in [0, 1]$ , and  $c \geq 1$ , we define *approximate submodular sharpness* as follows,

**Definition 6** (Approximate Submodular Sharpness). *A non-negative monotone submodular function  $f : 2^V \rightarrow \mathbb{R}_+$  is said to be approximate  $(\delta, c, \theta)$ -sharp, if there exists an optimal solution  $S^*$  for Problem (5.1) such that for any subset  $S \subseteq V$  with  $|S| \leq K$  the function satisfies*

$$\sum_{e \in S^* \setminus S} f_S^\delta(e) \geq \left[ \frac{|S^* \setminus S|}{K \cdot c} \right]^{\frac{1}{\theta}} \cdot f(S^*)$$

We get the following result for approximate sharp objective functions

**Theorem 18.** *Consider a non-negative monotone submodular function  $f : 2^V \rightarrow \mathbb{R}_+$  that is approximate  $(\delta, c, \theta)$ -sharp. Then, for Problem (5.1) the greedy algorithm returns a feasible set  $S^g$  such that*

$$f(S^g) \geq \frac{1}{1 - \delta + \delta K c} \left[ 1 - (1 - \delta)^{\frac{1}{\theta}} \left( 1 - \frac{\theta}{c} \right)^{\frac{1}{\theta}} \right] \cdot f(S^*).$$

Observe that for  $\delta = 0$ , we recover Theorem 16. In the following proof we will observe that the greedy algorithm automatically adapts to parameters  $\delta$ ,  $\theta$ , and  $c$ , and they will not be required to be part of the input.

*Proof of Theorem 18.* Similarly to the proof of Theorem 16, let us denote by  $S_i$  the set we obtained in the  $i$ -th iteration of the greedy algorithm. By using the properties of the function  $f$ , we obtain the following sequence of inequalities

$$\begin{aligned}
f(S_i) - (1 - \delta)f(S_{i-1}) &= \frac{\sum_{e \in S^* \setminus S_{i-1}} f(S_i) - (1 - \delta)f(S_{i-1})}{|S^* \setminus S_{i-1}|} \\
&\geq \frac{\sum_{e \in S^* \setminus S_{i-1}} f(S_{i-1} + e) - (1 - \delta)f(S_{i-1})}{|S^* \setminus S_{i-1}|} && \text{(choice of greedy)} \\
&= \frac{\sum_{e \in S^* \setminus S_{i-1}} f_{S_{i-1}}^\delta(e)}{|S^* \setminus S_{i-1}|} \\
&\geq \frac{f(S^*)^\theta}{Kc} \cdot \left( \sum_{e \in S^* \setminus S_{i-1}} f_{S_{i-1}}^\delta(e) \right)^{1-\theta} && \text{(sharpness)} \\
&\geq \frac{f(S^*)^\theta}{Kc} \cdot [f(S_{i-1} \cup S^*) - f(S_{i-1}) + \delta|S^* \setminus S_{i-1}|f(S_{i-1})]^{1-\theta} \\
&&& \text{(submodularity)} \\
&\geq \frac{f(S^*)^\theta}{Kc} \cdot [f(S_{i-1} \cup S^*) - (1 - \delta)f(S_{i-1})]^{1-\theta} \\
&&& (|S^* \setminus S_{i-1}| \geq 1) \\
&\geq \frac{f(S^*)^\theta}{Kc} \cdot [f(S^*) - (1 - \delta)f(S_{i-1})]^{1-\theta}. && \text{(monotonicity)}
\end{aligned}$$

Therefore, we need to solve the following recurrence

$$a_i \geq (1 - \delta)a_{i-1} + \frac{a^\theta}{Kc} \cdot [a - (1 - \delta)a_{i-1}]^{1-\theta} \quad (5.14)$$

where  $a_i = f(S_i)$ ,  $a_0 = 0$  and  $a = f(S^*)$ . We will show that the solution of this recurrence  $a_i$  satisfies for all  $i \in \{1, \dots, K\}$

$$a_i \geq \frac{a}{1 - \delta + \delta Kc} \cdot \left[ 1 - (1 - \delta)^{\frac{1}{\theta}} \left( 1 - \frac{\theta}{c} \cdot \frac{i}{K} \right)^{\frac{1}{\theta}} \right], \quad (5.15)$$

which finishes the proof of Theorem 18 since  $f(S^g) = a_K$ .

Let us prove (5.15). Define  $h(x) = (1 - \delta)x + \frac{a^\theta}{Kc} \cdot [a - (1 - \delta)x]^{1-\theta}$ , where  $x \in [0, a]$ . Observe that  $h'(x) = (1 - \delta) - \frac{a^\theta(1-\theta)(1-\delta)}{Kc} \cdot [a - (1 - \delta)x]^{-\theta}$ . Therefore,  $h$  is increasing inside the interval  $I := \left\{x : 0 \leq x \leq \frac{a}{1-\delta} \cdot \left(1 - \left(\frac{1-\theta}{Kc}\right)^{1/\theta}\right)\right\}$ . Let us define

$$b_i := \frac{a}{1 - \delta + \delta Kc} \cdot \left[1 - (1 - \delta)^{\frac{1}{\theta}} \left(1 - \frac{\theta}{c} \cdot \frac{i}{K}\right)^{\frac{1}{\theta}}\right].$$

First, let us check that  $b_i \in I$  for all  $i \in \{1, \dots, K\}$ . Namely, for any  $i$  we need to show that

$$\frac{a}{1 - \delta + \delta Kc} \cdot \left[1 - (1 - \delta)^{\frac{1}{\theta}} \left(1 - \frac{\theta}{c} \cdot \frac{i}{K}\right)^{\frac{1}{\theta}}\right] \leq \frac{a}{1 - \delta} \cdot \left(1 - \left(\frac{1 - \theta}{Kc}\right)^{1/\theta}\right).$$

Since  $1 - \delta \leq 1 - \delta + \delta Kc$ , then it is sufficient to prove that

$$1 - (1 - \delta)^{\frac{1}{\theta}} \left(1 - \frac{\theta}{c} \cdot \frac{i}{K}\right)^{\frac{1}{\theta}} \leq 1 - \left(\frac{1 - \theta}{Kc}\right)^{1/\theta} \Leftrightarrow (1 - \delta)(Kc - i\theta) \geq 1 - \theta$$

The expression  $Kc - i\theta$  is decreasing on  $i$ . Hence, we just need the inequality for  $i = K$ , namely  $(1 - \delta)K(c - \theta) \geq 1 - \theta$ , which is true since  $c \geq 1$ ,  $K \geq 1$  and  $\delta \leq 1 - \frac{1}{K}$ .

Our goal is to prove that  $a_i \geq b_i$ , so by induction let us assume that  $a_{i-1} \geq b_{i-1}$  is true. By using monotonicity of  $h$  on the interval  $I$ , we get  $h(a_{i-1}) \geq h(b_{i-1})$ . Also, observe that recurrence (5.14) is equivalent to write  $a_i \geq h(a_{i-1})$  which implies that  $a_i \geq h(b_{i-1})$ . To finish the proof we will show that  $h(b_{i-1}) \geq b_i$ .

Assume for simplicity that  $a = 1$ . For  $x \in [1, K]$ , define

$$g(x) := (1 - \delta)^{1/\theta} \left(1 - \frac{\theta}{Kc} \cdot x\right)^{1/\theta}.$$

Note that  $g'(x) = -\frac{1-\delta}{Kc} \cdot g(x)^{1-\theta}$  and  $g''(x) = \frac{(1-\delta)(1-\theta)}{(Kc)^2} \cdot g(x)^{1-2\theta}$ . Observe that  $g$  is convex, so for any  $x_1, x_2 \in [0, K]$  we have  $g(x_2) \geq g(x_1) + g'(x_1) \cdot (x_2 - x_1)$ . By

considering  $x_2 = i$  and  $x_1 = i - 1$ , we obtain

$$g(i) - g(i - 1) - g'(i - 1) \geq 0 \quad (5.16)$$

On the other hand,  $h(b_{i-1}) - b_i$  is equivalent to

$$(1 - \delta)[1 - g(i - 1)] + \frac{(1 - \delta + \delta Kc)}{Kc} \left[ 1 - \frac{1 - \delta}{1 - \delta + \delta Kc} (1 - g(i - 1)) \right]^{1-\theta} - 1 + g(i)$$

which is the same as

$$g(i) - g(i - 1) - \delta + \delta g(i - 1) + \frac{(1 - \delta + \delta Kc)^\theta}{kc} [\delta Kc + (1 - \delta)g(i - 1)]^{1-\theta}. \quad (5.17)$$

By Hölder's inequality we know that

$$(1 - \delta + \delta Kc)^\theta [\delta Kc + (1 - \delta)g(i - 1)]^{1-\theta} \geq \delta Kc + (1 - \delta)g(i - 1)^{1-\theta}.$$

Then, by using this bound we get

$$\begin{aligned} (5.17) &\geq g(i) - g(i - 1) - \delta + \delta g(i - 1) + \frac{1}{Kc} [\delta Kc + (1 - \delta)g(i - 1)]^{1-\theta} \\ &= g(i) - g(i - 1) - g'(i - 1) + \delta g(i - 1) \\ &\geq 0, \end{aligned}$$

where we used the definition of  $g'$  and inequality (5.16). This proves that  $h(b_{i-1}) - b_i \geq 0$ , concluding the proof of the recurrence.  $\square$

### 5.3 Computational Study

In this section, we provide an exhaustive computational study of the sharpness criteria in four real-world applications: movie recommendation, non-parametric learning, sensor

location, and exemplar-based clustering. First, to exemplify how sharpness looks, we consider a concave over a linear function  $[l(S)]^\alpha$  and we empirically show how the sharpness varies with respect to different values of  $\alpha$ . At the same time, we will observe that the approximation guarantees increase as  $\alpha$  increases. In the following experiments, we aim to explicitly obtain the sharpness parameters of the objective function for different small ground sets. With these results, we will empirically show how the approximation factors vary with respect to the cardinality budget  $K$ . We contrast these results with the approximation ratios guaranteed by the curvature analysis [30], and we will observe that in some cases the sharpness criterion provides better results. We will also study objective functions that are not stable [25], for which we also numerically study their sharpness. Finally, we will analyze the concept of dynamic sharpness, in which we empirically obtain both set of sharpness parameters when a single switching moment is considered. We will observe that the approximation factors are considerable improved, when they are compared to not having a switching moment.

Consider a non-negative integer  $K$ , a non-negative monotone submodular function  $f$ , and an optimal solution  $S^*$  for Problem (5.1). Due to monotonicity of  $f$  we can assume that  $|S^*| = K$ . Throughout this section, we will denote by

$$\text{AgMarg}(S, S^*) = \frac{\sum_{e \in S^* \setminus S} f_S(e)}{f(S^*)}, \quad (5.18)$$

$$\text{LowEnv}_{c,\theta}(S, S^*) = \left[ \frac{1}{c} \cdot \left( 1 - \frac{|S^* \cap S|}{K} \right) \right]^{1/\theta} \quad (5.19)$$

Let us recall the sharpness criteria:  $f$  is  $(c, \theta)$ -sharp if there exists an optimal solution  $S^*$  for Problem (5.1) such that for any subset  $S \subseteq V$  with  $|S| \leq K$  satisfies

$$\text{AgMarg}(S, S^*) \geq \text{LowEnv}_{c,\theta}(S, S^*). \quad (\text{Sharpness criteria})$$

Observe the following important facts for this section:

1. We always have  $\text{LowEnv}_{c,\theta}(S, S^*) \leq 1$ , since  $c \geq 1$  and  $\theta \in [0, 1]$ . Moreover,  $\text{LowEnv}_{c,\theta}(S, S^*)$  is decreasing in  $c$ , increasing in  $\theta$ , and decreasing in  $|S^* \cap S|$ .
2. The (Sharpness criteria) is trivially satisfied for the empty set, since  $\text{LowEnv}_{c,\theta}(S, S^*) \leq 1$  and  $\text{AgMarg}(\emptyset, S^*) \geq 1$ , where the latter is due to subadditivity:  $\sum_{e \in S^*} f_\emptyset(e) \geq f(S^*)$ .
3. In Inequality (5.5) of the proof of Theorem 16, we divide by  $|S^* \setminus S_{i-1}|$  which is valid for iterations  $i \in \{1, \dots, K\}$ . Specifically, in iteration  $i = K$ , we will divide by  $|S^* \setminus S_{K-1}|$  which later will be replaced by the sharpness criteria. Note that  $|S_{K-1}| = K - 1$ , which implies that it is sufficient enough to have a sharpness criteria for subsets  $|S| \leq K - 1$ . In the original Definition 4, we use  $|S| \leq K$  just for the sake of the exposition and to be consistent with the cardinality constraint of Problem (5.1). Numerically, we only need subsets of size at most  $K - 1$ .
4. For any subset  $S$ , the expression (5.19) only depends on the quantity  $|S^* \cap S|$ , and not on the objective  $f$ . Therefore, the expression (5.19) has the same set of values among sets of the same size. Specifically, for any subset  $S$  with size exactly  $j$  we have  $0 \leq |S^* \cap S| \leq j$ , which implies that

$$\text{LowEnv}_{c,\theta}(S, S^*) \in \left\{ \left[ \frac{K-j}{K \cdot c} \right]^{1/\theta}, \left[ \frac{K-j+1}{K \cdot c} \right]^{1/\theta}, \dots, \left[ \frac{1}{c} \right]^{1/\theta} \right\}$$

5. Since  $f$  is submodular, then for any  $S_1 \subseteq S_2$  we have  $\text{AgMarg}(S_1, S^*) \geq \text{AgMarg}(S_2, S^*)$ . Therefore, numerically we need to check the sharpness criteria only for sets of size exactly  $K - 1$ .
6. When we consider a single switching moment  $t_0 \in \{1, \dots, K - 1\}$  (recall that  $t_0 = 0$  means there is no switch in parameters) we have two set of parameters  $(c_0, \theta_0)$  and  $(c_1, \theta_1)$ . Numerically, we will apply the same logic as 5.: for  $(c_0, \theta_0)$  we will check only subsets of size exactly  $t_0 - 1$  and for  $(c_1, \theta_1)$  we check subsets of size exactly

$K - 1$ . This will significantly reduce the computational time when checking the sharpness criteria.

7. The approximation ratio  $1 - \left(1 - \frac{\theta}{c}\right)^{1/\theta}$  is increasing in  $\theta$  and decreasing in  $c$ , so numerically we look for the largest  $\theta$  and the smallest  $c$ .

**Binary search for sharpness criteria.** To find parameters  $(c, \theta)$  we follow a simple binary search. For  $c$  we sequentially iterate over possible values in a fixed range  $[1, c_{max}]$  (we consider granularity 0.01 and  $c_{max} = 3$ ). Given  $c = 1$ , we do a binary search on  $[0, 1]$ : if we find a subset that does not satisfy the (Sharpness criteria), then we update the lowest value that  $\theta$  can take (expression (5.19) is increasing on  $\theta$ ). If we find a *good pair*  $(c, \theta)$  that satisfies the (Sharpness criteria), then we update the highest value that  $\theta$  can take. For that good pair of parameters we record the approximation ratio and  $\theta$ . For the next  $c' \geq c$  in the grid we do not need to start the binary search again in the whole interval of  $[0, 1]$ . Since the next  $c'$  is higher than the previous one, then the approximation factor can only decrease on  $c'$ , so the only option to get a better factor is to increase  $\theta$ . Therefore, for  $c'$  we do the binary search on  $[\theta, 1]$  where  $\theta$  is part of the last good pair of parameters in the previous iteration. When a switching moment  $t_0$  is considered, then we divide this procedure in two parts: one for subsets of size exactly  $t_0 - 1$  and one for subsets of size exactly  $K - 1$ . We stop the loop on  $c$  if the approximation factor has not been improved after some amount of iterations.

Finally, in all the experiments we normalize the objective function so  $f(S) \in [0, 1]$ .

### 5.3.1 Movie Recommendation

For this application we consider the MovieLens data-set [50] which consists of 7,000 users and 13,977 movies. Each user had to rank at least one movie with an integer value in  $\{0, \dots, 5\}$  where 0 denotes that the movies was not ranked by that user. Therefore, we have

a matrix  $[r_{ij}]$  of rankings for each user  $i$  and each movie  $j$ . The objective in this application is to select the  $K$  highest ranked movies among the users. To make the computations less costly in terms of time, we use only  $m = 1000$  users. In the same spirit, we will choose a small number  $n$  from the 13,977 movies.

In our first experiment, we consider the following function  $f(S) = \frac{1}{m} \sum_{i \in [m]} \left( \sum_{j \in S} r_{ij} \right)^\alpha$  where  $\alpha \in (0, 1]$ , with a ground set of  $n = 20$  random movies and  $K = 10$ . This function is non-negative, monotone, and submodular since  $g(y) = y^\alpha$  is concave and increasing for  $\alpha \in (0, 1]$ . In Figures 5.1-5.3, our goal is to empirically show how the sharpness of function varies with different values of  $\alpha$ . For this, we plot  $\text{AgMarg}(S, S^*)$  (in red) for all  $S \subseteq V$  with  $|S| \leq K - 1$  and the possible different levels of  $\text{LowEnv}_{c,\theta}(S, S^*)$  (in green). We consider  $\alpha = 0.1$  in Figure 5.1,  $\alpha = 0.5$  in Figure 5.2 and  $\alpha = 0.9$  in Figure 5.3.

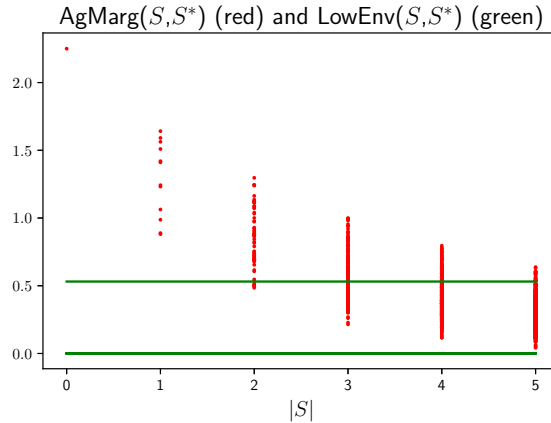


Figure 5.1: *How sharpness looks*:  $\alpha = 0.1$ ,  $c = 1.01$  and  $\theta = 0.0157$ .

The results are the following: for  $\alpha = 0.1$  we obtained  $c = 1.01$  and  $\theta = 0.0157$  which gives an approximation factor of 0.6321; for  $\alpha = 0.5$  we obtained  $c = 1.21$  and  $\theta = 0.6394$  which gives an approximation factor of 0.6958; for  $\alpha = 0.9$  we obtained  $c = 1.06$  and  $\theta = 0.8605$  which gives an approximation factor of 0.8564. We observe in Figure 5.1, that when  $\alpha$  is small, then the values in red concentrates all together and drop very fast as the size of  $S$  increases 0.6321. This forces the levels of  $\text{LowEnv}_{c,\theta}(S, S^*)$  to decrease faster, resulting in a worse approximation guarantee. On the other hand, when we consider

a larger  $\alpha$  as in Figure 5.3, values in red tend to be flat in certain values and well spread. In this case, it is easier to spread the levels of  $\text{LowEnv}_{c,\theta}(S, S^*)$ , which translates in a much better approximation guarantee 0.8564.

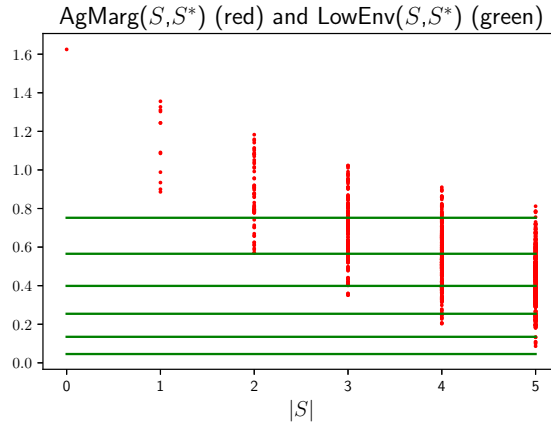


Figure 5.2: *How sharpness looks:*  $\alpha = 0.5$ ,  $c = 1.21$  and  $\theta = 0.6394$ .

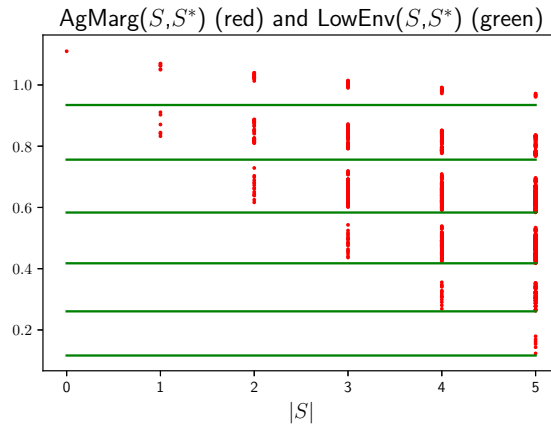


Figure 5.3: *How sharpness looks:*  $\alpha = 0.9$ ,  $c = 1.06$  and  $\theta = 0.8605$ .

In the following experiment let us fix  $\alpha = 0.5$ . We performed six random instances with  $n = 2K$  and different budgets  $K \in \{5, \dots, 10\}$ . Namely, out of the 13,977 movies we select uniformly at random  $n$  movies to form the ground set. We present our results in Figures 5.4, 5.5 and 5.6.

In Figure 5.4 we plot the variation of the approximation ratios guaranteed by the curva-

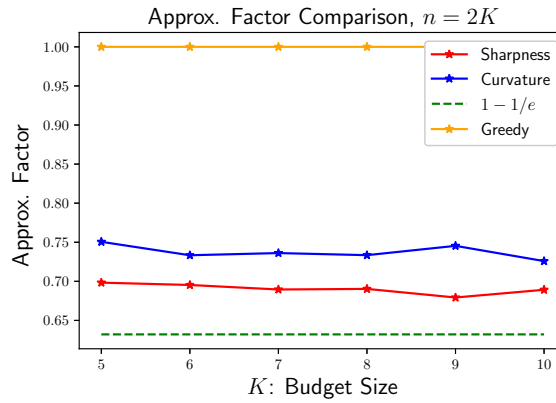


Figure 5.4: *Movie Recommendation*  $\alpha = 0.5$ : for different instances of size  $n = 2K$ , we plot the empirical ratio of the greedy algorithm and the approximation factors obtained from the curvature and sharpness analysis.

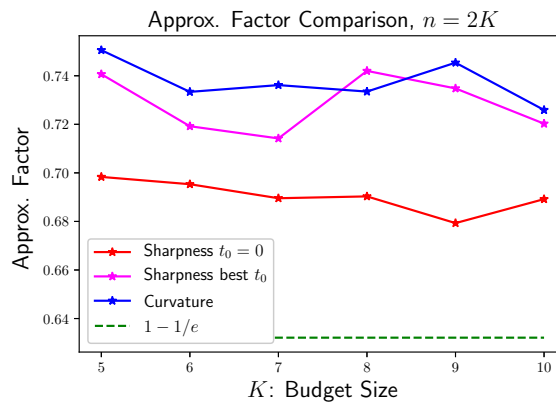


Figure 5.5: *Movie Recommendation*  $\alpha = 0.5$ : for different instances of size  $n = 2K$ , we plot the approximation factors obtained from the curvature analysis, the sharpness analysis with no switching point and the sharpness analysis with one single switching point.

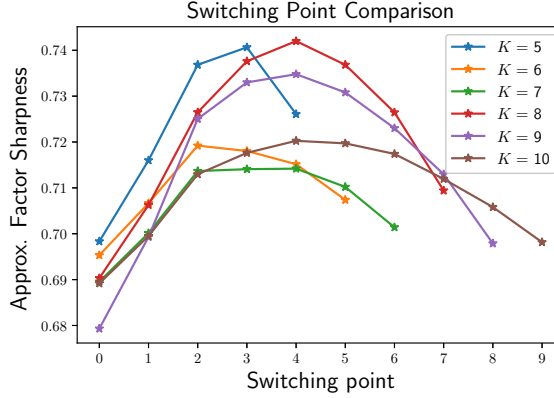


Figure 5.6: *Movie Recommendation*  $\alpha = 0.5$ : for different instances of size  $n = 2K$ , we plot the approximation factors obtained from the dynamic sharpness analysis with respect to the different candidates of switching point  $t_0$ .

ture analysis (blue,  $\frac{1-e^{-\gamma}}{\gamma}$ ), the standard greedy analysis ( $1 - 1/e$ ), the sharpness criterion (in red) with respect to the budget size  $K$ , and the empirical ratio obtained by the greedy algorithm (in orange). We observe that in this case the greedy algorithm always finds an optimal solution, which is expected given the objective function. We also note that the curvature analysis provides better results in general compared to the sharpness criterion. This might be due to the fact that the function is well curved, which translates in a better guarantee. At the same time the values of the function are well distributed (not sharp) which means worse guarantees. In Figure 5.5 we also plot (in magenta) the guarantee when we allow one switching moment in the sharpness parameters. For this we consider switching moments  $t_0 = \{1, \dots, K - 1\}$  and we plot for each  $K$  the best guarantee obtained among all options of  $t_0$ . To ease the comparison, we removed the ratios obtained by the greedy algorithm (orange in Figure 5.4). By considering a single switching moment we improve considerably the approximation guarantee, and in some cases we obtain better results than the curvature analysis (see  $K = 8$ ). Finally, in Figure 5.6 we plot the variation of the approximation guarantees with respect to  $t_0$  and we do this for every instance of size  $n = 2K$ . We observe that the best switching moment is generally around  $t_0 = K/2$ .

In the next experiment, we consider the function  $f(S) = \frac{1}{m} \sum_{i \in [m]} \max_{j \in S} r_{ij}$ . This

function is known to be non-negative, monotone, and submodular. Most of the time this function is not 2-stable (in the sense given in Section 5.1.3) since it may have multiple optimal solutions. We performed six random instances with  $n = 2K$  and different budgets  $K \in \{5, \dots, 10\}$ . Namely, out of the 13,977 movies we select uniformly at random  $n$  movies to form the ground set. We present our results in Figures 5.7, 5.8 and 5.9.

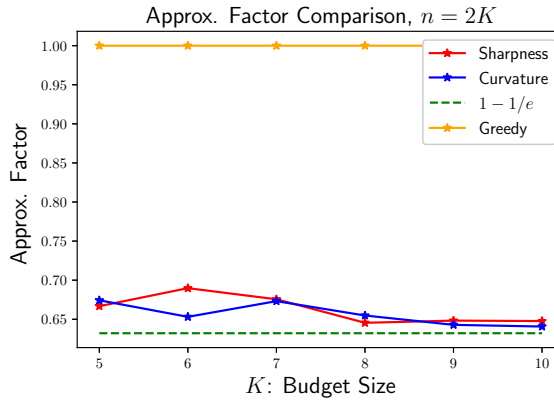


Figure 5.7: *Movie Recommendation - Facility Location*: for different instances of size  $n = 2K$ , we plot the empirical ratio of the greedy algorithm and the approximation factors obtained from the curvature and sharpness analysis.

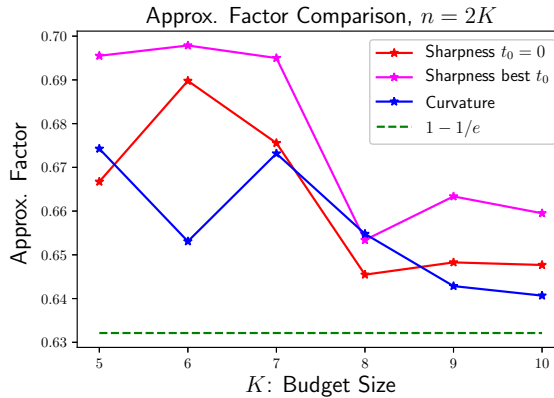


Figure 5.8: *Movie Recommendation - Facility Location*: for different instances of size  $n = 2K$ , we plot the approximation factors obtained from the curvature analysis, the sharpness analysis with no switching point and the sharpness analysis with one single switching point.

In Figure 5.7 we plot the variation of the approximation ratios guaranteed by the curvature analysis (blue,  $\frac{1-e^{-\gamma}}{\gamma}$ ), the standard greedy analysis ( $1 - 1/e$ ), the sharpness criterion

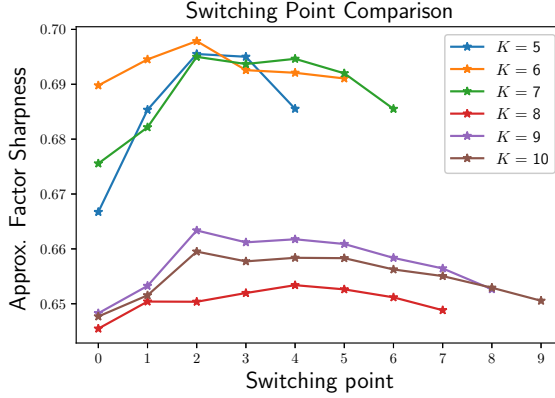


Figure 5.9: *Movie Recommendation - Facility Location*: for different instances of size  $n = 2K$ , we plot the approximation factors obtained from the dynamic sharpness analysis with respect to the different candidates of switching point  $t_0$ .

(in red) with respect to the budget size  $K$ , and the empirical ratio obtained by the greedy algorithm (in orange). We observe that in this case the greedy algorithm still finds an optimal solution. We observe that in some cases the sharpness analysis provides better guarantees than the curvature analysis. In general, this objective tends to be *flat* which for the curvature analysis translates in worse guarantees. When a single switching moment is considered (see Figure 5.8) we obtain significantly better results, beating the curvature’s guarantee in almost every instance. Finally, in Figure 5.9 we observe that the best switching moment is generally around  $t_0 = 2$ .

### 5.3.2 Nonparametric learning

For this application we follow the setup in [84]. Let  $X_V$  be a set of random variables corresponding to bio-medical measurements, indexed by a ground set of patients  $V$ . We assume  $X_V$  to be a Gaussian Process (GP), i.e., for every subset  $S \subseteq V$ ,  $X_S$  is distributed according to a multivariate normal distribution  $\mathcal{N}(\mu_S, \Sigma_{S,S})$ , where  $\mu_S = (\mu_e)_{e \in S}$  and  $\Sigma_{S,S} = [\mathcal{K}_{e,e'}]_{e,e' \in S}$  are the prior mean vector and prior covariance matrix, respectively. The covariance matrix is given in terms of a positive definite kernel  $\mathcal{K}$ , e.g., a common choice in practice is the squared exponential kernel  $\mathcal{K}_{e,e'} = \exp(-\|x_e - x_{e'}\|_2^2/h)$ . Most

efficient approaches for making predictions in GPs rely on choosing a small subset of data points. For instance, in the Informative Vector Machine (IVM) the goal is to obtain a subset  $A$  such that maximizes the information gain,  $f(A) = \frac{1}{2} \log \det(\mathbf{I} + \sigma^{-2} \Sigma_{A,A})$  which was shown to be monotone and submodular in [68]. In our experiment, we use the Parkinson Telemonitoring dataset [107] consisting of a total of 5,875 patients with early-stage Parkinson’s disease and the corresponding bio-medical voice measurements with 22 attributes (dimension of the observations). We normalize the vectors to zero mean and unit norm. With these measurements we computed the covariance matrix  $\Sigma$  considering the squared exponential kernel with parameter  $h = 0.75$ . For the objective function we consider  $\sigma = 1$ . As we mentioned before, the objective in this application is to select the  $k$  most informative patients.

We performed six random instances with  $n = 2K$  and different budgets  $K \in \{5, \dots, 10\}$ . Namely, out of 5,875 patients we randomly select  $n$  of them to construct our ground set. We present our results in Figures 5.10, 5.11 and 5.12.

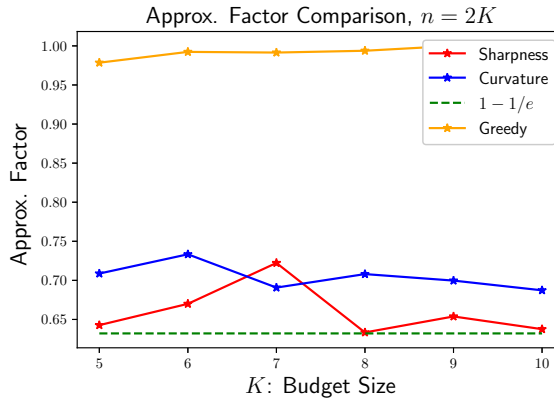


Figure 5.10: *Non-parametric Learning*: for different instances of size  $n = 2K$ , we plot the empirical ratio of the greedy algorithm and the approximation factors obtained from the curvature and sharpness analysis.

In Figure 5.10 we observe that the greedy algorithm still finds a nearly optimal solution. We also note that in most of the instances the curvature analysis provides better guarantees than the sharpness criterion. When a single switching moment is considered (see Figure

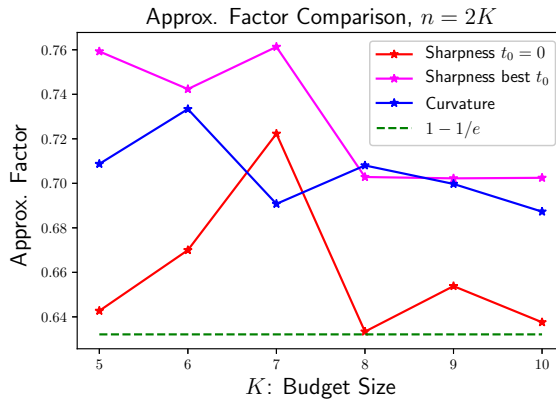


Figure 5.11: *Non-parametric Learning*: for different instances of size  $n = 2K$ , we plot the approximation factors obtained from the curvature analysis, the sharpness analysis with no switching point and the sharpness analysis with one single switching point.

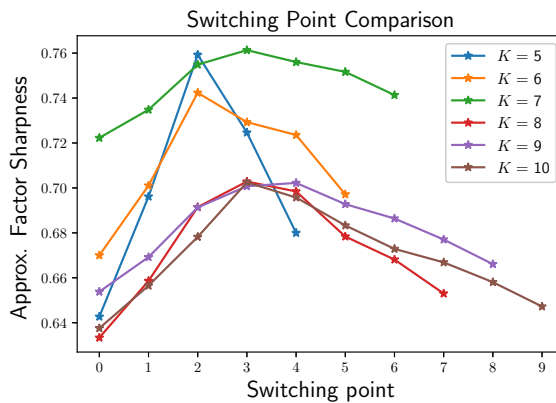


Figure 5.12: *Non-parametric Learning*: for different instances of size  $n = 2K$ , we plot the approximation factors obtained from the dynamic sharpness analysis with respect to the different candidates of switching point  $t_0$ .

5.11) we obtain significantly better results, beating the curvature’s guarantee in almost every instance. Finally, in Figure 5.12 we observe that the best switching moment is generally around  $t_0 = 3$ .

### 5.3.3 Sensor location

For this problem we follow the setup in [71]. Here, we are given a set of sensors  $V$  with fixed locations in a specific region. Each sensor  $s$  measures certain phenomena such as temperature, humidity and light, which define a random vector  $X_s$ . We assume that the set of random variables  $X_V$  is distributed according to a multivariate normal distribution, which corresponds to a Gaussian Process (GP). The predictive variance of sensor  $s$  after obtaining observations from a subset of sensors  $A \subseteq V$  is given by  $\sigma_{s|A}^2 = \sigma_s^2 - \Sigma_{sA} \Sigma_{AA}^{-1} \Sigma_{As}$ , where  $\Sigma_{AA}$  is the covariance matrix of the measurements at the chosen locations  $A$ ,  $\Sigma_{sA}$  is the row-vector in  $\Sigma$  with row  $s$  and columns  $A$ , and  $\sigma_s^2$  is the *a priori* variance of sensor  $s$ . Traditionally, the goal is to find a subset  $A$  that minimizes the predictive variance. However, let us assume that the *a priori* variance  $\sigma_s^2$  is constant for all locations  $s$  and define the variance reduction  $f_s(A) := \Sigma_{sA} \Sigma_{AA}^{-1} \Sigma_{As}$ . [32] show that  $f_s$  is monotone and submodular for certain distributions. Therefore, minimizing  $\sigma_{s|A}^2$  is equivalent to maximizing  $f_s$  when  $\sigma_s^2$  is assumed to be constant.

We use the Intel Research Berkeley dataset of  $n = 44$  sensors, which contains measurements of temperature (T), humidity (L), and light (L). We consider data of three consecutive days, and we construct the corresponding the covariance matrix  $\Sigma^T$  for the temperature measurements. We performed six random instances with  $n = 2K$  and different budgets  $K \in \{5, \dots, 10\}$ . Namely, out of 44 sensors we randomly select  $n$  of them to construct our ground set. We present our results in Figures 5.13, 5.14 and 5.15.

In Figure 5.13 we observe that the greedy algorithm still finds a nearly optimal solution. We also note that in every instance the curvature analysis provides better guarantees than the sharpness criterion (see 5.14), which does not improve beyond  $1 - 1/e$ . When a single

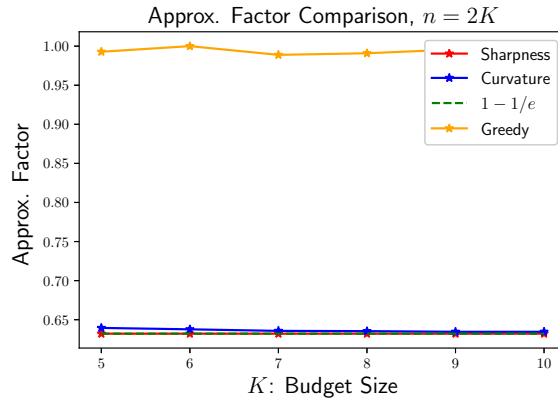


Figure 5.13: *Sensor Location*: for different instances of size  $n = 2K$ , we plot the empirical ratio of the greedy algorithm and the approximation factors obtained from the curvature and sharpness analysis.

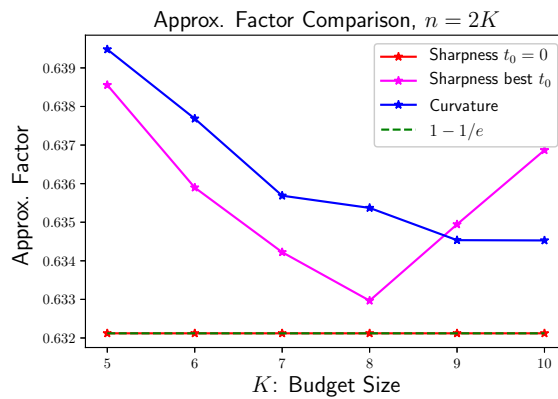


Figure 5.14: *Sensor Location*: for different instances of size  $n = 2K$ , we plot the approximation factors obtained from the curvature analysis, the sharpness analysis with no switching point and the sharpness analysis with one single switching point.

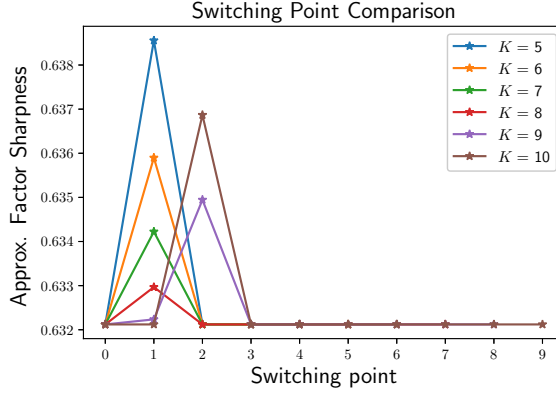


Figure 5.15: *Sensor Location*: for different instances of size  $n = 2K$ , we plot the approximation factors obtained from the dynamic sharpness analysis with respect to the different candidates of switching point  $t_0$ .

switching moment is considered (see Figure 5.14) we obtain significantly better results, beating the curvature’s guarantee in some cases. Finally, in Figure 5.15 we observe that the best switching moment is generally around  $t_0 = 1$  or  $t_0 = 2$ .

### 5.3.4 Exemplar-based clustering

We follow the setup in [84]. Solving the  $k$ -medoid problem is a common way to select a subset of exemplars that represent a large dataset  $V$  [62]. This is done by minimizing the sum of pairwise dissimilarities between elements in  $A \subseteq V$  and  $V$ . Formally, define  $L(A) = \frac{1}{V} \sum_{e \in V} \min_{v \in A} d(e, v)$ , where  $d : V \times V \rightarrow \mathbb{R}_+$  is a distance function that represents the dissimilarity between a pair of elements. By introducing an appropriate auxiliary element  $e_0$ , it is possible to define a new objective  $f(A) := L(\{e_0\}) - L(A + e_0)$  that is monotone and submodular [48], thus maximizing  $f$  is equivalent to minimizing  $L$ . In our experiment, we use the VOC2012 dataset [37] which contains around 10,000 images. The ground set  $V$  corresponds to images, and we want to select a subset of the images that best represents the dataset. Each image has several (possible repeated) associated categories such as person, plane, etc. There are around 20 categories in total. Therefore, images are represented by feature vectors obtained by counting the number of elements

that belong to each category, for example, if an image has 2 people and one plane, then its feature vector is  $(2, 1, 0, \dots, 0)$  (where zeros correspond to other elements). We choose the Euclidean distance  $d(e, e') = \|x_e - x_{e'}\|$  where  $x_e, x_{e'}$  are the feature vectors for images  $e, e'$ . We normalize the feature vectors to mean zero and unit norm, and we choose  $e_0$  as the origin.

We performed six random instances with  $n = 2K$  and different budgets  $K \in \{5, \dots, 10\}$ . Namely, out of 3,000 images we randomly select  $n$  of them to construct our ground set. We present our results in Figures 5.16, 5.17 and 5.18.

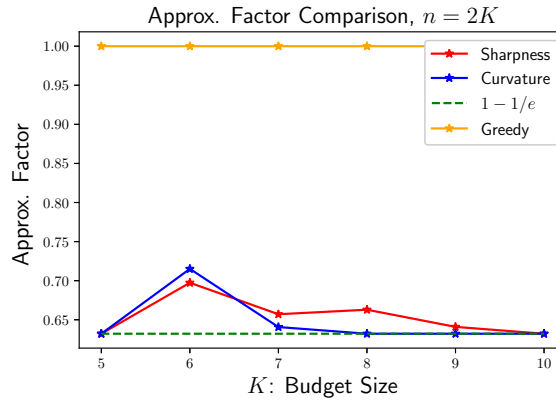


Figure 5.16: *Image Clustering*: for different instances of size  $n = 2K$ , we plot the empirical ratio of the greedy algorithm and the approximation factors obtained from the curvature and sharpness analysis.

In Figure 5.16 we observe that the greedy algorithm still finds an optimal solution. We also note that in some instances the sharpness criterion provides better guarantees than the curvature analysis. When a single switching moment is considered (see Figure 5.17) we obtain slightly better results. Finally, in Figure 5.18 we observe that there is no common best switching point.

## 5.4 Concluding Remarks

In this chapter, we introduced the concept of submodular sharpness as a novel candidate to explain the performance of the standard greedy algorithm. To our knowledge, this is

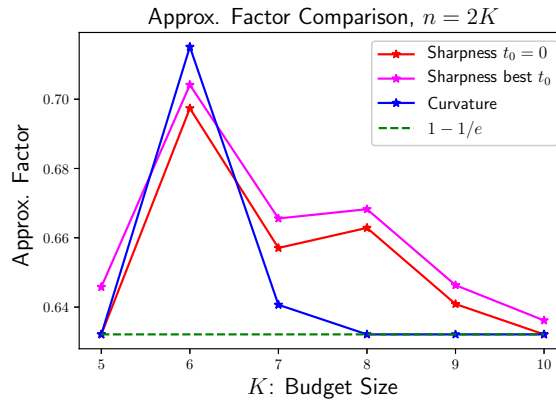


Figure 5.17: *Image Clustering*: for different instances of size  $n = 2K$ , we plot the approximation factors obtained from the curvature analysis, the sharpness analysis with no switching point and the sharpness analysis with one single switching point.

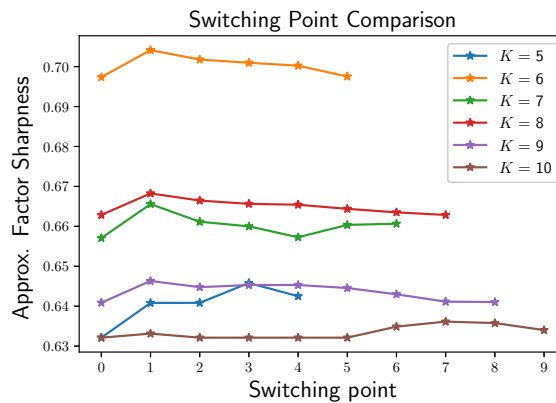


Figure 5.18: *Image Clustering*: for different instances of size  $n = 2K$ , we plot the approximation factors obtained from the dynamic sharpness analysis with respect to the different candidates of switching point  $t_0$ .

the first time that the sharpness criterion is used in the context of submodular optimization. In simple words, this property characterizes the behavior of a submodular function around an optimal solution. We show that when a monotone submodular function is sharper, then better approximation factors are guaranteed. In fact, for a  $(c, \theta)$ -sharp function, the standard greedy algorithm achieves a  $1 - (1 - \theta/c)^{1/\theta}$  ratio for the problem of maximizing a non-negative monotone submodular function subject to a single cardinality constraint. We propose two alternative generalizations: dynamic sharpness and approximate sharpness. For both criteria we show that the greedy algorithm provides provable guarantees. We stress that the standard greedy algorithm is adaptive to the sharpness of the objective and it does not need the sharpness parameters as part of the input. In other words, the algorithm does not need to have access to those parameters in order to provide provable guarantees. An open question that remains to be answered in this chapter is whether or not the approximation factor  $1 - (1 - \theta/c)^{1/\theta}$  is tight when the best set of parameters  $(c, \theta)$  is considered. Finally, we empirically contrast our approach with the curvature analysis and stability criterion. We demonstrate via four computational studies that the sharpness criterion provides in some cases better factor guarantees than these two previous approaches.

# Appendices

**APPENDIX A**  
**EXTRA MATERIAL FOR CHAPTER 3**

**A.1 Remaining Proofs**

*Proof of Theorem 7.* The case  $h = 1$  is already covered by the proof of Proposition 8. Consider the case  $h = n - 1$  and  $t = 1$ ; the other cases follow a similar construction of linearly independent points. Let  $J = \{0, \dots, n - 2\}$ , and assume without loss of generality that  $i_\tau = i$  for all  $\tau \in [n - 1]$ . The specific inequality is

$$Z_{N,J}^n + \sum_{\tau=1}^{n-1} n^{n-\tau} Z_{i,J}^\tau \leq 1 + \sum_{\tau=1}^{n-2} n^\tau. \quad (\text{A.1})$$

We know that  $z \in [0, 1]^{n^3}$ , but for the description of the points (and the proof) we will just consider the coordinates involved in the inequality, i.e.,  $z \in [0, 1]^p$ , where  $p := (2n - 1)(n - 1)$ . For the rest of the points, the construction is similar to the one in the proof of Proposition 8. Recall that  $e_{k,j}^\tau$  denotes the canonical vector in  $[0, 1]^p$ , i.e. a vector with a 1 in coordinate  $(k, j, \tau)$  and zero elsewhere, indicating a match of impression  $k$  with ad  $j$  in stage  $\tau$ . Consider the elements of  $J$  as an  $(n - 1)$ -tuple, i.e.,  $(0, \dots, n - 2)$ . For  $j \in J$ , we define

$$j + (0, \dots, n - 2) := (j, \dots, j + n - 2) \pmod{(n - 1)}.$$

Any addition or subtraction with  $j \in J$  is modulo  $(n - 1)$  for the remainder of the proof. We denote the circulation of  $J$  as the following set of  $(n - 1)$ -tuples:

$$\begin{aligned} \text{circ}(J) &:= \{j + (0, \dots, n - 2)\}_{j \in J} \\ &= \{(0, \dots, n - 2), (1, \dots, n - 2, 0), \dots, (n - 2, 0, \dots, n - 3)\}. \end{aligned}$$

Note that  $\text{circ}(J)$  can be viewed as a matrix. Each element of  $\text{circ}(J)$  corresponds to a sequence of ads in the process from stage  $n$  to stage 1. Since we have  $n$  stages and any of those sequences has size  $n - 1$ , then clearly there is no matching in some stage or an element repeats. We now describe the family of linearly independent points.

- I. Fix  $k \in N$  and  $j \in J$ . In stage  $n$ , if node  $k$  appears, then match it to node  $j$ , with probability  $1/n$ . For the remaining stages match according to  $(j, j + 1, \dots, j + n - 2) \in \text{circ}(J)$ . In terms of probability, if  $i$  appears in stage  $n - 1$ , then it is matched to  $j$  with probability  $(1 - 1/n) \cdot 1/n$ . For the rest, the probability is  $1/n$ . So, we have the point

$$\frac{1}{n}e_{k,j}^n + \frac{1}{n} \left(1 - \frac{1}{n}\right) e_{i,j}^{n-1} + \frac{1}{n} \sum_{\tau=1}^{n-2} e_{i,j+\tau}^{n-1-\tau}. \quad (\text{A.2})$$

By a simple calculation, it is easy to see that each of these points achieves the right-hand side of (A.1). Since we chose an arbitrary  $k \in N$  and  $j \in J$ , we have  $n(n - 1)$  points in this family.

- II. Fix  $j \in J$ . In this family we repeat  $j$  in stages  $n - 1$  and  $n - 2$ . If  $i$  appears in stage  $n - 1$ , match it to node  $j$  with probability  $1/n$ . If  $i$  appears in stage  $n - 2$  and it did not appear in  $n - 1$ , match it to  $j$  with probability  $(1 - 1/n) \cdot 1/n$ . For the remaining stages match according to  $(j + n - 2, j, j + 1, \dots, j + n - 3) \in \text{circ}(J)$ ; in particular, in stage  $n$  match any  $k \in N$  that appears with node  $j + n - 2$ , in stage  $n - 3$  match  $i$  to  $j + 1$  if it appears, and so forth. So, we have the point

$$\frac{1}{n} \sum_{k \in N} e_{k,j+n-2}^n + \frac{1}{n} e_{i,j}^{n-1} + \frac{1}{n} \left(1 - \frac{1}{n}\right) e_{i,j}^{n-2} + \frac{1}{n} \sum_{\tau=1}^{n-3} e_{i,j+\tau}^{n-\tau-2} \quad (\text{A.3})$$

By a simple calculation, we get the right-hand side of (A.1). Since we chose an arbitrary  $j \in J$ , we have  $n - 1$  points in this family.

- III. Fix  $j \in J$ ; in this family we have two different options in stage  $n - 3$ . If  $i$  appears in stage  $n - 1$ , match it to  $j$  with probability  $1/n$ . If  $i$  appears in stage  $n - 2$ , match

it to  $j + 1$ , also with probability  $1/n$ . If  $i$  appears in stage  $n - 3$ , match it to  $j + 1$  with probability  $(1 - 1/n) \cdot 1/n$ , or if  $j + 1$  was matched in stage  $n - 2$ , then to node  $j$  with probability  $(1 - 1/n) \cdot 1/n^2$ . For the remaining stages match according to  $(j + n - 2, j, j + 1, \dots, j + n - 3)$ ; in stage  $n$  match any  $k \in N$  that appears to  $j + n - 2$ , in stage  $n - 4$  match  $i$  to  $j + 2$  if it appears, and so forth. So, we have the point

$$\begin{aligned} \frac{1}{n} \sum_{k \in N} e_{k, j+n-2}^n + \frac{1}{n} e_{i, j}^{n-1} + \frac{1}{n} e_{i, j+1}^{n-2} \\ + \left(1 - \frac{1}{n}\right) \left[ \frac{1}{n^2} e_{i, j}^{n-3} + \frac{1}{n} e_{i, j+1}^{n-3} \right] + \frac{1}{n} \sum_{\tau=1}^{n-4} e_{i, j+\tau+1}^{n-\tau-3} \end{aligned} \quad (\text{A.4})$$

By a simple calculation, we get the right-hand side of (A.1). Since we chose an arbitrary  $j \in J$ , we have  $n - 1$  points in this family.

IV. Fix  $j \in J$  and stage  $s \in [n - 4]$ ; the previous family can be generalized for stage  $s$ , but increasing the number of options, i.e., in stage  $s$  we have  $n - s - 1$  options from the previous stages. If  $i$  appears in stage  $n - 1$ , match it to  $j$  with probability  $1/n$ , if  $i$  appears in stage  $n - 2$ , match it to  $j + 1$  with probability  $1/n$ , and continue in this way until stage  $s + 1$ , where if  $i$  appears, match it to node  $j + n - s - 2$  with probability  $1/n$ . If  $i$  appears in stage  $s$ , we consider ads  $(j + n - s - 2, \dots, j + 1, j)$  in this order of priority, so that  $i$  is matched to  $j + n - s - 2$  with probability  $(1 - 1/n) \cdot 1/n$ ; each subsequent ad's probability of being matched to  $i$  decreases exponentially until  $j$ , which has probability  $(1 - 1/n) \cdot 1/n^{n-s-1}$ . For the remaining stages (including stage  $n$ ) match according to  $(j + n - 2, j, j + 1, \dots, j + n - 3)$ ; in stage  $n$  match any  $k \in N$  that appears with  $j + n - 2$ , in  $s - 1$  match  $i$  to  $j + n - s - 1$  if it appears, etc. So, we have the point

$$\frac{1}{n} \sum_{k \in N} e_{k, j+n-2}^n + \frac{1}{n} \sum_{\tau=0}^{n-s-2} e_{i, j+\tau}^{n-\tau-1}$$

$$+ \left(1 - \frac{1}{n}\right) \left[ \sum_{\tau=0}^{n-s-2} \frac{1}{n^{n-\tau-s-1}} e_{i,j+\tau}^s \right] + \frac{1}{n} \sum_{\tau=1}^{s-1} e_{i,j+n-s-2+\tau}^{s-\tau} \quad (\text{A.5})$$

The left-hand side of (A.1) evaluated at this point is

$$1 + \sum_{\tau=s+1}^{n-1} \frac{n^{n-\tau}}{n} + \left(1 - \frac{1}{n}\right) \sum_{\tau=0}^{n-s-2} \frac{n^{n-s}}{n^{n-\tau-s-1}} + \sum_{\tau=1}^{s-1} \frac{n^{n-\tau}}{n} = 1 + \sum_{\tau=1}^{n-2} n^\tau.$$

Finally, since we chose an arbitrary  $j \in J$  and  $s \in [n-4]$ , we have  $(n-1)(n-4)$  points in this family.

V. Fix  $j \in J$ . For this family we do not match in stage  $n$ , and in the remaining stages we match according to  $(j, j+1, \dots, j+n-2) \in \text{circ}(J)$ . If  $i$  appears in stage  $n-1$  match it to  $j$  with probability  $1/n$ , if  $i$  appears in stage  $n-2$ , match it to  $j+1$ , and so on. So we have the point

$$\frac{1}{n} \sum_{\tau=0}^{n-2} e_{i,j+\tau}^{n-\tau-1} \quad (\text{A.6})$$

By a simple calculation, we get the right-hand side of (A.1). Finally, since we chose an arbitrary  $j \in J$ , then we have  $n-1$  points in this family.

With these families, we have  $p$  points in total. Denote by  $(k, j, \tau)$  the index of a vector  $z \in [0, 1]^p$ , which indicates that  $k \in N$  is matched to  $j \in J$  in stage  $\tau$ . In any of these points consider the following order of components (starting from the first one):  $(1, 0, n)$ ,  $(1, 1, n), \dots, (1, n-2, n), \dots, (n, 0, n), \dots, (n, n-2, n), (i, 0, n-1), \dots, (i, n-2, n-1), \dots, (i, 0, 1), \dots, (i, n-2, 1)$ .

The rest of the proof consists of showing that these families define a set of linearly independent points, and we prove this using Gaussian elimination. Arrange these points as column vectors in a matrix  $A$ ,

$$A = [\text{I}, \text{II}, \text{III}, \text{IV}, \text{V}] = \begin{pmatrix} B_1 & B_2 \\ B_3 & B_4 \end{pmatrix},$$

where  $B_1$  is a  $n(n-1) \times n(n-1)$  diagonal matrix with entries  $1/n$ . These columns can be used to make  $B_2$  a zero matrix, yielding

$$\bar{A} = \begin{pmatrix} B_1 & 0 \\ B_3 & C \end{pmatrix}.$$

Consider how the columns from families II, III, and IV look like after this elimination procedure (family V is not affected). Fix  $g \in J$  and sum every point (A.2) over  $k \in N$ ; this yields

$$\frac{1}{n} \sum_{k \in N} e_{k,g}^n + \left(1 - \frac{1}{n}\right) e_{i,g}^{n-1} + \sum_{\tau=1}^{n-2} e_{i,g+\tau}^{n-1-\tau}. \quad (\text{A.7})$$

II<sup>a</sup>. Pick the point (A.3) associated with  $g+1 \in J$ ,

$$\frac{1}{n} \sum_{k \in N} e_{k,g}^n + \frac{1}{n} e_{i,g+1}^{n-1} + \frac{1}{n} \left(1 - \frac{1}{n}\right) e_{i,g+1}^{n-2} + \frac{1}{n} \sum_{\tau=1}^{n-3} e_{i,g+1+\tau}^{n-1-\tau}. \quad (\text{A.8})$$

Subtract (A.7) from (A.8) to get

$$\frac{1}{n} e_{i,g+1}^{n-1} + \frac{1}{n} \left(1 - \frac{1}{n}\right) e_{i,g+1}^{n-2} + \frac{1}{n} \sum_{\tau=1}^{n-3} e_{i,g+1+\tau}^{n-1-\tau} - \left(1 - \frac{1}{n}\right) e_{i,g}^{n-1} - \sum_{\tau=1}^{n-2} e_{i,g+\tau}^{n-1-\tau},$$

which is equivalent to

$$\frac{1}{n} e_{i,g+1}^{n-1} + \left(-1 + \frac{1}{n}\right) e_{i,g}^{n-1} + \left(\frac{1}{n} - \frac{1}{n^2} - 1\right) e_{i,g+1}^{n-2} + \left(-1 + \frac{1}{n}\right) \sum_{\tau=2}^{n-2} e_{i,g+\tau}^{n-1-\tau}. \quad (\text{A.9})$$

III<sup>a</sup>. Pick the point (A.4) associated with  $g+1 \in J$ ,

$$\begin{aligned} & \frac{1}{n} \sum_{k \in N} e_{k,g}^n + \frac{1}{n} e_{i,g+1}^{n-1} + \frac{1}{n} e_{i,g+2}^{n-2} \\ & + \left(1 - \frac{1}{n}\right) \left[ \frac{1}{n^2} e_{i,g+1}^{n-3} + \frac{1}{n} e_{i,g+2}^{n-3} \right] + \frac{1}{n} \sum_{\tau=1}^{n-4} e_{i,g+\tau+2}^{n-1-\tau}. \end{aligned} \quad (\text{A.10})$$

Subtract (A.7) from (A.10) to get

$$\begin{aligned} & \left(-1 + \frac{1}{n}\right) e_{i,g}^{n-1} + \frac{1}{n} e_{i,g+1}^{n-1} + \frac{1}{n} e_{i,g+2}^{n-2} - e_{i,g+1}^{n-2} \\ & + \left(1 - \frac{1}{n}\right) \left[ \frac{1}{n^2} e_{i,g+1}^{n-3} + \frac{1}{n} e_{i,g+2}^{n-3} \right] - e_{i,g+2}^{n-3} + \left(-1 + \frac{1}{n}\right) \sum_{\tau=3}^{n-2} e_{i,g+\tau}^{n-1-\tau}. \end{aligned} \quad (\text{A.11})$$

IV<sup>a</sup>. Pick the point (A.5) associated with  $g+1 \in J$  and any  $s \in [n-4]$ ,

$$\begin{aligned} & \frac{1}{n} \sum_{k \in N} e_{k,g}^n + \sum_{\tau=0}^{n-s-2} \frac{1}{n} e_{i,g+\tau+1}^{n-\tau-1} \\ & + \left(1 - \frac{1}{n}\right) \left[ \sum_{\tau=0}^{n-s-2} \frac{1}{n^{n-\tau-s-1}} e_{i,g+\tau+1}^s \right] + \frac{1}{n} \sum_{\tau=1}^{s-1} e_{i,g+n-1-s+\tau}^{s-\tau}. \end{aligned} \quad (\text{A.12})$$

Subtract (A.7) from (A.12) to get

$$\begin{aligned} & \sum_{\tau=0}^{n-s-2} \frac{1}{n} e_{i,g+\tau+1}^{n-\tau-1} + \left(1 - \frac{1}{n}\right) \left[ \sum_{\tau=0}^{n-s-2} \frac{1}{n^{n-\tau-s-1}} e_{i,g+\tau+1}^s \right] \\ & + \frac{1}{n} \sum_{\tau=1}^{s-1} e_{i,g+n-1-s+\tau}^{s-\tau} - \left(1 - \frac{1}{n}\right) e_{i,g}^{n-1} - \sum_{\tau=1}^{n-2} e_{i,g+\tau}^{n-1-\tau}, \end{aligned}$$

which is equivalent to

$$\begin{aligned} & \left(-1 + \frac{1}{n}\right) e_{i,g}^{n-1} + \frac{1}{n} e_{i,g+1}^{n-1} + \sum_{\tau=1}^{n-s-2} \left[ \frac{1}{n} e_{i,g+\tau+1}^{n-\tau-1} - e_{i,g+\tau}^{n-\tau-1} \right] \\ & + \left(1 - \frac{1}{n}\right) \left[ \sum_{\tau=0}^{n-s-2} \frac{1}{n^{n-\tau-s-1}} e_{i,g+\tau+1}^s \right] - e_{i,g+n-s-1}^s + \left(-1 + \frac{1}{n}\right) \sum_{\tau=n-s}^{n-2} e_{i,g+\tau}^{n-1-\tau}. \end{aligned} \quad (\text{A.13})$$

Since  $B_1$  is a diagonal matrix, for the rest of the proof we focus on the matrix  $C$ , formed by points in families II<sup>a</sup>, III<sup>a</sup>, IV<sup>a</sup> and V. Next, we apply Gaussian elimination on  $C$ .

III<sup>b</sup>. Subtract (A.9) from (A.11),

$$\begin{aligned}
& \left(-1 + \frac{1}{n}\right) e_{i,g}^{n-1} + \frac{1}{n} e_{i,g+1}^{n-1} + \frac{1}{n} e_{i,g+2}^{n-2} - e_{i,g+1}^{n-2} + \left(1 - \frac{1}{n}\right) \left[ \frac{1}{n^2} e_{i,g+1}^{n-3} \right. \\
& \quad \left. + \frac{1}{n} e_{i,g+2}^{n-3} \right] - e_{i,g+2}^{n-3} + \left(-1 + \frac{1}{n}\right) \sum_{\tau=3}^{n-2} e_{i,g+\tau}^{n-1-\tau} \\
& \quad - \frac{1}{n} e_{i,g+1}^{n-1} - \left(-1 + \frac{1}{n}\right) e_{i,g}^{n-1} - \left(\frac{1}{n} - \frac{1}{n^2} - 1\right) e_{i,g+1}^{n-2} \\
& \quad - \left(-1 + \frac{1}{n}\right) \sum_{\tau=2}^{n-2} e_{i,g+\tau}^{n-1-\tau},
\end{aligned}$$

which is equivalent to

$$\left(-\frac{1}{n} + \frac{1}{n^2}\right) e_{i,g+1}^{n-2} + \frac{1}{n} e_{i,g+2}^{n-2} + \left(\frac{1}{n^2} - \frac{1}{n^3}\right) e_{i,g+1}^{n-3} - \frac{1}{n^2} e_{i,g+2}^{n-3}. \quad (\text{A.14})$$

IV<sup>b</sup>. For every  $s \in [n-4]$ , subtract (A.9) from (A.13),

$$\begin{aligned}
& \left(-1 + \frac{1}{n}\right) e_{i,g}^{n-1} + \frac{1}{n} e_{i,g+1}^{n-1} + \sum_{\tau=1}^{n-s-2} \left[ \frac{1}{n} e_{i,g+\tau+1}^{n-\tau-1} - e_{i,g+\tau}^{n-\tau-1} \right] \\
& \quad + \left(1 - \frac{1}{n}\right) \left[ \sum_{\tau=0}^{n-s-2} \frac{1}{n^{n-\tau-s-1}} e_{i,g+\tau+1}^s \right] - e_{i,g+n-s-1}^s \\
& \quad + \left(-1 + \frac{1}{n}\right) \sum_{\tau=n-s}^{n-2} e_{i,g+\tau}^{n-1-\tau} \\
& \quad - \frac{1}{n} e_{i,g+1}^{n-1} - \left(-1 + \frac{1}{n}\right) e_{i,g}^{n-1} - \left(\frac{1}{n} - \frac{1}{n^2} - 1\right) e_{i,g+1}^{n-2} \\
& \quad - \left(-1 + \frac{1}{n}\right) \sum_{\tau=2}^{n-2} e_{i,g+\tau}^{n-1-\tau},
\end{aligned}$$

which is equivalent to

$$\begin{aligned}
& \left(-\frac{1}{n} + \frac{1}{n^2}\right) e_{i,g+1}^{n-2} + \frac{1}{n} e_{i,g+2}^{n-2} + \frac{1}{n} \sum_{\tau=2}^{n-s-2} \left[ e_{i,g+\tau+1}^{n-\tau-1} - e_{i,g+\tau}^{n-\tau-1} \right] \\
& \quad + \left(1 - \frac{1}{n}\right) \left[ \sum_{\tau=0}^{n-s-3} \frac{1}{n^{n-\tau-s-1}} e_{i,g+\tau+1}^s \right] - \frac{1}{n^2} e_{i,g+n-s-1}^s. \quad (\text{A.15})
\end{aligned}$$

IV<sup>c</sup>. For every  $s \in [n - 4]$ , subtract (A.14) from (A.15),

$$\begin{aligned} & \left(-\frac{1}{n} + \frac{1}{n^2}\right) e_{i,g+1}^{n-2} + \frac{1}{n} e_{i,g+2}^{n-2} + \frac{1}{n} \sum_{\tau=2}^{n-s-2} [e_{i,g+\tau+1}^{n-\tau-1} - e_{i,g+\tau}^{n-\tau-1}] \\ & + \left(1 - \frac{1}{n}\right) \left[ \sum_{\tau=0}^{n-s-3} \frac{1}{n^{n-\tau-s-1}} e_{i,g+\tau+1}^s \right] - \frac{1}{n^2} e_{i,g+n-s-1}^s \\ & - \left(-\frac{1}{n} + \frac{1}{n^2}\right) e_{i,g+1}^{n-2} - \frac{1}{n} e_{i,g+2}^{n-2} - \left(\frac{1}{n^2} - \frac{1}{n^3}\right) e_{i,g+1}^{n-3} + \frac{1}{n^2} e_{i,g+2}^{n-3}, \end{aligned}$$

which is equivalent to

$$\begin{aligned} & \frac{1}{n} e_{i,g+3}^{n-3} - \left(\frac{1}{n^2} - \frac{1}{n^3}\right) e_{i,g+1}^{n-3} + \left(\frac{1}{n^2} - \frac{1}{n}\right) e_{i,g+2}^{n-3} + \frac{1}{n} \sum_{\tau=3}^{n-s-2} [e_{i,g+\tau+1}^{n-\tau-1} - e_{i,g+\tau}^{n-\tau-1}] \\ & + \left(1 - \frac{1}{n}\right) \left[ \sum_{\tau=0}^{n-s-3} \frac{1}{n^{n-\tau-s-1}} e_{i,g+\tau+1}^s \right] - \frac{1}{n^2} e_{i,g+n-s-1}^s. \end{aligned} \quad (\text{A.16})$$

IV<sup>d</sup>. For every  $s \in [n - 5]$ , subtract (A.16) corresponding to  $s + 1$  from (A.16) corresponding to  $s$ ,

$$\begin{aligned} & \frac{1}{n} e_{i,g+3}^{n-3} - \left(\frac{1}{n^2} - \frac{1}{n^3}\right) e_{i,g+1}^{n-3} + \left(\frac{1}{n^2} - \frac{1}{n}\right) e_{i,g+2}^{n-3} + \frac{1}{n} \sum_{\tau=3}^{n-s-2} [e_{i,g+\tau+1}^{n-\tau-1} - e_{i,g+\tau}^{n-\tau-1}] \\ & + \left(1 - \frac{1}{n}\right) \left[ \sum_{\tau=0}^{n-s-3} \frac{1}{n^{n-\tau-s-1}} e_{i,g+\tau+1}^s \right] - \frac{1}{n^2} e_{i,g+n-s-1}^s \\ & - \frac{1}{n} e_{i,g+3}^{n-3} + \left(\frac{1}{n^2} - \frac{1}{n^3}\right) e_{i,g+1}^{n-3} - \left(\frac{1}{n^2} - \frac{1}{n}\right) e_{i,g+2}^{n-3} \\ & - \frac{1}{n} \sum_{\tau=3}^{n-s-3} [e_{i,g+\tau+1}^{n-\tau-1} - e_{i,g+\tau}^{n-\tau-1}] \\ & - \left(1 - \frac{1}{n}\right) \left[ \sum_{\tau=0}^{n-s-4} \frac{1}{n^{n-\tau-s-2}} e_{i,g+\tau+1}^{s+1} \right] + \frac{1}{n^2} e_{i,g+n-s-2}^{s+1}, \end{aligned}$$

which is equivalent to

$$\frac{1}{n} e_{i,g+n-s-1}^{s+1} + \left(\frac{1}{n^2} - \frac{1}{n}\right) e_{i,g+n-s-2}^{s+1} + \left(-1 + \frac{1}{n}\right) \left[ \sum_{\tau=0}^{n-s-4} \frac{1}{n^{n-\tau-s-2}} e_{i,g+\tau+1}^{s+1} \right]$$

$$\begin{aligned}
& -\frac{1}{n^2}e_{i,g+n-s-1}^s + \left(\frac{1}{n^2} - \frac{1}{n^3}\right)e_{i,g+n-s-2}^s \\
& + \left(1 - \frac{1}{n}\right) \left[ \sum_{\tau=0}^{n-s-4} \frac{1}{n^{n-\tau-s-1}} e_{i,g+\tau+1}^s \right].
\end{aligned} \tag{A.17}$$

For  $s = n - 4$ , we do not need this step, since from (A.16) we have

$$\begin{aligned}
& \frac{1}{n}e_{i,g+3}^{n-3} + \left(\frac{1}{n^2} - \frac{1}{n}\right)e_{i,g+2}^{n-3} + \left(-\frac{1}{n^2} + \frac{1}{n^3}\right)e_{i,g+1}^{n-3} \\
& - \frac{1}{n^2}e_{i,g+3}^{n-4} + \left(\frac{1}{n^2} - \frac{1}{n^3}\right)e_{i,g+2}^{n-4} + \left(\frac{1}{n^3} - \frac{1}{n^4}\right)e_{i,g+1}^{n-4}.
\end{aligned}$$

Observe that for any  $s \in [n - 4]$  and  $g \in J$ , we can multiply row  $(i, g, s + 1)$  by  $-1/n$  and we get the entry in row  $(i, g, s)$ .

$\Pi^b$ . Finally, pick a point (A.6) in family V for  $g \in J$ ,

$$\frac{1}{n}e_{i,g}^{n-1} + \frac{1}{n}e_{i,g+1}^{n-2} + \cdots + \frac{1}{n}e_{i,g+n-2}^1. \tag{A.18}$$

Now multiply (A.18) by  $(1 - n)$  and subtract it from (A.9) for  $g \in J$ , yielding

$$\begin{aligned}
& \frac{1}{n}e_{i,g+1}^{n-1} + \left(-1 + \frac{1}{n}\right)e_{i,g}^{n-1} + \left(\frac{1}{n} - \frac{1}{n^2} - 1\right)e_{i,g+1}^{n-2} \\
& + \left(-1 + \frac{1}{n}\right) \sum_{\tau=2}^{n-2} e_{i,g+\tau}^{n-1-\tau} - \frac{1-n}{n} \sum_{\tau=0}^{n-2} e_{i,g+\tau}^{n-\tau-1},
\end{aligned}$$

which is equivalent to

$$\frac{1}{n}e_{i,g+1}^{n-1} - \frac{1}{n^2}e_{i,g+1}^{n-2}.$$

Since we have  $g + 1$  in those two stages, we have a general expression for any  $g \in J$ ,

$$\frac{1}{n}e_{i,g}^{n-1} - \frac{1}{n^2}e_{i,g}^{n-2}. \tag{A.19}$$

As before, we can multiply row  $(i, g, n - 1)$  by  $-1/n$  to get the entry in row  $(i, g, n -$

2).

Now, we can organize the points in  $C$  as

$$C = [\mathbf{V}, \mathbf{II}^b, \mathbf{III}^b, \mathbf{IV}_{n-4}^d, \dots, \mathbf{IV}_s^d, \dots, \mathbf{IV}_1^d],$$

where  $\mathbf{IV}_s^d$  corresponds to the block of points ( $g \in J$ ) with  $s \in [n-4]$ .  $C$  has the form

$$C = \begin{pmatrix} C_{n-1} & D_{n-2} & 0 & 0 & 0 & \dots & 0 & 0 \\ C_{n-2} & -\frac{1}{n}D_{n-2} & D_{n-3} & 0 & 0 & \dots & 0 & 0 \\ C_{n-3} & 0 & -\frac{1}{n}D_{n-3} & D_{n-4} & 0 & \dots & 0 & 0 \\ \vdots & & & \ddots & & & \vdots & \\ C_2 & 0 & 0 & 0 & 0 & \dots & -\frac{1}{n}D_2 & D_1 \\ C_1 & 0 & 0 & 0 & 0 & \dots & 0 & -\frac{1}{n}D_1 \end{pmatrix},$$

where every  $C_i$  and  $D_i$  are *circulant* matrices [67] of size  $(n-1) \times (n-1)$ . Since the determinant is invariant under elementary row and column operations, we can perform Gaussian elimination (of rows) from bottom to top, and we get

$$\bar{C} = \begin{pmatrix} \bar{C}_{n-1} & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ \bar{C}_{n-2} & -\frac{1}{n}D_{n-2} & 0 & 0 & 0 & \dots & 0 & 0 \\ \bar{C}_{n-3} & 0 & -\frac{1}{n}D_{n-3} & 0 & 0 & \dots & 0 & 0 \\ \vdots & & & \ddots & & & \vdots & \\ \bar{C}_2 & 0 & 0 & 0 & 0 & \dots & -\frac{1}{n}D_2 & 0 \\ C_1 & 0 & 0 & 0 & 0 & \dots & 0 & -\frac{1}{n}D_1 \end{pmatrix},$$

where

$$\begin{aligned} \bar{C}_{n-1} &= C_{n-1} + n(C_{n-2} - \dots n(C_2 + nC_1) \dots) \\ &= \text{circ}(1/n, 1, n, n^2, \dots, n^{n-3}), \end{aligned}$$

$$\begin{aligned}
D_{n-2} &= \text{circ}(1/n, 0, \dots, 0), \\
D_{n-3} &= \text{circ}\left(\frac{1}{n}, -\frac{1}{n} + \frac{1}{n^2}, 0, \dots, 0\right) \\
&\vdots \\
D_s &= \text{circ}\left(\frac{1}{n}, -\frac{1}{n} + \frac{1}{n^2}, -\frac{1}{n^2} + \frac{1}{n^3}, \dots, -\frac{1}{n^{n-s-2}} + \frac{1}{n^{n-s-1}}, 0, \dots, 0\right) \\
&\vdots \\
D_1 &= \text{circ}\left(\frac{1}{n}, -\frac{1}{n} + \frac{1}{n^2}, -\frac{1}{n^2} + \frac{1}{n^3}, \dots, -\frac{1}{n^{n-3}} + \frac{1}{n^{n-2}}\right)
\end{aligned}$$

For  $\bar{C}_{n-1}$ , the last entry,  $n^{n-3}$ , is greater than the sum of the remaining entries. The same applies for  $D_s$ , with entry  $1/n$ . Due to Proposition 18 in [67] all these matrices are nonsingular, so  $\bar{C}$  is nonsingular, and therefore  $C$  is nonsingular. This implies that  $A$  is nonsingular, showing that these points are linearly independent, and proceeding in the same way as we did in the proof of Proposition 8 for the remaining components, we can show (A.1) is facet-defining.  $\square$

*Proof of Theorem 8.* The proof is similar to Theorem 7. Again, assume  $h = n - 1$  and  $t = 1$ ; the remaining cases follow a similar construction of linearly independent points. Without loss of generality we can assume that  $i_\tau = i$  for all  $\tau \in [n - 2]$ . So, we have an inequality of the form

$$rZ_{N,J}^n + nZ_{I,J}^{n-1} + \sum_{\tau=1}^{n-2} n^{n-\tau} Z_{i,J}^\tau \leq r + \sum_{\tau=1}^{n-2} n^\tau. \quad (\text{A.20})$$

We construct the following linearly independent points.

- I. Fix  $k \in N$  and  $j \in J$ . In stage  $n$ , if node  $k$  appears, match it to node  $j$ , with probability  $1/n$ . For the remaining stages match according to  $(j, j + 1, \dots, j + n - 2) \in \text{circ}(J)$ . In terms of probability, if any  $k' \in I$  appears in stage  $n - 1$ , then it is matched to  $j$  with probability  $(1 - 1/n) \cdot 1/n$ , so the probability of matching  $j$  in stage  $n - 1$  is  $(1 - 1/n) \cdot r/n$ . For the rest, the probability is  $1/n$ . So, we have the

point

$$\frac{1}{n}e_{k,j}^n + \frac{1}{n} \left(1 - \frac{1}{n}\right) \sum_{k' \in I} e_{k',j}^{n-1} + \frac{1}{n} \sum_{\tau=1}^{n-2} e_{i,j+\tau}^{n-1-\tau}. \quad (\text{A.21})$$

By a simple calculation, it is easy to see that each of these points achieves the right-hand side of (A.20). Since we chose any arbitrary  $k \in N$  and  $j \in J$ , we have  $n(n-1)$  points in this family.

- II. Fix  $j \in J$  and  $k \in I$ . In this family we repeat the same ad to match in stages  $n-1$  and  $n-2$ . If  $k$  appears in stage  $n-1$ , match it to  $j$  with probability  $1/n$ . Then, if  $i$  appears in stage  $n-2$  and  $k$  did not appear in  $n-1$ , match it to  $j$  with probability  $(1-1/n) \cdot 1/n$ . For the remaining stages match according to  $(j+n-2, j, j+1, \dots, j+n-3) \in \text{circ}(J)$ ; in stage  $n$  match any  $k' \in N$  that appears with  $j+n-2$ , in stage  $n-3$  match  $i$  to  $j+1$  if it appears, and so on. So, we have the point

$$\frac{1}{n} \sum_{k' \in N} e_{k',j+n-2}^n + \frac{1}{n} e_{k,j}^{n-1} + \frac{1}{n} \left(1 - \frac{1}{n}\right) e_{i,j}^{n-2} + \frac{1}{n} \sum_{\tau=1}^{n-3} e_{i,j+\tau}^{n-\tau-2}. \quad (\text{A.22})$$

By a simple calculation, we get the right-hand side of (A.1). Finally, since we chose an arbitrary  $j \in J$  and  $k' \in I$ , we have  $r(n-1)$  points in this family.

- V. Fix  $j \in J$ . In this family we do not match in stage  $n$ , and in the remaining stages we match according to a vector in  $\text{circ}(J)$ . If any  $k \in I$  appears in stage  $n-1$ , match it to  $j$  with probability  $1/n$ , if  $i$  appears in stage  $n-2$ , match it to node  $j+1$ , and so forth. So we have the point

$$\frac{1}{n} \sum_{k \in I} e_{k,j}^{n-1} + \frac{1}{n} \sum_{\tau=1}^{n-2} e_{i,j+\tau}^{n-\tau-1}. \quad (\text{A.23})$$

By a simple calculation, we get the right-hand side of (A.1). Since we chose an arbitrary  $j \in J$ , we have  $n-1$  points in this family.

Families III, and IV remain the same as in the proof of Theorem 7, so in total we have

$(n - 1)(r + 2n - 2)$  points. The rest of the proof follows the same argument as Theorem 7. □

*Proof of Theorem 9.* The proof follows the same argument as the previous two theorems, the only difference being that the collection of points given by policies corresponding to  $I$  is bigger; however, they still form a diagonal block, so we can apply the same procedure. □

## A.2 Detailed Experiment Results

Table A.1: Experiment results for small instances.

Instance	(2.5) + (2.7)	(3.7)	(3.7) + (3.10)	Exp. Matching	(3.1)	(3.9)	(2.12)
S1	8.9613	8.7261	8.7170	8.5818	8.3976	8.3562	8.3452
S2	8.0125	7.7354	7.7147	7.4635	7.4055	7.4095	7.3980
S3	7.0252	6.8849	6.8795	6.8109	6.7241	6.7239	6.6546
S4	8.4799	8.1415	8.1282	8.1360	7.8468	7.8142	7.7895
S5	7.9805	7.4972	7.4654	7.3191	7.0849	7.0590	6.8223
S6	9.2650	8.5601	8.5289	8.4907	8.1364	8.1057	8.0802
S7	7.1327	6.9320	6.9155	6.7885	6.6895	6.6699	6.6709
S8	8.2993	7.8525	7.8181	7.6588	7.4762	7.4631	7.4203
S9	7.0193	6.6523	6.6320	6.4263	6.3451	6.3364	6.2574
S10	7.1206	6.9756	6.9700	6.9593	6.8103	6.7790	6.7588
S11	8.9684	8.5841	8.5519	8.3237	8.1223	8.0996	7.9973
S12	6.7110	6.4682	6.4537	6.3265	6.2555	6.2542	6.2542
S13	7.8013	7.5626	7.5543	7.7048	7.3845	7.3547	7.3122
S14	8.0801	7.7047	7.6880	7.5756	7.4003	7.3787	7.2813
S15	8.2220	8.0324	7.9922	7.9251	7.7369	7.7026	7.7021
S16	9.3714	8.8684	8.8560	8.9708	8.4948	8.4672	8.4034
S17	6.9684	6.8016	6.7739	6.6437	6.5785	6.5717	6.4977
S18	9.3936	8.8414	8.8248	8.8218	8.4484	8.4247	8.3458
S19	7.0304	6.7305	6.7138	6.5861	6.4776	6.4778	6.4772
S20	8.7322	8.4326	8.4139	8.2763	8.0769	8.0485	7.9651

Table A.2: Experiment results for large, dense instances.

Instance	(2.5) + (2.7)	(3.7)	Exp. Matching	(3.9)	(2.12)
LD1	99.8550	97.0381	99.8153	94.8957	94.8471
LD2	99.8035	97.1728	99.7964	95.5454	95.0983
LD3	99.9368	97.2471	99.8325	95.4772	95.1928
LD4	99.9083	97.3543	99.8998	95.7123	95.4918
LD5	99.9135	97.1967	99.9033	95.5889	95.3354
LD6	99.9533	97.3301	99.8303	95.5145	95.2998
LD7	99.9528	97.3809	99.9314	95.7929	95.3830
LD8	99.9100	97.3675	99.9006	95.7148	95.3640
LD9	99.7849	97.1600	99.7414	95.2710	95.0698
LD10	99.8933	97.2112	99.8805	95.6271	95.2136
LD11	99.9175	97.3080	99.8908	95.6526	95.1530
LD12	99.8668	97.3086	99.8597	95.5097	95.5914
LD13	99.7805	97.1135	99.6728	94.9733	94.6578
LD14	99.8641	97.3257	99.7695	95.3197	95.4821
LD15	99.9079	97.1715	99.8723	95.2706	95.2057
LD16	99.7806	96.8779	99.7579	94.8829	94.6222
LD17	99.7620	97.2239	99.7554	95.5676	95.4672
LD18	99.9504	97.3627	99.8417	95.7888	95.5311
LD19	99.9603	97.4715	99.9530	95.8706	95.7686
LD20	99.9392	97.1932	99.9263	95.2861	95.1938

Table A.3: Experiment results for large, sparse instances.

Instance	(2.5) + (2.7)	(3.7)	(3.7) + (3.10)	Exp. Matching	(3.9)	(2.12)
LS1	79.1015	74.7635	74.7260	72.3904	69.7186	69.0988
LS2	79.0703	75.3885	75.3555	73.3026	70.8925	70.2621
LS3	78.0157	74.3402	74.3051	72.0072	69.6943	69.1178
LS4	78.6983	74.6555	74.6181	72.3612	69.7633	69.1302
LS5	81.8877	76.7406	76.7085	74.9893	71.6122	70.7586
LS6	81.4750	77.6830	77.6517	75.5756	72.8110	72.3019
LS7	84.0754	79.2133	79.1825	77.4263	73.6374	72.3566
LS8	71.9270	69.5745	69.5391	67.6728	65.7709	65.4557
LS9	77.0648	73.8295	73.7947	71.4234	69.3153	68.5672
LS10	82.5481	77.2318	77.2028	75.3500	72.2398	71.4496
LS11	73.6532	70.8502	70.8139	68.2690	66.2972	64.9590
LS12	81.1134	77.2298	77.1911	74.8116	71.9705	71.3263
LS13	75.2453	72.2625	72.2297	70.0278	67.6604	66.7906
LS14	80.7172	75.8009	75.7654	74.1015	70.7923	70.3709
LS15	74.7313	72.2967	72.2627	69.7801	68.0195	67.1434
LS16	75.9530	72.2955	72.2691	70.2883	67.5794	66.6571
LS17	78.9469	74.2399	74.2093	72.6022	69.4289	68.6303
LS18	79.2340	74.8535	74.8261	73.3324	70.1805	69.6408
LS19	78.6156	75.6028	75.5745	73.7973	71.2080	70.6332
LS20	82.8804	78.4299	78.3911	76.6972	73.1519	72.9224

**APPENDIX B**  
**EXTRA MATERIAL FOR CHAPTER 4**

**B.1 Remaining Proofs**

*Proof of Lemma 4.* For every  $t \in [T]$  define a matroid  $\mathcal{M}_t = (V \times \{t\}, \mathcal{I} \times \{t\}) = (V_t, \mathcal{I}_t)$ . Given this, the union matroid is given by a ground set  $V^{[T]} = \bigcup_{t=1}^T V_t$ , and independent set family  $\mathcal{I}^{[T]} = \{S \subseteq V^{1:T} : S \cap V_t \in \mathcal{I}_t\}$ . Define  $\mathbb{H}(X) := \sum_{t \in [T]} H^t(x^t)$  for any matrix  $X \in \mathcal{P}(\mathcal{M})^T$ , where  $x^t$  denotes the  $t$ -th column of  $X$ . Clearly,  $\nabla_{(e,t)} \mathbb{H}(X) = \nabla_e H^t(x^t)$ . Moreover, the Hessian corresponds to

$$\nabla_{(e_1,t),(e_2,s)}^2 \mathbb{H}(X) = \begin{cases} 0 & \text{if } t \neq s \\ \nabla_{e_1,e_2}^2 H^t(x^t) & \text{if } t = s \end{cases}$$

Consider any  $X, Y \in \mathcal{P}(\mathcal{M})^T$  with  $|y_e^t - x_e^t| \leq \delta$ . Therefore, a Taylor's expansion of  $\mathbb{H}$  gives

$$\mathbb{H}(Y) = \mathbb{H}(X) + \nabla \mathbb{H}(X) \cdot (Y - X) + \frac{1}{2} (Y - X)^\top \nabla^2 \mathbb{H}(\xi) \cdot (Y - X)$$

where  $\xi$  is on the line between  $X$  and  $Y$ . If we expand the previous expression we obtain

$$\mathbb{H}(Y) - \mathbb{H}(X) = \sum_{e \in V} \sum_{t \in [T]} \nabla_e H^t(x^t) (y_e^t - x_e^t) + \frac{1}{2} \sum_{e_1, e_2 \in V} \sum_{t \in [T]} (y_{e_1}^t - x_{e_1}^t) \nabla_{e_1, e_2}^2 H^t(\xi) (y_{e_2}^t - x_{e_2}^t)$$

Finally, by using property 3 in Lemma 3 and by bounding the Hessian (and using the fact that  $g_i^t(x) \in [0, 1]$ ) we get

$$\mathbb{H}(Y) - \mathbb{H}(X) \geq \sum_{e \in V} \sum_{t=1}^T \nabla_e H^t(x^t) (y_e^t - x_e^t) - O(Tn^2 \delta^2 \alpha),$$

which is equivalent to

$$\sum_{t \in [T]} H^t(y^t) - \sum_{t \in [T]} H^t(x^t) \geq \sum_{e \in V} \sum_{t \in [T]} \nabla_e H^t(x^t)(y_e^t - x_e^t) - O(Tn^2\delta^2\alpha).$$

□

## B.2 Extended Stochastic Greedy for Partition Matroid

Consider a partition  $\{P_1, \dots, P_q\}$  on ground set  $V$  with  $n_j := |P_j|$  for all  $j \in [q]$  and a family of feasible sets  $\mathcal{I} = \{S \subseteq V : |S \cap P_j| \leq k_j \forall j \in [q]\}$  which for a matroid  $\mathcal{M} = (V, \mathcal{I})$ . We can construct a heuristic based on the stochastic greedy algorithm [84] and adapted to partition constraints (Algorithm 12): given  $\epsilon' > 0$ , in each round it uniformly samples  $\frac{n_j}{b} \log \frac{1}{\epsilon'}$  elements from each part  $R_j \sim P_j$ , where  $n_j := |P_j|$ . And then, it obtains the element with the largest marginal value among elements in  $\cup_{j \in [q]} R_j$ .

Even though, we are not able to state any provable guarantee, we use Algorithm 12 as inner loop for solving the robust problem (4.1) with  $\ell = \lceil \log \frac{2k}{\epsilon} \rceil$ .

---

### Algorithm 12 Extended Stochastic-Greedy for Partition Matroid

---

**Input:**  $\ell \geq 1$ , monotone submodular function  $g : 2^V \rightarrow \mathbb{R}_+$ , partition matroid  $\mathcal{M} = (V, \mathcal{I})$ ,  $\epsilon' > 0$ .

**Output:** sets  $S_1, \dots, S_\ell \in \mathcal{I}$ .

- 1: **for**  $\tau = 1, \dots, \ell$  **do**
  - 2:      $S_\tau \leftarrow \emptyset$
  - 3:     **while**  $S_\tau$  is not basis in  $\mathcal{M}$  **do**
  - 4:         For each  $j \in [q]$ , uniformly sample  $R_j \sim P_j \setminus S_\tau$  with  $\frac{n_j}{k_j} \log \frac{1}{\epsilon'}$  elements.
  - 5:          $e^* \leftarrow \operatorname{argmax}_{e \in R_1 \cup \dots \cup R_r} \left\{ g_{\cup_{j=1}^r S_j}(e) \right\}$ .
  - 6:          $S_\tau \leftarrow S_\tau + e^*$ .
- 

## B.3 Pseudo-Code for the Offline Bi-criteria Algorithm

In this section, we present the pseudo-code of the main algorithm that we use for the experiments in Section 4.2.2. Algorithm 13 works as follows: in an outer loop we obtain an esti-

mate  $\gamma$  on the value of the optimal solution OPT via a binary search. For each guess  $\gamma$  we define a set function  $g^\gamma(S) := \frac{1}{k} \sum_{i \in [k]} \min\{f_i(S), \gamma\}$ . Then, we run an extended algorithm  $\mathcal{A}$  (either the standard greedy, threshold greedy, or the adapted version of the stochastic greedy) on  $g^\gamma$ . If at some point the solution  $S$  satisfies  $\min_{i \in [k]} f_i(S) \geq (1 - \epsilon/2)\gamma$ , we stop and update the lower  $\text{LB} = \min_{i \in [k]} f_i(S)$ , since we find a good candidate. Otherwise, we continue. After finishing round  $\tau$ , we check if we realize the guarantee  $g^\gamma(\cup_{j=1}^\tau S_j) \geq \alpha_\tau \cdot \gamma$ . If not, then we stop and update the upper bound  $\text{UB} = \gamma$ , otherwise we continue. Finally, we stop the binary search whenever LB and UB are sufficiently close. We consider factor guarantees (line 17 in Algorithm 13)  $\alpha_\tau = 1 - 1/2^\tau$  for E-G and or E-StochG, and  $\alpha_\tau = 1 - 1/(2 - \delta)^\tau$  for E-ThG.

---

**Algorithm 13** Pseudo-code to get bi-criteria solutions

---

**Input:**  $\epsilon > 0$ , monotone submodular functions  $\{f_i\}_{i \in [k]}$ , partition matroid  $P_1, \dots, P_q$ , and subroutine  $\mathcal{A}$ .

**Output:** sets  $S_1, \dots, S_\ell \in \mathcal{I}$ .

```
1: Compute LB and UB as stated above.
2: while  $\frac{UB-LB}{UB} > 2\epsilon$  do
3:    $\gamma = (UB + LB)/2$ 
4:   for  $\tau = 1, \dots, \ell$  do
5:      $S_\tau = \emptyset$ .
6:     Compute marginals  $\rho(e) = g^\gamma(S + e) - g^\gamma(S)$  for all  $e \in V$ .
7:     if  $\max_e \rho(e) \leq 0$  then
8:       if  $\min_i f_i(\cup_{j=1}^\tau S_j) \geq (1 - \epsilon)\gamma$  then
9:         Update  $LB = \min_i f_i(\cup_{j=1}^\tau S_j)$ 
10:        else
11:          Update  $UB = \gamma$ 
12:          Break
13:        else
14:          Obtain  $S_\tau \leftarrow \mathcal{A}(g^\gamma, \cup_{j=1}^{\tau-1} S_j)$ 
15:
16:          if  $g^\gamma(\cup_{j=1}^\tau S_j) < \alpha_\tau \cdot \gamma$  then
17:            Update  $UB = \gamma$ .
18:            Break
19:          else
20:            if  $\min_i f_i(\cup_{j=1}^\tau S_j) \geq (1 - \epsilon)\gamma$  then
21:              Update  $LB = \min_i f_i(\cup_{j=1}^\tau S_j)$ 
22:              Break
23:            else
24:              Continue
```

---

## REFERENCES

- [1] Atila Abdulkadiroğlu and Tayfun Sönmez. “House Allocation with Existing Tenants”. In: *Journal of Economic Theory* 88.2 (1999), pp. 233–260.
- [2] Daniel Adelman. “Dynamic Bid Prices in Revenue Management”. In: *Operations Research* 55 (2007), pp. 647–661.
- [3] A.A. Ageev and M.I. Sviridenko. “Pipe Rounding: A New Method of Constructing Algorithms with Proven Performance Guarantee”. In: *Journal of Combinatorial Optimization* 8.3 (2004), pp. 307–328.
- [4] Susanne Albers. “Online algorithms: a survey”. In: *Mathematical Programming* 97.1 (2003), pp. 3–26.
- [5] Ashwinkumar Badanidiyuru and Jan Vondrák. “Fast algorithms for maximizing submodular functions”. In: *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2014, pp. 1497–1514.
- [6] Bahman Bahmani and Michael Kapralov. “Improved Bounds for Online Stochastic Matching”. In: *Proceedings of the 18th Annual European Conference on Algorithms: Part I*. 2010, pp. 170–181.
- [7] Maria-Florina Balcan, Avrim Blum, and Anupam Gupta. “Approximate Clustering Without the Approximation”. In: *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA ’09. Society for Industrial and Applied Mathematics, 2009, pp. 1068–1077.
- [8] Maria-Florina Balcan and M. Braverman. “Nash equilibria in perturbation-stable games”. In: *Theory of Computing* 13.13 (2017), pp. 1–31.
- [9] S. Ben-David, A. Borodin, R. Karp, G. Tardos, and A. Wigderson. “On the Power of Randomization in Online Algorithms”. In: *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing (STOC)*. 1990, pp. 379–386.
- [10] Shai Ben-David. “Computational Feasibility of Clustering under Clusterability Assumptions”. In: *CoRR* abs/1501.00437 (2015).
- [11] Dimitri Bertsimas and José Niño-Mora. “Conservation Laws, Extended Polymatroids and Multi-Armed Bandit Problems; A Polyhedral Approach to Indexable Systems”. In: *Mathematics of Operations Research* 21 (1996), pp. 257–306.

- [12] Yonatan Bilu and Nathan Linial. “Are Stable Instances Easy?” In: *Combinatorics, Probability and Computing* 21.5 (2012), 643–660.
- [13] Benjamin Birnbaum and Claire Mathieu. “On-Line Bipartite Matching Made Simple”. In: *ACM SIGACT News* 39 (2008), pp. 80–87.
- [14] Ilija Bogunovic, Slobodan Mitrovic, Jonathan Scarlett, and Volkan Cevher. “Robust Submodular Maximization: A Non-Uniform Partitioning Approach”. In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 2017, pp. 508–516.
- [15] Jérôme Bolte, Aris Daniilidis, and Adrian Lewis. “The Lojasiewicz Inequality for Nonsmooth Subanalytic Functions with Applications to Subgradient Dynamical Systems”. In: *SIAM Journal on Optimization* 17.4 (2007), pp. 1205–1223.
- [16] Jérôme Bolte, Shoham Sabach, and Marc Teboulle. “Proximal alternating linearized minimization for nonconvex and nonsmooth problems”. In: *Mathematical Programming* 146.1 (2014), pp. 459–494.
- [17] Brian Brubach, Karthik A. Sankararaman, Aravind Srinivasan, and Pan Xu. “On-line Stochastic Matching: New Algorithms and Bounds”. arXiv. 2017.
- [18] Sébastien Bubeck and Nicolò Cesa-Bianchi. “Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems”. In: *Foundations and Trends® in Machine Learning* 5.1 (2012), pp. 1–122.
- [19] Niv Buchbinder and Moran Feldman. “Deterministic Algorithms for Submodular Maximization Problems”. In: *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2016, pp. 392–403.
- [20] Niv Buchbinder, Moran Feldman, and Roy Schwartz. “Comparing Apples and Oranges: Query Trade-off in Submodular Maximization”. In: *Mathematics of Operations Research* 42.2 (2016), pp. 308–329.
- [21] Niv Buchbinder, Moran Feldman, Joseph Seffi, and Roy Schwartz. “A tight linear time (1/2)-approximation for unconstrained submodular maximization”. In: *SIAM Journal on Computing* 44.5 (2015), pp. 1384–1402.
- [22] Niv Buchbinder, Moran Feldman, Joseph Seffi Naor, and Roy Schwartz. “Submodular maximization with cardinality constraints”. In: *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2014, pp. 1433–1452.
- [23] Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. “Maximizing a monotone submodular function subject to a matroid constraint”. In: *SIAM Journal on Computing* 40.6 (2011), pp. 1740–1766.

- [24] Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.
- [25] Vaggos Chatziafratis, Tim Roughgarden, and Jan Vondrák. “Stability and Recovery for Independence Systems”. In: *Proceedings of the 25th European Symposium on Algorithms (ESA)*. 2017.
- [26] Chandra Chekuri, Jan Vondrak, and Rico Zenklusen. “Dependent randomized rounding via exchange properties of combinatorial structures”. In: *Proceedings of the 51st Annual Symposium on Foundations of Computer Science (FOCS)*. 2010, pp. 575–584.
- [27] Robert S. Chen, Brendan Lucier, Yaron Singer, and Vasilis Syrgkanis. “Robust optimization for non-convex objectives”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*. 2017, pp. 4708–4717.
- [28] Wei Chen, Tian Lin, Zihan Tan, Mingfei Zhao, and Xuren Zhou. “Robust Influence Maximization”. In: *Proceedings of the 22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. 2016, pp. 795–804.
- [29] Edward G. Coffman and Isi Mitrani. “A Characterization of Waiting Time Performance Realizable by Single-Server Queues”. In: *Operations Research* 28 (1980), pp. 810–821.
- [30] Michele Conforti and Gérard Cornuéjols. “Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the Rado-Edmonds theorem”. In: *Discrete Applied Mathematics* 7.3 (1984), pp. 251–274.
- [31] Amit Daniely, Nati Linial, and Michael E. Saks. “Clustering is difficult only when it does not matter”. In: *CoRR* abs/1205.4891 (2012).
- [32] Abhimanyu Das and David Kempe. “Algorithms for subset selection in linear regression”. In: *Proceedings of the 40th Annual ACM Symposium on the Theory of Computing (STOC)*. 2008, pp. 45–54.
- [33] Milind Dawande, Pinar Keskinocak, and Sridhar Tayur. “On the biclique problem in bipartite graphs”. GSIA Working Paper 1996-04, Carnegie Mellon University. 1996.
- [34] Gabrielle Demange, David Gale, and Marilda Sotomayor. “Multi-Item Auctions”. In: *Journal of Political Economy* 94.4 (1986), pp. 863–872.

- [35] Jack Edmonds. “Submodular Functions, Matroids, and Certain Polyhedra”. In: *Combinatorial Optimization — Eureka, You Shrink!: Papers Dedicated to Jack Edmonds 5th International Workshop Aussois, France, March 5–9, 2001 Revised Papers*. Springer Berlin Heidelberg, 2003, pp. 11–26.
- [36] Alina Ene and Huy L. Nguyen. “Constrained Submodular Maximization: Beyond  $1/e$ ”. In: *Proceedings of the 57th Annual Symposium on Foundations of Computer Science (FOCS)*. 2016, pp. 248–257.
- [37] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. <http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html>. 2012.
- [38] Uriel Feige. “A threshold of  $\log n$  for approximating set cover”. In: *Journal of the ACM (JACM)* 45.4 (1998), pp. 634–652.
- [39] Jon Feldman, Aranyak Mehta, Vahab Mirrokni, and Shan Muthukrishnan. “Online Stochastic Matching: Beating  $1 - 1/e$ ”. In: *Proceedings of the 50th Annual Symposium on Foundations of Computer Science (FOCS)*. 2009, pp. 117–126.
- [40] Moran Feldman. “Guess Free Maximization of Submodular and Linear Sums”. arXiv. 2018.
- [41] Moran Feldman, Joseph Naor, and Roy Schwartz. “A unified continuous greedy algorithm for submodular maximization”. In: *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science (FOCS)*. 2011, pp. 570–579.
- [42] Marshall L. Fisher, George L. Nemhauser, and Laurence A. Wolsey. “An analysis of approximations for maximizing submodular set functions–II”. In: *Polyhedral combinatorics*. Springer, 1978, pp. 73–87.
- [43] D. Gale and L. S. Shapley. “College Admissions and the Stability of Marriage”. In: *The American Mathematical Monthly* 69.1 (1962), pp. 9–15.
- [44] G. Goel and A. Mehta. “Online Budgeted Matching in Random Input Models with Applications to Adwords”. In: *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2008, pp. 982–991.
- [45] Michel X. Goemans, Nicholas J. A. Harvey, Satoru Iwata, and Vahab Mirrokni. “Approximating Submodular Functions Everywhere”. In: *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2009, pp. 535–544.

- [46] Daniel Golovin, Andreas Krause, and Matthew Streeter. “Online Submodular Maximization under a Matroid Constraint with Application to Learning Assignments”. Technical Report, arXiv. 2014.
- [47] Paul Gözl and Ariel D. Procaccia. “Migration as Submodular Optimization”. In: *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*. 2019.
- [48] Ryan Gomes and Andreas Krause. “Budgeted Nonparametric Learning from Data Streams”. In: *Proceedings of the 27th International Conference on Machine Learning (ICML)*. 2010, pp. 391–398.
- [49] B. Haeupler, Vahab Mirrokni, and Morteza Zadimoghaddam. “Online Stochastic Weighted Matching: Improved Approximation Algorithms”. In: *Proceedings of the 7th Workshop on Internet and Network Economics (WINE)*. 2011, pp. 170–181.
- [50] F. Maxwell Harper and Joseph A. Konstan. “The MovieLens Datasets: History and Context”. In: *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5.4 (2016), 19:1–19.
- [51] Elad Hazan. “Introduction to Online Convex Optimization”. In: *Foundations and Trends® in Optimization* 2.3-4 (2016), pp. 157–325.
- [52] Xinran He and David Kempe. “Robust Influence Maximization”. In: *Proceedings of the 22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. 2016, pp. 885–894.
- [53] Xinran He and David Kempe. “Stability of Influence Maximization”. In: *Proceedings of the 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. KDD ’14. ACM, 2014, pp. 1256–1265.
- [54] Alan J. Hoffman. “On approximate solutions of systems of linear inequalities”. In: *Mathematical Programming* 49.4 (1952), pp. 263–265.
- [55] Victor P. Il’ev. “An approximation guarantee of the greedy descent algorithm for minimizing a supermodular set function”. In: *Discrete Applied Mathematics* 114.1 (2001), pp. 131–146.
- [56] Rishabh K Iyer, Stefanie Jegelka, and Jeff A Bilmes. “Curvature and Optimal Algorithms for Learning and Minimizing Submodular Functions”. In: *Proceedings of the 27th International Conference on Neural Information Processing Systems (NeurIPS)*. 2013, pp. 2742–2750.
- [57] Patrick Jaillet and Xin Lu. “Online Stochastic Matching: New Algorithms with Better Bounds”. In: *Mathematics of Operations Research* 39 (2014), pp. 624–646.

- [58] Adam Kalai and Santosh Vempala. “Efficient algorithms for online decision problems”. In: *Journal of Computer and System Sciences* 71.3 (2005), pp. 291–307.
- [59] Chinmay Karande, Aranyak Mehta, and Pushkar Tripathi. “Online Bipartite Matching with Unknown Distributions”. In: *Proceedings of the 43rd Annual ACM Symposium on the Theory of Computing (STOC)*. 2011, pp. 587–596.
- [60] Hamed Karimi, Julie Nutini, and Mark Schmidt. “Linear Convergence of Gradient and Proximal-Gradient Methods Under the Polyak-Lojasiewicz Condition”. In: *Machine Learning and Knowledge Discovery in Databases*. 2016, pp. 795–811.
- [61] R.M. Karp, U.V. Vazirani, and V.V. Vazirani. “An Optimal Algorithm for On-line Bipartite Matching”. In: *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing (STOC)*. 1990, pp. 352–358.
- [62] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. Vol. 344. Wiley-Interscience, 2009.
- [63] Ehsan Kazemi, Morteza Zadimoghaddam, and Amin Karbasi. “Scalable Deletion-Robust Submodular Maximization: Data Summarization with Privacy and Fairness Constraints”. In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*. Vol. 80. 2018, pp. 2544–2553.
- [64] David Kempe, Jon Kleinberg, and Éva Tardos. “Maximizing the Spread of Influence through a Social Network”. In: *Theory of Computing* 11.4 (2003), pp. 105–147.
- [65] Thomas Kerdreux, Alexandre d’Aspremont, and Sebastian Pokutta. “Restarting Frank-Wolfe”. arXiv. 2018.
- [66] V. Kolmogorov and R. Zabini. “What energy functions can be minimized via graph cuts?” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.2 (2004), pp. 147–159.
- [67] Irwin Kra and Santiago Simanca. “On Circulant Matrices”. In: *Notices of the AMS* 59 (2012), pp. 368–377.
- [68] Andreas Krause and Carlos Guestrin. “Near-optimal Nonmyopic Value of Information in Graphical Models”. In: *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI)*. 2005, pp. 324–331.
- [69] Andreas Krause and Carlos Guestrin. “Near-optimal Nonmyopic Value of Information in Graphical Models”. In: *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI)*. 2005, pp. 324–331.

- [70] Andreas Krause, Ajit Singh, and Carlos Guestrin. “Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies”. In: *Journal of Machine Learning Research* 9 (2008), pp. 235–284.
- [71] Andreas Krause, H. Brendan McMahan, Carlos Guestrin, and Anupam Gupta. “Robust submodular observation selection”. In: *Journal of Machine Learning Research* 9 (2008), pp. 2761–2801.
- [72] Andreas Krause, Ram Rajagopal, Anupam Gupta, and Carlos Guestrin. “Simultaneous Placement and Scheduling of Sensors”. In: *Proceedings of the 8th ACM/IEEE Conference on Information Processing in Sensor Networks (IPSN)*. 2009, pp. 181–192.
- [73] Sumit Kunnumkal and Kalyan Talluri. “On a Piecewise-Linear Approximation for Network Revenue Management”. In: *Mathematics of Operations Research* 41.1 (2016), pp. 72–91.
- [74] Jon Lee, Vahab S Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. “Non-monotone submodular maximization under matroid and knapsack constraints”. In: *Proceedings of the 41st Annual ACM Symposium on the Theory of Computing (STOC)*. 2009, pp. 323–332.
- [75] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. “Cost-effective Outbreak Detection in Networks”. In: *Proceedings of the 13th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. 2007, pp. 420–429.
- [76] Hui Lin and Jeff A. Bilmes. “How to select a good training-data subset for transcription: submodular active selection for sequences”. In: *Proceedings of the 10th Annual Conference of the International Speech Communication Association (INTERSPEECH)*. 2009, pp. 2859–2862.
- [77] Stanislas Lojasiewicz. “Sur la géométrie semi-et sous-analytique”. In: *Annales de l’Institut Fourier* 43.5 (1993), pp. 1575–1595.
- [78] Stanislas Lojasiewicz. “Une propriété topologique des sous-ensembles analytiques réels”. In: *Les équations aux dérivées partielles* (1963), pp. 1575–1595.
- [79] Mohammad Mahdian and Qiqi Yan. “Online Bipartite Matching with Random Arrivals: An Approach Based on Strongly Factor-Revealing LPs”. In: *Proceedings of the 43rd Annual ACM Symposium on the Theory of Computing (STOC)*. 2011, pp. 597–606.

- [80] Vahideh H. Manshadi, Shayan Oveis Gharan, and Amin Saberi. “Online Stochastic Matching: Online Actions Based on Offline Statistics”. In: *Mathematics of Operations Research* 37 (2012), pp. 559–573.
- [81] Aranyak Mehta. “Online Matching and Ad Allocation”. In: *Foundations and Trends in Theoretical Computer Science* 8 (2013), pp. 265–368.
- [82] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. “Adwords and generalized online matching”. In: *Journal of the ACM* 54 (2007), 22:1–22:19.
- [83] M. Minoux. “Accelerated greedy algorithms for maximizing submodular set functions”. In: *Optimization Techniques, LNCS* (1978), pp. 234–243.
- [84] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. “Lazier Than Lazy Greedy”. In: *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*. 2015, pp. 1812–1818.
- [85] Marko Mitrovic, Ehsan Kazemi, Morteza Zadimoghaddam, and Amin Karbasi. “Data Summarization at Scale: A Two-Stage Submodular Approach”. In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*. 2018, pp. 3593–3602.
- [86] George L. Nemhauser and Laurence A. Wolsey. “Best Algorithms for Approximating the Maximum of a Submodular Set Function”. In: *Mathematics of Operations Research* 3.3 (1978), pp. 177–188.
- [87] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. “An analysis of approximations for maximizing submodular set functions—I”. In: *Mathematical Programming* 14.1 (1978), pp. 265–294.
- [88] Arkadi S. Nemirovskii and Yurii E. Nesterov. “Optimal methods of smooth convex minimization”. In: *USSR Computational Mathematics and Mathematical Physics* 25.2 (1985), pp. 21–30.
- [89] James B. Orlin, Andreas S. Schulz, and Rajan Udawani. “Robust Monotone Submodular Function Maximization”. In: *Proceedings of the 18th International Conference on Integer Programming and Combinatorial Optimization (IPCO)*. 2016, pp. 312–324.
- [90] B. Polyak. “Sharp minima”. Presented in Workshop on generalized Lagrangians and their applications (IIASA). 1979.
- [91] Thomas Powers, Jeff Bilmes, Scott Wisdom, David W Krout, and Les Atlas. “Constrained robust submodular optimization”. In: *NIPS OPT2016 workshop*. 2016.

- [92] Thomas Powers, Jeff Bilmes, David W. Krout, and Les Atlas. “Constrained robust submodular sensor selection with applications to multistatic sonar arrays”. In: *Proceedings of the 19th International Conference on Information Fusion (FUSION)*. 2016, pp. 2179–2185.
- [93] Sharath Raghvendra. “A Robust and Optimal Online Algorithm for Minimum Metric Bipartite Matching”. In: *Proceedings of the 19th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX) and the 20th International Workshop on Randomization and Computation (RANDOM)*. 2016, 18:1–18:16.
- [94] Alexander Rakhlin. “Lecture notes on online learning”. In: *Draft, April (2009)*.
- [95] Alvin E. Roth, Tayfun Sönmez, and M. Utku Ünver. “Pairwise kidney exchange”. In: *Journal of Economic Theory* 125.2 (2005), pp. 151–188.
- [96] Vincent Roulet and Alexandre d’Aspremont. “Sharpness, Restart and Acceleration”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*. 2017, pp. 1119–1129.
- [97] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*. Vol. 24. Springer Science & Business Media, 2003.
- [98] Shai Shalev-Shwartz. “Online Learning and Online Convex Optimization”. In: *Found. Trends Mach. Learn.* 4.2 (2012), pp. 107–194.
- [99] Daniel D. Sleator and Robert E. Tarjan. “Amortized Efficiency of List Update and Paging Rules”. In: *Commun. ACM* 28.2 (1985), pp. 202–208.
- [100] Matthew Staib, Bryan Wilder, and Stefanie Jegelka. “Distributionally Robust Submodular Maximization”. In: *CoRR* abs/1802.05249 (2018).
- [101] Matthew Streeter and Daniel Golovin. “An Online Algorithm for Maximizing Submodular Functions”. In: *Proceedings of the 21st International Conference on Neural Information Processing Systems (NeurIPS)*. 2008, pp. 1577–1584.
- [102] Maxim Sviridenko. “A note on maximizing a submodular set function subject to a knapsack constraint”. In: *Operations Research Letters* 32.1 (2004), pp. 41–43.
- [103] Maxim Sviridenko, Jan Vondrák, and Justin Ward. “Optimal Approximation for Submodular and Supermodular Optimization with Bounded Curvature”. In: *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2015, pp. 1134–1148.

- [104] Kalyan Talluri and Garrett van Ryzin. “An analysis of bid-price controls for network revenue management”. In: *Management Science* 44 (1998), pp. 1577–1593.
- [105] Anthony Thompson. *Minkowski Geometry*. Cambridge University Press, 1996.
- [106] Chaoxu Tong and Huseyin Topaloglu. “On the approximate linear programming approach for network revenue management problems”. In: *INFORMS Journal on Computing* 26 (2014), pp. 121–134.
- [107] A. Tsanas, M. A. Little, P. E. McSharry, and L. O. Ramig. “Enhanced classical dysphonia measures and sparse regression for telemonitoring of Parkinson’s disease progression”. In: *Proceedings of the 35th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2010, pp. 594–597.
- [108] Jan Vondrák. “Optimal approximation for the submodular welfare problem in the value oracle model”. In: *Proceedings of the 40th Annual ACM Symposium on the Theory of Computing (STOC)*. 2008, pp. 67–74.
- [109] Jan Vondrak. “Submodularity and curvature: the optimal algorithm”. In: *RIMS Kokyuroku Bessatsu* 23 (2010).
- [110] Thomas Vossen and Dan Zhang. “Reductions of Approximate Linear Programs for Network Revenue Management”. In: *Operations Research* 63 (2015), pp. 1352–1371.
- [111] Bryan Wilder. “Equilibrium Computation and Robust Optimization in Zero Sum Games with Submodular Structure”. arXiv. 2017.
- [112] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.
- [113] Laurence A. Wolsey. “An analysis of the greedy algorithm for the submodular set covering problem”. In: *Combinatorica* 2.4 (1982), pp. 385–393.
- [114] Dan Zhang and Daniel Adelman. “An Approximate Dynamic Programming Approach to Network Revenue Management with Customer Choice”. In: *Transportation Science* 43 (2009), pp. 381–394.