Combinatorial Structures in Online and Convex Optimization

by

Swati Gupta

B. Tech & M. Tech (Dual Degree), Computer Science and Engineering,
Indian Institute of Technology (2011)

Submitted to the
Sloan School of Management
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Operations Research
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2017

Author . . . .
Sloan School of Management
May 19, 2017

Certified by . . . .
Michel X. Goemans
Leighton Family Professor
Department of Mathematics
Thesis Supervisor

Certified by . . . .
Patrick Jaillet
Dugald C. Jackson Professor
Department of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by . . . .
Dimitris Bertsimas
Boeing Leaders for Global Operations
Co-director, Operations Research Center

Combinatorial Structures in Online and Convex Optimization

by

Swati Gupta

Submitted to the Sloan School of Management
on May 19, 2017, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Operations Research

Abstract

Motivated by bottlenecks in algorithms across online and convex optimization, we consider three fundamental questions over combinatorial polytopes.

First, we study the minimization of separable strictly convex functions over polyhedra. This problem is motivated by first-order optimization methods whose bottleneck relies on the minimization of a (often) separable, convex metric, known as the Bregman divergence. We provide a conceptually simple algorithm, Inc-Fix, in the case of submodular base polyhedra. For cardinality-based submodular polytopes, we show that Inc-Fix can be speeded up to be the state-of-the-art method for minimizing uniform divergences. We show that the running time of Inc-Fix is independent of the convexity parameters of the objective function.

The second question is concerned with the complexity of the parametric line search problem in the extended submodular polytope P: starting from a point inside P, how far can one move along a given direction while maintaining feasibility. This problem arises as a bottleneck in many algorithmic applications like the above-mentioned Inc-Fix algorithm and variants of the Frank-Wolfe method. One of the most natural approaches is to use the discrete Newton's method, however, no upper bound on the number of iterations for this method was known. We show a quadratic bound resulting in a factor of n^6 reduction in the worst-case running time from the previous state-of-the-art. The analysis leads to interesting extremal questions on set systems and submodular functions.

Next, we develop a general framework to simulate the well-known multiplicative weights update algorithm for online linear optimization over combinatorial strategies $\mathcal{U}$ in time polynomial in $\log |\mathcal{U}|$, using efficient approximate general counting oracles. We further show that efficient counting over the vertex set of any 0/1 polytope P implies efficient convex minimization over P. As a byproduct of this result, we can approximately decompose any point in a 0/1 polytope into a product distribution over its vertices.

Finally, we compare the applicability and limitations of the above results in the context of finding Nash-equilibria in combinatorial two-player zero-sum games with bilinear loss functions. We prove structural results that can be used to find certain Nash-equilibria with a single separable convex minimization.

Thesis Supervisor: Michel X. Goemans
Title: Leighton Family Professor
Department of Mathematics

Thesis Supervisor: Patrick Jaillet
Title: Dugald C. Jackson Professor
Department of Electrical Engineering and Computer Science

Acknowledgments

"As we express our gratitude, we must never forget that the highest appreciation is not to utter words, but to live by them." - John F. Kennedy.

My journey at MIT would not have been this wonderful without selfless mentorship, close friendships, and the love of many.

My deepest gratitude goes to my advisors Michel Goemans and Patrick Jaillet. Throughout the past six years, Michel has amazed me with his enthusiasm for mathematics, for proving things in the best way possible, and his patience in improving my technical writing. I deeply appreciate the considerable amount of time and effort that he has put forth while working with me. I really admire Patrick for providing direction to research, his approachability and extraordinary work ethic. His mentorship and positivity has been an inspiration and I would like to thank Patrick for always lending me a friendly ear whenever I was in doubt or needed any help. It has been an absolute pleasure to learn from Michel and Patrick and I will always cherish our research meetings together. I am really thankful to them for giving me the freedom to pursue different research ideas and a lot of extremely valuable advice in making critical career decisions. I can think of no two other faculty members who would be such great co-advisors, and I will always look up to you both for advice and guidance! I would also like to thank Rico Zenklusen for his mentorship and friendship during my first year at MIT, and will always remember fondly our conversation about calling professors by their first name.

I will forever be grateful to Jim Orlin for providing me valuable feedback on my writing and presentation skills, for being on my thesis committee and advising me about the academic job market. I had the opportunity to be a teaching assistant for a course taught by Jim, and this was a great experience for me that helped me strengthen my decision to be in academia. I feel fortunate to have been a student at MIT during Sasha Rakhlin's sabbatical here. I want to extend a special thanks to him for his infectious enthusiasm for online learning, for being on my thesis committee and for being an amazing mentor. I would like to thank Sasha for several exciting mathematical discussions and his unique perspective on the connections between learning and optimization.

I would like to wholeheartedly thank Georgia Perakis for always looking out for me, advising me and being the strong female role model I needed. I want to thank her for introducing me to the wonderful field of revenue management and pricing. Working with Georgia has made me think about OR practices feasible for the industry, given their practical business considerations. I am really touched that I made it to your tree of students Georgia, I have always felt like the "unofficial" member of your wonderful research group!

I would like to give heartfelt thanks to Dimitris Bertsimas for also looking out for me, checking up with me multiple times throughout graduate school, and for always being straight with me. I had the opportunity of working with Dimitris on a vehicle routing research project. My interactions with him have always given me something to think about, beyond solving the bottlenecks in our project. If I may say so, Dimitris had the strongest opinion orthogonal to my taste in research, but it has definitely expanded the convex hull of problems I care about and I want to really thank you for that! I will always look up to you and Georgia for advice in years to come.

I am forever grateful to Martin Demaine for being my external voice of reason and support. I love him for making me believe in myself, for inspiring me to think outside the box, for making me expand what I perceived as the boundary of my abilities. I am so thankful that you stopped by my ambigram stall at the art fair and we started this wonderful friendship. I will always cherish our brainstorming sessions on installations and art projects, and I hope that I can bring some of these to life one day.

There are many faculty members at the Operations Research Center who I have learnt from a lot, both in classes and during the seminars. I would like to especially thank Rob Fruend for being a great teacher, mentor and for his useful advice on convex optimization algorithms. The research presented in this thesis started with a question posed by Costis Daskalakis: whether the multiplicative weights update algorithm can be simulated for a large number of strategies, and I would like to give him heartfelt thanks for this beginning. Thank you for teaching us linear programming from a polyhedral perspective, Andreas Schulz, we miss you at MIT! I would also like to sincerely thank Laura Rose and Andrew Carvahlo for managing the deadlines and course requirements extremely well, despite my absent-mindedness. I had the opportunity to get to know Suvrit Sra and Stefanie Jegelka towards the end of my graduate studies. They were my eyes into the world of machine learning and it is thanks to them that I had the courage of sending a submission to NIPS and the workshops there. I would like to also thank all the wonderful seminar speakers at ORC, LIDS and CSAIL for thought provoking discussions.

Collaborations with multiple people have been one of the highlights of my journey at MIT. As someone once told me, make the most of your time in graduate school by talking to as many people as you can, and I am really glad I was able to. I would like to thank my wonderful collaborators John Silberholz and Iain Dunning for our work on *the graph conjecture generator*, Maxime Cohen and Jeremy Kalas for their insights into the pricing world, Joel Tay for our adventures with various formulations of the vehicle routing problem. I have learnt a lot from you guys and really want to thank you for that! Also, thanks Lennart Baardman for helping me with some computations even though we have not collaborated directly on a project.

Patrick's research group has been like my academic family here at MIT, and I would like to thank Max, Virgille, Konstantina, Maokai, Andrew, Xin, Dawsen, Chong, Nikita, Sebastien and Arthur for enlightening discussions on technical ideas. I would also like to thank Juliane Dunkel, Jacint Szabo and Marco Laummans at IBM Research Zurich for an exciting summer of railway scheduling! The mountain climbing trips to the Braunwald Klettersteig and the one in Brunnistöckli have been one of the most amazing experiences of my life, and would like to thank the business optimization group for taking me there, especially Ulrich Schimpel for literally pushing me to climb!

I would like to thank faculty at IIT Delhi, especially Naveen Garg for making me get addicted to the traveling salesman problem, Amitabha Tripathi for inculcating a love for graph theory; thank you both for encouraging me to pursue graduate studies. I would also like to thank my uncle, Atul Prakash, for inviting me to the University of Michigan for a summer project that sparked my enthusiasm for research.

I would like to take this opportunity to document some invaluable advice and guidance I have received in my research career thus far and hope that this serves as a reminder for me in the years to come: One should only write papers when they think that have an interesting idea to share. One needs to be critical of every step in their proof, think of why each step is needed and if the proof can be said in simpler terms. One should question every assumption in their work: either give an example of why their argument would not hold without the assumption or try to remove the assumption to obtain a more general statement. It helps to have a bigger question in your mind, and solve smaller more feasible questions that might help you solve the big one. It is important to make your work accessible to people and it is okay to add simplified lemmas for useful special cases. It is okay to think of many questions and ideas at a time, just like art these ideas evolve and influence each other. Everyone's taste in research can be different, some people might share the same enthusiasm for your work, some may not. When selecting which problem you want to work on, it is good to think of why solving the problem is important in the first place. You do not have to be like anyone else, you can be your own unique self!

No PhD can be completed without the support and love of friends. I want to first and foremost thank my friends, Nataly and John. I will always fondly remember our homework solving sessions in our first year with a ready supply of John's candy and Nataly's delicious lebanese food, our power of exponentials lesson, uber competitive badminton matches with John, and Nataly's infinite wisdom on worldly matters. I will cherish most our first ORC retreat together and all of the stories from that party that have been told uncountable number of times in the past years. I would like to also thank my amazing friends Joel, Iain, Paul, Rajan, Nishanth, Leon, He, David, Rhea, Kris, Angie, Fernanda, Vishal, Adam, Ross, Will, Phebe, Miles, Joline, Nishanth, Allison, Velibor, Andrew, Chaitanya, Yehua, Xin, Yaron, Ilias, Michael, Miles, Nathan, Rim, Ross, Tamar, Lennart, Alex, Wei, Deeksha, Jacopo, Amanda, Sarah(s), Divya and Soumya for their love and friendship.

I would like to thank Praneeth for inspiring me with his dedication to his research and standing by me come what may, Vibhuti for being my roommate and for exciting discussions ranging from viruses to cancer cells to writing sci-fi stories, Shreya for being my mate at LIDS and for our lunches together. I would like to thank Siddharth and Harshad for listening to my proofs even though nothing made sense to them, and Abhinav and Anushree for their sound advice. Through ups and downs of the graduate student life, I have shared many wonderful memories with my roommates - Will, Elyot, Shefali, Padma and Anirudh - and I would like to thank them all. A big shout out to teams Sangam and AID MIT and all

my friends there for the wonderful times! Even while being much away from me and far from a PhD life, I would like to thank from the bottom of my heart my friend Abhiti, who has always been just a phone call away to cheer me on. Thank you Abha, Ekta, Shashank, Raman and Anshul for exciting discussions, adventures and board games!

Last but definitely not the least, I would like to thank my family who have been my pillars of support throughout. I would first and foremost like to thank my parents, Sujata and Shyam, for providing me unconditional love and belief in my abilities during my graduate studies, and for doing everything in their power to help me achieve what I wanted. Words cannot express what your support has meant to me. My sister Sakshi for her much-needed wise cracks and fierceness for me, Amit bhaiya for being my sounding board specially ordered from New York, mausi for her infinite worldly wisdom and circle theory, mausa for his unique commentary on life, Ashu bhaiya and Aarti for their cool management advice and support, didi and jiju for inspiring me with your creativity, and a big hug to my extended family for being there for me always.

I would like to give my heartfelt thanks to my husband and best friend, Tushar, for literally being there for me, for believing in me, for inspiring me, for making me laugh, for remembering mathematical terms from my thesis and throwing them back at me in random sentences - I cannot even begin to imagine my thesis (and my life) without him. Whenever I would feel low for any reason, he had the special skill of somehow coming up with an anecdote to lessen my worries and cheer me up. *Nenu ninnu chhala premistunnanu!*

Finally, I would like to dedicate this thesis to my grandparents: Tirath nana who never understood why I would want to solve the problems of traveling salesmen yet always keeps me in his prayers, Anand nana who has always advised me to direct my thoughts into writing, babaji who taught me to how to compute fractions, dadi who always proud to see a bit of my father in me, and Satyavathi nani who has always blessed me and loved me as her own.

Lastly I would like to acknowledge various sources of funding that have supported my research during graduate school: NSF grant OR-1029603, ONR grant N00014-12-1-0999, ONR grant N00014-15-1-2083, ONR contract N00014-14-1-0072, ONR contract N00014-17-1-2177, ONR grant N00014-16-1-2786 and NSF contract CCF-1115849.

Contents

List of Figures

List of Tables

To my grandparents

asato ma sad gamaya

tamaso ma jyotir gamaya

From delusion lead me to truth

From ignorance lead me to knowledge

–Brhadaranyaka Upanisad, I.iii.28

Chapter 1

Introduction

"Facebook defines who we are, Amazon defines what we want,
Google defines what we think."
- George Dyson, *Turing's Cathedral.*

Algorithms shape almost all aspects of modern life - search, social media, news, e-commerce, finance and urban transportation, to name a few. At the heart of most algorithms today is an optimization engine trying to provide the best feasible solution with the information observed thus far in time. For instance, a recommendation engine repetitively shows a list of items to incoming customers, observes which items they clicked on, and updates the list by placing the more popular items higher for subsequent customers. A routing engine suggests routes that have historically had the least amount of network congestion, observes the congestion on the selected route, and updates its recommendation for subsequent users. What makes this optimization with partial information even more challenging is the effect of competition from other algorithms on users or shared resources. For instance, two search engines, like Google and Bing, might compete for the same set of users and try to attract them with appropriate page rankings.

The space of feasible solutions that these algorithms have to operate within, needs to respect various combinatorial constraints. For instance, when displaying a list of n objects each object must have a unique position from $\{1, \dots, n\}$ or when selecting roads in a network they must link to form a path from the specified origin of request to its destination. This inherent combinatorial structure in the feasible solutions often results in certain computational bottlenecks. In this thesis, we consider three fundamental questions over combinatorial polytopes that help in improving these bottlenecks that arise in various algorithms across convex optimization, game theory and online learning due to the combinatorial nature of the feasible solution set. The first is concerned with how to minimize a separable strictly convex function over submodular polytopes (in Section 1.1), the second is regarding the complexity of the parametric line search problem over extended submodular polyhedra (in Section 1.2), and the third is dealing with the implication of efficient generalized approximate counting over convex optimization and online learning (in Section 1.3). Finally, we give an overview of our results in terms of applications to two-player games and online learning in Section 1.4 and a roadmap of the thesis in Section 1.5.

1.1 Separable convex minimization

In Chapter 3, we consider the fundamental problem of minimizing separable strictly convex functions over submodular polytopes. This problem is motivated by first-order optimization methods that only assume access to a first order oracle: in the case of minimizing a function $h(\cdot)$, a first order oracle reports the value of $h(x)$ and a sub-gradient in $\partial h(x)$ when queried at any given point x. An important class of first-order methods is projection-based: they require to minimize a (often) separable convex function over the set of feasible solutions. This minimization is referred to as a projection and it is usually the computational bottleneck in these methods whenever the feasible set is constrained. In spite of this bottleneck, projection-based first-order methods often have near-optimal convergence guarantees, thus motivating our search for efficient algorithms to minimize separable convex functions.

To make this more tangible, let us consider a projection-based first-order method, called *mirror descent*, that can be used for minimizing a convex function, $h(\cdot)$, over a convex set P. Mirror descent is based on a strongly-convex[1] function $\omega(\cdot)$, known as the mirror map. Let us consider $\omega(x) = \frac{1}{2}\|x\|^2$ as an example. Then, the iterations of the mirror descent

[1] $h : X \to \mathbb{R}$ is κ-strongly convex w.r.t. $\|\cdot\|$ if $h(x) \geq h(y) + g^T(x-y) + \frac{\kappa}{2}\|x-y\|^2, \forall x, y \in X, g \in \partial h(x)$.

algorithm are as follows,

$$x^{(0)} = \arg\min_{x \in P} \omega(x) = \arg\min_{x \in P} \frac{1}{2}\|x\|^2,$$

and for each $t \geq 1$:

$$\begin{aligned} x^{(t)} &= \min_{x \in P} D_\omega(x, y), \text{ where } y = (\nabla\omega)^{-1}(\nabla\omega(x^{(t-1)}) - \eta\nabla g(x^{(t-1)})), && (1.1) \\ &= \min_{x \in P} \|x - (x^{(t-1)} - \eta\nabla g(x^{(t-1)}))\|^2. && (1.2) \end{aligned}$$

Here, $D_\omega(x, y) = \omega(x) - \omega(y) - \nabla\omega(y)^T(x - y)$ is a convex metric called the Bregman divergence of the mirror map ω, and η is a pre-defined step-size. For $\omega(x) = \frac{1}{2}\|x\|^2$, $\nabla\omega(x) = x$ and $D_\omega(x, y) = \frac{1}{2}\|x - y\|^2$, resulting in the simplified gradient-descent step[2] (1.2). As the algorithm progresses, $x^{(t)}$ approaches the $\arg\min_{x \in P} g(x)$. Note that mirror descent requires only the computation of the gradient of $g(\cdot)$ at a given point $x^{(t-1)}$, along with a separable convex minimization over P (independent of the global properties of the function $g(\cdot)$). The rate of convergence of mirror descent depends on the choice of the mirror map $\omega(\cdot)$, the convex set P, and the convexity constants of $g(\cdot)$. We are concerned with computing (1.1) efficiently, for a broad range of mirror maps $\omega(\cdot)$ and convex sets P.

In order to capture a large variety of combinatorial structures, we consider the class of *submodular* polytopes. Submodularity is a discrete analogue of convexity and naturally occurs in several real-world applications ranging from clustering, experimental design, sensor placement to structured regression. Submodularity captures the property of diminishing returns: given a ground set of elements E ($n = |E|$), each subset S of E is associated with a value $f(S)$ such that the increase in function value obtained by adding an element to a smaller set is more than the increase in value obtained by adding to a larger set. To be precise, submodular set functions $f : 2^n \to \mathbb{R}$ satisfy the property

$$f(S \cup \{e\}) - f(S) \geq f(T \cup \{e\}) - f(T) \text{ for all } S \subseteq T, e \notin T. \tag{1.3}$$

Given such a function f, a submodular base polytope $B(f) = \{x \in \mathbb{R}^n_+ \mid \sum_{e \in E} x(e) =$

[2] Under $\omega(x) = \frac{1}{2}\|x\|^2$, mirror descent is equivalent to the well-known gradient descent algorithm.

$f(E),\ \sum_{e\in S} x(e) \leq f(S)\ \forall S \subseteq E\}$ is the convex hull of combinatorial objects such as spanning trees, permutations, k-experts, and so on [Edmonds, 1970]. In Chapter 3, we consider the problem of minimizing separable strictly convex functions $h(\cdot)$ over submodular base polytopes $B(f)$ of non-negative submodular functions $f(\cdot)$, defined over a ground set E:

$$(\text{P1}) : \min_{x\in B(f)} h(x) := \sum_{e\in E} h_e(x(e)). \tag{1.4}$$

We propose a novel algorithm, Inc-Fix, for solving problem (P1) by deriving it directly from first-order optimality conditions. The algorithm is iterative and maintains a sequence of points in the submodular polytope $P(f) = \{x \in \mathbb{R}^n_+ \mid \sum_{e\in S} x(e) \leq f(S)\ \forall S \subseteq E\}$ while moving towards the base polytope $B(f)$, which is a face of $P(f)$. Successive iterates in the Inc-Fix algorithm are obtained by a greedy increase in the element values in the gradient space. Note that a submodular set function $f(\cdot)$ requires an exponential input (one value for each subset of E). Thus, to obtain meaningful guarantees of running time for any algorithm on submodular polytopes, a natural assumption is to allow oracle access for submodular function evaluation. Inc-Fix operates under the oracle model and uses known submodular function minimization algorithms as subroutines. We show that Inc-Fix is an exact algorithm under the assumption of infinite precision arithmetic, and its worst-case running time requires $O(n)$ submodular function minimizations[3]. Note that this running time does not depend on the convexity constants of $h(\cdot)$.

When more information is known about the structure of the submodular function (as opposed to only an oracle access to the function value), one can significantly speed up the running time of Inc-Fix. We specifically consider cardinality-based submodular functions, where the function value $f(S)$ only depends on the cardinality of set S and not on the choice of elements in S. Although simple in structure, base polytopes of cardinality-based functions are still interesting and relevant, for instance, the probability simplex is obtained by setting $f(S) = 1$ for all subsets S and the convex hull of permutations is obtained by setting $f(S) = \sum_{s=1}^{|S|}(n-1+s)$ for all $S \subseteq E$. For minimizing Bregman divergences arising

[3]Each submodular function minimization also requires the computation of the maximal minimizer, we will make these details clearer in Chapter 3.

from uniform mirror maps $\omega(x) = \sum_e w(x_e)$, onto cardinality-based submodular polytopes, we give the fastest-known running time of $O(n(\log n + d))$, where d ($d \leq n$) is the number of unique submodular function values. This subsumes some of the recent results of Yasutake et al. [Yasutake et al., 2011], Suehiro et al. [Suehiro et al., 2012] and Krichene et al. [Krichene et al., 2015].

1.2 Parametric line search

In Chapter 4, we consider the parametric line search problem in the extended submodular polytope $EP(f) = \{x \mid x(S) \leq f(S) \; \forall S \subseteq E\}$, that computes how far one can move in $EP(f)$ starting from $x_0 \in EP(f)$, along the direction $a \in \mathbb{R}^n$:

$$\text{(P2): given } a \in \mathbb{R}^n, x_0 \in EP(f), \text{ compute } \max\{\delta \mid x_0 + \delta a \in EP(f)\}. \tag{1.5}$$

In this chapter, we do not impose a non-negativity condition on the submodular function $f(\cdot)$. The parametric line search is a basic subproblem needed in many algorithmic applications. For example, in the above-mentioned Inc-Fix algorithm, the required subproblem of a parametric increase on the value of elements in the gradient space is equivalent to parametric line searches when minimizing squared Euclidean distance or KL-divergence. The Carathéodory's theorem states that given any point in a polytope $K \subseteq \mathbb{R}^n$, it can be expressed as a convex combination of at most $n+1$ vertices of K. For the algorithmic version of Carathéodory's theorem, one typically performs a line search from a vertex of the face being considered in a direction within the same face. Line searches also arise as subproblems in various variants of the Frank-Wolfe algorithm (for instance, see Algorithm 2 in [Freund et al., 2015]) that is used for convex minimization over convex sets that admit efficient linear optimization. Since computing the maximum movement along a direction $a \in \mathbb{R}^n$ while being feasible in $EP(f)$ entails solving $\min_{\{S \mid a(S)>0\}} \frac{f(S)}{a(S)}$, line searches are closely related to minimum-ratio problems (for e.g. [Cunningham, 1985b]).

One of the most natural approaches for performing a line search is the cutting-plane based approach of the discrete Newton's algorithm: repeatedly generate the most violated inequality starting from a point on the line outside the polytope and subsequently consider next the point where the line intersects this hyperplane. This method would terminate for any polytope that admits an efficient way of generating the most violated hyperplane. Surprisingly, for extended submodular polytopes no bound on the number of iterations was known for the case of an arbitrary line direction, while the nonnegative case has been well understood (the Inc-Fix algorithm requires movement along nonnegative directions only).

In Chapter 4, we provide a quadratic bound on the number of iterations of the discrete Newton's algorithm to solve the general line search problem. The only other strongly-polynomial method known that can be used to compute line searches prior to our work utilized Megiddo's parametric search framework [Nagano, 2007c]. In this framework, values of all the elements are maintained as linear expressions of the parametric variable δ, and a fully-combinatorial[4] submodular function minimization algorithm is used to find maximum δ such that the minimum of the submodular function $f - x_0 - \delta a$ is still zero (a submodular function plus a linear function is still submodular). Each comparison in the fully-combinatorial SFM algorithm requires another submodular function minimization (SFM). The fastest known fully-combinatorial algorithm for SFM is that of [Iwata and Orlin, 2009] that requires $\tilde{O}(n^8)$ fully-combinatorial operations. Thus, such a parametric search framework requires $\tilde{O}(n^8)$ SFMs to perform line searches.

As a byproduct of our study, we prove (tight) bounds on the length of certain chains of ring families[5] and geometrically increasing sequences of sets. We first show a tight (quadratic) bound on the length of a sequence $T_1, \cdots, T_k$ of sets such that no set in the sequence belongs to the smallest ring family generated by the previous sets. One of the key ideas in the proof of the quadratic bound is to consider a sequence of sets (each set corresponds to an iteration in the discrete Newton's method) such that the value of a submodular function on these sets increases geometrically (to be precise, by a factor of 4). We show a quadratic bound on the length of such sequences for any submodular function and construct two examples to show that this bound is tight. These results might be of independent interest.

[4]An algorithm is called fully-combinatorial if it requires to compute only the fundamental operations of additions, subtractions and comparisons.

[5]A ring family $\mathcal{R}$ is a family of sets closed under taking unions and intersections.

1.3 Generalized approximate counting

In Chapter 5, we consider a popular online learning algorithm, the multiplicative weights update method, and ask how it can be used to learn over combinatorial strategies and for efficient convex minimization. Online learning is a sequential framework of decision-making under partial information. A decision or a feasible solution is selected (for example, which products to display to a user, given that we do not know their preferences), using knowledge of previous outcomes (some similar users clicked on product A more than B), as that knowledge is acquired. Contrary to machine learning models, where one sees all the data, fits a model to it, and uses the model to find future decisions or solutions, in online learning the data arrives sequentially, for each data point x_t a decision is selected, and then the model is updated after observing feedback due to x_t. The online learning framework also does not require any assumptions on how the data is generated (as opposed to statistical learning models).

Multiplicative weights update algorithm (MWU) is one of the most intuitive online learning methods (and has been discovered in game theory, machine learning and optimization independently under various names) [Arora et al., 2012]. It maintains a probability distribution over the set of decisions and samples a decision according to this probability distribution. For each decision that incurs a loss (in some metric), the probability is reduced by a multiplicative factor, and for each decision that incurs a gain its probability is increased by the same factor. Repeating this process over time, one can show that the MWU algorithm is competitive with respect to a fixed decision in hindsight (when all the losses and gains are known at some point in the future) even though this fixed decision may not be known a priori. Intuitively, this algorithm works because higher paying decisions are sampled with a higher probability in the long run.

We are interested in online learning over combinatorial decisions. However, the running time of each iteration of the MWU algorithm over N decisions is $O(N)$ due to the updates required on the probabilities of each strategy. Note that the number of combinatorial strategies is typically exponential in the input parameters of the problem, for instance, the number of spanning trees of a complete graph with a vertex set of size $|V|$ is $|V|^{(|V|-2)}$ (Cayley's Theorem). However, we would like to simulate the MWU algorithm in time polynomial in the input size of the problem, i.e. $|V|$. In Chapter 5, we first consider the question of

(P3.1): *Under what conditions, can the MWU algorithm be simulated in logarithmic time in the number of combinatorial strategies?*

We develop a general framework for simulating the MWU algorithm over the set of vertices $\mathcal{U}$ of a 0/1 polytope P efficiently by updating product distributions. A product distribution p over $\mathcal{U} \subseteq \{0,1\}^m$ is such that $p(u) \propto \prod_{e:u(e)=1} \lambda(e)$ for some multiplier vector $\lambda \in \mathbb{R}^m_+$. Product distributions allow us to maintain a distribution on (the exponentially sized) $\mathcal{U}$ by simply maintaining $\lambda \in \mathbb{R}^m_+$. We show that whenever there is an efficient *generalized (approximate) counting oracle* which, given $\lambda \in \mathbb{R}^m_+$, (approximately) computes $\sum_{u\in\mathcal{U}} \prod_{e:u_e=1} \lambda(e)$ and also, for any element s, computes $\sum_{u\in\mathcal{U}:u_s=1} \prod_{e:u_e=1} \lambda(e)$ allowing the derivation of the corresponding marginals $x \in P$, then the MWU algorithm can be efficiently simulated to learn over combinatorial sets $\mathcal{U}$. This generalizes known results for learning over spanning trees [Koo et al., 2007] where a generalized exact counting oracle is available using the matrix tree theorem, and bipartite matchings [Koolen et al., 2010] where a randomized approximate counting oracle can be used [Jerrum et al., 2004].

Recall that in Section 1.1, we discussed briefly the first-order optimization method, mirror descent, which is based on a strongly-convex function known as the mirror map. Online mirror descent is an online variant of the offline version where the (sub)gradients are generated externally (by the environment, users or adversary) and the updates are similar to those of the mirror descent algorithm. As we noticed in (1.2), selecting the mirror map $\omega(x) = \frac{1}{2}\|x\|^2$, shows that the gradient descent method is a special case of the mirror descent algorithm. Similarly, it is known that selecting $\omega(x) = \sum(x_e \ln(x_e/y_e) - x_e + y_e)$ to perform convex minimization over an d-dimensional simplex results in the multiplicative weights update algorithm over d strategies [Beck and Teboulle, 2003]. Given a polytope $P \in \mathbb{R}^n$, one can consider the space of (an exponential number) the vertex set $\mathcal{U}$, and probability distributions over $\mathcal{U}$. The representation of the polytope changes (now it uses an exponential number of variables), however, the above-mentioned approximate counting oracles give a way of computing projections efficiently (these correspond to computing the normalization constant of the probability distribution). We next ask the following question:

(P3.2): *What are the implications of being able to compute projections efficiently in a different representation of the polytope?*

By moving to a large space with an exponential number of dimensions, we see that it is straightforward to compute projections (via approximate counting). This is reminiscent of the theory of extended formulations, where polynomial number of variables are added to a formulation with the hope of reducing the number of facets of the raised polytope (and thereby improve the running time of linear optimization). With this point of view, we show that convex functions over the marginals of a polytope P can be minimized efficiently by moving to the space of vertices and exploiting approximate counting oracles. Note that this results holds irrespective of the convex function being separable or not (recall that in Chapter 3 we minimize *separable* convex functions). This leads to interesting connections and questions about different representations of combinatorial polytopes, while drawing a connection to approximate counting and sampling results from the theoretical computer science literature. As a corollary, we show that using the MWU algorithm we can decompose any point in a 0/1 polytope P into a product distribution over the vertex set of P.

1.4 Nash-equilibria in two-player games

In Chapter 6, we discuss the above-mentioned results in the context of finding optimal strategies (Nash-equilibria) for two-player zero-sum games, as well as prove structural properties of equilibria that help in computing these using convex minimization. Two-player zero-sum games (or more generally saddle point problems) allow us to mathematical model many interesting scenarios involving interdiction, competition, robustness, etc. We are interested in games where each player plays a combinatorial strategy[6], and the loss of one player can be modeled as a bilinear function of their strategies (the loss of the other player is negative the loss of the former player). As an example, consider a spanning tree game in which most of the results of the thesis will apply, pure strategies correspond to spanning trees T_1 and T_2

[6]We consider simultaneous-move and single round games. Note that the number of pure strategies for each player is then exponential in the input of the game.

selected by the two players in a given graph G. We can model intersection losses as bilinear functions: whenever their strategies T_1 and T_2 intersect at an edge, there is a payoff from one player to the other, i.e. say the first (row) player looses $\sum_{e \in T_1 \cap T_2} L_e$ to the other player. Selecting $L_e > 0$ can be used to model an interdiction scenario where the first player is trying to avoid detection (by minimizing the intersection $T_1 \cap T_2$), while the other player is trying to maximize detection (by maximizing the intersection). Another example is that of dueling search engines, as described in a paper by Immorlica et al. [Immorlica et al., 2011]. Suppose two search engines A and B would like to select an ordering of webpages to display to a set of users, where both the search engines know a distribution p over the webpages $i \in \mathcal{I}$ such that $p(i)$ is the fraction of users looking for a page i. Consider a scenario in which the users prefer the search engine that displays the page they are looking for earlier in the ordering. Note that if a search engine displays a greedy ordering $G_r = (1, 2, 3, \ldots, |\mathcal{I}|)$ where $p(i) \geq p(j)$ for $i < j$ (which is optimal if the goal is to maximize relevance of results given p), then the other search engine can attract $1 - p(1)$ fraction of the users by displaying a modified ordering $G'_r = (2, 3, \ldots, |\mathcal{I}|, 1)$. This competitive scenario between two search engines can again be modeled as a two-player zero-sum game, where each player plays a bipartite matching (vertices corresponding to pages are matched to vertices corresponding to the position in the ordering) with a bilinear loss function[7].

In Chapter 6, we first discuss the well-known von Neumann linear program to find Nash-equilibria[8] for the above-mentioned two-player zero-sum games. Under bilinear loss functions, the von Neumann linear program has a compact form, and this can be solved using the ellipsoid algorithm. Next, any online learning algorithm can be used to converge to Nash-equilibria for two-player zero-sum games, a well-studied connection that we discuss in this chapter. This allows us to make use of either online mirror descent (along with the computation of Bregman projections, as discussed in Chapter 3) or the multiplicative weights update (along with approximate generalized counting oracles, as discussed in Chapter 5). We discuss the convergence rates to approximate Nash-equilibria in the case of a

[7]To obtain a bilinear loss function, one must use the representation of bipartite matchings using doubly stochastic matrices.

[8]A pair of strategies such that neither player has an incentive to deviate from their strategy if the other player commits to his/her strategy.

spanning tree game as all the results apply to this case, using entropic mirror descent, gradient descent and the multiplicative weights update algorithm. We further discuss limitations of these approaches in the context of the results presented in this thesis, for instance, our projection algorithms would not work for bipartite matchings (although one could use the ellipsoid algorithm). Finally, we show certain structural results that hold for (symmetric) Nash-equilibria of two-player zero-sum matroid games[9] (where each player plays bases[10] of the same matroid). These results enable us to find equilibria using a single separable convex minimization under some conditions over the loss matrix.

1.5 Roadmap of thesis

This thesis is organized as follows. In Chapter 2, we discuss some background for the problems and related work for the above mentioned questions.

In Chapter 3, we consider the problem of separable convex minimization over submodular base polytopes. We give our algorithm, Inc-Fix, for minimizing separable convex functions over these base polytopes (Section 3.1). We show that the Inc-Fix computes exact projections and prove correctness of our algorithm in Section 3.2. We next show equivalence of various convex problems (Section 3.2.1), as well as discuss a natural way to round intermediate iterates to the base polytope (Section 3.2.2). In Section 3.3, we discuss two ways of implementing the Inc-Fix method using either $O(n)$ parametric line searches (Section 3.3.1) or $O(n)$ submodular function minimizations (Section 3.3.2). Further, we develop a variant of the Inc-Fix algorithm, called Card-Inc-Fix, that works in nearly linear to quadratic time for minimizing divergences arising from uniform separable mirror maps onto base polytopes of cardinality-based functions (Section 3.4).

Next, in Chapter 4, we consider the problem of finding the maximum possible movement along a direction while staying feasible in the extended submodular polytope. In Section 4.1, we review some background related to ring families and Birkhoff's representation theorem, as well as a key result on the length of a certain sequence of sets that is restricted due to

[9]Matroids abstract and generalize the notion of linear independence in vector spaces.

[10]These are the maximal independent sets in a matroid. For example, spanning trees of a given graph are the bases of the graphic matroid.

the structure of ring families. This result plays an important part in proving the main result in this chapter. We next show a cubic bound on the number of iterations of the discrete Newton's algorithm, in Section 4.2.1, and the stronger quadratic bound (Theorem 11, in Section 4.2.2). One of the key ideas in the proof for Theorem 11 is to consider a sequence of sets (each set corresponds to an iteration in the discrete Newton's method) such that the value of a submodular function on these sets increases geometrically (to be precise, by a factor of 4). We show a quadratic bound on the length of such sequences for any submodular function and construct two examples to show that this bound is tight, in Section 4.3.

Chapter 5 concerns with a general recipe for simulating the multiplicative weights update algorithm in polynomial time (logarithmic in the number of combinatorial strategies) (in Section 5.1). We show how this framework can be used to compute convex minimizers over combinatorial polytopes that admit efficient approximate counting oracles over their vertex set (Section 5.2). As a byproduct of this result, we show that the MWU algorithm can be used to decompose any point in a 0/1 polytope (that admits approximate counting) into a product distribution over the vertex set.

In Chapter 6, we view the above discussed results in the context of finding Nash-equilibria for two-player zero-sum games where each player plays a combinatorial strategy and the losses are bilinear in the two strategies. After reviewing the ellipsoid algorithm for solving the von Neumann linear program for finding Nash-equilibria (Section 6.1), we show that on one hand, the mirror descent algorithm can be used in conjunction with projections over submodular polyhedra, and on the other hand, the multiplicative weights update algorithm can be used in conjunction with approximate counting oracles (Section 6.2). We also show that symmetric Nash-equilibria for certain games can be computed by minimizing a single separable convex function (Section 6.3).

Finally, in Chapter 7, we summarize the results in this thesis and discuss research directions that emerge out of this work. We survey important projection-based first-order optimization methods in Appendix A and include some examples of Nash-equilibria of the spanning tree game under identity loss matrices in Appendix B.

Chapter 2

Background

"If I have seen further, it is by standing on the shoulders of giants."

- Isaac Newton.

We present in this chapter the notation used throughout the thesis, some useful references for certain theoretical concepts and machinery required to understand the results (in Sections 2.1 and 2.2). None of the theorems discussed in this chapter are our own. We give attributions in almost all cases, unless the results have been known in the community as folklore results. Our development of the background material is in no way comprehensive. We give most attention to the results required in the subsequent chapters. Further we also discuss important related work pertaining to the results in each chapter in Section 2.3.

2.1 Notation

We first discuss the notation used in this thesis. We use $\mathbb{R}^n_+$ to denote the space of vectors in n-dimensions that are non-negative in each coordinate and $\mathbb{R}^n_{>0}$ is the space of vectors that are positive (non-zero) in each coordinate. In Chapter 3, we minimize differentiable separable convex functions $h(\cdot)$ and we refer to their gradients as ∇h. Throughout the thesis, we focus on combinatorial strategies that are a selection of elements of a ground set E, for instance, given a graph $G = (V, E)$ the ground set E is the set of edges and combinatorial strategies are spanning trees, matchings, paths, etc. We let the cardinality of the ground set be $|E| = n$. We will often represent these combinatorial strategies by n-dimensional $0/1$ vectors and use the shorthand $e \in u$ to imply $e : u(e) = 1$ for any $0/1$ vector u. We use $\mathbb{R}^{|E|}$ and $\mathbb{R}^E$ interchangeably. For a vector $x \in \mathbb{R}^E$, we use the shorthand $x(S)$ for $\sum_{e \in S} x(e)$. For readability, we use $x(e)$ and x_e interchangeably. To represent a vector of ones, we use $\mathbf{1}$ (when the dimension is clear from context) or $\chi(E)$ (to specify the dimension to be $|E|$). By $\arg\min_{x \in P} h(x)$, we mean the set of all minimizers of $h(\cdot)$ over $x \in P$. This set is just the unique minimizer when $h(\cdot)$ is a strictly convex function.

2.2 Background

We next discuss some important concepts and theorems required to understand the results of this thesis. In Chapters 3, 4 and 6, we work with submodular polyhedra and review some important concepts related to submodularity in Section 2.2.1. As a motivation for considering the bottleneck of computing projections (i.e. convex minimization) over submodular polytopes, we often refer to projection-based first-order methods like the mirror descent and its variants. We discuss these in Section 2.2.2, along another first-order optimization method, Frank-Wolfe, that does not require projections. Chapter 5 deals predominantly with an online learning algorithm and its usefulness in online linear optimization and convex optimization. We therefore discuss some background on the online learning framework in Section 2.2.3.

2.2.1 Submodular functions and their minimization

Submodularity is a discrete analogue of convexity and is a property often used to handle combinatorial structure. Given a ground set E ($n = |E|$) of elements, for e.g., the edge set of a given graph, columns of a given matrix, objects to be ranked, a set function $f : 2^n \to \mathbb{R}$ is said to be submodular if

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B), \tag{2.1}$$

for all $A, B \subseteq E$. Another way of defining submodular set functions is by using the property of diminishing returns, i.e. adding an element to a smaller set results in a greater increase in the function value compared to adding an element to a bigger set. More precisely, a set function f is said to be submodular if

$$f(\{e\} \cup T) - f(T) \leq f(S \cup \{e\}) - f(S), \tag{2.2}$$

for every $S \subseteq T \subseteq E$ and $e \notin T$. The latter characterization is at times easier to verify than the sum of the function value over the intersection and unions of subsets, as in (2.1).

We can assume without loss of generality that f is normalized such that $f(\emptyset) = 0$ (suppose it is not, then one can consider $f' = f - f(\emptyset)$ instead). Given such a function f, the submodular polytope (or independent set polytope) is defined as $P(f) = \{x \in \mathbb{R}^n_+ : x(U) \leq f(U) \; \forall \; U \subseteq E\}$, the extended submodular polytope (or the extended polymatroid) as $EP(f) = \{x \in \mathbb{R}^n : x(U) \leq f(U) \; \forall \; U \subseteq E\}$, the base polytope as $B(f) = \{x \in P(f) \mid x(E) = f(E)\}$ and the extended base polytope as $B_{\text{ext}}(f) = \{x \in EP(f) \mid x(E) = f(E)\}$ [Edmonds, 1970]. The vertices of these base polytopes are often the combinatorial strategies that we care about, for instance, spanning trees, permutations of the ground set, etc. We list in Table 2.1 some interesting examples of base polytopes of submodular functions.

Combinatorial strategies represented by vertices of $B(f)$	Submodular function f, $S \subseteq E$ (unless specified)
One out of n elements, $E = \{1, \dots, n\}$	$f(S) = 1$
Subsets of size k, $E = \{1, \dots, n\}$	$f(S) = \min\{\lvert S\rvert, k\}$
Permutations over $E = \{1, \dots, n\}$	$f(S) = \sum_{s=1}^{\lvert S\rvert}(n + 1 - s)$
k-truncated permutations over $E = \{1, \dots, n\}$	$f(S) = (n - k)\lvert S\rvert$ for $\lvert S\rvert \leq k$, $f(S) = k(n - k) + \sum_{s=k+1}^{\lvert S\rvert}(n + 1 - s)$ if $\lvert S\rvert \geq k$
Spanning trees on $G = (V, E)$	$f(S) = \lvert V(S)\rvert - \kappa(S)$, $\kappa(S)$ is the number of connected components of S
Bases of matroids $M = (E, \mathcal{I})$ over ground set E, $\mathcal{I} \subseteq 2^E$	$f(S) = r_M(S)$, the rank function of the matroid

Table 2.1: Examples of common base polytopes and the submodular functions (on ground set of elements E) that give rise to them.

Given a vector $x \in EP(f)$ (or $x \in P(f)$), a subset $S \subseteq E$ is said to be tight if $x(S) = f(S)$. If the value of any element e in a tight set S is increased by some $\epsilon > 0$, then $x + \epsilon\chi(e)$ would violate the submodular constraint corresponding to the set S. We refer to the maximal subset of tight elements in x as $T(x)$. This is unique by submodularity of f, as is clear from the following lemma.

Lemma 2.1 ([Schrijver, 2003], Theorem 44.2)**.** *Let f be a submodular set function on E, and let $x \in EP(f)$. Then the collections of sets $S \subseteq E$ satisfying $x(S) = f(S)$ is closed under taking intersections and unions.*

Proof. Suppose S, T are tight sets with respect to $x \in EP(f)$. Note that $x(S \cup T) + x(S \cap T) = x(S) + x(T) \overset{(1)}{=} f(S) + f(T) \geq^{(2)} f(S \cup T) + f(S \cap T)$, where (1) follows from S and T being tight and (2) follows from submodularity of f. Since $x \in EP(f)$, $x(S \cap T) \leq f(S \cap T)$ and $x(S \cup T) \leq f(S \cup T)$, which in turn imply that $S \cup T$ and $S \cap T$ are also tight with respect to x. $\square$

The above lemma implies that the union of all tight sets with respect to $x \in EP(f)$ is also tight, and hence it is the unique maximal tight set $T(x)$.

We next discuss two operations, contractions and restrictions, that preserve submodularity of submodular set systems. This will be useful when we perform certain parametric gradient searches in Chapter 3 to implement the Inc-Fix algorithm. For a submodular function f on E with $f(\emptyset) = 0$, the pair (E, f) is called a submodular set system.

Definition 1. *For any $A \subseteq E$, a restriction of f by A is given by the submodular function $f^A(S) = f(S)$ for $S \subseteq A$.*

In the case of a restriction f^A, the ground set of elements is restricted to A, i.e. $E^A = A$. It is easy to see that (E^A, f^A) is also a submodular set system.

Definition 2. *For any $A \subseteq E$, a contraction of f by A is given by the submodular function $f_A(S) = f(A \cup S) - f(A)$ for all $S \supseteq A$.*

In the case of a contraction f_A, the ground set of elements is $E_A = E - A$. To check that (E_A, f_A) is a submodular set system, note that for any $S, T \subseteq E_A$, $f_A(S) + f_A(T) = f(S \cup A) + f(T \cup A) - 2f(A) \geq f((S \cup T) \cup A) + f((S \cap T) \cup A) - 2f(A)$ by submodularity of f.

To specify a submodular function f, we need the value of the function on every subset of the ground set. Since the input size would then be exponential in the size of the ground set, a natural model that helps in obtaining meaningful running time guarantees is to assume an oracle access for the submodular function value. Edmonds [Edmonds, 1970] showed that one can maximize any linear function $w^T x$ over $P(f)$ by using the greedy algorithm whenever f is monotone and normalized.

Next, note that to check for feasibility of x in the extended submodular polytope $EP(f)$, one needs to verify that $x(S) \leq f(S)$ for all subsets S. This can be done by minimizing the submodular function $f - x$ over all the subsets. If the minimum is at least zero, then $x \in EP(f)$ (if $x \in EP(f)$ and $x \geq 0$ then $x \in P(f)$). The first polynomial time algorithm for submodular function minimization (SFM) was due to [Grötschel et al., 1981], and was based on the ellipsoid method. Starting with the work of Cunningham [Cunningham, 1985a], there have been many combinatorial algorithms for SFM, like [Schrijver, 2000], [Fleischer and Iwata, 2003], [Orlin, 2009] and [Lee et al., 2015]. One of the crucial ingredient of these algorithms is the following min-max theorem:

Theorem 1 ([Edmonds, 1970]). *For a submodular function f with $f(\emptyset) = 0$, we have*

$$\min_{S \subseteq E} f(S) = \max_{x \in B_{ext}(f)} x^-(E),$$

where $x^-(E) = \sum_{e | x(e) \leq 0} x(e)$.

These combinatorial algorithms maintain a certificate of optimality $x \in B_{\text{ext}}(f)$ such that $x^-(E) = f(W)$ for some $W \subseteq E$. Existence of such $x \in B_{\text{ext}}(f)$ and W proves that W is a minimizer of the submodular function f using Edmonds' min-max theorem. Since checking for feasibility in a base polytope requires submodular function minimization itself, these certificates are often maintained as a convex combination of bases, i.e. $x = \sum_i \lambda_i v_i$ where v_i are vertices of $B_{\text{ext}}(f)$ and $\sum_i \lambda = 1, \lambda_i \geq 0$. There are also some projected stochastic subgradient descent based submodular function minimization algorithms (like [Chakrabarty et al., 2016]) that exploit the properties of a certain Lovász extension of the submodular function. The Lovász extension is a convex continuous extension of a submodular function to the interior of the n-dimensional hypercube, and its approximate minimizers can be used to approximate minimizers of the submodular function.

An important property of minimizers of submodular functions is that they form a distributive lattice with set union and intersection as the lattice operations.

Lemma 2.2. *Suppose S and T are minimizers of a submodular function f, $S \cap T$ and $S \cup T$ are also minimizers of f.*

Proof. This is easy to see since $f(S) + f(T) \geq^{(1)} f(S \cup T) + f(S \cap T) \geq^{(2)} f(S) + f(T)$, where (1) follows from submodularity of f and (2) follows since S and T are minimizers of f. Therefore, equality must hold throughout implying $S \cap T$ and $S \cup T$ are also minimizers of f. □

The above lemma also implies that submodular functions have unique minimal and maximal minimizers. Next, we consider minimizing sequences of submodular functions that satisfy the property of being strong maps.

Definition 3. *A submodular function $\bar{f}$ is said to be a strong quotient of another submodular function f if $Z \subseteq Y$ implies*

$$f(Z) - f(Y) \geq \bar{f}(Z) - \bar{f}(Y).$$

In other words, $\bar{f}$ is a strong quotient of f if $f - \bar{f}$ is monotone non-decreasing, and this relation, denoted by $f \rightarrow \bar{f}$, is called a strong map. A fundamental property of any strong map $f \rightarrow \bar{f}$ is that the minimizers of f and $\bar{f}$ are related in the following sense:

Lemma 2.3 ([Iwata et al., 1997], [Topkis, 1978])**.** *The minimal (maximal) minimizer of f is contained in the minimal (maximal) minimizer of $\bar{f}$.*

Proof. We will first show that if S and T are minimizers of f and $\bar{f}$ respectively, then $S \cap T$ and $S \cup T$ are also minimizers of f and $\bar{f}$ respectively.

$$\begin{aligned} f(S) - \bar{f}(S) &\geq f(S \cap T) - \bar{f}(S \cap T) && (2.3)\\ &\geq f(S) - \bar{f}(S \cap T) && (2.4)\\ &\geq f(S) + \bar{f}(S \cup T) - \bar{f}(S) - \bar{f}(T) && (2.5)\\ &\geq f(S) - \bar{f}(S), && (2.6)\end{aligned}$$

where (2.3) follows from monotonicity of $f - \bar{f}$, (2.4) follows from S being a minimizer of $f(\cdot)$, (2.5) follows from submodularity of $\bar{f}$ and (2.6) follows from T being a minimizer of $\bar{f}$. Therefore, equality holds throughout and $S \cap T$ is indeed a minimizer of f. One can similarly show that $S \cup T$ is a minimizer of $\bar{f}$. Replacing S by minimal (maximal) minimizer of f and T by minimal (maximal) minimizer of $\bar{f}$, we get that $S \cap T$ is a minimizer of f ($S \cup T$ is a minimizer of $\bar{f}$). Using Lemma 2.2 on the uniqueness of minimal and maximal minimizers of submodular functions, we get the statement of the lemma. □

A strong map sequence is a sequence of submodular functions (on the same ground set) such that $f_1 \rightarrow f_2 \rightarrow \ldots \rightarrow f_k$. A parametric submodular function minimization entails minimizing all the functions f_i parametrized by $i \in K$. When f_i, $i \in K$ form a strong map sequence (of length k), then combinatorial algorithms for SFM can be adapted to perform a parametric submodular function minimization in the same worst-case running time as that for a single SFM [Nagano, 2007a]. The key component of the analysis uses the final combinatorial state (of the distance functions or orders) in the minimization of f_i as a starting state for the minimization of f_{i+1}. Strong map sequences were first formulated in the paper by [Iwata et al., 1997], and have been used by [Fleischer and Iwata, 2003] and [Nagano, 2007a] to achieve faster running times in the context of minimizing separable convex functions.

We further refer the interested reader to [Fujishige, 2005] (for background on submodular set systems), [Schrijver, 2003] (for background on submodular polyhedra and submodular function minimization), [Iwata, 2008] (for a survey on algorithms for submodular function minimization) and references therein.

2.2.2 First-order optimization methods

Before reviewing background on first-order optimization methods, we discuss some definitions. Given an arbitrary norm $\|\cdot\|$ on $\mathbb{R}^n$, its dual norm $\|\cdot\|_*$ is defined as $\|u\|_* = \sup\{u^T x \mid \|x\| \leq 1\}$. Let X be a closed convex set. A function $h : X \rightarrow \mathbb{R}$ is convex iff $\partial h(x)$ is non-empty for all $x \in X$. If h is differentiable at x then $\partial h(x)$ consists of a single vector which amounts to the gradient of h at x, and it is denoted by $\nabla h(x)$. A convex function

$h : X \to \mathbb{R}$ is

(i) G-Lipschitz w.r.t. $\|\cdot\|$ if $\forall x \in X, g \in \partial h(x), \|g\|_* \le G$,

(ii) strictly convex if $h(\theta x + (1-\theta)y) < \theta h(x) + (1-\theta)h(y)$ for any $0 < \theta < 1$ and $x, y \in X, x \neq y$, or equivalently, $h(x) > h(y) + g^T(x-y), \forall x, y \in X, g \in \partial h(x)$,

(iii) κ-strongly convex w.r.t. $\|\cdot\|$ if $h(x) \ge h(y) + g^T(x-y) + \frac{\kappa}{2}\|x-y\|^2, \forall x, y \in X, g \in \partial h(x)$, and

(iv) β-smooth w.r.t. $\|\cdot\|$ if $\|\nabla h(x) - \nabla h(y)\|_* \le \beta\|x-y\|$ for all $x, y \in X$.

First-order optimization methods for minimizing a convex function[1], say $h(\cdot) : X \to \mathbb{R}^n$, rely on a black-box first-order oracle for h, which only reports the value of $h(x)$ and an arbitrary sub-gradient $g(x) \in \partial h(x)$ given an input vector $x \in X$.

We first discuss briefly the mirror descent algorithm [Nemirovski and Yudin, 1983] for minimizing an arbitrary convex function $h(\cdot) : X \to \mathbb{R}$ that is G-Lipschitz on a closed convex set X with respect to $\|\cdot\|$. The presentation of the mirror descent algorithm is inspired from [Bubeck, 2014]. The mirror descent algorithm is defined with the help of a strictly-convex and differentiable function $\omega : \mathcal{D} \to \mathbb{R}$, known as the mirror map, that is defined on a convex set $\mathcal{D}$ such that $X \subseteq \overline{\mathcal{D}}$. A mirror map is required to satisfy additional properties of divergence of the gradient on the boundary of $\mathcal{D}$, i.e., $\lim_{x \to \partial \mathcal{D}} \|\nabla \omega(x)\| = \infty$ (for details, refer to [Bubeck, 2014]). The algorithm is iterative and it starts with the first iterate $x^{(1)}$ as the ω-center of $\mathcal{D}$, given by $x^{(1)} = \arg\min_{x \in X \cap \mathcal{D}} \omega(x)$. Subsequently, for $t > 1$, the algorithm first moves in an unconstrained way using

$$\nabla \omega(y^{(t+1)}) = \nabla \omega(x^{(t)}) - \eta g_t, \text{ where } g_t \in \partial h(x^{(t)}). \tag{2.7}$$

Then, the next iterate $x^{(t+1)}$ is obtained by a projection step:

$$x^{(t+1)} = \arg\min_{x \in X \cap \mathcal{D}} D_\omega(x, y^{(t+1)}), \tag{2.8}$$

[1]In this section, we deviate from the notation of calling the domain of the convex function h to be $\mathcal{D}$ and let the domain of function to be minimized be X, and reserve $\mathcal{D}$ to be the domain of the mirror map.

where $D_\omega(x,y) = \omega(x) - \omega(y) - \nabla\omega(y)^T(x-y)$ is the Bregman divergence with respect to $\omega(\cdot)$ [Bregman, 1967]. Note that the Bregman divergence need not be symmetric, i.e. $D_\omega(x,y) \neq D_\omega(y,x)$. Also, $D_\omega(x,y) \geq 0$ since $\omega(\cdot)$ is strictly-convex, and it is zero iff $x = y$. Further, it is convex in the first argument, as $\omega(x)$ is convex and $\nabla\omega(y)^T x$ is linear in x. The Bregman divergence is in fact strictly-convex in x given y, and therefore has a unique minimizer over any convex set (the proof is straight-forward and follows from the strict-convexity of the mirror map). Bregman divergences also satisfy the generalized Pythagorean theorem,

$$D_\omega(u,x) \geq D_\omega(u,\Pi(x)) + D_\omega(\Pi(x),x) \ \ \forall u \in X \cap \mathcal{D},$$

where $\Pi(x) = \arg\min_{w \in X\cap\mathcal{D}} D_\omega(w,x)$ is the Bregman projection of x onto $X \cap \mathcal{D}$. This property is useful in proving the convergence of the mirror descent algorithm. Note that the partial derivative of the Bregman divergence with respect to x is $\partial_x D_\omega(x,y) = \nabla\omega(x) - \nabla\omega(y)$. Since we care about the divergences as a function of the first argument, we will overload the notation $\nabla D_\omega(x,y)$ to mean $\partial_x D_\omega(x,y)$.

Examples of two important mirror maps that we consider in this thesis are the Euclidean mirror map and the unnormalized entropy mirror map. The Euclidean mirror map is given by $\omega(x) = \frac{1}{2}\|x\|^2$, for $\mathcal{D} = \mathbb{R}^E$ and is 1-strongly convex with respect to the L_2 norm. The unnormalized entropy map is given by $\omega(x) = \sum_{e\in E} x(e)\ln(x(e)) - \sum_{e\in E} x(e)$, for $\mathcal{D} = \mathbb{R}^E_+$ and is known to be 1-strongly convex over the n-dimensional simplex with respect to the L_1 norm. The Bregman divergence with respect to the Euclidean mirror map is $D_\omega(x,y) = \frac{1}{2}\|x-y\|^2$, i.e. the squared Euclidean distance, and the divergence with respect to the unnormalized entropy mirror map is $D_\omega(x,y) = \sum_e (x_e \ln(x_e/y_e) - x_e + y_e)$, i.e. the KL-divergence. We summarize a few examples of mirror maps and their corresponding divergences in Table 2.2. The Bregman divergence corresponding to a κ-strongly convex function is also κ-strongly convex in the first parameter. It is straightforward to check that the squared Euclidean distance is 1-strongly convex with respect to the L_2 norm. The strong convexity of the KL-divergence and the Itakura-Saito divergence follows from Pinsker's inequality, after normalizing $B(f)$ by $f(E)$ (such that $x \in B(f)$ implies $\|x\|_1 = 1$), under the choice of the L_1 norm. Last, the Itakura-Saito divergence corresponds to a strictly convex function, $\omega(x) = -\log(x)$. However, one can still bound its strong convexity coefficient with respect to the L_2 norm whenever $\|x\|_\infty$ is bounded for $x \in P$, by using the fact that if $\nabla^2 h \succeq \kappa I$ for twice-differentiable functions $h(\cdot)$, then $h(\cdot)$ is κ-strongly convex. We summarize the strong-convexity properties of the above mentioned divergences in Table 3.1.

$\omega(\mathbf{x}) = \sum \mathbf{w}(\mathbf{x_e})$	$\mathbf{D}_\omega(\mathbf{x}, \mathbf{y})$	**Divergence**
$\|x\|^2/2$	$\sum_e (x_e - y_e)^2/2$	Squared Euclidean Distance
$\sum_e (x_e \log x_e - x_e)$	$\sum_e \left(x_e \log(x_e/y_e) - x_e + y_e\right)$	Generalized KL-divergence
$-\sum_e \log x_e$	$\sum_e \left(x_e/y_e - \log(x_e/y_e) - 1\right)$	Itakura-Saito Distance
$\sum_e x_e \log x_e + \sum_e (1 - x_e)\log(1 - x_e)$	$\sum_e x_e \log(x_e/y_e) + (1 - x_e)\log((1 - x_e)/(1 - y_e))$	Logistic Loss

Table 2.2: Examples of some uniform separable mirror maps and their corresponding divergences. Itakura-Saito distance [Itakura and Saito, 1968] has been used in processing audio signals and clustering speech data (for e.g. in [Banerjee et al., 2005]).

The rate of convergence of the mirror descent algorithm depends on the radius of the set X with respect to ω, where the radius R is defined using $R^2 = \max_{x\in X} \omega(x) - \min_{x\in X} \omega(x)$. We include the formal statement regarding the rate of convergence of the mirror descent algorithm:

Theorem 2 (see for e.g. [Bubeck, 2014])**.** *Let ω be a mirror map κ-strongly convex on $X \cap \mathcal{D}$ w.r.t. $\|\cdot\|$. Let $R^2 = \max_{x\in X} \omega(x) - \min_{x\in X} \omega(x)$ and h be convex and G-Lipschitz w.r.t. $\|\cdot\|$. Then, the mirror descent algorithm with $\eta = \frac{R}{G}\sqrt{\frac{2\kappa}{t}}$ satisfies*

$$h\Big(\frac{1}{t}\sum_{s=1}^{t} x^{(s)}\Big) - h(x^*) \leq RG\sqrt{\frac{2}{\kappa t}}.$$

Even though in the description of the algorithm, we required a weaker condition of the mirror map to be strictly convex, rate of convergence depends on the strong-convexity parameter of the mirror map. In many cases it is possible to get a bound on the strong-convexity parameter when considering strictly-convex mirror maps over a bounded set. For instance, the Itakura-Saito divergence is generated from a strictly convex mirror map, $\omega(x) = -\sum_e \log x_e$. However, it is easy[2] to show that the divergence is 1-strongly convex over $(0, 1]^n$ under the $\|\cdot\|_2$ norm.

[2]If a function h is twice-differentiable, then it is m-strongly convex with respect to the L_2 norm if $\nabla^2 h \succeq mI$ (for e.g. [Boyd and Vandenberghe, 2009], Chapter 9).

Next, if the function h is smooth, then one can use a variant of the mirror descent algorithm to obtain a faster convergence rate of $O(1/t)$. This method is called the mirror-prox algorithm [Nemirovski, 2004] and it is described by the following iterations starting with $x^{(1)} = \arg\min_{x \in X \cap \mathcal{D}} \omega(x)$:

$$\nabla\omega(y^{(t+1)\prime}) = \nabla\omega(x^{(t)}) - \eta\nabla h(x^{(t)}), \tag{2.9}$$

$$y^{(t+1)} = \arg\min_{x \in X \cap \mathcal{D}} D_\omega(x, y^{(t+1)\prime}), \tag{2.10}$$

$$\nabla\omega(x^{(t+1)\prime}) = \nabla\omega(x^{(t)}) - \eta\nabla h(y^{(t+1)}), \tag{2.11}$$

$$x^{(t+1)} = \arg\min_{x \in X \cap \mathcal{D}} D_\omega(x, x^{(t+1)\prime}). \tag{2.12}$$

Mirror prox will be helpful in 5 in showing a faster convergence when minimizing smooth functions over a 0/1 polytope with the help of the MWU algorithm over the simplex of the vertices. The rate of convergence of the mirror-prox algorithm for minimizing smooth convex functions is given by the following theorem.

Theorem 3 (see for e.g. [Bubeck, 2014]). *Let ω be a mirror map κ-strongly convex on $X \cap \mathcal{D}$ w.r.t. $\|\cdot\|$. Let $R^2 = \max_{x \in X} \omega(x) - \min_{x \in X} \omega(x)$ and h be convex and $\beta-$smooth w.r.t. $\|\cdot\|$. Then, the mirror-prox algorithm with $\eta = \frac{\kappa}{\beta}$ satisfies*

$$h\left(\frac{1}{t}\sum_{s=1}^{t} y_{s+1}\right) - h(x^*) \leq \frac{\beta R^2}{\kappa t}.$$

We give many other variants of the mirror descent algorithm and their iterations in Tables A.1, A.2 and A.3. Note that all of them involve a projection step (in each iteration), and this is a separable convex minimization in many cases specifically for divergences listed in Table 2.2. Computing this step efficiently when X is a submodular base polytope is the main question answered in Chapter 3.

We next discuss another first-order optimization method, Frank-Wolfe that does not rely on the computation of projections [Frank and Wolfe, 1956]. The following presentation of the Frank-Wolfe method (also known as the conditional gradient method) is inspired by the work of [Jaggi, 2013]. The vanilla Frank-Wolfe method for minimizing a convex and differentiable function $h(\cdot)$ over a compact convex set X starts with an arbitrary $x^{(0)} \in X$, and for each iteration $t \geq 0$, repeats

$$s^{(t)} \in \arg\min_{s \in X} \langle s, \nabla h(x^{(t)}) \rangle, \tag{2.13}$$

$$x^{(t+1)} = (1-\gamma)x^{(t)} + \gamma s^{(t)}, \text{ where } \gamma = \frac{2}{2+t}. \tag{2.14}$$

The rate of convergence of the Frank-Wolfe method depends on a parameter C_h, the curvature constant of the function h. It is defined for convex and differentiable functions $h(\cdot)$ with respect to a compact domain $\mathcal{X}$, as

$$C_h := \sup_{x,s \in X, \gamma \in [0,1], y = x + \gamma(s-x)} \frac{2}{\gamma^2}(h(y) - h(x) - \langle y - x, \nabla h(x) \rangle).$$

For instance, for $h(x) := \frac{1}{2}\|x\|_2^2$, the curvature C_h is simply the squared Euclidean diameter $\max_{x \in X, s \in X} \frac{1}{2}\|s-x\|^2$ of the domain X. We next state the rate of converge of the vanilla Frank-Wolfe method to approximate minimizers of the function $h(\cdot)$.

Theorem 4 ([Jaggi, 2013]). *For each $t \geq 1$, the iterates $x^{(t)}$ of the vanilla Frank-Wolfe algorithm satisfy*

$$h(x^{(t)}) - \min_{x \in X} h(x) \leq \frac{2C_h}{t+2}.$$

The step-size γ can be either pre-determined (as in (2.14)), or can be selected using inexact or exact line search (along the line segment joining $s^{(t)}$ and $x^{(t)}$). Note that unlike the mirror descent algorithm (and its variants), the Frank-Wolfe algorithm does not depend on the choice of a norm.

Even though the Frank-Wolfe method is simple to state and computationally inexpensive in many cases (especially for submodular polytopes since the linear optimization step can be computed using Edmonds' greedy algorithm), the mirror descent algorithm is more general, and it is optimal (upto a constant factor) in terms of the convergence rate achievable by any first-order optimization method, as can be observed from the following theorem (it is attributed to [Nemirovski and Yudin, 1983]).

Theorem 5 ([Nesterov, 2013], Chapter 3). *Let $k \leq n$, $G, R > 0$. There exists a convex* *function $h(\cdot)$ with $\|\nabla h\|_2 \leq G$ such that for any first-order algorithm that only uses sub-gradients that outputs $x^{(i)}$ in iteration i,*

$$\min_{1\leq i\leq k} h(x^{(i)}) - \min_{\|x\|\leq R} h(x) \geq \frac{RG}{2(1+\sqrt{k})}.$$

We refer the interested reader to [Ben-Tal and Nemirovski, 2001] (for details on first-order methods, especially the mirror descent algorithm), [Nesterov, 2013] (for lower bounds on rate of convergence of first-order convex minimization methods under different settings), [Bubeck, 2014] (for a compilation of the mirror descent algorithm and its variants), [Grigas, 2016] (for analysis of the Frank-Wolfe method) and [Boyd and Vandenberghe, 2009] (for background on convex optimization).

2.2.3 Online learning framework

We next review the basics of the online learning framework and algorithms, as required for interpreting the results of Chapter 5. The online learning framework can be described as a repeated game between a decision maker (or simply a learner) and an adversary as follows: at each time step $t = 1, \ldots, n$, the learner selects, possibly in a randomized way, a decision or a feasible solution $x^{(t)}$ from a given bounded set $X \subseteq \mathbb{R}^n$. Next after potentially observing the learner's decision, the adversary chooses a loss vector $l^{(t)} : X \to \mathbb{R}$, and the loss incurred by the learner is $l^{(t)}(x^{(t)})$. Note that there is no assumption on the distribution from which the loss functions are drawn (as opposed to statistical learning models). The goal of online learning is to minimize the "regret": the difference between the total cost incurred by the algorithm and that of the best fixed decision in hindsight:

$$R_T = \sum_{t=1}^{T} l^{(t)}(x^{(t)}) - \min_{x\in X} \sum_{t=1}^{T} l^{(t)}(x), \tag{2.15}$$

To make this framework meaningful, the loss functions chosen by the adversary can not be allowed to be unbounded (otherwise the adversary can choose a high loss in the first time step, and subsequently select small losses to never allow the algorithm to recover from the loss of the first round). Loss functions $l^{(t)}$ can be convex in learner's strategy (the framework is then referred to as online convex optimization if X is also convex), or linear (online linear optimization), or can come from a fixed loss function $l^{(t)}(x^{(t)}) = l(x^{(t)}, y^{(t)})$ where $y^{(t)} \in Z$ is played by the adversary (online prediction, where $y^{(t)}$ is the true parameter that the algorithm is trying to predict and $l(\cdot, \cdot)$ is the loss that captures how good the prediction is). We are interested in the setting where $X = \mathcal{U}$ is the set of combinatorial strategies or the vertex set of a $0/1$ polytope and the losses are linear functions of the combinatorial strategies.

An algorithm is said to perform well if its regret is sublinear as a function of the T, i.e. $\lim_{T\to\infty} R_T = 0$, since it means that on an average the algorithm performs as well as the best fixed decision in hindsight. Such an online learning algorithm is said to have low regret or is simply called Hannan-consistent.

To develop some intuition, we first review a standard example of the online learning framework: prediction from experts advice. The decision maker or learner has to choose (possibly randomly) from the advice of n given experts. Thus, the decision set $X = \Delta_n = \{x \mid \sum_i x_i = 1, x \geq 0\}$. After selecting $x^{(t)} \in X$, a loss in $[0, 1]$ is revealed for each expert, i.e. $l^{(t)} \in [0, 1]^n$ is revealed, and the learner incurs a loss of $x^{(t)T} l^{(t)}$. Here, $x^{(t)T} l^{(t)}$ can be interpreted as the expected loss of the learner (under randomization to the $x^{(t)}$ over $[n]$). The goal of the learner is to perform as well as the best expert in hindsight, i.e. minimize $\sum_{t=1}^{T} x^{(t)T} l^{(t)} - \min_{i\in[n]} \sum_{t=1}^{T} l^{(t)}(i)$. A very intuitive weighted majority algorithm, also called the multiplicative weights update (MWU) algorithm, is known to achieve sublinear regret for this setting. The MWU algorithm starts with a uniform probability over all the experts. As losses for each expert are observed in the subsequent rounds, the algorithm multiplicatively reduces the probabilities such that the advice of experts with larger losses is taken with lower probability. We review the algorithm in more detail in Chapter 5. In the above example, one can also think of the experts being an exponential number of combinatorial strategies like paths, matchings, permutations, spanning trees. In this setting, the losses are often selected to be linear and can model congestion on the path, percentage clicks on a permutations, etc.

Next, it is useful to recall the online mirror descent algorithm, which is a variant of the previously mentioned mirror descent algorithm, and extends to many important settings within online learning (for instance, when only estimates of the gradient are available). The online adaptation is often attributed to Zinkevich [Zinkevich, 2003] and mirror descent is due to the seminal work of Nemirovski and Yudin in 1983 [Nemirovski and Yudin, 1983]. As in the case of mirror descent, this algorithm is also defined with respect to a mirror map $\omega : \mathcal{D} \to \mathbb{R}$ that is strictly-convex with respect to $\|\cdot\|$. The learner selects $x^{(t)} \in X$ ($X \subseteq \overline{\mathcal{D}}$) where we can think of X as a combinatorial polytope and the adversary is allowed to select G-Lipschitz convex loss functions $l^{(t)}$ in each round. The algorithm is the same as mirror descent, except that the gradient step is now computed with respect to the gradients of the loss functions $l^{(t)}$ (as opposed to a fixed convex function). The first iterate $x^{(1)} = \operatorname{argmin}_{x \in X} \omega(x)$. Subsequently, for $t > 1$, the algorithm first moves in an unconstrained way using

$$\nabla\omega(y^{(t+1)}) = \nabla\omega(x^{(t)}) - \eta\nabla l^{(t)}(x^{(t)}),$$

and the next iterate $x^{(t+1)}$ is obtained by the Bregman projection step:

$$x^{(t+1)} = \underset{x \in X \cap \mathcal{D}}{\operatorname{argmin}} D_\omega(x, y^{(t+1)}), \tag{2.16}$$

The regret of the online mirror descent algorithm is known to scale as $O(RG\sqrt{t})$ where recall that $R^2 = \max_{x\in X} \omega(x) - \min_{x \in X} \omega(x)$. We restate the theorem about the regret of the online mirror-descent algorithm.

Theorem 6 (see for e.g. [Rakhlin and Sridharan, 2014])**.** *Consider online mirror descent based on a κ-strongly convex (with respect to $||\cdot||$) and differentiable mirror map $\omega : \mathcal{D} \to \mathbb{R}$ on a closed convex set X ($X \subseteq \overline{\mathcal{D}}$). Let each loss function $l^{(t)} : X \to \mathbb{R}$ be convex and G-Lipschitz, i.e. $||\nabla l^{(t)}||_* \leq G \quad \forall t \in \{1, \dots, T\}$ and let the radius $R^2 = \max_{x \in X} \omega(x) - \min_{x\in X} \omega(x)$. Further, we set the learning rate $\eta = \frac{R}{G}\sqrt{\frac{2k}{T}}$ then:*

$$\sum_{t=1}^{T} l^{(t)}(x^{(t)}) - \sum_{t=1}^{T} l^{(t)}(x^*) \leq RG\sqrt{\frac{2T}{\kappa}} \quad \textit{for all } x^* \in X.$$

Even though the convex function is allowed to change in each round, the analysis of the algorithm does not change much compared to that of mirror descent, as in Theorem 2. In fact, setting each $l^{(t)} = h(\cdot)$ recovers the mirror descent algorithm for minimizing a convex function $h(\cdot)$. Further, we will see in Chapter 5 that the multiplicative weights update algorithm can also be recovered by performing online mirror descent with the unnormalized entropy mirror map over the simplex of experts ([Beck and Teboulle, 2003], also see [Bubeck, 2011] for a short proof).

We refer the interested reader to [Hazan, 2012] (for an overview of online convex optimization) and [Cesa-Bianchi and Lugosi, 2006] and [Audibert et al., 2013] (for background on online combinatorial optimization).

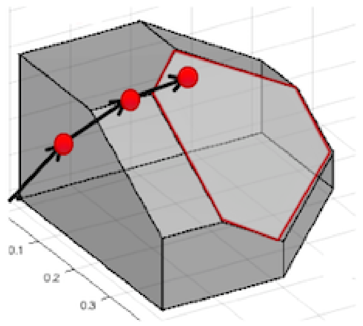
2.3 Related Work

We now discuss briefly the related work concerned with each chapter, and go into more details within each chapter. We start with summarizing the related work for minimizing separable convex functions over submodular base polytopes (P1), the key question considered in Chapter 3:

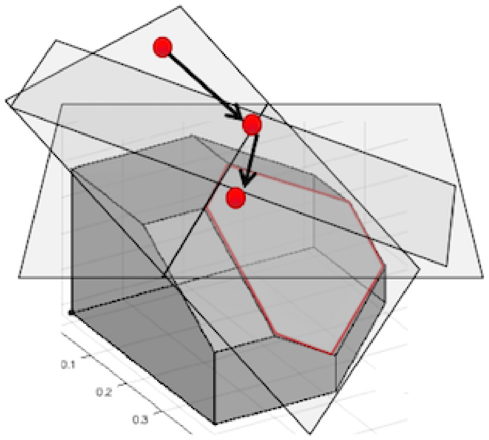
$$(\text{P1}) : \min_{x \in B(f)} h(x) := \sum_{e \in E} h_e(x(e)). \tag{2.17}$$

Separable convex minimization The related work on exact separable convex minimization (under infinite precision arithmetic) can be broadly characterized into primal-style approaches that always maintain a feasible point in the submodular polytope and dual-style approaches that work by finding violated inequalities while moving towards the submodular polytope.

In 1980, Fujishige gave a primal-style method, the *monotone* algorithm, to find the minimizer of $\min_{x \in B(f)} \sum_e x_e^2/w_e$ for a positive weight vector $w \in \mathbb{R}^E_{>0}$ [Fujishige, 1980]. Our algorithm Inc-Fix can be viewed as a generalization of the monotone algorithm, that works for minimizing any differentiable strictly convex and separable function. In 1991, Fujishige and Groenevelt developed a dual-style method, decomposition algorithm, for separable convex minimization over submodular base polytopes [Groenevelt, 1991]. It generates a sequence of violated inequalities and computes a feasible solution only at the completion of the algorithm. There has been a lot of work since then to speed up the decomposition algorithm and



(a) Primal-style methods



(b) Dual-style methods

Figure 2-1: Primal-style algorithms always maintain a feasible point in the submodular polytope $P(f)$. Dual-style algorithms work by finding violated constraints till they find a feasible point in $B(f)$.

show rationality of its solutions (for e.g. see [Nagano, 2007b]). Some other recent primal-style methods for minimizing specific convex functions over cardinality-based submodular polytopes include algorithms by Yasutake et al. [Yasutake et al., 2011] (for minimizing KL-divergence over the permutations base polytope), Suehiro et al. [Suehiro et al., 2012] (for minimizing KL-divergence and squared Euclidean distance over cardinality-based base polytopes) and Krichene et al. [Krichene et al., 2015] (for minimizing ϕ-divergences over the simplex). We give a modification of Inc-Fix, called Card-Fix, for minimizing separable convex functions over cardinality-based polytopes that subsumes these latter results.

One can also use general purpose projection-free convex minimization methods to find minimizers of these separable convex functions. One such alternative is to use the conditional gradient method or the Frank-Wolfe method [Frank and Wolfe, 1956]. The Frank-Wolfe method is attractive as it only requires to solve linear optimization as a subproblem, however it generates approximate minimizers, whereas the above mentioned algorithms, Inc-Fix, decomposition method, monotone algorithm are exact in nature (assuming infinite precision arithmetic). We discuss the tradeoffs of these approaches compared to our algorithm in more detail in Chapter 3.

Next, in Chapter 4, we consider the problem of computing maximum feasible movement along a direction starting with a point inside an extended submodular polytope:

$$\text{(P2): given } a \in \mathbb{R}^n, x_0 \in EP(f), \text{ compute } \max\{\delta \mid x_0 + \delta a \in EP(f)\},$$

and next, we summarize the related work for the parametric line search problem.

Parametric line search As we discussed in the introduction, a natural way to solve the parametric line search problem (P2) is to use a cutting plane approach: Dinkelbach's method or the discrete Newton's method. While a bound of n iterations was known when $a \geq 0$ (for e.g. [Topkis, 1978]), no bound better than exponential iterations was known for general directions before our work. We show a quadratic bound on the number of iterations of the discrete Newton's algorithm, which implies a worst-case running time for the parametric line search problem of $O(n^2)$ submodular function minimizations. The only other strongly polynomial algorithm for the parametric line search problem was due to Nagano et al. [Nagano, 2007b] that relies on Megiddo's parametric search framework and requires $\tilde{O}(n^8)$ submodular function minimizations. Some of our analysis draws ideas from Radzik's analysis of the discrete Newton's method for a related problem of $\max \delta : \min_{S\in\mathcal{S}} b(S) - \delta a(S) \geq 0$ where both a and b are modular functions and $\mathcal{S}$ is an arbitrary collection of sets [Radzik, 1998]. Our setting is both more general (since we consider submodular functions as opposed to modular functions) and restrictive (since we consider the power set of E as opposed to an arbitrary collection of sets) compared to his. We highlight similarities and differences from Radzik's analysis in Chapter 4.

The focal point of Chapter 5 is the multiplicative weights update (MWU) algorithm and its application to do online linear optimization over combinatorial strategies and for convex minimization over combinatorial polytopes. Next, we review the background for the multiplicative weights update method in this context.

Approximate generalized counting The multiplicative weights update algorithm has been rediscovered for different settings in game theory, machine learning, and online learning with a large number of applications (see [Arora et al., 2012] and the references therein). Most of the applications of the MWU algorithm have running times polynomial in the number of pure strategies of the learner, an observation also made in [Blum et al., 2008]. In order to simulate this algorithm efficiently for combinatorial strategies, it does not take much to see that for linear losses one can use product distributions over the combinatorial set and update them efficiently in each iteration. These product distributions have been used by [Helmbold and Schapire, 1997] (for learning over bounded depth binary decision trees), [Takimoto and Warmuth, 2003] (for learning over simple paths in directed graphs), [Koo et al., 2007] (for learning over spanning trees) to give a few examples. However, the analysis of prior works was very specific to the structure of the problem. We generalize and abstract the analysis to enable learning over vertices of 0/1 polytopes as long as there exists an efficient generalized approximate counting oracle. As a result, we can add to the list of problems where the MWU can be simulated efficiently by compiling known existing counting oracles.

The second part of the chapter talks about performing convex minimization over any 0/1 polytope P using the MWU algorithm that maintains a (product) probability distributions over its vertex set. We extend the framework for online linear optimization to minimize convex functions over combinatorial polytopes using approximate counting oracles. This generalizes known results where the MWU algorithm has been used to minimize convex functions over the n-dimensional simplex (however the simplex we consider lies in the space of an exponential number of vertices of the 0/1 polytope).

Finally, in Chapter 6, we discuss techniques for finding Nash-equilibria in two-player zero-sum games where each player plays a combinatorial object and discuss the applications of the above mentioned results. The use of online learning for finding Nash-equilibria in two-player zero-sum games has been known, as early as the work of Robinson [Robinson, 1951]. Under positive diagonal loss matrices for matroid games, where each player plays bases of a matroid, we show that the symmetric Nash-equilibria coincide with lexicographically optimal bases (studied in [Fujishige, 1980]). To the best of our knowledge, this connection has not been made before, and this results in another way of computationally finding symmetric Nash-equilibria (if they exist) using a single convex minimization.

Chapter 3

Separable Convex Minimization

"Whatever affects one directly, affects all indirectly."
- Martin Luther King, Jr.

Motivated by bottlenecks in various first-order optimization methods across game theory, online learning and convex optimization, in this chapter we consider the fundamental question of minimizing a separable strictly convex function over a submodular base polytope. Given a ground set E ($n = |E|$) of elements, we consider submodular set functions $f(\cdot)$ (refer to (2.1) for definition) that are monotone non-decreasing, i.e., $f(A) \leq f(B)$ for all $A \subseteq B \subseteq E$, normalized such that $f(\emptyset) = 0$ and non-negative such that $f(A) > 0$ for all $\emptyset \neq A \subseteq E$ (without loss of generality). As discussed in Chapter 2, the submodular polytope is defined as $P(f) = \{x \in \mathbb{R}^n_+ : \sum_{e \in U} x(e) \leq f(U) \; \forall \; U \subseteq E\}$ and the base polytope as $B(f) = \{x \in \mathbb{R}^n_+ : \sum_{e \in E} x(e) = f(E), x \in P(f)\}$. We consider the problem of minimizing separable strictly[1] convex and differentiable functions over submodular base polytopes:

$$(\mathrm{P1}) : \min_{x \in B(f)} h(x) := \sum_{e \in E} h_e(x(e)). \tag{3.1}$$

First-order projection-based optimization methods like mirror descent or online mirror descent are required to solve (P1), for computing a projection with respect to a certain convex metric called Bregman divergence. We refer the reader to Section 2.2.2 for background on

[1] Recall that $h : X \to \mathbb{R}$ is strictly convex if X is convex, and $h(\theta x + (1-\theta)y) < \theta h(x) + (1-\theta)h(y)$ for any $0 < \theta < 1$ and $x, y \in X, x \neq y$ (refer to Section 2.2.2).

these divergences and useful references on first-order methods. Some important examples of Bregman divergences that we will refer to throughout the chapter are:

(i) the squared Euclidean distance, $h(x) = \frac{1}{2}\|x-y\|_2^2$, for a given $y \in \mathbb{R}^E$,

(ii) KL-divergence, $h(x) = \sum_e (x_e \ln(x_e/y_e) - x_e + y_e)$, for a given[2] $y \in \mathbb{R}^E_{>0}$,

(iii) Logistic loss, $h(x) = \sum_e x_e \ln \frac{x_e}{y_e} + \sum_e (1-x_e) \ln \frac{1-x_e}{1-y_e}$, for $y \in (0,1)^E$ and,

(iv) Itakura-Saito distance, $h(x) = \sum_e (x_e/y_e - \ln(x_e/y_e) - 1)$, for $y \in \mathbb{R}^E_{>0}$.

Note that all the above-mentioned divergences are separable over the ground set. We review their domain and convexity properties in Table 3.1.

The main result of this chapter is a novel algorithm Inc-Fix for solving (P1). The key idea of the algorithm comes from first order optimality conditions, i.e. if a point x^* is a minimizer of a convex function $h : X \rightarrow \mathbb{R}$ over a convex set X, then it must hold that $\nabla h(x^*)^T(x^* - z) \leq 0$ for all points $z \in X$. Read differently, if one somehow knew the value of $\nabla h(x^*) = c$ (say), then x^* would minimize the linear function $c^T z$ over $z \in X$. This point is subtle, yet crucial, so we state it again as a question.

"Can one construct a gradient vector $\nabla h(x^*)$ such that the corresponding point x^* minimizes the corresponding first-order approximation of the convex function at x^*?"

This implies that perhaps for problems where linear optimization is well understood, one can devise a specialized convex minimization method by considering the first-order optimality conditions[3]. Linear optimization for submodular base polytopes is given by the well-known Edmonds' greedy algorithm [Edmonds, 1970]. We use a greedy increase in the gradient space to construct a point x^* that satisfies the first-order optimality condition. To be more specific, we start with 0 (or a point in the submodular polytope such that the partial derivatives with respect to all the elements are equal), and increase the value on elements with the lowest partial derivative. As these element values are increased, the corresponding partial derivative also increases (since h is strictly convex). By carefully maintaining the ordering of the partial

[2]By $y \in \mathbb{R}^E_{>0}$, we mean $y \in \mathbb{R}^E$ such that $y(e) > 0$ for all $e \in E$.

[3]This observation is independent of whether the polytope is submodular or not.

derivatives at every iterate of the algorithm as well as feasibility inside the submodular polytope $P(f)$, we ensure that the first-order approximation of the convex function at the constructed point is in fact minimized by that point itself. Informally, our main result in this chapter is the following.

Theorem 7 (informal). *Consider a strictly convex and differentiable separable function $\sum_{e\in E} h_e(\cdot) : \mathcal{D} \rightarrow \mathbb{R}$ such that mild technical conditions over the domain are satisfied. Then, the* Inc-Fix *algorithm starting with $0 \in \mathcal{D}$ or some $x_0 \in P(f)$ such that $\nabla h(x_0) = c\mathbf{1}$ for some c, results in $x^* = \arg\min_{z\in B(f)} \sum_e h_e(z(e))$.*

The rest of the chapter is organized as follows. We discuss the precise algorithm Inc-Fix in Section 3.1 and its proof of correctness in Section 3.2, along with equivalence of convex minimization problems and provable gaps from optimality in case of early termination. Inc-Fix requires to compute the maximum feasible increase in the partial derivatives of elements, and this is not quite straightforward to compute. It entails finding maximum δ such that $(\nabla h)^{-1}(\nabla h(x_0) + \delta\chi(M)) \in P(f)$, given $x_0 \in P(f), M \subseteq E$. We present a parametric gradient search method in Section 3.3.1, and show that the Inc-Fix algorithm can be implemented using $O(n)$ parametric submodular function minimizations (PSFM). We further show, in Section 3.3.2, that the Inc-Fix algorithm can also be implemented in overall $O(n)$ calls to submodular function minimizations (that returns the maximal minimizer), which is currently faster than performing $O(n)$ PSFM. The running time of our method does not depend on the convexity constants (smoothness or strong-convexity constants) of the convex function h.

Inc-Fix only requires oracle access to the value of the submodular function $f(\cdot)$. However, if some more information about the structure of the submodular function is known, then it can be exploited for obtaining faster running times. We specifically consider cardinality-based submodular functions that can be defined as $f(S) = g(|S|)$ for some concave function $g(\cdot)$. We show that a variant of the Inc-Fix algorithm, Card-Fix, can be implemented overall in $O(n(\log n + d))$ time (Section 3.4) for minimizing uniform divergences, where d is the number of distinct values of the submodular function. This gives the fastest known running time for separable convex minimization over cardinality-based submodular polytopes.

Both Inc-Fix and Card-Fix require to find the zero of a univariate monotone function as a subproblem. This can be as simple as dividing two sums (in the case of minimizing the squared Euclidean distance) or might require the use of a binary search or Newton's method (in the case of minimizing the Itakura-Saito divergence). In all our running times, we assume a constant time oracle for computing this zero.

3.1 The Inc-Fix algorithm

In this section, we discuss our algorithm Inc-Fix to minimize any strictly convex and differentiable separable function $h : \mathcal{D} \to \mathbb{R}$, defined over a convex set $\mathcal{D} \subseteq \mathbb{R}^E$. Separability and strict convexity allow us to work in the space of gradients such that increasing the partial derivatives with respect to any element results in a well-defined increase on the value of the corresponding element. Since our function h is separable, its domain $\mathcal{D}$ is the product of domains $\mathcal{D}_e$ for each h_e. In the Inc-Fix algorithm, we increase the value of the elements starting with a feasible point $x^{(0)} \in P(f)$, such that feasibility in $P(f)$ is always maintained. Thus, we require that $P(f) \subseteq \mathcal{D}$, i.e. $[0, f(\{e\})] \in \mathcal{D}_e$ for all $e \in E$. We can relax this condition to allow for $P(f) \subseteq \overline{\mathcal{D}}$ (i.e., the closure of $\mathcal{D}$). This is useful, for instance, for minimizing the KL-divergence over the base polytopes with respect to some $y \in \mathbb{R}^E_{>0}$, as the domain of the KL-divergence is $\mathbb{R}^E_{>0}$, however $0 \in P(f)$. We next require that $B(f) \cap \mathcal{D}$ must be non-empty, otherwise the minimization over $B(f)$ is not well-defined. There are a very few corner cases when $B(f) \cap \mathcal{D} = \emptyset$ while $P(f) \subseteq \overline{\mathcal{D}}$. Since $[0, f(\{e\})] \subseteq \overline{\mathcal{D}}_e$ for all e, and $f(\{e\}) > 0$ by assumption, the only way that $B(f) \cap \mathcal{D} = \emptyset$ is if $f(\{e\}) \notin \mathcal{D}_e$ for some e and $x_e = f(e)$ for all $x \in B(f)$, i.e. $f(E) = f(\{e\}) + f(E \setminus \{e\})$. Finally, for the ease of exposition of the proofs in the chapter, we assume that $\nabla h(\mathcal{D}) = \mathbb{R}^E$ (this condition is not restrictive). To summarize the above conditions, we require (i) $P(f) \subseteq \overline{\mathcal{D}}$, (ii) $B(f) \cap \mathcal{D} \neq \emptyset$, and (iii) $\nabla h(\mathcal{D}) = \mathbb{R}^E$.

Our next condition is to help in the choice of the starting point for the algorithm. We require that either $0 \in \mathcal{D}$ (observe that $0 \in P(f) \subseteq \overline{\mathcal{D}}$) or there exists some $x \in P(f)$ such that $\nabla h(x) = c\chi(E), c \in \mathbb{R}$, where $\chi(S)$ denotes the characteristic vector of a set $S \subseteq E$. This is useful in selecting a starting point $x^{(0)}$ such that $x^{(0)}$ has a lower partial-derivative element-wise compared to the optimal solution (even if the optimal solution is not known). For instance, for minimizing the squared Euclidean distance, $h(x) = \frac{1}{2}\|x-y\|^2$ with $\mathcal{D} = \mathbb{R}^E$, and the starting point of Inc-Fix can be $0 \in P(f)$. For minimizing the KL-divergence with respect to some $y \in \mathbb{R}^E_{>0}$, we note that $\mathcal{D} = \mathbb{R}^E_{>0}$ and hence $0 \notin \mathcal{D}$. However, we can select the starting point to be cy for some $0 < c < 1$ such that $cy \in P(f)$ (this ensures that the partial derivative is the same for all elements). It is easy[4] to see that such a constant c exists due to our assumption on f that $f(A) > 0$ for $\emptyset \neq A \subseteq E$.

We list some valid choices for the starting point $x^{(0)}$ for minimizing various uniform divergences in Table 3.2. As we assume h to be separable, we use $(\nabla h(x))_e$ and $h'_e(x(e))$ interchangeably.

$\omega(x) = \sum w(x_e)$	$\mathcal{D}$	w'	$(w')^{-1}$	$\nabla\omega(\mathcal{D})$	strong-convexity parameter κ of $\omega(\cdot)$
$\omega(x) = \|x\|^2/2$	$\mathbb{R}^n$	x	x	$\mathbb{R}^n$	$\kappa = 1$ w.r.t. $\|\cdot\|_2$
$\omega(x) = \sum_e (x_e \log x_e - x_e)$	$\mathbb{R}^n_{>0}$	$\log x$	e^x	$\mathbb{R}^n$	$\kappa = 1/f(E)$ w.r.t. $\|\cdot\|_1$ over base polytopes $B(f)$
$\omega(x) = -\sum_e \log x_e$	$\mathbb{R}^n_{>0}$	$-1/x$	$-1/x$	$\mathbb{R}^n_{>0}$	$\kappa \geq 1/M^2$ w.r.t. $\|\cdot\|_2$ for $\|x\|_\infty \leq M$
$\omega(x) = \sum_e x_e \log x_e + \sum_e (1-x_e)\log(1-x_e)$	$(0,1)^n$	$\ln(\frac{x}{1-x})$	$\frac{e^x}{1+e^x}$	$\mathbb{R}^n$	$\kappa = 2/f(E)$, w.r.t. $\|\cdot\|_1$ over base polytopes $B(f)$

Table 3.1: Examples of strictly convex functions and their domains, derivatives with their domains, inverses and their strong-convexity parameters. Refer to Section 2.2.2 for a discussion.

$\omega(x)$	$D_\omega(x,y)$	Choice for $x^{(0)}$ such that $x^{(0)} \in P(f)$
$\|x\|^2/2$	$\sum_e (x_e - y_e)^2$	$x^{(0)} = 0$
$\sum_e (x_e \log x_e - x_e)$	$\sum_e \big(x_e \log(x_e/y_e) - x_e + y_e\big)$	$x^{(0)} = e^\delta y$, $\delta \leq 0$
$\sum_e - \log x_e$	$\sum_e \big(x_e/y_e - \log(x_e/y_e) - 1\big)$	$x_e^{(0)} = y_e/(1-\delta y_e)$, $\delta \leq 0$ $\forall e \in E$
$\sum_e x_e \log x_e + \sum_e (1-x_e)\log(1-x_e)$	$\sum_e x_e \log(x_e/y_e) + (1-x_e)\log((1-x_e)/(1-y_e))$	$x_e^{(0)} = e^\delta y_e/(1-y_e+e^\delta y_e)$, $\delta \leq 0$ $\forall e \in E$

Table 3.2: Valid choices for the starting point $x^{(0)}$ when minimizing $D_\omega(x,y)$ using the Inc-Fix algorithm, such that either $x^{(0)} = 0$ or $\nabla h(x^{(0)}) = \delta\chi(E)$. In each case, we can select δ to be sufficiently negative such that $x^{(0)} \in P(f)$.

The Inc-Fix algorithm The algorithm is iterative and it maintains a vector $x \in P(f) \cap \mathcal{D}$. During the execution of the algorithm, some elements will get tight and thus we will *fix* them

[4]Since we assume in this chapter that f is monotone and $f(A) > 0$ for all non-empty subsets A, we can define $x \in \mathbb{R}^n$ as $x(e) = \frac{1}{n}f(\{e\})$ for all $e \in E$. Note that $x \in P(f)$ as it is the average of n points in $P(f)$. One way to select c such that $cy \in P(f)$, is to set $c = \min_e x(e)/y(e)$.

so that we do not change their value any more. We increase the values on only the *non-fixed* elements. When considering x, we associate a weight vector given by $\nabla h(x)$, let M be the set of minimum weight elements that have not been fixed and refer to the maximal tight set with respect to x as $T(x)$ (unique by submodularity of f, Lemma 2.1). We move x within $P(f)$ in a direction such that $h'_e(x_e)$ increases uniformly on elements in M, until one of two things happen: (i) either continuing further would violate a constraint defining $P(f)$, i.e. $T(x)$ changes or (ii) the set M of elements of minimum weight changes. If the former happens, we *fix* the tight elements and continue the process on non-fixed elements. If the latter happens, then we continue to *increase* the value of the elements in the modified set of minimum weight elements. Starting with an appropriate $x = x^{(0)} \in P(f)$, Inc-Fix algorithm can be stated simply as follows:

(1.) $M = \arg\min_{e \in E \setminus T(x)} h'_e(x_e)$

(2.) While maintaining feasibility in $P(f)$, uniformly increase the value of the partial derivative of the elements in M, until (i) $T(x)$ changes, or (ii) M changes.

(3.) If $T(x) \neq E$, go to Step (1.).

The complete description of the Inc-Fix algorithm with the help of a pseudocode is given in Algorithm 1. The additional accounting of tight elements as $M_i \cap T(x^{(i)})$ in step (14) helps in proving the correctness of the algorithm. Step (8) computes the second highest partial derivative value amongst non-fixed elements (to track changes in M). Step (9) computes the maximum possible increase, ϵ_2, in the partial derivatives of elements in M, while staying in P(f). Note that even though ϵ_1 might be unbounded, ϵ_2 is always bounded as ∇h is a strictly increasing function. As $h'_e(x(e))$ is increased, the corresponding $x(e)$ increases while being bounded by the base polytope.

We next discuss Examples 1 and 2 to illustrate how the gradients are increased in each iteration for developing intuition.

Example 1. Consider minimizing squared Euclidean distance $h(x) = \frac{1}{2}\|x - y\|^2$, $x \in B(f) \subseteq \mathbb{R}^3_{>0}$ from a given point $y = (0.05, 0.07, 0.6)$. Here, we consider a cardinality-based submod-

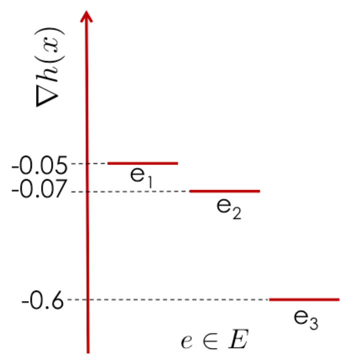
Algorithm 1: Inc-Fix

```
input  : $f : 2^E \to \mathbb{R}$, $h = \sum_{e \in E} h_e$, and $x^{(0)} \in P(f)$
output: $x^* = \arg\min_{z \in B(f)} \sum_e h_e(z(e))$
   $N_0 = E, i = 0$
   repeat                                                        // Main loop starts
       $i \leftarrow i + 1$, $x = x^{(i-1)}$
6      $M = \mathrm{argmin}_{e \in N_{i-1}} h'_e(x_e)$
7      while $T(x) \cap M = \emptyset$ do                         // Inner loop starts
8          $\epsilon_1 = \min_{e \in N_{i-1} \setminus M} h'_e(x_e) - \min_{e \in N_{i-1}} h'_e(x_e)$
9          $\epsilon_2 = \max\{\delta : (\nabla h)^{-1}(\nabla h(x) + \delta \chi(M)) \in P(f)\}$
10         $x \leftarrow (\nabla h)^{-1}(\nabla h(x) + \min(\epsilon_1, \epsilon_2)\chi(M))$
11         $M \leftarrow \mathrm{argmin}_{e \in N_{i-1}} h'_e(x_e)$
       end
13     $x^{(i)} = x$, $M_i = M$                                    // Bookkeeping
14     $N_i = N_{i-1} \setminus (M_i \cap T(x^{(i)}))$              // Fix tight elements in $M_i$
   until $N_i = \emptyset$;                            // Until no element can be increased
   Return $x^* = x^{(i)}$.
```

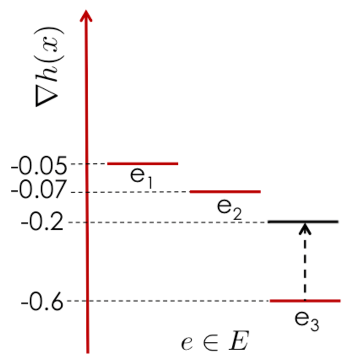
ular function[5] $f : f(S) = g(|S|)$ where $g(\emptyset) = 0$, $g(1) = 0.4$, $g(2) = 0.6$, $g(3) = 0.7$. Inc-Fix starts with $x^{(0)} = 0$. Note that $\nabla h(x^{(0)}) = 0 - y$, thus the set of elements with the minimum partial derivative at the start is $M = \{e_3\}$. Increase in gradient space by ϵ corresponds to an increase in the value of the element by ϵ as well. Thus, $x(e_3)$ is increased till M changes or a tight constraint is hit. At $x(e_3) = 0.4$, the submodular constraint $x(e_3) \leq f(\{e_3\}) = g(1)$ becomes tight, and the algorithm fixes the value of e_3. Thus, $x^{(1)} = (0, 0, 0.4)$. The set of minimum gradient elements that are not yet fixed is now $M = \{e_2\}$, and e_2 is raised until M changes. Thus, $x^{(2)} = (0, 0.02, 0.4)$ when M increases to $\{e_1, e_2\}$. In the last iteration, e_1 and e_2 are increased uniformly, to obtain $x^{(3)} = (0.14, 0.16, 0.4)$. We illustrate the different states of the computation in Figure 3-1, in the gradient space as well as in the submodular polytope.

Example 2. Next, let us consider the case of minimizing KL-divergence from the same point, as in Example 1, $y = (0.05, 0.07, 0.6)$ over the base polytope $B(f)$, as in Example 1. We start the algorithm with $x^{(0)} = e^c y$ (we pick $c = -3$ so that $x^{(0)} \in P(f)$) and thus $\nabla h(x^{(0)}) = (\ln(x_e^{(0)}/y_e))_e = -3$. Since each element has equal partial derivative value, $M = \{e_1, e_2, e_3\}$. Increase in gradient space by ϵ, corresponds to an increase in the value of the elements proportional to y. The first increase results in $x(e_3) = 0.4$, thus setting the corresponding

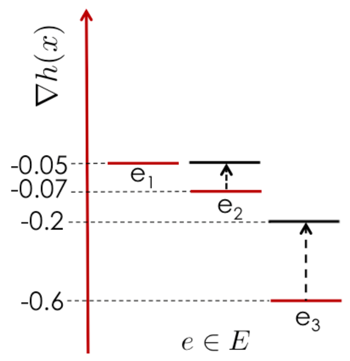
[5]Refer to Section 3.4 for more details on cardinality-based functions.



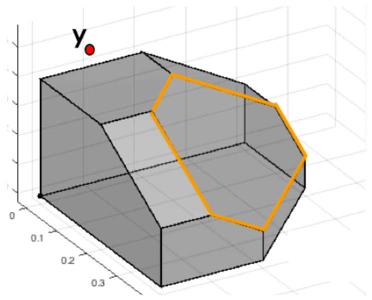
(a) Initial gradients at $x^{(0)}$



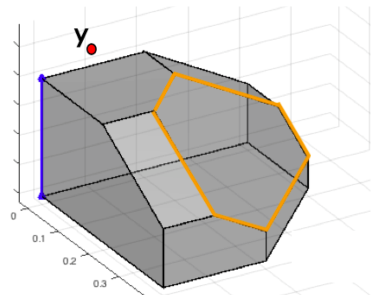
(b) Increase to $x^{(1)}$, Fix e_3



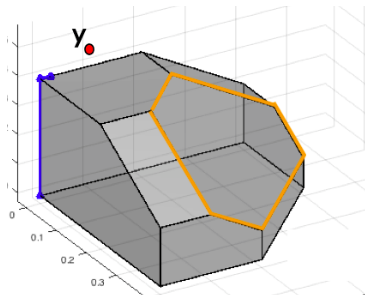
(c) Increase to $x^{(2)}$



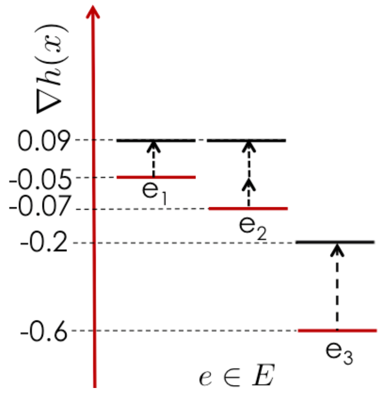
(d) $x^{(0)} = 0$, $B(f)$ is the highlighted face of $P(f)$ as described in the main caption.



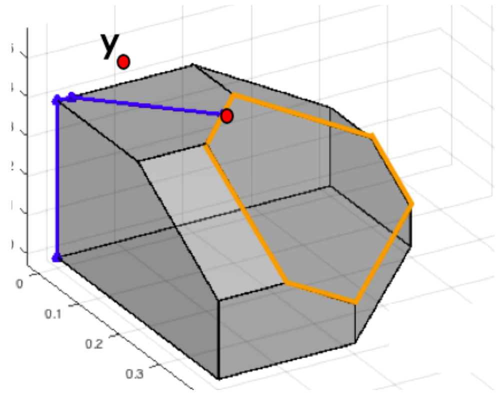
(e) $x^{(1)}$ is obtained by increasing e_3. Fix e_3 due to tight constraint.



(f) $x^{(2)}$ is obtained by increasing e_2. M changes.



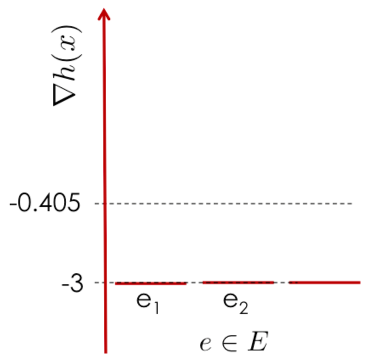
(g) Increase to $x^{(3)} = x^*$



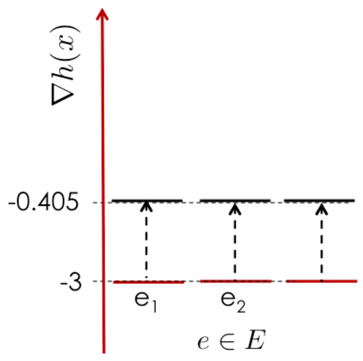
(h) The optimal solution $x^{(3)} \in B(f)$ is obtained by increasing both e_1, e_2.

Figure 3-1: Illustrative gradient space and polytope view of Example 1 that shows Inc-Fix computations for projecting $y = (0.05, 0.07, 0.6)$ under the squared Euclidean distance onto $B(f)$, where $f(S) = g(|S|)$ and $g = [0.4, 0.6, 0.7]$. Projected point is $x^{(3)} = (0.14, 0.16, 0.4)$.

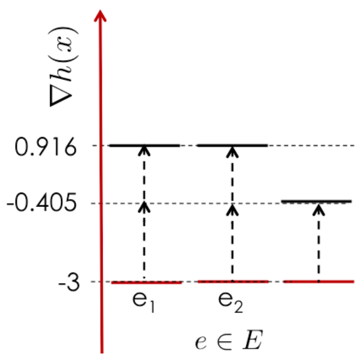
submodular constraint tight. We get $x^{(1)} = (\nabla h)^{-1}(-0.405, -0.405, -0.405) \Rightarrow x^{(1)}(e_3) = e^{-0.405}0.6 = 0.4$. The set of minimum gradient elements that are not yet fixed is now



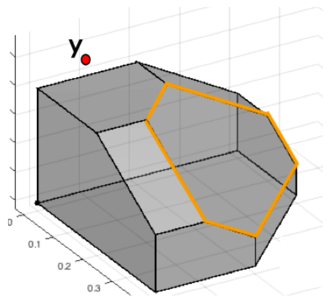
(a) Initial partial derivatives at $x^{(0)}$



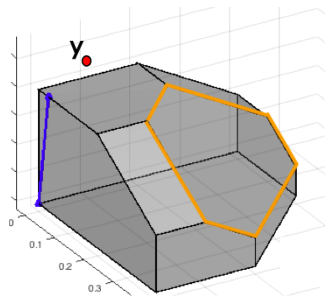
(b) Increase to $x^{(1)}$, Fix e_3



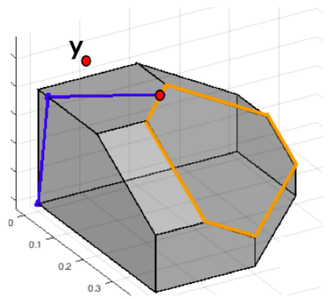
(c) Increase to $x^{(2)} = x^*$.



(d) $x^{(0)} = 0$, $B(f)$ is the highlighted face of $P(f)$ as in Figure 3-1.



(e) $x^{(1)}$ is obtained by increasing all elements proportional to y. Fix e_3.



(f) $x^{(2)} = x^*$ is obtained by increasing e_1, e_2 proportional to $y(e_1, e_2)$.

Figure 3-2: Illustrative gradient space and polytope view of Example 2 that shows Inc-Fix computations for projecting $y = (0.05, 0.07, 0.6)$ under KL-divergences onto $B(f)$, where $f(S) = g(|S|)$, $g = [0.4, 0.6, 0.7]$. Projected point is $x^{(3)} = (0.125, 0.175, 0.4)$.

$M = \{e_1, e_2\}$. The next increase in value of e_1 and e_2 proportional to y gives the optimal solution $x^* = x^{(2)} = (0.125, 0.175, 0.4)$. We illustrate the different states of the computation in Figure 3-2, in the gradient space as well as in the submodular polytope.

3.2 Correctness of Inc-Fix

The correctness of the algorithm follows from the first-order optimality conditions and Edmonds' greedy algorithm. It crucially relies on the following theorem (that holds irrespective of $h(\cdot)$ begin separable).

Theorem 8. *Consider any differentiable convex function $h : \mathcal{D} \to \mathbb{R}$, and a monotone* *submodular function $f : 2^E \to \mathbb{R}$ with $f(\emptyset) = 0$. Let $B(f) \cap \mathcal{D} \neq \emptyset$. For $x^* \in \mathbb{R}^E$, let $F_1, F_2, \ldots, F_k$ be a partition of the ground set E such that $(\nabla h(x^*))_e = c_i$ for all $e \in F_i$ and $c_i < c_j$ for $i < j$. Then, $x^* = \arg\min_{z \in B(f)} h(z)$ if and only if x^* lies in the face H_{opt} of $B(f)$ given by $H_{opt} := \{z \in B(f) | \; z(F_1 \cup \ldots \cup F_i) = f(F_1 \cup \ldots \cup F_i) \; \forall \, 1 \leq i \leq k\}$.*

Proof. By first-order optimality conditions, we know that $x^* = \text{argmin}_{z \in B(f)} \sum_e h_e(x(e))$ if and only if $\nabla h(x^*)^T(z - x^*) \geq 0$ for all $z \in B(f)$. This is equivalent to x^* being in $\text{argmin}_{z \in B(f)} \nabla h(x^*)^T z$. Now consider the partition $F_1, F_2, \ldots, F_k$ as defined in the statement of the theorem. Using Edmonds' greedy algorithm [Edmonds, 1971], we know that any $z^* \in B(f)$ is a minimizer of $\nabla h(x^*)^T z$ if and only if it is tight (i.e., full rank) on each $F_1 \cup \ldots F_i$ for $i = 1, \ldots, k$, i.e., z^* lies in the face H_{opt} of $B(f)$ given by

$$H_{opt} := \{z \in B(f) | \; z(F_1 \cup \ldots \cup F_i) = f(F_1 \cup \ldots \cup F_i) \; \forall \, 1 \leq i \leq k\}.$$

□

At the end of the Inc-Fix algorithm, there may be some elements at zero value (specifically in cases where $x^{(0)} = 0$). In our proof for correctness for the algorithm, we use the following simple lemma about zero-valued elements that requires the submodular function to be monotone non-decreasing.

Lemma 3.1. *For $x \in P(f)$, if a subset S of elements is tight then so is $S \setminus \{e : x(e) = 0\}$.*

Proof. Let $S = S_1 \cup S_2$ such that $x(S_2) = 0$ and $x(e) > 0$ for all $e \in S_1$. Then, $f(S_1) \geq x(S_1) = x(S_1 \cup S_2) = f(S_1 \cup S_2) \geq f(S_1)$, where the last inequality follows from monotonicity of f, implying that we have equality throughout. Thus, $x(S_1) = f(S_1)$. □

We now state the main theorem about correctness of the algorithm.

Theorem 9. *Consider a differentiable strictly convex and separable function $\sum_{e \in E} h_e(\cdot) : \mathcal{D} \to \mathbb{R}$ and a monotone submodular function $f : 2^E \to \mathbb{R}$ with $f(\emptyset) = 0$. Assume $P(f) \subseteq \overline{\mathcal{D}}$, $B(f) \cap \mathcal{D} \neq \emptyset$ and $\nabla h(\mathcal{D}) = \mathbb{R}^E$ such that either $0 \in \mathcal{D}$ or there exists $x^{(0)} \in P(f)$ such that $\nabla h(x^{(0)}) = c\chi(E)$ for $c \in \mathbb{R}$. Then, the output of* Inc-Fix *algorithm is $x^* = \text{argmin}_{z \in B(f)} \sum_e h_e(z(e))$.*

Proof. Note that since $h(x) = \sum_e h_e(x(e))$ is separable and strictly convex, h'_e is a strictly increasing function for each $e \in E$. Note that $(\nabla h)^{-1}$ is well-defined for all points in $\mathbb{R}^E$ since we assume $\nabla h(\mathcal{D}) = \mathbb{R}^E$ (this will be useful in implementing step (9)). Consider the optimal solution $x^* = \arg\min_{x \in B(f)} \sum_e h_e(x_e)$ and let us partition the elements of the ground set E into $F_1, F_2, \ldots, F_k$ such that $h'_e(x^*(e)) = c_i$ for all $e \in F_i$ and $c_i < c_j$ for $i < j$. We will show that $F_i = M_i \cap T(x^{(i)})$ and that k is the number of main loop iterations of Algorithm 1.

We first claim that in each iteration $i \geq 1$ of the main loop, if $x = x^{(i-1)} \in P(f)$ and $N_{i-1} \neq \emptyset$ at the start of the inner loop (steps (7)-(11)) then the inner loop terminates. At the start of the inner loop, $M = \arg\min_{e \in N_{i-1}} h'_e(x_e^{(i-1)})$, the set of non-fixed elements N_{i-1} with the minimum partial derivative. Since $N_{i-1} \neq \emptyset$, M is well-defined. By the definition of ϵ_1, it is the partial derivative value at which we must update the minimum set of elements M. ϵ_2 ensures that the potential increase in the partial derivatives is such that the corresponding point $x \in P(f)$ ($\epsilon_2 \geq 0$ since $x^{(i-1)} \in P(f)$). Finally, x is obtained by increasing the partial derivatives on M by $\min(\epsilon_1, \epsilon_2)$ amount and M is updated correctly. Observe that in each iteration of the inner loop either the size of M increases (in the case when $\epsilon_1 = \min(\epsilon_1, \epsilon_2)$) or the size of $T(x)$ increases (in the case when $\epsilon_2 = \min(\epsilon_1, \epsilon_2)$). Therefore, the inner loop must terminate with $T(x) \cap M \neq \emptyset$.

Next, we show that if $x = x^{(i-1)} \in P(f)$, $N_{i-1} \neq \emptyset$ and $M = \arg\min_{e \in N_{i-1}} h'_e(x_e^{(i-1)})$ at the start of the inner loop (steps (7)-(11)) then the inner loop returns $x = x^{(i)} \in P(f)$ such that the following properties hold:

(a.i) $x^{(i)} \geq x^{(i-1)}$. This claim holds as we always increase the partial derivatives on elements in M, resulting in the corresponding element values to increase as $h'_e(\cdot)$ is strictly increasing for each $e \in E$.

(a.ii) $x^{(i)}(e) = x^{(i-1)}(e)$ for $e \in E \setminus N_{i-1}$. This claim holds since we never increase the value on elements not in N_{i-1}. M is always a subset of N_{i-1}.

(a.iii) The set of minimum elements grows as the inner loop proceeds. Whenever an increase in the partial derivatives corresponds to the choice of ϵ_1 (the second lowest partial derivative value in N_{i-1}), the set of elements with the minimum partial derivative increases.

(a.iv) $x^{(i)} \in P(f)$. This is easy to see as we never increase the tight elements $T(x)$.

(a.v) $x^{(i)}(e) = x^{(i-1)}(e)$ for $e \in N_{i-1} \setminus M_i$. This claim holds since M_i must contain all elements that were potentially increased in value (using (a.iii)).

We refer to the new value of the partial derivatives of elements in M_i as $\epsilon^{(i)}$, i.e. $\epsilon^{(i)} = \min_{e \in N_{i-1}} h'_e(x_e^{(i)})$. Let the set of elements fixed at the end of each iteration be $L_i = M_i \cap T(x^{(i)})$. We next prove the following claims at the end of each iteration $i \geq 1$ of the main loop.

(b). Since the inner loop exits when $L_i = M_i \cap T(x^{(i)}) \neq \emptyset$, we get that $N_i \subset N_{i-1}$. Note that the set of elements fixed by the end of iteration i is $E \setminus N_i = L_1 \dot\cup \ldots \dot\cup L_i$[6].

(c). We claim that the set of minimum elements M_i at the end of any iteration i always contains the non-fixed minimum elements from the previous iteration, i.e., $M_{i-1} \setminus L_{i-1} \subseteq M_i$. This is clear if $L_{i-1} = M_{i-1}$, so consider the case when $L_{i-1} \subset M_{i-1}$. At the beginning of the inner loop of iteration i, $M = \text{argmin}_{e \in N_{i-1}} h'_e(x^{(i-1)}(e)) = M_{i-1} \setminus L_{i-1}$. Subsequently, in the inner loop the set of minimum elements can only increase and thus, $M_{i-1} \setminus L_{i-1} \subseteq M_i$.

(d). We next show that the partial derivatives of elements in M_i can only increase, i.e. $\epsilon^{(i-1)} < \epsilon^{(i)}$ for $i \geq 2$. Consider an arbitrary iteration $i \geq 2$. If $L_{i-1} = M_{i-1}$, then

$$\epsilon^{(i)} = \min_{e \in N_{i-1} = N_{i-2} \setminus L_{i-1}} h'_e(x^{(i)}(e)) \tag{3.2}$$

$$\geq \min_{e \in N_{i-2} \setminus M_{i-1}} h'_e(x^{(i-1)}(e)) \tag{3.3}$$

$$> \min_{e \in M_{i-1}} h'_e(x^{(i)}(e)) = \epsilon^{(i-1)}, \tag{3.4}$$

where (3.3) follows from (a.i). Otherwise, we have $L_{i-1} \subset M_{i-1}$. This implies that $M_{i-1} \setminus L_{i-1}$ is not tight on $x^{(i-1)}$, and it is in fact equal to $\text{argmin}_{e \in N_{i-1}} h'_e(x^{i-1}(e))$ ($= M$ at the beginning of the inner loop in iteration i). As this set is not tight, the partial derivatives can be increased and therefore $\epsilon^{(i)} > \epsilon^{(i-1)}$.

[6]By $\dot\cup$ we mean disjoint union of sets.

Claim (b) implies the termination of the algorithm when for some t, $N_t = \emptyset$. We also obtain a partition of E into disjoint sets $\{L_1, L_2, \ldots, L_t\}$ using (b). From claims (a) and (b), we get $x^{(t)}(e) = \epsilon^{(i)}$ for $e \in L_i$. Claim (d) implies that the partition in the theorem $\{F_1, \ldots, F_k\}$ is identical to the partition obtained via the algorithm $\{L_1, \ldots, L_t\}$ (hence $t = k$).

To show that $x^{(t)}$ is indeed tight on each $H_i = L_1 \cup \ldots L_i$ for $1 \leq i \leq t$, we consider two cases (e) and (e′), depending on how the starting point $x^{(0)}$ is initialized.

(e). When $x^{(0)} = 0 \in P(f)$, we claim that $x^{(i)}(e) = 0$ for $e \in N_i \setminus M_i$. Using (c) we get $N_i \setminus M_i = N_{i-1} \setminus M_i \subseteq N_{i-1} \setminus (M_{i-1} \setminus L_i)$. Since $L_i \cap N_i = \emptyset$, we get $N_i \setminus M_i \subseteq N_{i-1} \setminus M_{i-1}$. Using induction (a.v), we get $x^{(i)}(e) = x^{(i-1)}(e) = \ldots = x^{(0)}(e) = 0$ for all $e \in N_i \setminus M_i$.

(e′) When $x^{(0)} = \nabla h^{-1}(c\chi(E)) \in P(f)$, we claim that $M_i = N_{i-1}$. For iteration $i = 1$, it is clear that $M_1 = E = N_0$ since $h'_e(x^{(0)}(e)) = c$ for all $e \in E$. For iteration $i > 1$, we have $N_{i-1} = N_{i-2} \setminus L_{i-1} = M_{i-1} \setminus L_{i-1}$ (by induction on $i - 2$). This implies that $h'_e(x^{(i-1)}(e)) = \epsilon^{(i-1)}$ for all the elements in N_{i-1}.

Finally, we will show by induction that for any iteration $i \geq 1$, $x^{(i)}(H_i) = f(H_i)$. This will complete the proof of correctness of Algorithm 1, by using Theorem 8.

Base case: To show $x^{(1)}(L_1) = f(L_1)$ when $x^{(0)} = 0$, note $T(x^{(1)}) = L_1 \dot{\cup} (T(x^{(1)}) \setminus M_1)$. Since $T(x^{(1)}) \setminus M_1 \subseteq N_1 \setminus M_1$, we get $x^{(1)}(T(x^{(1)}) \setminus M_1) = 0$ using (e). Using Lemma 3.1, we get that $x^{(1)}$ is tight on L_1. When $x^{(0)} = \nabla h^{-1}(c\chi(E))$, $M_1 = E$ using (e′), therefore $T(x^{(1)}) = L_1$, proving the claim in this case.

Assume the induction hypothesis to be true for $1 \leq j < i$ and consider the case of $j = i$. Since $x^{(i)} \geq x^{(i-1)}$, $x^{(i)}$ must also be tight on $L_1 \cup \ldots \cup L_{i-1}$. Note that $T(x^{(i)})$ can be partitioned into $\{\big(T(x^{(i)}) \cap (E \setminus N_{i-1}\big), \big(T(x^{(i)}) \cap (N_{i-1} \setminus M_i)\big), \big(T(x^{(i)}) \cap M_i\big)\} = \{\big(L_1 \cup \ldots \cup L_{i-1}\big), \big(T(x^{(i)}) \cap (N_i \setminus M_i)\big), L_i\}$ using (b). Note that $x^{(i)}$ is either zero-valued on $N_i \setminus M_i$ using (e) or $N_i \setminus M_i = \emptyset$ using (e′). In either case using Lemma (3.1), we get that $x^{(i)}$ is also tight on $\big(L_1 \cup \ldots \cup L_i\big)$.

Finally, by induction this implies that $x^* = x^{(t)}$ lies in the face H_{opt} as defined in Theorem 8. □

The analysis of the Inc-Fix algorithm uses the fact that the submodular function is monotone non-decreasing (in Lemma 3.1). However, we can relax this condition to consider non-negative submodular functions by using a correspondence between nonempty polymatroids and monotone non-decreasing submodular functions.

Non-monotone non-negative submodular functions: Consider the case when f is a non-negative submodular function that is not necessarily monotone. For any non-negative submodular function f, there exists a monotone submodular function $\bar{f}$ such that $P(f) = P(\bar{f})$ (see e.g., Section 44.4 of [Schrijver, 2003]). The Inc-Fix algorithm can use $\bar{f}$ instead of f, however it never needs to compute $\bar{f}$ explicitly. The parametric gradient search (step (9) in Algorithm 1) can be equivalently performed on $P(f)$, and it is easy to show that the maximum tight set of any $x \in P(f)$ is the same as the maximum tight set with respect to $\bar{f}$.

3.2.1 Equivalence of problems

A useful corollary of Theorem 8 is the equivalence of various convex minimization problems such that the partial derivatives of the corresponding minimizers partition the ground set into the same sets $F_1, \ldots, F_k$. The following result generalizes the equivalence known for certain convex functions, as discussed in [Nagano and Aihara, 2012].

Corollary 1. *Consider any two strictly-convex separable functions $h : \mathcal{D}_h \to \mathbb{R}$ and $g : \mathcal{D}_g \to \mathbb{R}$ where $h = \sum_{e \in E} h_e(x(e))$, $g(x) = \sum_{e \in E} g_e(x(e))$. Consider any normalized, monotone submodular function $f : 2^E \to \mathbb{R}$. Assume $\nabla h(\mathcal{D}_h) = \mathbb{R}^E$, $P(f) \subseteq \overline{\mathcal{D}_h}$ and $B(f) \cap \mathcal{D}_h \neq \emptyset$ (respectively for g). If there exists a strictly monotone (increasing) function $\eta(\cdot) : \mathbb{R} \to \mathbb{R}$, such that $\eta(h'_e(x)) = g'_e(x)$ $\forall x \in P(f) \cap \mathcal{D}_h \cap \mathcal{D}_g$ then* $\operatorname{argmin}_{z \in B(f)} \sum_{e \in E} h_e(z(e)) = \operatorname{argmin}_{z \in B(f)} \sum_{e \in E} g_e(z(e))$.

Proof. Consider $k, l \in E$ such that $h'_k(x) < h'_l(x)$ then $\eta(h'_k(x)) < \eta(h'_l(x)) \Rightarrow g'_k(x) < g'_l(x)$ for all $x \in P(f) \cap \mathcal{D}_h \cap \mathcal{D}_g$. Hence, the partition $F_1, \ldots, F_k$ defined in Theorem 8 is the same for both the convex functions h, g implying the equivalence of optimal solutions. □

We would like to highlight the fact that the iterate $x^{(i)}$ follows the same trajectory in the algorithm Inc-Fix for equivalent convex optimization problems in the sense described above. We can further simplify the exposition of Algorithm 1 for two important special cases of Bregman divergences: the KL-divergence and the squared Euclidean distance.

KL-divergence is given by $D_\omega(x, y) = \sum_{e\in E} x(e)\ln(x(e)/y(e)) - \sum_{e\in E} x(e) + \sum_{e\in E} y(e)$ with respect to $y \in \mathbb{R}^E_{>0}$, and we need to find the maximum possible increase in the gradient[7] $\nabla D_\omega = \big(\ln(x(e)/y(e))\big)_{e\in E}$ while remaining in $P(f)$. It is easy to show that step (9) in Inc-Fix corresponds to finding $\max\{\delta : x + z \in P(f), z(e) = (e^\delta - 1)x(e) \text{ for } e \in M, z(e) = 0 \text{ for } e \notin M\}$. Another way to project under the KL-divergence is to consider the convex function $g_y(x) = \sum_{e\in E} x(e)^2/2y(e)$ such that $\nabla g_y(x) = x/y$. By Corollary 1, the minimum of the KL-divergence $D_\omega(\cdot, y)$ and $g_y(\cdot)$ is obtained by the same point in the base polytope. It is interesting that minimizing g_y allows the choice of starting point $x^{(0)} = 0$, even though 0 is not in the domain of $D_\omega(\cdot, y)$. We describe the simplified algorithm in Algorithm[8] 2.

Algorithm 2: Inc-Fix For Minimizing KL-Divergence

```
input : $f : 2^E \to \mathbb{R}$, $f$ nonnegative and monotone, $y \in \mathbb{R}^E_{>0}$
output: $x^* = \text{argmin}_{z\in B(f)} \sum_e z_e \ln(z_e/y_e)$
3  $N_0 = E, i = 0, x^{(0)} = 0$
   ... same as Algorithm 1, except simplifying lines 6-13 as follows...
6  $M = \text{argmin}_{e\in N_{i-1}} x_e/y_e$
7  while $T(x) \cap M = \emptyset$ do
9      $\epsilon_2 = \max\{\delta : x + \delta y \cdot \chi(M)) \in P(f)\}$
10     $x \leftarrow x + \epsilon_2 y \cdot \chi(M)$;    ... since $\epsilon_1 = \infty$
11     $M = \text{argmin}_{e\in N_{i-1}} x_e/y_e$
   end
13 $x^{(i)} = x$, $M_i = \text{argmin}_{e\in N_{i-1}} x_e/y_e$
```

Squared Euclidean distance given a $y \in \mathbb{R}$, is $D_\omega(x, y) = \frac{1}{2}\|x - y\|^2$. Here, $\nabla D_\omega(x, y) = x - y$, which simplifies the step (9) in Inc-Fix to $\max\{\delta : x + \delta\chi(M) \in P(f)\}$. We describe the simplified algorithm in Algorithm 3.

3.2.2 Rounding to approximate solutions

Note that whenever the Inc-Fix method is terminated with an $x^{(i)}$ (after completing iteration i), the values on the tight set of elements $T(x^{(i)})$ remains the same throughout the

[7]Recall that we overload the notation $\nabla D_\omega(x, y)$ to denote $\partial_x D_\omega(x, y) = \nabla\omega(x) - \nabla\omega(y)$ (Section 2.2.2).
[8]In step (8), by $y \cdot \chi(M)$ we mean the vector $d(e) = y(e)$ if $e \in M$, $d(e) = 0$ otherwise.

Algorithm 3: Inc-Fix For Minimizing Euclidean Distance

```
input  : $f : 2^E \to \mathbb{R}, f$ nonnegative and monotone, $y \in \mathbb{R}^n$
output: $x^* = \text{argmin}_{z \in B(f)} \sum_e \|z - y\|^2$
3  $N_0 = E, i = 0, x^{(0)} = 0$
   ... same as Algorithm 1, except simplifying lines 6-13 as follows...
6  $M = \text{argmin}_{e \in N_{i-1}}(x_e - y_e)$
7  while $T(x) \cap M = \emptyset$ do
8      $\epsilon_1 = \min_{e \in N_{i-1} \setminus M}(x_e - y_e) - \min_{e \in N_{i-1}}(x_e - y_e)$
9      $\epsilon_2 = \max\{\delta : x + \delta\chi(M) \in P(f)\}$
10     $x \leftarrow x + \min(\epsilon_1, \epsilon_2)\chi(M)$
11     $M = \text{argmin}_{e \in N_{i-1}}(x_e - y_e)$
   end
13 $x^{(i)} = x$, $M_i = \text{argmin}_{e \in N_{i-1}}(x_e - y_e)$
```

algorithm. This results in multiple ways in which intermediate iterates of the algorithm can be round to the base polytope to obtain approximate solutions to the convex minimization problem. One can order elements in $N = E \setminus T(x^{(i)})$ arbitrarily as $e_1, \ldots, e_{|N|}$. Let the approximate solution $\tilde{x}$ be $\tilde{x}(e_j) = f(T(x^{(i)}) \cup \{e_1, \ldots, e_j\}) - f(T(x^{(i)}) \cup \{e_1, \ldots, e_{j-1}\})$ for all $e_j \in N$, $\tilde{x}(e) = x^{(i)}(e)$ otherwise. It is easy to check that $\tilde{x} \in B(f)$. Another way to think about this rounding process is to consider any $x_N \in B(f_{T(x^{(i)})})$, the base polytope of the contracted submodular function $f_{T(x^{(i)})}$, such that $f_{T(x^{(i)})}(S) = f(S \cup T(x^{(i)})) - f(T(x^{(i)}))$ (refer to Definition 2). Then, $\tilde{x}$ is given by $\tilde{x}(e) = x_N(e)$ for $e \in N$, $\tilde{x}(e) = x^{(i)}(e)$ otherwise.

Gap from optimality Let x^* be the unique minimum of the convex function $h(\cdot)$ minimized over a base polytope $B(f)$ using the Inc-Fix algorithm. Intermediate iterates $x^{(i)}$ in the algorithm enjoy the property that once an element is tight, its value does not change throughout the algorithm. This helps in bounding the gap from the optimal solution value $h(x^*)$. Next we discuss three ways to obtain lower bounds, each with a different computational requirement and tightness of the bound.

We know that $x^{(i)}(e) = x^*(e)$ for all $e \in T(x^{(i)})$ and $x^{(i)}(e) \geq x^*(e)$ for $e \in E \setminus T(x^{(i)})$. Using convexity of the function $h(\cdot)$, we get the first lower bound:

$$h(x^*) \geq h(x^{(i)}) + \nabla h(x^{(i)})^T(x^* - x^{(i)}) \tag{3.5}$$

$$\geq h(x^{(i)}) - \nabla h(x^{(i)})^T x^{(i)} + \min_{z \in B(f),\, \ell \leq z \leq u} z^T \nabla h(x^{(i)}), \tag{3.6}$$

where $\ell, u \in \mathbb{R}^E$ such that $\{\ell_e, u_e\}$ are the best lower and upper bounds computed on the value of $x^*(e)$. At the start of the Inc-Fix algorithm, one can set $\ell_e = 0, u_e = f(\{e\})$ for each $e \in E$. However these bounds can be updated as more information is obtained, for instance $\ell_e \geq x^{(i)}(e)$ for any intermediate iterate $x^{(i)}$ of the Inc-Fix algorithm (we discuss later in Section 3.3.2 how the upper bound u_e can be updated as the algorithm progresses). A submodular polytope intersected with box constraints $\{z \mid l \leq z \leq u\}$ results in a polymatroid (see for e.g. Theorem 3.3 in [Fujishige, 2005]), and therefore the minimization in (3.6) can be computed using Edmonds' greedy algorithm.

The second lower bound can be obtained by relaxing (3.6) and optimizing $z^T \nabla h(x^{(i)})$ only over the box constraints $z_e \in [l_e, u_e]$ (and not intersect with the base polytope $B(f)$):

$$h(x^*) \geq h(x^{(i)}) - \nabla h(x^{(i)})^T x^{(i)} + \sum_{e \in E \setminus T(x^{(i)})} d_e h'_e(x_e^{(i)}), \tag{3.7}$$

where $d_e = l_e$ when $h'_e(x_e^{(i)}) > 0$ and $d_e = u_e$ otherwise. This bound can be computed in $O(1)$ time, however it is much weaker than (3.6).

Instead of using the first-order approximation of the convex function, given lower and upper bounds $[\ell_e, u_e]$ on the optimal value of each element e, we can obtain another lower bound by simply minimizing the convex functions h_e over $[\ell_e, u_e]$:

$$h(x^*) \geq \min_{z \mid \ell_e \leq z_e \leq u_e} \sum_e h_e(z_e). \tag{3.8}$$

The time required to compute this bound depends on the complexity of the convex function, however this results in a tighter bound compared to (3.7).

Let the lower bound on the value of $h_e(x_e^*)$ be $h_L^{(i)}(e)$ obtained using (3.6), (3.7) or (3.8) after iteration i. Suppose $x^{(i)}$ is rounded to $\tilde{x} \in B(f)$ as described above, we can bound its gap from optimality in a straightforward manner:

$$\frac{h(\tilde{x}) - h(x^*)}{h(x^*)} \leq \frac{h(\tilde{x}) - \sum_e h_L^{(i)}(e)}{\sum_e h_L^{(i)}(e)} = \frac{\sum_{e \in E \setminus T(x^{(i)})} h(\tilde{x}_e) - \sum_{e \in E \setminus T(x^{(i)})} h_L^{(i)}(e)}{\sum_{e \in T(x^{(i)})} h(x_e^{(i)}) + \sum_{e \in E \setminus T(x^{(i)})} h_L^{(i)}(e)}. \tag{3.9}$$

As the tight set of the current iterate $x^{(i)}$ increases, the gap closes to zero.

3.3 Implementing the Inc-Fix algorithm

A parametrized increase in the gradient space in the Inc-Fix algorithm (step (9) in Algorithm 1, see (3.10) below) will, in general, result in a movement along a piecewise smooth curve in the submodular polytope $P(f)$, which is non-trivial to compute. In this section, we show how each maximum possible increase in the gradient space, i.e. step (9), can be computed with the help of $O(1)$ parametric submodular function minimizations (SFMs). This implies a worst-case overall running time of $O(n)$ parametric SFMs for the Inc-Fix algorithm. Using properties of convex minimizers over base polytopes, we further improve the overall running time of the Inc-Fix method to require only $O(n)$ submodular function minimizations in Section 3.3.2.

3.3.1 $O(n)$ parametric gradient searches

In this section, we discuss a parametric gradient search method to solve for step (9) of Inc-Fix (Algorithm 1):

$$\delta^* = \max \delta \text{ such that } (\nabla h)^{-1}(\nabla h(x_0) + \delta \chi(M)) \in P(f), \tag{3.10}$$

for a given $x_0 \in P(f)$ and $M \subseteq E$ is the subset of non-fixed elements with the minimum partial derivative with respect to x_0 (all elements in M have the same partial derivative value). Recall that $h(\cdot)$ is differentiable, strictly convex and separable. Let $\bar{x}_\delta$ correspond to a vector with gradient δ over elements in E, i.e. $\bar{x}_\delta(e) = (h'_e)^{-1}(\delta)$ for $e \in E$. Since h'_e is a strictly increasing function, $\bar{x}_\delta(e) = (h'_e)^{-1}(\delta)$ increases monotonically with increasing δ for all $e \in E$. Suppose we were to minimize Bregman divergences $D_\omega(x, y)$ corresponding to *uniformly separable* mirror maps $\omega(x) = \sum_e w(x_e)$ where $w : \mathcal{D}_w \to \mathbb{R}$ is a strictly-convex function (see Table 2.2). In this case, $h(x) = D_\omega(x, y) = \sum_e (w(x_e) - w(y_e) - w'(y_e)(x_e - y_e))$, $h'_e(x_e) = w'(x_e) - w'(y_e)$, and thus $\bar{x}_\delta(e) = (w')^{-1}(\delta + w'(y_e))$. We give the closed form expressions of $\bar{x}_\delta(e)$ for popular uniform divergences:

$$\bar{x}_\delta(e) = \begin{cases} \delta + y_e & \text{for } w(x) = \|x\|^2/2, D_\omega(x,y) = \frac{1}{2}\|x-y\|^2, \\ e^\delta y_e & \text{for } w(x) = x\log x - x, D_w(x,y) = \sum_e (x_e \ln(x_e/y_e) - x_e + y_e), \\ -1/(\delta - 1/y_e) & \text{for } w(x) = -\log x, D_w(x,y) = \sum_e \big(x_e/y_e - \log(x_e/y_e) - 1\big), \\ \frac{e^\delta y_e}{1-y_e+e^\delta y_e} & \text{for } w(x) = x\log x + (1-x)\log(1-x), \\ & \quad D_w(x,y) = \sum_e (x_e \log(x_e/y_e) + (1-x_e)\log(\frac{(1-x_e)}{(1-y_e)}). \end{cases}$$

In what follows, we are required to find δ such that $\sum_{e\in S} \bar{x}_\delta(e) = f(S\cup T) - f(T)$, for $S, T \subseteq E$. By our assumption that $\nabla h(\mathcal{D}) = \mathbb{R}^E$, we know that these univariate equations always have a solution. Note that for the squared Euclidean distance, $\sum_{e\in S} \bar{x}_\delta(e) = \sum_{e\in S}(\delta + y_e)$, and therefore the solution is simply $\delta = \frac{f(S\cup T) - f(T) - y(S)}{|S|}$. For the KL-divergence, it is easy to check that the solution is $\delta = \log((f(S\cup T) - f(T))/y(S))$. In general, we know that $\sum_e \bar{x}_\delta(e)$ is an increasing function of δ, and therefore one can use binary search or Newton's method to find the solution. We will henceforth assume a constant time oracle to solve equations of the form $\sum_{e\in S} \bar{x}_\delta(e) = f(S\cup T) - f(T)$.

We now discuss how to compute the maximal feasible increase in the gradient space, i.e. (3.10). Recall that during each iteration of the inner loop in the Inc-Fix algorithm, we either increase the number of non-fixed elements with the minimum partial derivative value (i.e., the size of M), or set at least one more element to be tight, i.e.

$$T(x^*) \supset T(x_0), \text{ where } x^* = \nabla h^{-1}(\nabla h(x_0) + \delta^* \chi(M)).$$

Using a single submodular function minimization (to compute the maximal minimizer), one can check if $T(x_0) \cap M = \emptyset$. If the intersection is not empty, then $\delta = 0$. Therefore, let us consider the case when $\delta^* > 0$.

Let $x|_S$ be the vector restricted to elements in S and 0 otherwise. We want to find maximum δ such that $x_0|_{E\setminus M} + \bar{x}_\delta|_M \in P(f)$ given that $x_0 \in P(f)$. Equivalently, we can find maximum δ such that $\bar{x}_\delta|_M \in P(f - x_0|_{E\setminus M})$ or

$$\text{find maximum } \delta \text{ such that } \bar{x}_\delta|_M \in P(f'), \tag{3.11}$$

for a non-negative submodular function f' (since $x_0|_{E\setminus M} \in P(f)$).

One can solve (3.11) as follows: start with a point $\bar{x}_{\delta_1}|_M$ with δ_1 such that the entire ground set E is tight. If the corresponding $\bar{x}_{\delta_1}|_M$ is feasible for $P(f')$, we stop. Otherwise, we repeatedly find the maximally violated submodular constraint for set S_i (say) and construct $\bar{x}_{\delta_{i+1}}|_M$ such that S_i is tight, until we find some $\bar{x}_{\delta_k}|_M \in P(f)$. We refer to this process as a parametric gradient search and describe it formally in Algorithm 4. Note that step (a) requires solving univariate equations of the form $\sum_{e\in S} \bar{x}_\delta(e) = f(S \cup T) - f(T)$ as discussed above.

Algorithm 4: PARAMETRIC GRADIENT SEARCH

```
input  : non-negative submodular $f'$, $M \subseteq E$
output: $\max \delta : \bar{x}_\delta|_M \in P(f')$
    $S_0 = E$, $i = 0$;
    repeat
        $i \leftarrow i + 1$
(a)     $\delta_i : \bar{x}_{\delta_i}|_M(S_{i-1}) = f'(S_{i-1})$
        $S_i$ = maximal minimizer of $(f' - \bar{x}_{\delta_i}|_M)$
    until $f'(S_i) - \bar{x}_{\delta_i}|_M(S_i) \geq 0$;
    Return $\delta_i$
```

We claim that Algorithm 4 finds maximum δ in at most n iterations, each requiring a submodular function minimization. Correctness of the algorithm is trivial, since $\{\delta_i\}$ is a strictly decreasing sequence and for the last iteration k, $\delta > \delta_k$ would violate the submodular constraint for set S_{k-1}. Next, we claim that $k \leq n$. As is mentioned above, $\bar{x}_\delta$ is an increasing function of δ. Thus, $f' - \bar{x}_{\delta_1}|_M \leftarrow f' - \bar{x}_{\delta_2}|_M \cdots \leftarrow f' - \bar{x}_{\delta_k}|_M$ is strong map sequence (refer to Definition 3 in Chapter 2) for $\delta_1 > \delta_2 > \ldots > \delta_k$. Therefore, the maximal minimizer S_i of $f' - \bar{x}_{\delta_i}|_M$ contains in the maximal minimizer S_j of $f' - \bar{x}_{\delta_j}|_M$ for $\delta_i > \delta_j$ (Lemma 2.3). This bounds the number of iteration k by the size n of the ground set.

All of the submodular function minimizations in Algorithm 4 can be computed using a single parametric submodular function minimization, for example, using the result of [Nagano, 2007a] that requires $O(n^6 + \gamma n^5)$ running time, where γ is the time required for a single submodular function evaluation. Note that it is an open question to adapt the fastest known strongly polynomial submodular function minimization algorithm of [Lee et al., 2015] to do parametric minimization.

Running time for Inc-Fix using gradient searches Using parametric gradient searches for each gradient increase in the Inc-Fix algorithm, we get an overall running time of $O(n)$ parametric submodular function minimizations. Using the current fastest known parametric SFM method of [Nagano, 2007a], we get a worst-case running time of $O(n^7 + \gamma n^6)$ for Inc-Fix. We believe that it should be possible to implement the Inc-Fix algorithm using $O(1)$ parametric minimizations since we only increase the values on the elements in a well-defined way, however, we leave it as an open question (see Chapter 7). We summarize the running times for the Inc-Fix method in Table 3.3 using different ways of solving the gradient search subproblem.

3.3.2 $O(n)$ submodular function minimizations

We next discuss a more refined implementation of the Inc-Fix algorithm using $O(n)$ submodular functions minimizations overall. Note that these minimizations can be solved using the fastest known strongly polynomial algorithm of [Lee et al., 2015], without requiring to solve parametric submodular function minimizations.

Consider $x^* = \arg\min_{x\in B(f)} \sum_e h_e(x(e))$. Let $F_1, F_2, \ldots, F_k$ be a partition of the ground set E such that $h'_e(x^*(e)) = c_i$ for all $e \in F_i$ and $c_i < c_j$ for $i < j$. Let $H_s = F_1 \cup \ldots \cup F_s$ for $s = 1, \ldots, k$. As before, we let $\bar{x}_\delta$ be the point such that the partial derivative of each element is δ, i.e. $\bar{x}_\delta(e) = (h'_e)^{-1}(\delta)$ for $e \in E$. The key observation required to achieve an $O(n)$ SFM running time for Inc-Fix is an extension of the following lemma from [Nagano, 2007b]. We include its proof here for completeness.

Lemma 3.2 ([Nagano, 2007b])**.** *Consider δ: $c_s \le \delta < c_{s+1}$. If $c_s < \delta$ then H_s is the unique minimizer of $f_\delta := f - \bar{x}_\delta$. If $\delta = c_s$ then H_{s-1} is the unique minimal minimizer and H_s is the unique maximal minimizer of f_δ.*

Proof. Suppose $c_s < \delta < c_{s+1}$. As h'_e is strictly increasing, $x^*(e) - \bar{x}_\delta(e) < 0$ if $e \in H_s$ and $x^*(e) - \bar{x}_\delta(e) > 0$ otherwise. By Theorem 8, we know that $x^*(H_s) = f(H_s)$ for all s. Thus, for each $X \subseteq E$, with $X \neq H_s$, we have

$$f_\delta(X) = f(X) - \bar{x}_\delta(X) \ge x^*(X) - \bar{x}_\delta(X)$$

$$= (x^* - \bar{x}_\delta)(H_s) + (x^* - \bar{x}_\delta)(X \setminus H_s) - (x^* - \bar{x}_\delta)(H_s \setminus X)$$
$$> x^*(H_s) - \bar{x}_\delta(H_s) = f(H_s) - \bar{x}_\delta(H_s) = f_\delta(H_s).$$

On the other hand, suppose $\delta = c_s$ holds. Then $e \in H_{s-1}$ if and only if $x^*(e) - \bar{x}_\delta(e) < 0$ and $e \in H_s$ if and only if $x^*(e) - \bar{x}_\delta(e) \leq 0$. So, H_{s-1} and H_s minimize f_δ and any minimizer X satisfies $H_{s-1} \subseteq X \subseteq H_s$. □

We prove an extension of the above lemma. This will be useful in each iteration $i+1$ after we have obtained $x^{(i)}|_{H_i} = x^*|_{H_i}$, and we want to find the next increase up to $x^{(i+1)}$ in Algorithm 1.

Lemma 3.3. *Consider $i \leq s$ and $\delta : c_s \leq \delta < c_{s+1}$. If $c_s < \delta$ then H_s is the unique minimizer of $f_\delta := f - x^*|_{H_i} - \bar{x}_\delta|_{E \setminus H_i}$. Otherwise, if $\delta = c_s$ then H_{s-1} is the unique minimal minimizer and H_s is the unique maximal minimizer of $f - x^*|_{H_i} - \bar{x}_\delta|_{E \setminus H_i}$.*

The proof is along similar lines as Lemma 3.2, as we show here.

Proof. For $c_s < \delta < c_{s+1}$, we know that $x^*(e) - \bar{x}_\delta|_{E \setminus H_i}(e) > 0$ for $e \in E \setminus H_s$, and $x^*(e) - \bar{x}_\delta|_{E \setminus H_i}(e) \leq 0$ for $e \in H_s \cap (E \setminus H_i)$. Consider any $X \subseteq E$, such that $X \neq H_s$.

$$\begin{aligned} f_\delta(X) &= f(X) - x^*|_{H_i}(X) - \bar{x}_\delta|_{E \setminus H_i}(X) \\ &\geq x^*(X) - x^*(X \cap H_i) - \bar{x}_\delta(X \cap (E \setminus H_i)) \\ &= \sum_{e \in X \setminus H_s} (x^*(e) - \bar{x}_\delta(e)) + \sum_{e \in (X \setminus H_i) \cap H_s} (x^*(e) - \bar{x}_\delta(e)). \end{aligned} \tag{3.12}$$

However, the value of $f_\delta(H_s)$ can be broken down as follows -

$$\begin{aligned} f_\delta(H_s) &= f(H_s) - x^*|_{H_i}(H_s) - \bar{x}_\delta|_{E \setminus H_i}(H_s) \\ &= x^*(H_s) - x^*(H_i) - \bar{x}_\delta(H_s \cap (E \setminus H_i)) \\ &= \sum_{e \in H_s \setminus (X \setminus H_i)} (x^*(e) - \bar{x}_\delta(e)) + \sum_{e \in H_s \cap (X \setminus H_i)} (x^*(e) - \bar{x}_\delta(e)). \end{aligned} \tag{3.13}$$

Comparing (3.12) and (3.13), we get that $f_\delta(X) \geq f_\delta(H_s)$ and the inequality is strict if $\delta < c_{s+1}$ and $X \setminus H_s \neq \emptyset$. □

Algorithm 5: INC-FIX using $O(n)$ submodular function minimizations

```
input  : $f : 2^E \to \mathbb{R}$, $h = \sum_{e \in E} h_e$, and a valid $x^{(0)} \in P(f)$
output: $x^* = \arg\min_{z \in B(f)} \sum_e h_e(z(e))$
3    $N_0 = E, i = 0, \mathcal{C} = \{\emptyset, E\}$
     ... same as Algorithm 1, except implement the inner loop as follows...
7    while $T(x) \cap M = \emptyset$ do                                  // Inner loop starts
8        $\epsilon = \min_{e \in N_{i-1} \setminus M} h'_e(x_e)$
9.a      Let $H$ be the smallest set in $\mathcal{C}$ that strictly contains $E \setminus N_{i-1}$
9.b      Set $\delta_H : \bar{x}_{\delta_H}(M) = f(H) - f(E \setminus N_{i-1})$
9.c      $(\delta, S_1, \dots, S_{k_i}) = $ PARAMETRICGRADIENTSEARCH'$(f - x|_{E \setminus N_{i-1}}, M, \min\{\epsilon, \delta_H\})$
9.d      $\mathcal{C} \leftarrow \mathcal{C} \cup \{S_1, \dots, S_{k_i}\}$
10.e     $x(e) \leftarrow \bar{x}_\delta(e)$ for $e \in M$
11       $M \leftarrow \operatorname{argmin}_{e \in N_{i-1}} h'_e(x_e)$
     end
     ... where PARAMETRICGRADIENTSEARCH' (Algorithm 6) is a slight modification of Algorithm 4.
```

Algorithm 6: PARAMETRICGRADIENTSEARCH$'$

```
input  : $f' : 2^E \to \mathbb{R}$, $M \subseteq E$ and $\delta_1 \in \mathbb{R}$
output: $(\delta_k, S_1, \dots, S_k)$: $\delta_k$ is the maximum $\delta \le \delta_1$ such that $\bar{x}_\delta|_M \in P(f')$
    $i = 1$
    $S_1 = $ maximal minimizer of $f' - \bar{x}_{\delta_1}|_M$
    while $f'(S_i) - \bar{x}_{\delta_i}|_M(S_i) < 0$ do
        $\delta_{i+1} : \bar{x}_{\delta_{i+1}}|_M(S_i) = f'(S_i)$
        $S_{i+1} = $ maximal minimizer of $(f' - \bar{x}_{\delta_{i+1}}|_M)$
        $i \leftarrow i + 1$
    end
    Return $(\delta_i, S_1, \dots, S_i)$
```

Consider the optimal solution $x^* = \arg\min_{x \in B(f)} h(x)$ and let $\{F_1, F_2, \dots, F_k\}$ be a partition of the ground set E such that $h'_e(x^*_e) = c_i$ for all $e \in F_i$ and $c_i < c_j$ for $i < j$. In the INC-FIX algorithm, at the end of iteration i of the main loop, we know that we have fixed elements in $H_i = F_1 \cup \ldots \cup F_i$. When doing a parametric gradient search to compute the next feasible increase, every submodular function minimization of the form $f - x^*|_{H_i} - \bar{x}_\delta|_{E \setminus H_i}$ helps us either check for feasibility up to the second highest gradient level or discover some prefixes of the optimal partition $\{F_1, \dots, F_k\}$, using Lemma 3.3. Thus we can re-use the information obtained from parametric gradient search in an iteration of the INC-FIX algorithm in future iterations as well. This helps prove a linear bound on the total number of submodular function minimizations required in INC-FIX.

We give the complete description of INC-FIX while maintaining known level sets $H_s = F_1 \cup \ldots F_s$ and using them in future computations, in Algorithm 5. We start the algorithm with $\mathcal{C} = \{\emptyset, E\}$, the trivial level sets in the optimal partition, that we know of. The algorithm proceeds by maintaining the set M of elements with the minimum partial derivative value that are not fixed at the start of iteration i, i.e. $M \subseteq N_{i-1}$ (with $N_0 = E$). INC-FIX increases the partial derivative value of elements in M either up to the second highest partial derivative in N_{i-1} (i.e. line 8, ϵ) or the highest partial derivative while maintaining feasibility in $P(f)$ using a parametric gradient search, whichever increase is smaller. We compute the partial derivative value δ_H required on elements in M such that the next known level set H is tight ($H \in \mathcal{C}$ such that it strictly contains the current fixed set $E \setminus N_{i-1} = H_{i-1}$). Then a parametric gradient search using Algorithm 6 is performed, starting with the partial derivative value $\min\{\epsilon, \delta_H\}$.

Algorithm 6 is the same as Algorithm 4 except that it starts with a given partial derivative value δ_1 and returns the entire sequence of minimizers found during the parametric search. Given a non-negative submodular function f', a subset $M \subseteq E$ and a partial derivative value δ_1, it returns the sequence of maximal minimizers $S_1, \ldots, S_k$ obtained by first minimizing $f' - \bar{x}_{\delta_1}|_M$. It also returns δ_k that is the maximum $\delta \leq \delta_1$ such that $\bar{x}_\delta|_M \in P(f')$.

We call Algorithm 6 to do a parametric gradient search with submodular function $f' = f - x|_{E \setminus N_{i-1}}$, over the set M of non-fixed elements with minimum partial-derivative value as described above, starting with $\delta_1 = \min\{\epsilon, \delta_H\}$. It returns the sequence of minimizers $S_1, \ldots, S_{k_i}$ maximum $\delta \leq \min\{\epsilon, \delta_H\}$ such that $\bar{x}_\delta|_M \in P(f - x|_{E \setminus N_{i-1}})$. One of the following cases will hold:

(i) $\bar{x}_{\delta_1}|_M \in P(f - x|_{E \setminus N_{i-1}})$: In this case, the parametric gradient search will return $\delta = \delta_1$ and only one set S_1 that is the maximal minimizer of $f - x|_{E \setminus N_{i-1}} - \bar{x}_{\delta_1}|_M$ ($k_i = 1$). By Lemma 3.3, we know that S_1 is a level set in the optimal solution. If $S_1 = E \setminus N_{i-1}$, then $S_1 \cap M = \emptyset$. In this case, we only update M and continue the inner loop (raising elements to the second highest partial derivative did not increase the size of the tight set). Otherwise, we discovered the next level set $S_1 \supset E \setminus N_{i-1}$. In this case, we update M, fix elements in S_1, add S_1 to $\mathcal{C}$ and break the inner loop.

(ii) $\bar{x}_{\delta_1}|_M \notin P(f - x|_{E \setminus N_{i-1}})$: In this case, the parametric gradient search returns a sequence of minimizers $S_1, \ldots, S_{k_i}$ (recall $S_1 \supset S_2 \supset \ldots S_{k_i}$) and the highest δ ($< \delta_1$) such that

$\bar{x}_\delta|_M \in P(f - x|_{E\setminus N_{i-1}})$. We know that $S_1 \subseteq H$ since H is the smallest known level set of the optimal solution that strictly contains $E \setminus N_{i-1}$. Note that each of $S_2, \ldots, S_{k_i}$ (and S_1 as well if $S_1 \neq H$) are new level sets of the optimal solution that we have discovered from this computation (using Lemma 3.3), and therefore we can add these to $\mathcal{C}$. We thus increase the value of elements $e \in M$ to $\bar{x}_\delta(e)$, update M, and break the inner loop as the maximal tight set $S_{k_i} \cap M \neq \emptyset$.

Since we record the discovered level sets $\mathcal{C}$ in the optimal solution, we can start subsequent parametric gradient searches with a much tighter bound on partial derivative values and as a result provide a linear bound on the overall number of submodular function minimizations.

Running time The number of submodular function minimizations that result in only increasing the gradient level to the second highest partial derivative value while not fixing any new elements is at most n (case (i) when $S_1 = E \setminus N_{i-1}$). In every other scenario, the tight set increases. We get either $S_1 = H$ or $S_1 \subset H$ as $\delta \leq \delta_H$. The former can happen at most n times (the maximum number of possible level sets). In the latter case when $S_1 \subset H$, we discover new level sets $S_1, \ldots, S_{k_i}$, where all of the sets were not known to be prefixes of the optimal partition before this computation. Thus, across all the iterations of Inc-Fix we know that $\sum_i k_i \leq n$, thereby providing an overall linear bound on the number of submodular function minimizations required.

Updating the bounds on the optimal value During a parametric gradient search, suppose we minimize $f - x^*|_{H_i} - \bar{x}_\delta|_{E\setminus H_i}$ and obtain H_s as the maximal minimizer. We know from Lemma 3.3 that the elements in $E \setminus H_s$ must have a partial derivative greater than δ in the optimal solution, and elements in H_s can have the corresponding partial derivative at most δ. Thus, in each iteration of the parametric gradient search, the lower bound l_e, and the upper bound u_e, on each $e \in H$ can be updated as follows:

$$l_e^{new} = \max\{l_e^{old}, \bar{x}_\delta(e)\}, \text{ for } e \in E \setminus H_s,$$
$$u_e^{new} = \min\{u_e^{old}, \bar{x}_\delta(e)\}, \text{ for } e \in H_s.$$

3.3.3 Running time for Inc-Fix

In Sections 3.3.1 and 3.3.2, we discussed two ways of implementing the Inc-Fix algorithm. One way requires $O(n)$ parametric SFM and the other requires $O(n)$ SFMs by amortizing the computation across the iterations of the Inc-Fix algorithm. In Table 3.3, we document the running times for the state-of-the-art strongly-polynomial (parametric) SFM algorithms that can be used to implement these approaches. The algorithms of [Iwata and Orlin, 2009], [Orlin, 2009] and [Fleischer and Iwata, 2003] can be adapted to do parametric SFM using the results of [Nagano, 2007a]. However, it is an open question to show parametric function minimization using the algorithm of [Lee et al., 2015]. As is clear from Table 3.3, the current best running time of the Inc-Fix algorithm is $O(n^5 \log^{O(1)} n + \gamma n^4 \log^2 n)$.

Algorithms for SFM	**Running Time**	**Inc-Fix Time**
[Iwata and Orlin, 2009]	$O(n^6 + \gamma n^5)$	$O(n^7 + \gamma n^6)$
[Lee et al., 2015]	$O(n^4 \log^{O(1)} n + \gamma n^3 \log^2 n)$	$O(n^5 \log^{O(1)} n + \gamma n^4 \log^2 n)$
Algorithms for PSFM		
[Nagano, 2007a]	$O(\gamma(n^5 + kn^3) + n^6)$	$O(\gamma(n^6 + kn^4) + n^7)$

Table 3.3: Running times for the Inc-Fix method using different algorithms for submodular function minimization. In the running time for [Nagano, 2007a], k is the length of the strong map sequence.

One could potentially use faster polynomial or pseudopolynomial SFM algorithms (for e.g. [Chakrabarty et al., 2016]) to do these function minimizations. Recall that we minimize repeatedly submodular functions of the form $f - \bar{x}_\delta$ to compute the maximum feasible increase in the gradient space. Therefore, in order to get a meaningful bound on the running time of Inc-Fix using (pseudo) polynomial SFM algorithms, one would need to bound the size of $f - \bar{x}_\delta$ (or perhaps find another implementation of the Inc-Fix method). Further, note that we also require the computation of maximal minimizers in the Inc-Fix algorithm. One way to compute the maximal minimizer of an integral submodular function f is to minimize instead $f'(S) = f(S) - \epsilon|S|$ for $\epsilon < 1/n$ (resulting in an increase in the size by a factor of n). Then, the unique minimizer of f' is the maximal minimizer for f. Since the running time strongly polynomial SFM algorithms does not dependent on the size of the submodular function, these computations can be done at no additional cost. For combinatorial algorithms that maintain the certificate of optimality as a convex combination of bases in $B_{ext}(f)$ (see Theorem 1 in Chapter 2), one could also use the classical result of [Bixby et al., 1985] to compute the maximal minimizer at an additional cost of $O(n^3\gamma)$ time.

Comparison with Related Work In 1980, Fujishige gave the *monotone* algorithm, to find the minimum norm point, i.e., $\min_{x\in B(f)} \sum_{e\in E} x_e^2/w_e$ over the submodular base polytope $B(f)$ and $w \in \mathbb{R}^E_{>0}$ [Fujishige, 1980]. This algorithm starts with $x^{(0)} = 0$ and iteratively moves proportional to w_N where N is the set of non-fixed elements till it hits a tight constraint. Inc-Fix can be viewed as a generalization of this method.

In 1991, Fujishige and Groenevelt developed a decomposition algorithm, for minimizing separable convex functions $h(\cdot)$ over submodular base polytopes [Groenevelt, 1991] (the exact setting that we consider). This algorithm starts by finding any vector $z \in \mathbb{R}^E_+$ that sets $z(E) = f(E)$ and minimizes $h(z)$. If z is feasible, then z is the minimizer. Otherwise, the problem is decomposed into two subproblems: one with the submodular function restricted to the maximally violated constraint S for z and the other with the contracted submodular function over $E \setminus S$. This process repeats recursively, until each subproblem returns the optimal solution. There has been a large volume of work since 1991 to speed up the decomposition algorithm and show rationality of its solutions for certain convex functions (for e.g. see [Nagano and Aihara, 2012]). The current best known running times are $O(n)$ submodular function minimizations (along with maximal minimizer computation) or a single parametric submodular function minimization [Nagano, 2007b]. Thus, Inc-Fix has the same worst-case running time as the decomposition algorithm since there exist faster methods for submodular function minimization (compared to parametric SFM).

The above mentioned algorithms are exact, under infinite precision arithmetic. However, general convex optimizations can also be for approximately minimizing separable convex functions (in fact, even non-separable convex functions) over submodular (base) polytopes. One such method is another first-order constrained optimization method, Frank-Wolfe [Frank and Wolfe, 1956], that does not require the computation of projections. Frank-Wolfe is an iterative procedure that considers, in each step, a linear approximation of the convex function and moves towards the minimizer by a small step. We review the vanilla Frank-Wolfe method in Section 2.2.2 and provide useful references for its variants. Each step of the Frank-Wolfe method only requires a linear optimization, which is quite inexpensive for submodular polytopes (only $n\gamma + n \log n$ time, where γ is the time for a single function evaluation) thus making Frank-Wolfe method an attractive way to tradeoff running time for accuracy when Inc-Fix requires the full machinery of the oracle-model submodular function minimization. The rate of convergence of Frank-Wolfe however, depends on the curvature[9] C_h of $h(\cdot)$, and $O(C_h/\epsilon)$ iterations are required to achieve an optimality gap of $O(\epsilon)$. Moreover, as we will see in the next section, for cardinality-based submodular polytopes, we can obtain running times that are competitive with the Frank-Wolfe method while computing exact solutions.

3.4 Cardinality-based submodular functions

A submodular function is cardinality-based if $f(S) = g(|S|)$ $(S \subseteq E)$ for some concave function $g : \mathbb{N} \to \mathbb{R}$ (e.g., corresponding to the simplex, k-sets, permutations, in Table 2.1). We use the notation $P(g)$ and $B(g)$ to refer to the cardinality-based submodular polytope and the base polytope corresponding to the concave function g.

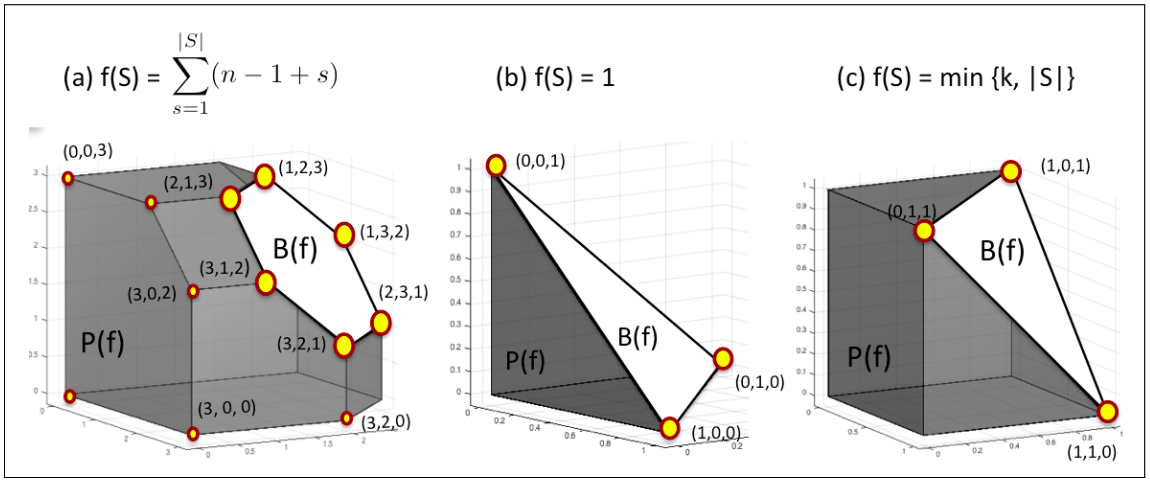


Figure 3-3: Different choices of concave functions $g(\cdot)$, such that $f(S) = g(|S|)$, result in different cardinality-based polytopes; (a) permutations if $f(S) = \sum_{s=1}^{|} S|(n-1+s)$, (b) probability simplex if $f(S) = 1$, (c) k-subsets if $f(S) = \min\{k, |S|\}$.

Define $g'(i) = \min_{j \geq i} g(j)$ and note that $g'(\cdot)$ is non-decreasing. It is easy to check that $P(g) = \{x \in \mathbb{R}^E_+ \mid x(S) \leq g(|S|) \; \forall S \subseteq E\} = P(g')$. Thus, without loss of generality, we

[9] Curvature $C_h := \sup_{x,s \in \mathcal{D}, \gamma \in [0,1], y = x + \gamma(s-x)} \frac{2}{\gamma^2}(h(y) - h(x) - \langle y - x, \nabla h(x) \rangle)$ where $\mathcal{D}$ is the domain of the convex function $h(\cdot)$ (the convex function to be minimized). Refer to Section 2.2.2 for more details.

can assume that the concave function $g(\cdot)$ itself is non-decreasing. Further, we assume that $g(0) \geq 0$ so that $P(g)$ (as well as the base polytope $B(g)$) is non-empty. In this section, we will present an efficient adaptation of the INC-FIX algorithm to compute projections onto cardinality-based submodular base polytopes $B(g)$ under divergences arising from uniformly separable mirror maps, i.e.

$$(\text{P1})' : \min_{x \in B(g)} \sum_{e \in E} w(x_e) - \sum_{e \in E} w(y_e) - \sum_{e \in E} w'(y_e)(x_e - y_e). \tag{3.14}$$

Recall that we defined $\bar{x}_\delta(e) = (h'_e)^{-1}(\delta)$ that is the point corresponding to a gradient value of δ, and in the case of uniform divergences $\bar{x}_\delta(e) = (w')^{-1}(\delta + w'(y_e))$. We first show that projection of any constant vector $c\chi(E)$ has a closed form expression, for any choice of the cardinality-based submodular function f and any uniformly separable mirror map.

Lemma 3.4. *Consider a cardinality-based submodular function $f : f(S) = g(|S|)$ $(S \subseteq E)$ for some concave function g with $g(0) \geq 0$. Then, the projection of a constant vector $y = c\chi(E) \in \mathbb{R}^E$ onto $B(g)$ under the Bregman divergence of any uniform separable mirror map $\omega(x) = \sum_{e \in E} w(x(e))$ is $\frac{g(|E|)}{|E|}\chi(E)$.*

Proof. Consider $\delta^* = \max \delta : \bar{x}_\delta \in P(g)$. By definition of δ^*, we get $T(\bar{x}_{\delta^*}) \neq \emptyset$. This in turn implies that E is tight on $\bar{x}_{\delta^*}$ since the function is cardinality-based and $\bar{x}_{\delta^*}(e) = (w')^{-1}(\delta^* + c)$ for all $e \in E$. Since $B(g) \neq \emptyset$, we have $\bar{x}_{\delta^*}(E) = g(|E|) \Rightarrow \bar{x}_{\delta^*}(e) = g(|E|)/|E|$ for all $e \in E$. Finally, using Theorem 8, we have that $\bar{x}_{\delta^*} = \text{argmin}_{z \in B(g)} D_\omega(z, y)$. $\square$

An alternate proof of the above lemma is the following: observe first that the minimizer x^* is unique since the objective function is strictly convex. Next, since the objective function is symmetric, all x^*_e are equal (since any permutation of them would also give an optimum solution). The only point in $B(g)$ with all components equal is given by $x_e = g(|E|)/|E|$ (since $x(E) = g(|E|)$). In other words, given a cardinality-based submodular polytope, the projection of the constant vector $c\chi(E)$ with respect to the Bregman divergence of any uniform mirror map is the same. However, in general, the projected vectors can be very different depending on the choice of the mirror map. To give an example, we constructed eight different concave functions $g(\cdot)$ by sampling $k \in [0,1]^{100}$ from different probability

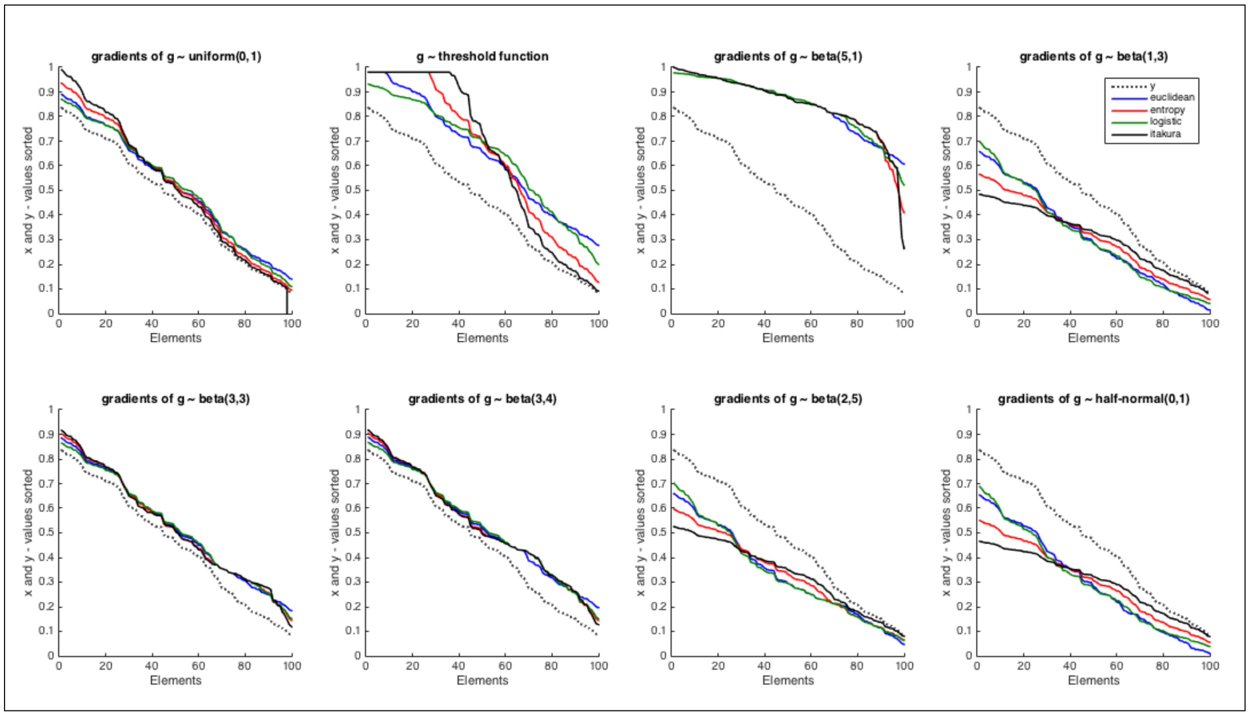


Figure 3-4: Squared Euclidean, entropic, logistic and Itakura-Saito Bregman projections of the (dotted) vector y onto the cardinality-based submodular polytopes given by different randomly selected concave functions $g(\cdot)$. We refer to the corresponding projected vector in each case by x. The threshold function is of the form $g(i) = \min\{\alpha i, \tau\}$ constructed by selecting a slope α and a threshold τ both uniformly at random.

distributions, sorting them as $k_1 \geq k_2 \geq \ldots k_{100}$, and setting $g(0) = 0, g(s) = \sum_{i=1}^{s} k_s$. We also sampled a vector $y \in [0,1]^{100}$ from the uniform distribution on [0,1], and sorted the elements of y to be in decreasing order (for illustration purposes). We then computed projections (denoted by $x \in \mathbb{R}^{100}$) of the sorted y vector onto cardinality-based polytopes corresponding to each of the concave functions $g(\cdot)$. Figure 3-4 illustrates the values of the projected elements (ordered according to the sorted y vector) corresponding to different divergences.

3.4.1 Card-Fix algorithm

We next discuss a modification of the Inc-Fix algorithm to solve problem P1′ (3.14). Since it relies on properties of cardinality-based polytopes, we call the method Card-Fix. Let $\omega(x) = \sum_{e \in E} w(x(e))$ be a mirror map where $w : \mathcal{D}_w \rightarrow \mathbb{R}$ is strongly convex. We want to minimize the function $h(x) := D_\omega(x, y)$ over $x \in B(g)$ for some $y \in \mathbb{R}^E$ with $y(e) \in \mathcal{D}_w$.

We can simplify the conditions on the convex function in the INC-FIX algorithm to be the following: (i) $[0, g(1)] \subseteq \overline{\mathcal{D}_w}$ (i.e. $P(g)$ must be contained in the closure of the domain of h), (ii) $g(n)/n \in \mathcal{D}_w$ (i.e., $B(g)$ must have a non-empty intersection with the domain of h), and (iii) $w'(\mathcal{D}_w) = \mathbb{R}$ (i.e., image of the gradients of h must be $\mathbb{R}$).

Similar to the INC-FIX algorithm, we start CARD-FIX with $x^{(0)} = 0$ or $x^{(0)} = (\nabla\omega)^{-1}(\delta + \omega(y)) \in P(g)$. Note that if $0 \in \mathcal{D}_w$, then a valid starting point is $x^{(0)} = 0$. Otherwise, since $w'(\mathcal{D}_w) = \mathbb{R}$, we know that $\lim_{\delta \to -\infty}(w')^{-1}(\delta + w'(y(e))) = 0$. Therefore, there always exists $\delta < 0$ such that $x^{(0)} = (\nabla\omega)^{-1}(\delta + \omega(y)) \in P(g)$.

We sort the elements in E as $e_1, e_2, \ldots, e_n$ such that $y(e_s) > y(e_t)$ whenever $s < t$ (breaking ties arbitrarily). The key observation that helps in speeding up the INC-FIX algorithm is that whenever the elements are increased to a gradient value in the INC-FIX algorithm to obtain an iterate $x^{(i)}$, $x^{(i)}(e_s) \geq x^{(i)}(e_t)$ for $s < t$. Since the polytope is cardinality-based, an efficient way to check for feasibility in $P(g)$ is to check if the sum of the highest k elements is less than $g(k)$ for each $1 \leq k \leq n$. We show that each gradient increase allows the elements to maintain the decreasing order in their values, and therefore we only need to check for at most n constraints for feasibility without requiring to sort the elements after each increase in the gradient space. This helps achieve the speed up in the running time to $O(n(\log n + n))$. We give the complete description of the CARD-FIX algorithm in Algorithm 7. The maximal tight set is simply a prefix of the ordered elements $(e_1, \ldots, e_t)$ as $x^{(i)}(e_u) \geq x^{(i)}(e_v)$ for $u < v$ and is maintained using the index t (Lemma 3.6).

Note that for an arbitrary k, one can compute ϵ_k in step (8) of Algorithm 7 by solving a univariate (often non-linear) equation. We discuss the form of these non-linear equations for the previously mentioned set of divergences. For squared Euclidean distance, $x_\delta(e) = \delta + y_e$, thus ϵ_k can be computed using a closed-form expression:

$$\sum_{j=t+1}^{k} (\epsilon_k + y(e_j)) = g(k) - g(t) \Rightarrow \epsilon_k = \frac{g(k) - g(t) - \sum_{j=t+1}^{k} y(e_j)}{k - t}.$$

Algorithm 7: Card-Fix

```
input : $f(S) = g(|S|)$ for $S \subseteq E$, $g$ non-decreasing and concave, $g(0) \geq 0$, $\omega(x) = \sum_e w(x(e))$,
        $y \in \mathbb{R}^E$, and $x^{(0)} \in P(g)$
output: $x^* = \operatorname{argmin}_{z \in B(g)} D_\omega(z, y)$
3  Sort elements in $E$ as $\{e_1, e_2, \ldots, e_n\}$ such that $y(e_s) > y(e_t)$ whenever $s < t$.
4  $i = 0$, $t = 0$
5  repeat
6      $i \leftarrow i + 1$
7      for $k \in \{t+1, \ldots, n\}$:
8          Set $\epsilon_k : \sum_{j=t+1}^{k} \bar{x}_{\epsilon_k}(e_j) = g(k) - g(t)$
9      $\epsilon^{(i)} = \min_{t+1 \leq k \leq n} \epsilon_k$;                      // maximal feasible increase
10     $x^{(i)}(e) \leftarrow \begin{cases} x^{(i-1)}(e_j) & \text{for } j \leq t, \\ \max\{\bar{x}_{\epsilon^{(i)}}(e_j), x^{(i-1)}(e_j)\} & \text{for } j > t \end{cases}$
11     $t = \max\{k \mid t+1 \leq k \leq n, \epsilon_k = \epsilon^{(i)}\}$ ;          // bookkeeping maximal tight set
   until $t = n$;
13 Return $x^* = x^{(i)}$.
```

For minimizing KL-divergence, we have $x_\delta(e) = e^\delta y_e$, thus step (8) reduces to -

$$\sum_{j=t+1}^{k} e^{\epsilon_k} y(e_j) = g(k) - g(t) \Rightarrow e^{\epsilon_k} = \frac{g(k) - g(t)}{\sum_{j=t+1}^{k} y(e_j)}.$$

However, this computation becomes more involved for Itakura-Saito divergence and the logistic loss. For the former,

$$\sum_{j=t+1}^{k} -1/(\epsilon_k - 1/y(e_j)) = g(k) - g(t) \tag{3.15}$$

and for logistic loss, we have

$$\sum_{j=t+1}^{k} \frac{e^{\epsilon_k} y_{e_j}}{1 - y_{e_j} + e^{\epsilon_k} y_{e_j}} = g(k) - g(t). \tag{3.16}$$

Note that for both (3.15) and (3.16), the expressions on the left hand side are strictly increasing functions of ϵ_k. Thus, ϵ_k can be found by using methods like binary search or Newton's algorithm. We assume a constant time oracle for solving these equations (as in the more general setting of Inc-Fix).

We now give the main theorem about the correctness of the Card-Fix algorithm.

Theorem 10. *Consider a cardinality-based submodular function corresponding to a non-*

decreasing concave function g with $g(0) \geq 0$. Let $\omega(x) = \sum_{e \in E} w(x(e))$ be a mirror map where $w : \mathcal{D}_w \to \mathbb{R}$ is strongly convex. We assume that (i) $[0, g(1)] \subseteq \overline{\mathcal{D}_w}$, (ii) $g(n)/n \in \mathcal{D}_w$ and (iii) $w'(\mathcal{D}_w) = \mathbb{R}$. Then, given $y \in \mathbb{R}^E$ with $y(e) \in \mathcal{D}_w$ for all $e \in E$, the output of Card-Fix *algorithm is $x^* = \operatorname{argmin}_{z \in B(g)} D_\omega(z, y)$.*

The Card-Fix algorithm can be interpreted as a simplified version of the Inc-Fix algorithm, that exploits the properties of cardinality-based submodular functions to compute parametric gradient searches efficiently. However, instead of relying on the correctness of Inc-Fix, we give an independent proof of correctness. Before we prove Theorem 10, we discuss some useful lemmas in order to simplify the exposition of the main proof. Recall that since we are minimizing $h(x) = D_\omega(x, y)$, we have that $h'_e(x_e) = w'(x_e) - w'(y_e)$.

Lemma 3.5. *For $e_s, e_t \in E$, if $y(e_s) > y(e_t)$ then $\bar{x}_\delta(e_s) > \bar{x}_\delta(e_t)$ for any δ. Further, if $x(e_s) > x(e_t)$ and $h'_s(x_{e_s}) = h'_t(x_{e_t})$ then $\bar{x}_\delta(e_s) > \bar{x}_\delta(e_t)$ for arbitrary δ.*

Proof. If $y(e_s) > y(e_t)$, then $\bar{x}_\delta(e_s) = (w')^{-1}(\delta + y(e_s)) > (w')^{-1}(\delta + y(e_t)) = \bar{x}_\delta(e_t)$, as w' is the gradient of a strongly-convex function $w(\cdot)$ and hence strictly increasing. Further, $x(e_s) > x(e_t)$ and $h'_s(x_{e_s}) = h'_t(x_{e_t})$ imply that $y(e_s) > y(e_t)$ and hence the lemma follows. □

At the beginning of the algorithm, we fix order the elements $\{e_1, e_2, \ldots, e_n\}$ such that whenever $y(e_i) > y(e_j)$ we have $i < j$ (breaking ties arbitrarily). For any vector $x \in \mathbb{R}^E$, we say that it is satisfies the order on E if $x(e_i) \geq x(e_j)$ whenever $i < j$.

Lemma 3.6. *Consider $x \in P(g)$. Let x satisfy the order on $E = \{e_1, e_2 \ldots e_n\}$. Then, the maximal tight set $T(x)$ of elements of x is a prefix of the ordering, i.e. $T(x) = \{e_1, \ldots, e_{|T(x)|}\}$. Further, $x \in P(g)$ if and only if $\sum_{i=1}^{k} x(e_i) \leq g(k)$ for all $k \in \{1, \ldots, n\}$.*

This follows from the cardinality-based constraints defining $P(g)$. We next show that steps (7)-(10) of the Card-Fix algorithm generate a feasible point in the submodular polytope $P(g)$.

Lemma 3.7. *Consider $x \in P(g)$ for $g(\cdot)$ concave and non-decreasing, $g(0) \geq 0$. Let $T(x)$ be the maximal set of tight elements of x and $|T(x)| = t$. Further, let x satisfy the order on E. For each $k = t+1, \ldots, n$, let ϵ_k be the gradient value at which the* *cardinality constraint for size k becomes tight, i.e.* $\sum_{i=t+1}^{k} \bar{x}_{\epsilon_k}(e_i) = g(k) - g(t)$. *Then,* $x' = (x|_{T(x)}, \max\{0, \bar{x}_\epsilon|_{E \setminus T(x)}\}) \in P(g)$, *where* $\epsilon = \min_{t+1 \leq k \leq n} \epsilon_k$.

Proof. We first point out that, by definition of x' and Lemma 3.5,

$$x' = (x(e_1), \ldots, x(e_t), \bar{x}_\epsilon(e_{t+1}), \ldots, \bar{x}_\epsilon(e_s), 0, \ldots, 0) \in \mathbb{R}^n_+$$

for some $t \leq s \leq n$ such that $\bar{x}_\epsilon(e_s) > 0, \bar{x}_\epsilon(e_{s+1}) \leq 0$. Further, by assumptions on x, we have $x'(e_i) \leq x'(e_j)$ for $1 \leq i \leq j \leq t$ and using Lemma 3.5 on x', we get $x'(e_i) \geq x'(e_j)$ for $t+1 \leq i \leq j \leq n$.

Since $x \in P(g)$, we know that $\sum_{j=1}^{t-1} x(e_j) = \sum_{j=1}^{t-1} x'(e_j) \leq g(t-1)$ which in turn implies that $x(e_t) = x'(e_t) = g(t) - \sum_{j=1}^{t-1} x'(e_j) \geq g(t) - g(t-1)$. Suppose first that $x'(e_{t+1}) = x_\epsilon(e_{t+1})$. Then,

$$\begin{aligned} \bar{x}_\epsilon(e_{t+1}) = x'(e_{t+1}) &\leq g(t+1) - g(t) \ldots \text{ since } \epsilon \leq \epsilon_{t+1} \\ &\leq g(t) - g(t-1) \ldots \text{ by concavity of } g \\ &\leq x'(e_t). \end{aligned}$$

Thus, x' also satisfies the order on E, i.e. $x'(e_i) \geq x'(e_j)$ whenever $i < j$. This holds trivially if $x'(e_{t+1}) = 0$. We can check if $\sum_{i=1}^{k} x'(e_i) \leq g(i)$ for all $i \geq t$ to check for feasibility, as stated in Lemma 3.6. Suppose that $x \notin P(g)$, we will show that we get a contradiction. Consider the first index $j > t$ such that $\sum_{i=1}^{j} x'(e_i) > g(j)$. Since $g(\cdot)$ is non-decreasing, we know that $x'(e_j) > 0$ otherwise $\sum_{i=1}^{j} x'(e_i) = \sum_{i=1}^{j-1} x'(e_i) \leq g(j-1) \leq g(j)$. However, this is a contraction since $\sum_{i=1}^{j} \bar{x}_{\epsilon_j}(e_i) = g(j)$ and $\epsilon \leq \epsilon_j$ implies $\sum_{i=1}^{j} \bar{x}_\epsilon(e_i) = \sum_{i=1}^{j} x'(e_i) \leq g(j)$. □

We are now ready to prove Theorem 10. We will show that the output x^* of Algorithm 7 satisfies first-order optimality conditions of Theorem 8, i.e. suppose $F_1, F_2, \ldots, F_k$ is a partition of the ground set E such that $w'(x^*_e) - w'(y_e) = c_i$ for all $e \in F_i$ and $c_i < c_j$ for $i < j$. Then, x^* lies in the face H_{opt} of $B(g)$ given by $H_{opt} := \{z \in B(g) | \ z(F_1 \cup \ldots \cup F_i) = g(|F_1 \cup \ldots \cup F_i|) \ \forall \ 1 \leq i \leq k\}$.

Proof. (Proof of Theorem 10.) Let elements in E be ordered as $\{e_1, e_2, \ldots\}$ such that $y(e_s) > y(e_t)$ for $s < t$, breaking ties arbitrarily. We start the algorithm with $x^{(0)} = 0 \in P(f)$ or $x_e^{(0)} = (w')^{-1}(c + w'(y_e))$ for all e, for some $c \in \mathbb{R}$ such that $x^{(0)} \in P(f)$. In either case, it is easy to check that $x^{(0)}$ satisfies the order on E.

For $i \geq 1$, we show that given $x = x^{(i-1)} \in P(f)$ such that x satisfies the order on E, its maximal tight set $T(x)$ is of size t, and $w'(x_e) - w'(y_e) = \epsilon^{(i-1)}$ or $x(e) = 0$ for $e \in T(x)$, the algorithm computes

$$x^{(i)} = x' = (x|_{T(x)}, \max\{\bar{x}_{\epsilon^{(i)}}|_{E \setminus T(x)}, x|_{E \setminus T(x)}\})$$

where $\epsilon^{(i)} = \min_{t+1 \leq k \leq n} \epsilon_k$, and ϵ_k is such that $\sum_{i=t+1}^{k} \bar{x}_{\epsilon_k}(e_i) = g(k) - g(t)$ for $k = t+1, \ldots, n$. We show by induction that the following hold for $x^{(i)}$ $(= x')$ and $\epsilon^{(i)}$:

(i) $x' = x^{(i)} \in P(f)$ and satisfies the order on E,

(ii) $T(x') \supset T(x)$ and $\epsilon^{(i)} > \epsilon^{(i-1)}$,

(iii) For $e \in E \setminus T(x')$, $w'(x'_e) - w'(y_e) = \epsilon^{(i)}$ or $x'(e) = 0$.

Proof for (i). Suppose $\epsilon^{(i)} \leq \epsilon^{(i-1)}$, then $x' = x \in P(f)$ and x' satisfies the order on E. Otherwise suppose $\epsilon^{(i)} > \epsilon^{(i-1)}$. Then, using Lemma 3.7 and the assumption on x that $w'(x_e) - w'(y_e) = \epsilon^{(i-1)}$ or $x(e) = 0$ for $e \in T(x)$, we get $x' \in P(f)$ and x' satisfies the order on E.

Proof for (ii). Consider $k = \arg\min_{t+1 \leq j \leq n} \epsilon_k$. We know that $\sum_{i=t+1}^{k} \bar{x}_\epsilon(e_i) = g(k) - g(t)$. However, $x' \geq (x|_{T(x)}, \bar{x}_\epsilon|_{E \setminus T(x)})$. Thus, $\{e_1, \ldots, e_k\} \in T(x')$. This also implies that $\epsilon^{(i)} > \epsilon^{(i-1)}$ otherwise $x = x'$.

Proof for (iii). For $e \in E \setminus T(x')$, $x'_e = \max\{0, \bar{x}_{\epsilon^{(i)}}(e)\}$. This implies $x'_e = 0$ or $w'(x'_e) - w'(y_e) = \epsilon^{(i)}$.

Note that whenever $T(x^{(i)})$ contains 0 element, the algorithm stops as $T(x^{(i)})$ must be E. Let us partition the ground set according to the gradient value of elements. Let $F_1, F_2, \ldots, F_k$ is a partition of the ground set E such that $w'(x_e^*) - w'(y_e) = c_i$ for all $e \in F_i$ and $c_i < c_j$ for $i < j$. We claim that $F_i = T(x^{(i)}) \setminus T(x^{(i-1)})$, and $w'(x_e^*) - w'(y_e) = \epsilon^{(i)}$ for $e \in F_i$. Moreover,

$x^*(F_1, \ldots, F_i) = x^*(T(x^{(i)})) = f(T(x^{(i)}))$ for each i, which using Theorem 8 proves the main claim.

$\square$

Running Time Algorithm 7 starts with sorted elements $\{e_1, e_2, \ldots, e_n\}$ such that $y(e_s) > y(e_t)$ for $s < t$. The number of iterations in the algorithm is at most n, since in each iteration the size of the maximal tight set increases. Each iteration requires the solution of at most n equations in a single variable ϵ_k, for which we assume an oracle access with constant query time (recall that this is just a fraction in the case of squared Euclidean distance and KL-divergence). The worst case running time of Card-Fix algorithm is $O(n \log n + n^2)$. For cardinality-based submodular functions $f(\cdot)$, based on a concave function $g(\cdot)$ we in fact need to check for the cardinality constraints only at the unique values of g. Consider $U = \{1, \ldots, j, n\}$ where j is the minimum value such that $g(j) = g(n)$. Then, steps (7-9) can be simplified to be:

```
(7) for $k \in \{t+1, \ldots, n\} \cap U$ :
(8)     set $\epsilon_k : \sum_{j=t+1}^{k} \bar{x}_{\epsilon_k}(e_j) = g(k) - g(t)$
(9) $\epsilon^{(i)} = \min_{t+1 \le k \le n, k \in U} \epsilon_k.$
```

This modification reduces the worst-case running time $O(n(\log n + d))$ where $d = |U|$. This subsumes some recent results of Yasutake et al. (for minimizing Euclidean and KL-divergence on the permutahedron) [Yasutake et al., 2011], Suehiro et al. (for minimizing Euclidean and KL-divergence on to cardinality-based polytopes) [Suehiro et al., 2012] and Krichene et al. (for minimizing ϕ-divergences onto the simplex). Our work, however, applies to the divergence generated from any uniformly separable mirror map and any cardinality-based submodular function.

Chapter 4

Parametric Line Search

"A sequence works in a way a collection never can." - George Murray.

In this chapter, we would like to solve the fundamental problem of a parametric line search in an extended submodular polytope $EP(f) = \{x \in \mathbb{R}^E \mid x(S) \leq f(S) \ \forall S \subseteq E\}$. Given $x_0 \in EP(f)$ (this condition can be verified by performing a single submodular function minimization) and $a \in \mathbb{R}^n$, we would like to find the largest δ such that $x_0 + \delta a \in EP(f)$. The only assumption we make on the submodular function $f(\cdot)$ in this chapter is that $f(\emptyset) \geq 0$ (otherwise $EP(f)$ will be empty). By considering the submodular function f' taking the value $f'(S) = f(S) - x_0(S)$ for any set S, we can equivalently find largest δ such that $\delta a \in EP(f')$. Since $x_0 \in EP(f)$ we know that $0 \in EP(f')$ and thus f' is nonnegative. Thus, without loss of generality, we consider the problem

$$\delta^* = \max\left\{\delta : \min_{S \subseteq E} f(S) - \delta a(S) \geq 0\right\}, \tag{4.1}$$

for nonnegative submodular functions f. Geometrically, the problem of finding δ^* can also be interpreted as: as we go along the line segment $\ell(\delta) = x_0 + \delta a$ (or just δa if we assume $x_0 = 0$), when do we exit the extended submodular polyhedron $EP(f)$?

Line searches arise as subproblems in many algorithmic applications. For example, in the previous chapter, we noted that the INC-FIX algorithm requires to solve the line search problem when computing projections under the squared Euclidean distance and KL-divergence

(Section 3.2.1). For the algorithmic version of Carathéodory's theorem[1] (over any polytope), one typically performs a line search from a vertex of the face being considered in a direction within the same face. This is, for example, also the case for variants of the Frank-Wolfe algorithm (see for instance [Freund et al., 2015]). Line searches over extended submodular polyhedra are also intimately related to minimum ratio problems that seek to minimize $\min_S \frac{f(S)}{g(S)}$ for some submodular function $f(\cdot)$ and a linear function $g(\cdot)$ [Cunningham, 1985b].

Since $x_0 = 0 \in EP(f)$ we know that $\delta^* \geq 0$ and that the minimum over S could be taken only over the sets S with $a(S) > 0$, although we will not be using this fact. To make this problem nontrivial, we assume that there exists some i with $a_i > 0$. A natural way to solve the line search problem is to use a cutting plane approach. Start with any upper bound $\delta_1 \geq \delta^*$ and define the point $x^{(1)} = \delta_1 a$. One can then generate a most violated inequality for $x^{(1)}$, where most violated means the one minimizing $f(S) - \delta_1 a(S)$ over all sets S. The hyperplane corresponding to a minimizing set S_1 intersects the line in $x^{(2)} = \delta_2 a$. Proceeding analogously, we obtain a sequence of points and eventually will reach the optimum δ.

This cutting-plane approach is equivalent to Dinkelbach's method or the discrete Newton's algorithm for solving (4.1). Let δ_1 be large enough so that $\delta_1 a \notin EP(f)$. For example we could set $\delta_1 = \min_{e \in E, a(\{e\})>0} f(\{e\})/a_e$. At iteration $i \geq 1$ of Newton's algorithm, we consider the submodular function $k_i(S) = f(S) - \delta_i a(S)$, and compute

$$h_i = \min_S k_i(S),$$

and define S_i to be *any* minimizer of $k_i(S)$. Now, let $f_i = f(S_i)$ and $g_i = a(S_i)$. As long as $h_i < 0$, we proceed and set

$$\delta_{i+1} = \frac{f_i}{g_i}.$$

As soon as $h_i = 0$, Newton's algorithm terminates and we have that $\delta^* = \delta_i$. We give the full description of the discrete Newton's algorithm in Algorithm 8.

When $a \geq 0$ (as is the case in the Inc-Fix algorithm), it is known that Newton's

[1]The Carathéodory's theorem states that given any point in a polytope $P \subseteq \mathbb{R}^n$, it can be expressed as a convex combination of at most $n+1$ vertices of P.

Algorithm 8: Discrete Newton's algorithm

```
input  : submodular $f : 2^E \to \mathbb{R}$, $f$ nonnegative, $a \in \mathbb{R}^n$
output: $\delta^* = \max\{\delta : \min_S f(S) - \delta a(S) \geq 0\}$
$i = 0, \delta_1 = \min_{e \in E, a_e > 0} f(\{e\})/a_e$;
repeat
    $i = i + 1$;
    $h_i = \min_{S \subseteq E} f(S) - \delta_i a(S)$;
    $S_i \in \arg\min_{S \subseteq E} f(S) - \delta_i a(S)$;
    $\delta_{i+1} = \frac{f(S_i)}{a(S_i)}$;
until $h_i = 0$;
Return $\delta^* = \delta_i$.
```

algorithm terminates in at most n iterations (for e.g. [Topkis, 1978]). Even more, the function $g(\delta) := \min_S f(S) - \delta a(S)$ is a concave, piecewise affine function with at most n breakpoints (and $n+1$ affine segments) since for any set $\{\delta_i\}_{i \in I}$ of δ values, the submodular functions $f(S) - \delta_i a(S)$ for $i \in I$ form a sequence of strong quotients (ordered by the δ_i's), and therefore the minimizers form a chain of sets. Refer to Section 2.2.1 for definitions of strong quotients and details.

When a is arbitrary (not necessarily nonnegative), little is known about the number of iterations of the discrete Newton's algorithm. The number of iterations can easily be bounded by the number of possible distinct positive values of $a(S)$, but this is usually very weak (unless, for example, the support of a is small as is the case in the calculation of exchange capacities[2]). A weakly polynomial bound involving the sizes of the submodular function values is easy to obtain (by doing a binary search on $[0, \delta_1]$ and checking for feasibility), but no strongly polynomial bound was known as mentioned as an open question in [Nagano, 2007b], [Iwata, 2008]. In this chapter, we show that the number of iterations is quadratic. This is the first strongly polynomial bound in the case of an arbitrary a.

Theorem 11. *For any submodular function $f : 2^{[n]} \to \mathbb{R}_+$ and an arbitrary direction a, the discrete Newton's algorithm takes at most $n^2 + O(n \log^2(n))$ iterations.*

Previously, the only strongly polynomial algorithm to solve the line search problem in the case of an arbitrary $a \in \mathbb{R}^n$ was an algorithm of Nagano et al. [Nagano, 2007b] relying on Megiddo's parametric search framework. This requires $\tilde{O}(n^8)$ submodular function

[2]For $x_0 \in EP(f)$, the exchange capacity of an element e with respect to $e' \in E$ ($e' \neq e$) is the maximum δ: $x_0 + \delta(\chi(e) - \chi(e')) \in EP(f)$.

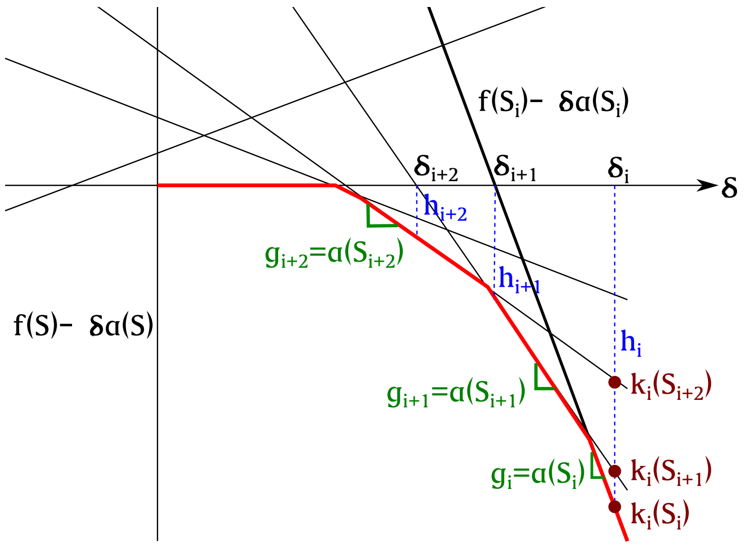


Figure 4-1: Illustration of Newton's iterations and notation in Lemma 4.1.

minimizations, where $\tilde{O}(n^8)$ corresponds to the current best running time known for *fully combinatorial* submodular function minimization [Iwata and Orlin, 2009]. On the other hand, our main result in Theorem 11 shows that the discrete Newton's algorithm takes $O(n^2)$ iterations, i.e. $O(n^2)$ submodular function minimizations, and we can use any submodular function minimization algorithm. Each submodular function minimization can be computed, for example, in $\tilde{O}(n^4 + \gamma n^3)$ time using a result of [Lee et al., 2015], where γ is the time for an evaluation of the submodular function.

Radzik [Radzik, 1998] provides an analysis of the discrete Newton's algorithm for the related problem of $\max \delta : \min_{S \in \mathcal{S}} b(S) - \delta a(S) \geq 0$ where both a and b are modular functions and $\mathcal{S}$ is an arbitrary collection of sets. He shows that the number of iterations of the discrete Newton's algorithm is at most $O(n^2 \log^2(n))$. Our analysis does not handle an arbitrary collection of sets, but generalizes his setting as it applies to the more general case of submodular functions f. Note that considering submodular functions (as opposed to modular functions) makes the problem considerably harder since the number of input parameters for modular functions is only $2n$, whereas in the case of submodular functions the input is exponential (we assume oracle access for function evaluation).

Apart from the main result of bounding the number of iterations of the discrete Newton's algorithm for solving $\max \delta : \min_S f(S) - \delta a(S) \geq 0$ in Section 4.2, we prove results on ring families and geometrically increasing sequences of sets, which may be of independent interest. As part of the proof of Theorem 11, we first show a tight (quadratic) bound on the length of a sequence $T_1, \cdots, T_k$ of sets such that no set in the sequence belongs to the smallest ring family generated by the previous sets (Section 4.1). Further, one of the key ideas in the proof of Theorem 11 is to consider a sequence of sets (each set corresponds to an iteration in the discrete Newton's algorithm) such that the value of a submodular function on these sets increases geometrically (to be precise, by a factor of 4). We show a quadratic bound on the length of such sequences for any submodular function and construct two (related) examples to show that this bound is tight, in Section 4.3. Interestingly, one of these examples is a construction of intervals and the other example is a weighted directed graph where the cut function already gives such a sequence of sets.

4.1 Ring families

A *ring family* $\mathcal{R} \subset 2^V$ is a family of sets closed under taking unions and intersections[3]. From Birkhoff's representation theorem, we can associate to a ring family a directed graph $D = (V, E)$ in the following way. Let $A = \bigcap_{R \in \mathcal{R}} R$ and $B = \bigcup_{R \in \mathcal{R}} R$. Let $E = \{(i, j) \mid R \in \mathcal{R}, i \in R \Rightarrow j \in R\}$. Then for any $R \in \mathcal{R}$, we have that (i) $A \subseteq R$, (ii) $R \subseteq B$ and (iii) $\delta^+(R) = \{(i, j) \in E \mid i \in R, j \notin R\} = \emptyset$. But, conversely, any set R satisfying (i), (ii) and (iii) must be in $\mathcal{R}$. Indeed, for any $i \neq j$ with $(i, j) \notin E$, there must be a set $U_{ij} \in \mathcal{R}$ with $i \in U_{ij}$ and $j \notin U_{ij}$. To show that a set R satisfying (i), (ii) and (iii) is in $\mathcal{R}$, it suffices to observe that

$$R = \bigcup_{i \in R} \bigcap_{j \notin R} U_{ij}, \tag{4.2}$$

and therefore R belongs to the ring family.

Given a collection of sets $\mathcal{T} \subseteq 2^V$, we define $\mathcal{R}(\mathcal{T})$ to be the smallest ring family containing $\mathcal{T}$. The directed graph representation of this ring family can be obtained by defining A, B and E directly from $\mathcal{T}$ rather than from the larger $\mathcal{R}(\mathcal{T})$, i.e. $A = \bigcap_{R \in \mathcal{T}} R = \bigcap_{R \in \mathcal{R}(\mathcal{T})} R$,

[3]We depart in this section from the notation used otherwise in this thesis, and refer to the ground set of elements as V instead of E. Here, we call $V = \{1, \ldots, n\}$ and reserve E for encoding pairwise relation between the elements of the ground set.

$B = \bigcup_{R\in\mathcal{T}} R = \bigcup_{R\in\mathcal{R}(\mathcal{T})} R$, and $E = \{(i,j) \mid R \in \mathcal{T}, i \in R \Rightarrow j \in R\}$. Further, in the expression (4.2) of any set $R \in \mathcal{R}(\mathcal{T})$, we can use sets $U_{ij} \in \mathcal{T}$.

Given a sequence of subsets $T_1, \cdots, T_k$ of V, define $\mathcal{L}_i := \mathcal{R}(\{T_1, \cdots, T_i\})$ for $1 \le i \le k$. Assume that for each $i > 1$, we have that $T_i \notin \mathcal{L}_{i-1}$. We should emphasize that this condition depends on the *ordering* of the sets, and not just on this collection of sets. For instance, $\{1\}, \{1,2\}, \{2\}$ is a valid ordering whereas $\{1\}, \{2\}, \{1,2\}$ is not. We have thus a chain of ring families: $\mathcal{L}_1 \subset \mathcal{L}_2 \subset \cdots \subset \mathcal{L}_k$ where all the containments are proper. The question is how large can k be, and the next theorem shows that it can be at most quadratic in n.

Theorem 12. *Consider a chain of ring families, $\mathcal{L}_0 = \emptyset \ne \mathcal{L}_1 \subsetneq \mathcal{L}_2 \subsetneq \cdots \subsetneq \mathcal{L}_k$ within 2^V with $n = |V|$. Then*

$$k \le \binom{n+1}{2} + 1$$

.

Before proving this theorem, we show that the bound on the number of sets is tight.

Example 1. Let $V = \{1, \cdots, n\}$. For each $1 \le i \le j \le n$, consider intervals $[i,j] = \{k \mid i \le k \le j\}$. Add also the empty set $\emptyset$ as the trivial interval $[0,0]$ (as $0 \notin E$). We have just defined $k = \binom{n+1}{2} + 1$ sets. Define a complete order on these intervals in the following way: $(i,j) \prec (s,t)$ if $j < t$ or ($j = t$ and $i < s$). We claim that if we consider these intervals in the order given by $\prec$, we satisfy the main assumption of the theorem that $[s,t] \notin \mathcal{R}(\mathcal{T}_{st})$ where $\mathcal{T}_{st} = \{[i,j] \mid (i,j) \prec (s,t)\}$. Indeed, for $s = 1$ and any t, we have that $[1,t] \notin \mathcal{R}(\mathcal{T}_{1t})$ since $\bigcup_{I\in\mathcal{T}_{1t}} I = [1, t-1] \not\supset [1,t]$. On the other hand, for $s > 1$ and any t, we have that $[s,t] \notin \mathcal{R}(\mathcal{T}_{st})$ since for all $I \in \mathcal{T}_{st}$ we have ($t \in I \Rightarrow s-1 \in I$) while this is not the case for $[s,t]$.

Proof. For each $1 \le i \le k$, let $T_i \in \mathcal{L}_i \setminus \mathcal{L}_{i-1}$. We can assume that $\mathcal{L}_i = \mathcal{R}(\{T_1, \cdots T_i\})$ (otherwise a longer chain of ring families can be constructed). If none of the T_i's is the empty set, we can increase the length of the chain by considering (the ring families generated by) the sequence $\emptyset, T_1, T_2, \cdots, T_k$. Similarly if V is not among the T_i's, we can add V either in first or second position in the sequence. So we can assume that the sequence has $T_1 = \emptyset$ and $T_2 = V$, i.e. $\mathcal{L}_1 = \{\emptyset\}$ and $\mathcal{L}_2 = \{\emptyset, V\}$.

When considering $\mathcal{L}_2$, its digraph representation has $A = \emptyset$, $B = V$ and the directed graph $D = (V, E)$ is the bi-directed complete graph on V. To show a weaker bound of $k \leq 2 + n(n-1)$ is easy: every T_i we consider in the sequence will remove at least one arc of this digraph and no arc will get added.

To show the stronger bound in the statement of the theorem, consider the digraph D' obtained from D by contracting every strongly connected component of D and discarding all but one copy of (possibly) multiple arcs between two vertices of D'. We keep track of two parameters of D': s is its number of vertices and a is its the number of arcs. Initially, when considering $\mathcal{L}_2$, we have $s = 1$ strongly connected component and D' has no arc: $a = 0$. Every T_i we consider will either keep the same strongly connected components in D (i.e. same vertices in D') and remove (at least) one arc from D', or will break up at least one strongly connected component in D (i.e. increases vertices in D'). In the latter case, we can assume that only one strongly connected component is broken up into two strongly connected components and the number of arcs added is at most s since this newly formed connected component may have a single arc to every other strongly connected component. Thus, in the worst case, we move either from a digraph D' with parameters (s, a) to one with $(s, a-1)$ or from (s, a) to $(s+1, a+s)$. By induction, we claim that if the original one has parameters (s, a) then the number of steps before reaching the digraph on V with no arcs with parameters $(n, 0)$ is at most

$$a + \binom{n+1}{2} - \binom{s+1}{2}.$$

Indeed, this trivially holds by induction for any step $(s, a) \to (s, a-1)$ and it also holds for any step $(s, a) \to (s+1, a+s)$ since:

$$(a+s) + \binom{n+1}{2} - \binom{s+2}{2} + 1 = a + \binom{n+1}{2} - \binom{s+1}{2}.$$

As the digraph corresponding to $\mathcal{L}_2$ has parameters $(1, 0)$, we obtain that $k \leq 2 + \binom{n+1}{2} - 1 = \binom{n+1}{2} + 1$. $\square$

4.2 Analysis of discrete Newton's Algorithm

To prove Theorem 11, we start by recalling Radzik's analysis of Newton's algorithm for the case of modular functions ([Radzik, 1998]). First of all, the discrete Newton's algorithm, as stated in Algorithm 8 for solving $\max \delta : \min_{S \subseteq E} f(S) - \delta a(S) \geq 0$ terminates (Lemma 4.1). Recall that $h_i = \min_S f(S) - \delta_i a(S)$, $S_i \in \arg\min_S f(S) - \delta_i a(S)$, $g_i = a(S_i)$ and $\delta_{i+1} = \frac{f(S_i)}{a(S_i)}$. Let $f_i = f(S_i)$ and $g_i = a(S_i)$. Figure 4-1 illustrates the discrete Newton's algorithm and the notation.

Lemma 4.1. *Newton's algorithm as described in Algorithm 8 terminates in a finite number of steps t and generate sequences:*

(i) $h_1 < h_2 < \cdots < h_{t-1} < h_t = 0$,

(ii) $\delta_1 > \delta_2 > \cdots > \delta_{t-1} > \delta_t = \delta^* \geq 0$,

(iii) $g_1 > g_2 > \cdots > g_{t-1} > g_t \geq 0$.

Furthermore, if $g_t > 0$ then $\delta^* = 0$.

The first proof of the above lemma is often attributed to McCormick and Ervolina [McCormick and Ervolina, 1994] and we present it here for completeness.

Proof. Notice first that by the choice of $\delta_1 = \min_{e \in E, a(e) > 0} f(\{e\})/a_e$, $h_1 \leq 0$. Since we start with a feasible point in the extended submodular polytope $EP(f)$, $f(\cdot)$ can be assumed to be non-negative, and thus, $\delta_1 \geq 0$. Further, let S_1 be a minimizer of $\min_{S \subseteq E} f(S) - \delta_1 a(S)$. We know that the minimum of $f - \delta a$ is at most 0 (by the choice of δ_1), therefore $f(S_1) \leq \delta_1 a(S_1)$, therefore, $g_1 = a(S_1) \geq 0$. Thus, the claim of the lemma holds for the first iteration.

Assume by induction that the claim holds for all iterations i, for $1 \leq i \leq k$. Consider iteration $i = k + 1$, and let us suppose that the algorithm has not terminated yet. Then, using the definition of δ_{k+1} we get:

$$\delta_{k+1} = \frac{f_k}{g_k} = \frac{h_k + \delta_k g_k}{g_k} \ldots \text{ since } h_k = f_k - \delta_k g_k. \tag{4.3}$$

$$= \delta_k + \frac{h_k}{g_k}. \tag{4.4}$$

By induction we know that $\delta_k > 0$, $h_k < 0$, $g_k > 0$. Therefore, $\delta_{k+1} < \delta_k$. Moreover, $\delta_{k+1} \geq \delta^* \geq 0$, since otherwise the constraint with respect to the set S_k would be violated. Note that $h(\delta) = \min_{S \subseteq E} f(S) - \delta a(S)$ is the lower envelope of a number of linear functions, and therefore $h(\cdot)$ is a concave function. Moreover, $h(\delta)$ is a strictly decreasing function for $\delta \geq \delta_{k+1}$, therefore, $h_{k+1} < h_k$ given $\delta_{k+1} < \delta_k$.

Finally to show that $g_{k+1} < g_k$, consider the following two inequalities obtained by the minimality of S_{k+1} and S_k at δ_{k+1} and δ_k respectively:

$$f(S_{k+1}) - \delta_k a(S_{k+1}) \geq f(S_k) - \delta_k a(S_k) \tag{4.5}$$

$$f(S_{k+1}) - \delta_{k+1} a(S_{k+1}) \leq f(S_k) - \delta_{k+1} a(S_k) \tag{4.6}$$

Subtracting (4.6) from (4.5), we get:

$$\delta_k a(S_{k+1}) - \delta_{k+1} a(S_{k+1}) \geq \delta_{k+1} a(S_k) - \delta_k a(S_k) \tag{4.7}$$

$$\Rightarrow (\delta_k - \delta_{k+1}) g_k \geq (\delta_k - \delta_{k+1}) g_{k+1}. \tag{4.8}$$

Since $\delta_k > \delta_{k+1}$, we get $g_k \geq g_{k+1}$ and the inequality is tight whenever there exists iteration $k+1$, i.e. $f(S_k) - \delta_{k+1} a(S_k) = 0 > f(S_{k+1}) - \delta_{k+1} a(S_{k+1})$. Since the sequence of $\{g_i\}$ is strictly decreasing, all the elements in the sequence are distinct. Thus, the length of the sequence (hence the number of iterations of the algorithm) has to be finite as each $g_i = a(S_i)$ for some set S_i. □

As in Radzik's analysis, we use the following lemma, illustrated in Figure 4-2, and we reproduce here its proof.

Lemma 4.2. *For any $i < t$, we have $\frac{h_{i+1}}{h_i} + \frac{g_{i+1}}{g_i} \leq 1$.*

Proof. By definition of S_i, we have that

$$\begin{aligned} h_i = f(S_i) - \delta_i a(S_i) = f_i - \delta_i g_i \leq f(S_{i+1}) - \delta_i a(S_{i+1}) = f_{i+1} - \delta_i g_{i+1} \\ = h_{i+1} + \frac{f_i}{g_i} g_{i+1} - \frac{f_i - h_i}{g_i} g_{i+1} = h_{i+1} + h_i \frac{g_{i+1}}{g_i}. \end{aligned}$$

Since $h_i < 0$, dividing by h_i gives the statement. □

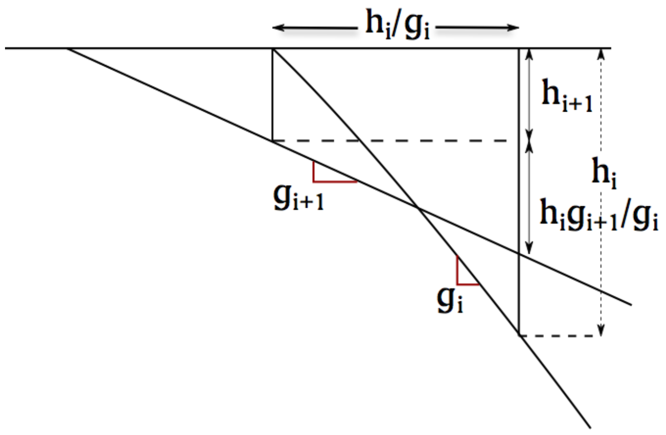


Figure 4-2: Illustration for showing that $h_{i+1} + h_i \frac{g_{i+1}}{g_i} \le h_i$, as in Lemma 4.2.

Thus, in every iteration, either g_i or h_i decreases by a constant factor smaller than 1. We can thus partition the iterations into two types, for example as

$$J_g = \left\{ i \mid \frac{g_{i+1}}{g_i} \le \frac{2}{3} \right\}$$

and $J_h = \{i \notin J_g\}$. Observe that $i \in J_h$ implies $\frac{h_{i+1}}{h_i} < \frac{1}{3}$. We first bound $|J_g|$ as was done in [Radzik, 1998].

Lemma 4.3. $|J_g| = O(n \log n)$.

Proof sketch. Let $J_g = \{i_1, i_2, \cdots, i_k\}$ and let $T_j = S_{i_j}$. From the monotonicity of g, these sets T_j are such that $a(T_{j+1}) \le \frac{2}{3} a(T_j)$. These can be viewed as linear inequalities with small coefficients involving the a_i's, and by normalizing and taking an extreme point of this polytope, Goemans (see [Radzik, 1998]) has shown that the number k of such sets is $O(n \log n)$.

Although we do not need this for the analysis, the bound of $O(n \log n)$ on the number of geometrically decreasing sets defined on n numbers is tight, as was shown by Mikael Goldmann in 1993 by a beautiful construction based on a Fourier-analytic approach of Håstad [Håstad, 1994]. We refer the interested reader to the conference paper version of this chapter that contains the full proof of this construction [Goemans et al., 2017].

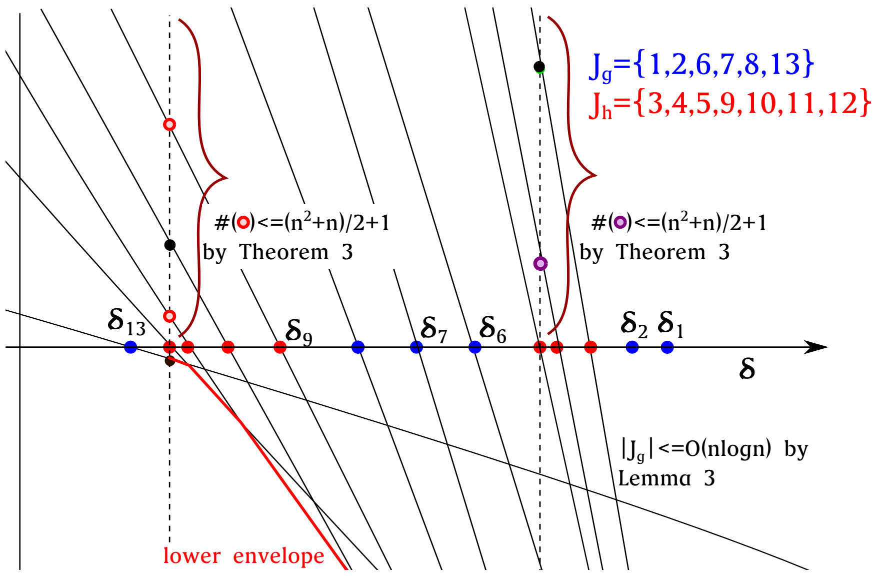


Figure 4-3: Illustration of the sets J_g and J_h and the bound on these required to show an $O(n^3 \log n)$ bound on the number of iterations of the discrete Newton's algorithm.

4.2.1 Weaker cubic upper bound

Before deriving the bound of $O(n^2)$ on $|J_g| + |J_h|$ for Theorem 11, we show how to derive a weaker bound of $O(n^3 \log n)$. For showing the $O(n^3 \log n)$ bound, first consider a block of *consecutive* iterations $[u, v] := \{u, u+1, \cdots, v\}$ within J_h.

Theorem 13. *Let $[u, v] \subseteq J_h$. Then $|[u, v]| \leq n^2 + n + 1$.*

Figure 4-3 illustrates an example of sets J_g and J_h along with the bound on their size.

The strategy of the proof is to show (i) that, for the submodular function $k_v(S) = f(S) - \delta_v a(S)$, the values of $k_v(S_i)$ for $i \in [u, v-1]$ form a geometrically decreasing series (Lemma 4.4), (ii) that each S_i cannot be in the ring family generated by $S_{i+1}, \ldots, S_{v-1}$ (Lemma 4.5 and Theorem 14), and (iii) then conclude using our Theorem 12 on the length of a chain of ring families.

Lemma 4.4. *Let $[u, v] \subseteq J_h$. Then for $k_v(S) = f(S) - \delta_v a(S)$, we have (i) $k_v(S_v) = \min_S k_v(S) = h_v$, (ii) $k_v(S_{v-1}) = 0$, (iii) $k_v(S_{v-2}) > 2|h_v|$ and (iv) $k_v(S_{i-1}) > 2k_v(S_i)$ for $i \in [u+1, v-1]$.*

Proof. Since $\frac{g_{i+1}}{g_i} > \frac{2}{3}$ for all $i \in [u, v]$, Lemma 4.2 implies that $\frac{h_{i+1}}{h_i} \leq \frac{1}{3}$, and thus

$$\frac{|h_{i+1}|}{g_{i+1}} \leq \frac{1}{2}\frac{|h_i|}{g_i}.$$

Since $\delta_{i+1} - \delta_i = \frac{f_i}{g_i} - \frac{f_i - h_i}{g_i} = \frac{h_i}{g_i}$. We deduce that

$$\delta_{i+1} - \delta_{i+2} = -\frac{h_{i+1}}{g_{i+1}} \leq \frac{1}{2}(\delta_i - \delta_{i+1}), \tag{4.9}$$

for all $i \in [u, v]$. Now, observe that for any $i \in [u, v-2]$, we have

$$\delta_{i+1} - \delta_v = \sum_{k=i+1}^{v-1} \delta_k - \delta_{k+1} \leq \frac{1}{2}\sum_{k=i+1}^{v-1} (\delta_{k-1} - \delta_k) = \frac{1}{2}(\delta_i - \delta_{v-1}) < \frac{1}{2}(\delta_i - \delta_v).$$

Thus

$$\delta_{i+1} - \delta_v < \frac{1}{2}(\delta_i - \delta_v), \tag{4.10}$$

and we can even extend the range of validity to $i \in [u, v]$ since for $i = v - 1$ or $i = v$, this follows from Lemma 4.1.

Consider the submodular function $k_v(S) = f(S) - \delta_v a(S)$. We have denoted its minimum value by $h_v < 0$ and S_v is one of its minimizers. For each $i \in [u, v-1]$ we have

$$k_v(S_i) = f_i - \delta_v g_i = g_i(\delta_{i+1} - \delta_v),$$

and therefore $k_v(S_{v-1}) = 0$ while $k_v(S_i) > 0$ for $i \in [u, v-2]$. Furthermore, (4.10) implies that

$$k_v(S_i) = g_i(\delta_{i+1} - \delta_v) < \frac{1}{2}\frac{g_i}{g_{i-1}}g_{i-1}(\delta_i - \delta_v) < \frac{1}{2}g_{i-1}(\delta_i - \delta_v) = \frac{1}{2}k_v(S_{i-1}),$$

and this is valid for $i \in [u, v-1]$. Thus the $k_v(S_i)$'s decrease geometrically with increasing i. In addition, we have $k_v(S_{v-2}) = g_{v-2}(\delta_{v-1} - \delta_v)$ while (by (4.9) and Lemma 4.1)

$$-k_v(S_v) = |h_v| = -h_v = g_v(\delta_v - \delta_{v+1}) < \frac{1}{2}g_{v-2}(\delta_{v-1} - \delta_v) = \frac{1}{2}k_v(S_{v-2}).$$

Summarizing, we have $k_v(S_v) = \min_S k_v(S) = h_v$, $k_v(S_{v-1}) = 0$, $k_v(S_{v-2}) > 2|h_v|$ and

$k_v(S_{i-1}) > 2k_v(S_i)$ for $i \in [u, v-1]$. ☐

We now show that for any submodular function and any ring family on the same ground set, the values attained by the submodular function cannot increase much when the ring family is increased to the smallest ring family including a single additional set. This lemma follows from the submodularity of f and Birkhoff's representation theorem for subsets contained in a ring family.

Lemma 4.5. *Let $f : 2^E \to \mathbb{R}$ be a submodular function with $f_{min} = \min_{S \subseteq E} f(S) \le 0$. Let $\mathcal{L}$ be any ring family over E and $T \notin \mathcal{L}$. Define $\mathcal{L}' := \mathcal{R}(\mathcal{L} \cup \{T\})$, $m = \max_{S \in \mathcal{L}} f(S)$ and $m' = \max_{S \in \mathcal{L}'} f(S)$. Then*

$$m' \le 2(m - f_{min}) + f(T).$$

Proof. Consider $S \in \mathcal{L}'$. Using (4.2), we can express S as $S = \bigcup_{i \in S} S_i$ where S_i can be either (i) T, or (ii) R for some $R \in \mathcal{L}$, or (iii) $R \cap T$ for some $R \in \mathcal{L}$. Taking the union of the sets R of type (ii), resp. (iii), into P, resp. Q, we can express S as $S = P \cup T$ or as $S = P \cup (Q \cap T)$ where $P, Q \in \mathcal{L}$ (since the existence of any case (i) annihilates the need for case (iii)).

Now using submodularity, we obtain that

$$f(P \cup T) \le f(P) + f(T) - f(P \cap T) \le m + f(T) - f_{min},$$

in the first case and

$$\begin{aligned} f(P \cup (Q \cap T)) &\le f(P) + f(Q \cap T) - f(P \cap Q \cap T) \\ &\le f(P) + f(Q) + f(T) - f(Q \cup T) - f(P \cap Q \cap T) \\ &\le 2m + f(T) - 2f_{min}. \end{aligned}$$

In either case, we get the desired bound on $f(S)$ for any $S \in \mathcal{R}'$. ☐

We will now use the bound in Lemma 4.5 to show that if a sequence of sets increases in their submodular function value by a factor of 4, then any set in the sequence is not contained in the ring family generated by the previous sets.

Theorem 14. *Let $f : 2^E \to \mathbb{R}$ be a submodular function with $f_{min} = \min_{S \subseteq E} f(S) \leq 0$. Consider a sequence of distinct sets $T_1, T_2, \cdots, T_q$ such that $f(T_1) = f_{min}$, $f(T_2) \geq -2f_{min}$, and $f(T_i) \geq 4f(T_{i-1})$ for $3 \leq i \leq q$. Then $T_i \notin \mathcal{R}(\{T_1, \cdots, T_{i-1}\})$ for all $1 < i \leq q$.*

Proof. This is certainly true for $i = 2$. For any $i \geq 1$, define $\mathcal{L}_i = \mathcal{R}(\{T_1, \cdots, T_i\})$ and $m_i = \max_{S \in \mathcal{L}_i} f(S)$. We know that $m_1 = f_{min} \leq 0$ and $m_2 = f(T_2)$ since $T_1 \cap T_2$ and $T_1 \cup T_2$ cannot have larger f values than T_2 by submodularity of f and minimality of T_1.

We claim by induction that $m_k \leq 2f(T_k) + 2f_{min}$ for any $k \geq 2$. This is true for $k = 2$ since $m_2 = f(T_2) \leq 2f(T_2) + 2f_{min}$. Assume the induction claim to be true for $k - 1$.

We get that $m_{k-1} \leq 2f(T_{k-1}) + 2f_{min} < 4f(T_{k-1})$. Since $f(T_k) > m_{k-1}$, $T_k \notin \mathcal{L}_{k-1} = \mathcal{R}(T_1, \cdots, T_{k-1})$. Using Lemma 4.5, we get that

$$\begin{aligned} m_k &\leq 2(m_{k-1} - f_{min}) + f(T_k) \\ &\leq 2(2f(T_{k-1}) + 2f_{min} - f_{min}) + f(T_k) \\ &\leq 2f(T_k) + 2f_{min}. \end{aligned}$$

Thus proving the induction step for k, and hence the statement of the theorem. □

We are now ready to prove Theorem 13.

Proof. (of Theorem 13) Apply Theorem 14 to the submodular function k_v given in Lemma 4.4. Let $T_1 = S_v$ and skip every other set to define $T_i = S_{v-2(i-1)}$ for $v - 2(i-1) \geq u$ i.e. $i \leq q := 1 + (v - u)/2$. Then the conditions of Theorem 14 are satisfied (thanks to Lemma 4.4), and we obtain a sequence of sets $T_1, \cdots, T_q$ such that $T_i \notin \mathcal{R}(T_1, \cdots, T_{i-1})$. Therefore, Theorem 12 on the length of a chain of ring families implies that $q \leq \binom{n+1}{2} + 1$, or $v - u \leq (n+1)n$. This means $|[u, v]| \leq n^2 + n + 1$. □

Since Lemma 4.3 shows that $|J_g| = O(n \log n)$ and we know from Theorem 13 that the intervals between two indices of J_g have length $O(n^2)$, this implies that $|J_g| + |J_h| = O(n \log n) \cdot O(n^2) = O(n^3 \log n)$.

4.2.2 Quadratic upper bound

The analysis of Theorem 13 can be improved by showing that we can extract a chain of ring families not just from one interval of J_h but from all of J_h. Instead of discarding every other set in J_h, we also need to discard the first $O(\log n)$ sets in every interval of J_h. This helps prove the main result of the paper that bounds the number of iterations in the discrete Newton's algorithm by at most $n^2 + O(n \log^2 n)$.

Theorem 15. *We have $|J_h| = n^2 + O(n \log^2 n)$.*

Before proving this, we need a variant of Lemma 4.5. The proof of the lemma again follows from the submodularity of f and Birkhoff's representation theorem for subsets contained in a ring family.

Lemma 4.6. *Let $\mathcal{T} \subseteq 2^E$ and assume that $f(S) \leq M$ for all $S \in \mathcal{T}$. Then for all $S \in \mathcal{R}(\mathcal{T})$*

$$f(S) \leq \frac{n^2}{4}(M - f_{min}).$$

Proof. Consider any $S \in \mathcal{R}(\mathcal{T})$. We know that $S = \bigcup_{i \in S} \bigcap_{j \notin S} U_{ij}$, for some $U_{ij} \in \mathcal{T}$. Define $S_i = \bigcap_{j \notin S} U_{ij}$; thus $S = \bigcup_{i \in S} S_i$.

We first claim that, for any k sets $T_1, T_2, \cdots, T_k \in \mathcal{T}$, we have that

$$f(\bigcap_{i=1}^{k} T_i) \leq kM - (k-1) f_{min}.$$

This is proved by induction on k, the base case of $k = 1$ being true by our assumption on f. Indeed, applying submodularity to $P = \bigcap_{i=1}^{k-1} T_i$ and T_k (and the inductive hypothesis), we get

$$\begin{aligned} f(\bigcap_{i=1}^{k} T_i) &= f(P \cap T_k) \leq f(P) + f(T_k) - f(P \cup T_k) \\ &\leq (k-1)M - (k-2) f_{min} + M - f_{min} = kM - (k-1) f_{min}. \end{aligned}$$

Using this claim, we get that for any $i \in S$, we have

$$\begin{aligned} f(S_i) = f(\bigcap_{j \notin S} U_{ij}) &\leq |E \setminus S|M - (|E \setminus S| - 1)f_{min} \\ &\leq |E \setminus S|(M - f_{min}). \end{aligned}$$

By a similar argument on the union of the S_i's, we derive that

$$\begin{aligned} f(S) &\leq |S|\left(|E \setminus S|M - (|E \setminus S| - 1)f_{min})\right) - (|S| - 1)f_{min} \\ &\leq |S||E \setminus S|M - (|S||E \setminus S| - 1)f_{min} \\ &\leq \frac{n^2}{4}(M - f_{min}). \end{aligned}$$

□

We are now ready to prove Theorem 15.

Proof. (of Theorem 15) Let $J_h = \bigcup_{i=1}^{\ell}[u_i, v_i]$ where $u_{i-1} > v_i + 1$ for $1 < i \leq \ell$. Notice that these intervals are ordered in a *reverse* order (compared to the natural ordering). We construct a sequence of sets $T_1, \cdots$ such that each set in the sequence is not in the ring closure of the previous ones. The first sets are just every other set S_i from $[u_1, v_1]$ obtained as before by using Theorem 14 and Lemma 4.4 with the submodular function k_{v_1}. Let $\mathcal{T}_1$ denote this sequence of sets.

Suppose now we have already considered the intervals $[u_j, v_j]$ for $j < i$ and have extracted a (long) sequence of sets $\mathcal{T}_{i-1}$ such that each set in the sequence is not in the ring closure of the previous ones. Consider now the submodular function $f := k_{v_i}$, and let $f_{min} \leq 0$ be its minimum value. Notice that from the order of iterations in the discrete Newton's algorithm we have that $f(T) < 0$ for $T \in \mathcal{T}_{i-1}$. Therefore by Lemma 4.6 with $M = 0$ we have that $f(S) \leq -\frac{n^2}{4} f_{min}$ for all $S \in \mathcal{R}(\mathcal{T}_{i-1})$. Using Lemma 4.4 with $f = k_{v_i}$, we have that only sets S_k with $k > v_i - \log(n^2/4)$ could possibly be in $\mathcal{R}(\mathcal{T}_{i-1})$, and therefore we can safely add to $\mathcal{T}_{i-1}$ every other set in $[u_i, v_{i-O(\log n)}]$ while maintaining the property that every set is not in the ring closure of the previous ones. Over all i, we have thus constructed a chain of ring families of length $\frac{1}{2}|J_h| - O(\log n)\ell = \frac{1}{2}|J_h| - O(\log n)|J_g|$. The theorem now follows from

Lemma 4.3 and Theorem 12. $\square$

Finally, combining Theorem 15 and Lemma 4.3 proves Theorem 11.

Proof. (of Theorem 11.) In every iteration of discrete Newton's algorithm, either g_i or h_i decreases by a constant factor smaller than 1. Thus, the iterations can be partitioned into two types $J_g = \left\{ i \mid \frac{g_{i+1}}{g_i} \leq \frac{2}{3} \right\}$ and $J_h = \{i \notin J_g\}$. Lemma 4.3 shows that $|J_g| = O(n \log n)$ and Theorem 15 shows that $|J_h| = n^2 + O(n \log^2 n)$. Thus, the total number of iterations is $n^2 + O(n \log^2 n)$. $\square$

4.3 Geometrically increasing sequences

In the proof for Theorem 11, we considered a sequence of sets $S_1, \cdots, S_k$ such that $f(S_i) \geq 4f(S_{i-1})$ for all $i \leq k$ for submodular functions f. In the special case when f is modular, we know that the maximum length of such a sequence is at most $O(n \log n)$ (Lemma 4.3). When f is submodular, we show that the maximum length is at most $\binom{n+1}{2} + 1$ by applying Theorem 12 to Theorem 14. In this section, we show that the bound for the submodular case is tight by constructing two related examples: one that uses interval sets of the ground set $\{1, \cdots, n\}$, and the other that assigns weights to arcs in a directed graph such that the cut function already gives such a sequence of quadratic (in the number of vertices) number of sets.

4.3.1 Interval submodular functions

In this section, we show that the bound for the submodular case is tight be constructing a sequence of $\binom{n+1}{2} + 1$ sets $\emptyset, S_1, \cdots, S_{\binom{n+1}{2}}$ for a submodular function f, such that $f(S_i) = 4f(S_{i-1})$ for all $i \leq \binom{n+1}{2}$.

For each $1 \leq i \leq j \leq n$, consider intervals $[i, j] = \{k \mid i \leq k \leq j\}$ and let the set of all intervals be $\mathcal{I} = \bigcup_{i,j} \{[i, j]\}$. Let $[i, j] = \emptyset$ whenever $i > j$. Consider a set function $f : \mathcal{I} \to \mathbb{R}_+$ such that $f(\emptyset) = 0$. We say f is *submodular on intervals* if for any $S, T \in \mathcal{I}$

such that $S \cup T \in \mathcal{I}$ and $S \cap T \in \mathcal{I}$, we have

$$f(S) + f(T) \geq f(S \cup T) + f(S \cap T).$$

Lemma 4.7. *Let τ and κ be monotonically increasing, nonnegative functions on the set $[n]$, then f defined by $f([i,j]) = \tau(i)\kappa(j)$ is submodular on intervals.*

Proof. Consider two intervals S and T. The statement follows trivially if $S \subseteq T$, so consider this is not the case. Let $S = [s_i, s_j]$ and $T = [t_i, t_j]$ and assume w.l.o.g that $s_j \geq t_j$.

i. Case $S \cap T \neq \emptyset$. This implies $t_i < s_i$ and $s_i \leq t_j \leq s_j$. In this case, $f(S) + f(T) - f(S \cap T) - f(S \cup T) = \tau(s_i)\kappa(s_j) + \tau(t_i)\kappa(t_j) - \tau(s_i)\kappa(t_j) - \tau(t_i)\kappa(s_j) = (\tau(s_i) - \tau(t_i))(\kappa(s_j) - \kappa(t_j)) \geq 0$.

ii. Case $S \cap T = \emptyset, S \cup T = [t_i, s_j]$. In this case, $f(S) + f(T) - f(S \cup T) = \tau(s_i)\kappa(s_j) + \tau(t_i)\kappa(t_j) - \tau(t_i)\kappa(s_j) \geq \kappa(s_j)(\tau(s_i) - \tau(t_i)) \geq 0$.

□

We show that one can *extend* any function that is submodular on intervals to a submodular function (defined over the ground set). This construction is general, and might be of independent interest. For any set $S \subseteq E$, define $\mathcal{I}(S)$ to be the set of maximum intervals contained in S. For example, for $S = \{1, 2, 3, 6, 9, 10\}$, $\mathcal{I}(S) = \{[1, 3], [6, 6], [9, 10]\}$.

Lemma 4.8. *Consider a set function f defined over intervals such that (i) $f(\emptyset) = 0$, (ii) $f([i,j]) \geq 0$ for interval $[i,j]$, (iii) for any $S, T \in \mathcal{I}$ such that $S \cap T, S \cup T \in \mathcal{I}$, $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$. Then, $g(S) = \sum_{I \in \mathcal{I}(S)} f(I)$ is submodular over the ground set $\{1, \ldots, n\}$.*

Proof. We will show that g is submodular by proving that for any $T \subseteq S$ and any $k \notin S$, $g(S \cup \{k\}) - g(S) \leq g(T \cup \{k\}) - g(T)$. Let the marginal gain obtained by adding k to S be $g_k(S) = g(S \cup \{k\}) - g(S)$.

Note that $\mathcal{I}(S \cup k) \backslash \mathcal{I}(S)$ can either contain (i) $[s, k]$, for some $s \leq k$, or (ii) $[k, u]$, for some $u > k$, or (iii) $[s, u]$ for $s \leq k \leq u$. In case (i), $g_k(S) = f([s, k]) - f([s, k-1])$; in case (ii), $g_k(S) = f([k, u]) - f([k+1, u])$; and in case (iii), $g_k(S) = f([s, u]) - f([s, k-1]) - f([k+1, u])$.

Thus, when comparing the values of $g_k(S)$ and $g_k(T)$, we are only concerned with intervals that are modified due to the addition of k.

Let $S \cup \{k\}$ contain the interval $[s, k-1] \cup \{k\} \cup [k+1, u]$ and $T \cup \{k\}$ contain the interval $[t, k-1] \cup \{k\} \cup [k+1, v]$ where $s \leq t$, $v \leq u$ (as $T \subseteq S$) and $s \leq k \leq u$ ($s = k$ implies $[s, k-1] = \emptyset$ and $u = k$ implies that $[k+1, u] = \emptyset$) and $t \leq k \leq v$ ($t = k$ implies $[t, k-1] = \emptyset$ and $v = k$ implies that $[k+1, v] = \emptyset$).

$$\begin{aligned}
& g(S \cup \{k\}) - g(S) - g(T \cup \{k\}) + g(T) \\
&= f([s,u]) - f([s,k-1]) - f([k+1,u]) - (f([t,v]) - f([t,k-1]) - f([k+1,v])) \\
&= f([s,u]) - f([s,k-1]) - f([k+1,u]) - f([t,v]) + f([t,k-1]) + f([k+1,v]) \\
&\leq f([s,u]) - f([s,k-1]) - f([k+1,v]) - f([t,u]) + f([t,k-1]) + f([k+1,v]) \qquad (4.11) \\
&= f([s,u]) - f([s,k-1]) - f([t,u]) + f([t,k-1]) \leq 0. \qquad (4.12)
\end{aligned}$$

where (4.11) follows from submodularity of f on intervals $[k+1, u]$ and $[t, v]$, i.e., $f([k+1, u]) + f([t, v]) \geq f([t, u]) + f([k+1, v])$, and (4.12) follows from submodularity of f on intervals $[s, k-1]$ and $[t, u]$. □

Construction. Consider the function $f([i, j]) = 4^{\frac{j(j-1)}{2}} 4^i$ for $[i, j] \in \mathcal{I}$, obtained by setting $\tau(i) = 4^i$ and $\kappa(j) = 4^{\frac{j(j-1)}{2}}$. This is submodular on intervals from Lemma 4.7. This function defined on intervals can be extended to a submodular function g by Lemma 4.8. Consider the total order $\prec$ defined on intervals $[i, j]$ specified in example 1 (Section 2). By our choice of τ and κ we have that $S \prec T$ implies $4g(S) \leq g(T)$. The submodular function g thus contains a sequence of length $\binom{n+1}{2} + 1$ of sets that increase geometrically in their function values.

4.3.2 Cut functions

The example from the previous section and the Birkhoff representation theorem motivates a construction of a complete directed graph $G = (V, A)$ ($|V| = n$) and a weight vector $w \in \mathbb{R}_+^{|A|}$ such that there exists a sequence of $m = \binom{n}{2}$ sets $\emptyset, S_1, \cdots, S_m \subseteq V$ that has $w(\delta^+(S_i)) \geq 4w(\delta^+(S_{i-1}))$ for all $i \geq 2$.

Construction. The sets S_i are all intervals of $[n-1]$, and are ordered by the complete order $\prec$ as defined previously. One can verify that the kth set S_k in the sequence is $S_k = [i,j]$ where $k = i + j(j-1)/2$.

Note that, if $i > 1$, for each interval $[i,j]$, arc $e_{i,j} := (j, i-1) \in \delta^+([i,j])$ and $(j, i-1) \notin \delta^+([s,t])$ for any $(s,t) \prec (i,j)$. For any interval $[1,j]$, arc $e_{1,j} := (j, j+1) \in \delta^+([1,j])$ and $(j, j+1) \notin \delta^+([s,t])$ for any $(s,t) \prec (1,j)$. Define arc weights w by $w(e_{i,j}) = 5^{i+j(j-1)/2}$. Thus, the arcs $e_{i,j}$ corresponding to the intervals $[i,j]$ increase in weight by a factor of 5. We claim that $w(\delta^+(S_k)) \geq 4w(\delta^+(S_{k-1}))$. This is true because $4\sum_{e_{s,t}:(s,t)\prec(i,j)} w(e_{s,t}) \leq w(e_{i,j})$.

Chapter 5

Approximate Generalized Counting

"What we see depends mainly on what we look for."
- John Lubbock.

In this chapter, we consider a popular online learning algorithm, the multiplicative weights update method, and its application to online linear optimization over over combinatorial structures as well as to do convex optimization over combinatorial polytopes. In Chapters 3 and 4, we restricted our attention to submodular polytopes, however in this chapter our combinatorial decision sets need not be submodular. We still define the combinatorial structures over a ground set E, for instance one can think of matchings defined on a graph $G = (V, E)$ where E is the set of edges (and also the ground set for representing matchings). We refer the reader to Section 2.2.3 for background on online learning and review here the multiplicative weights update algorithm (MWU) for learning over $\mathcal{U}$ combinatorial strategies here[1]. For instance, $\mathcal{U}$ can be the set of matchings in a bipartite graph or the set of spanning trees in a given graph.

The multiplicative weights update is an extremely intuitive (see (2.15) for definition) online learning algorithm. It starts with the uniform distribution over all the strategies $\mathcal{U}$, and simulates an iterative procedure where the learner plays a mixed strategy $p^{(t)}$ in each round t. In response, the adversary (or the environment) selects a loss vector $L^{(t)} \in [-1, 1]^{|\mathcal{U}|}$

[1]We present here the full-information setting, where the losses for each strategy (whether played or not) are observed by the learner. The results would also go through in the semi-bandit case, where the losses corresponding to the elements (for e.g. edges) in the selected combinatorial strategy (for e.g. spanning trees) are observed.

for round t. The learner observes losses for all the pure strategies in $\mathcal{U}$ and incurs loss equal to the expected loss of their mixed strategy, i.e. $loss(t) = \sum_{u\in\mathcal{U}} p^{(t)}(u)L^{(t)}(u)$. Subsequently, the learner updates their mixed strategy by lowering the weight of each pure strategy $u \in \mathcal{U}$ by a factor of $\exp(-\eta L^{(t)}(u))$ for a fixed constant $\eta < 1$. That is, for each round $t \geq 1$, the updates in the MWU algorithm are as follows, starting with $w^{(1)}(u) = 1$ for all $u \in \mathcal{U}$:

$$w^{(t+1)}(u) = w^{(t)}(u)\exp(-\eta L^{(t)}(u)) \;\; \forall u \in \mathcal{U}.$$

Standard analysis of the MWU algorithm shows that the average regret over T rounds scales as $O(\sqrt{1/T})$ (see for e.g. [Arora et al., 2012]). We include a proof in Theorem 16 for completeness. However, as the algorithm is described, it requires $O(|\mathcal{U}|)$ updates to the probability distribution $p^{(t)}$ in each round t. We are concerned with simulating the MWU algorithm over combinatorial sets, such as spanning trees, bipartite matchings, and these are typically exponential in number in the input of the problem. We represent these strategies with a 0/1 polytope $P \subseteq \mathbb{R}^n$, where $\mathcal{U} = \text{vert}(P)$, the vertex set of P. Thus, having a running time of $O(|\mathcal{U}|)$ per iteration is not practical or polynomial in the input size. The first question we consider is if we can do better.

(P3.1): *Under what conditions, can the MWU algorithm be simulated in logarithmic time in the number of combinatorial strategies, i.e. polynomial in* $\log(|\mathcal{U}|)$*?*

Informally, our main result in Section 5.1 is that if there exists an efficient algorithm to compute (even approximately) the marginals corresponding to a product distribution over the vertex set $\mathcal{U}$, then one can simulate efficiently the MWU algorithm over the polytope P in time polynomial in n. A product distribution p over $\mathcal{U} \subseteq \{0,1\}^n$ is such that $p(u) \propto \prod_{e:u(e)=1} \lambda(e)$ for some vector $\lambda \in \mathbb{R}^n_{>0}$. To be able to compute the marginal point, we require access to a *generalized (approximate) counting oracle* $\mathbf{M}_\epsilon$, that given $\lambda \in \mathbb{R}^n_{>0}$, computes $\tilde{Z}_\lambda$ and $\tilde{x} \in \mathbb{R}^n$ such that the following hold:

$$\text{(i) } (1-\epsilon)Z_\lambda \leq \tilde{Z}_\lambda \leq (1+\epsilon)Z_\lambda, \tag{5.1}$$

$$\text{(ii) } \forall e \in E, (1-\epsilon)x_\lambda(e) \leq \tilde{x}_\lambda(e) \leq (1+\epsilon)x_\lambda(e), \tag{5.2}$$

where $Z_\lambda = \sum_{u\in\mathcal{U}} \prod_{e:u(e)=1} \lambda(e)$ and x_λ is the marginal point corresponding to the product distribution. Note that for any $s \in E$,

$$x_\lambda(s) = \sum_{u\in\mathcal{U}:u(s)=1} \prod_{e:u(e)=1} \lambda(e).$$

Next, we look deeper into the fact that the MWU algorithm over N experts is a special case of the online mirror descent algorithm on the N-dimensional simplex $\Delta_N = \{x \in \mathbb{R}^N_+ \mid \sum_e x(e) = 1\}$ under the entropic divergence (i.e. KL-divergence) and the L_1-norm (see Lemma 5.2). This equivalence follows from the observation that the KL-divergence projection of any vector $w \in \mathbb{R}^N_{>0}$ over an N-dimensional simplex is obtained by normalizing w by its L_1 norm, i.e.

$$\arg\min_{z\in\Delta_N} \sum_{i=1}^{N} (z_i \ln(\frac{z_i}{w_i}) - z_i + w_i) = w/||w||_1. \tag{5.3}$$

An approximate generalized counting oracle thus gives an efficient way of computing approximate projections onto a high-dimensional simplex. However, we know that any polytope can be equivalently expressed a convex hull of its vertices $\mathcal{U}$ using probability distributions over $\mathcal{U}$. In Section 5.2, we partially answer the following question:

(P3.2): *What are the implications of being able to compute projections efficiently in a different representation of the polytope?*

Our main result in Section 5.2, informally, is that efficient generalized counting oracles over the vertex set $\mathcal{U}$ of a 0/1 polytope P can be used to compute projections over $\Delta_{|\mathcal{U}|}$, and this in turn can be used in conjunction with mirror descent (and its variants) to minimize convex functions over P (without requiring to compute projections over P itself).

5.1 Online linear optimization

In order to simulate the MWU algorithm over an exponentially sized vertex set $\mathcal{U}$ of a 0/1 polytope $P \subseteq \mathbb{R}^n$, we should be able to (i) represent the loss vector compactly (in dimension n) or allow oracle access to the loss vector, (ii) update the probability distribution efficiently given the losses in any round t. Recently, in a work by [Hazan and Koren, 2015], it was shown that any online algorithm requires $\tilde{O}(\sqrt{N})$ time to approximate the value of an N-strategy two-player zero-sum game, even when given access to constant time best-response oracles. It was known as early as 1951 [Robinson, 1951] that Nash-equilibria for two-player zero-sum games can be found by simulating an online learning algorithm: one of the player acts as a learner while the other generates adversarial losses and the average of the strategies played by each player converges to an approximate equilibrium. The connection on online learning with two-player games is discussed in more detail in Chapter 6. However, what this implies for the MWU algorithm in our case, is that under no assumptions on the structure of the loss function it is not possible to achieve a running time for the algorithm better than $O(\sqrt{|\mathcal{U}|})$.

We assume here that the losses can be compactly represented as linear functions over the vertices, such that $L^{(t)}(u) = u^T l^{(t)}$ for all $u \in \mathcal{U}$, for some $l^{(t)} \in \mathbb{R}^n$. The marginal point corresponding to the probability distribution $p^{(t)}$ over the vertices is simply $x^{(t)} = \sum_{u\in\mathcal{U}} p^{(t)}(u)u$. Since $x^{(t)}$ is a convex combination of the vertices, it lies in P. Interestingly, the linearity of the loss functions extends to the marginal point and it is easy to show that the expected loss in round t is $p^{(t)T}L^{(t)} = \sum_{u\in\mathcal{U}} p^{(t)}(u)u^T l^{(t)} = x^{(t)T}l^{(t)}$.

Product distributions For linear loss functions, one can simulate the MWU algorithm in time polynomial in n, by the use of *product distributions*: $p \in [0,1]^{|\mathcal{U}|}$ over the set $\mathcal{U}$ such that $p(u) \propto \prod_{e\in u} \lambda_e$ for all $u \in \mathcal{U}$ and some vector $\lambda \in \mathbb{R}^n_{>0}$. We refer to the λ vector as the *multiplier vector* of the product distribution. The two key observations we make here are that *product distributions can be updated efficiently by updating only the multipliers (for linear loss functions)*, and *multiplicative updates on a product distribution result in a product distribution again.*

To argue that the MWU can work by updating only product distributions, suppose first that in some iteration t of the MWU algorithm, we are given a product distribution $p^{(t)}$ over the vertex set $\mathcal{U}$ implicitly by its multiplier vector $\lambda^{(t)} \in \mathbb{R}^n$, and a loss vector $l^{(t)} \in \mathbb{R}^n$ is revealed such that the loss of each vertex u is $u^T l^{(t)}$. In order to multiplicatively update the probability of each vertex u as

$$p^{(t+1)}(u) \propto p^{(t)}(u) \exp(-\eta u^T l^{(t)}),$$

note that we can simply update the multipliers with the loss of each component.

$$\begin{aligned} p^{(t+1)}(u) \propto p^{(t)}(u) \exp(-\eta u^T l^{(t)}) &\propto \left(\prod_{e \in u} \lambda^{(t)}(e) \right) \exp(-\eta u^T l^{(t)}) \\ &\propto \prod_{e \in u} \left(\lambda^{(t)}(e) \exp(-\eta l^{(t)}(e)) \right) \qquad \text{as } u \in \{0,1\}^n. \end{aligned} \tag{5.4}$$

Hence, the resulting probability distribution $p^{(t+1)}$ is also a product distribution, and we can implicitly represent it in the form of the multipliers $\lambda^{(t+1)}(e) = \lambda^{(t)}(e) \exp(-\eta l^{(t)}(e))$ for $e \in E$ in the next round of the MWU algorithm. It is easy to start with a uniform distribution over all vertices in this representation, by simply setting $\lambda^{(1)}(e) = 1$ for all $e \in E$. Thus, in different rounds of the MWU algorithm, we move from one product distribution to another. The proof follows from the standard regret analysis for the MWU algorithm, but we include it here for completeness.

Theorem 16. *Assume that all costs $L^{(t)} \in [-1,1]^{\mathcal{U}}$ such that $L^{(t)}(u) = u^T l^{(t)}$ for some $l^{(t)} \in \mathbb{R}^n$ and $\eta \leq 1$. Then, the* MWU *algorithm with product distributions guarantees that after T rounds, we have*

$$\sum_{t=1}^{T} x^{(t)T} l^{(t)} - \min_{x \in P} \sum_{t=1}^{T} x^T l^{(t)} \leq \eta T + \frac{\ln |\mathcal{U}|}{\eta}. \tag{5.5}$$

Proof. We want to show that the updates to the weights of each vertex $u \in \mathcal{U} = \mathrm{vert}(P)$ (recall $P \subseteq \mathbb{R}^n$) can be done efficiently. For the multipliers $\lambda^{(t)}$ in each round, let $w^{(t)}(u)$ be the unnormalized probability for each vertex u, i.e., $w^{(t)}(u) = \prod_{e:u(e)=1} \lambda^{(t)}(e)$. Let $Z^{(t)}$ be the normalization constant for round t, i.e., $Z^{(t)} = \sum_{u \in \mathcal{U}} w^{(t)}(u)$. Thus, the probability of each vertex u is $p^{(t)}(u) = w^{(t)}(u)/Z^{(t)}$. We assume that for each round t, the losses $L^{(t)}(u) \in [-1,1]$ for all $u \in \mathcal{U}$, or equivalently $u^T l^{(t)} \in [-1,1]$ for all $u \in \mathcal{U}$.

The algorithm starts with $\lambda^{(1)}(e) = 1$ for all $e \in E$ and thus $w^{(1)}(u) = 1$ for all $u \in \mathcal{U}$. First note that,

$$\begin{aligned}
w^{(t+1)}(u) &= \prod_{e \in u} \lambda^{(t+1)}(e) = \prod_{e \in u} \lambda^{(t)}(e) \exp(-\eta l^{(t)}(e)) \\
&= \exp(-\eta u^T l^{(t)}) \prod_{e \in u} \lambda^{(t)}(e) \qquad \ldots u \in \{0,1\}^m. \\
&= w^{(t)}(u) \exp(-\eta u^T l^{(t)}) = w^{(1)}(u) \exp(-\eta \sum_{t=1}^{T} u^T l^{(t)}). \qquad (5.6)
\end{aligned}$$

Next, we bound the partition function in round $t+1$.

$$\begin{aligned}
Z^{(t+1)} &= \sum_{u \in \mathcal{U}} w^{(t+1)}(u) = \sum_{u \in \mathcal{U}} w^{(t)}(u) \exp(-\eta u^T l^{(t)}) \qquad (5.7) \\
&\le \sum_{u \in \mathcal{U}} w^{(t)}(u)(1 - \eta u^T l^{(t)} + \eta^2 (u^T l^{(t)})^2) \quad \ldots e^x \le 1 + x + x^2 \text{ for all } x \in [-1,1]. \\
& \qquad (5.8) \\
&= \sum_{u \in \mathcal{U}} w^{(t)}(u) - \eta \sum_{u \in \mathcal{U}} w^{(t)}(u) u^T l^{(t)} + \eta^2 \sum_{u \in \mathcal{U}} w^{(t)}(u)(u^T l^{(t)})^2 \qquad (5.9) \\
&= Z^{(t)} - \eta Z^{(t)} x^{(t)T} l^{(t)} + \eta^2 Z^{(t)} \sum_{u \in \mathcal{U}} p^{(t)}(u)(u^T l^{(t)})^2 \qquad (5.10) \\
&\le Z^{(t)} - \eta Z^{(t)} x^{(t)T} l^{(t)} + \eta^2 Z^{(t)} \qquad (5.11) \\
&\le Z^{(t)} \exp(-\eta x^{(t)T} l^{(t)} + \eta^2) \quad \ldots (1 + x) \le e^x \ \forall \ x. \qquad (5.12)
\end{aligned}$$

Rolling out the above till the first round, we get

$$Z^{(t+1)} \le Z^{(1)} \exp(-\eta \sum_{t=1}^{T} x^{(t)T} l^{(t)} + T\eta^2). \qquad (5.13)$$

Since $w^{(t+1)}(u) \le Z^{(t+1)}$ for all $u \in \mathcal{U}$, using (5.6) we get

$$w^{(1)}(u) \exp(-\eta \sum_{t=1}^{T} u^T l^{(t)}) \le Z^{(1)} \exp(-\eta \sum_{t=1}^{T} x^{(t)T} l^{(t)} + T\eta^2) \qquad (5.14)$$

$$\Rightarrow \ln w^{(1)}(u) - \eta \sum_{t=1}^{T} u^T l^{(t)} \leq \ln Z^{(1)} - \eta \sum_{t=1}^{T} x^{(t)T} l^{(t)} + T\eta^2 \tag{5.15}$$

$$\Rightarrow \sum_{t=1}^{T} x^{(t)T} l^{(t)} \leq \sum_{t=1}^{T} u^T l^{(t)} + T\eta + \frac{\ln |\mathcal{U}|}{\eta} \quad \ldots \eta > 0, w^{(1)}(u) = 1, Z^{(1)} = |\mathcal{U}|. \tag{5.16}$$

and the statement of the theorem follows. □

Corollary 2. *Setting* $\eta = \sqrt{\frac{\ln |\mathcal{U}|}{T}}$ *in* (5.16)*, shows that the average regret scales as* $O(\sqrt{\frac{\ln |U|}{T}})$*:*

$$\frac{1}{T} \sum_{t=1}^{T} x^{(t)T} l^{(t)} - \frac{1}{T} \sum_{t=1}^{T} u^T l^{(t)} \leq 2\sqrt{\frac{\ln |\mathcal{U}|}{T}}.$$

We would like to draw attention to the fact that, by the use of product distributions, we are not restricting the online algorithms to search over a subset of marginal points. We can indeed restrict our attention to product distributions without loss of generality; any point in a 0/1 polytope can be decomposed into a product distribution. We include a proof of the following lemma for completeness. (In Section 5.2, we also show that the MWU algorithm can be used to (approximately) compute the product distribution corresponding to any given marginal point.)

Lemma 5.1 ([Asadpour et al., 2010], [Singh and Vishnoi, 2014])**.** *Given a vector z in the relative interior of a 0/1 polytope $P \subseteq \mathbb{R}^n$, there exist γ_e^* for all $e \in E$ such that if we sample a vertex u of P according to $p^*(u) = \exp(\gamma^*(u))$, then $\mathbb{P}(e \in u) = z(e)$ for every $e \in E$.*

Proof. The maximum entropy distribution $p^*(\cdot)$ with respect to given marginal probabilities $z \in P$ is the optimum solution of the following convex problem:

$$\text{(CP)} \;\; = \inf \sum_{u \in \mathcal{U}} p(u) \log p(u)$$

$$\text{s.t.} \sum_{T : e \in T} p(u) = z(e) \quad \forall e \in E,$$

$$\sum_{u \in \mathcal{U}} p(u) = 1, p(u) \geq 0 \quad \forall u \in \mathcal{U}.$$

This convex program is feasible whenever z belongs to the relative interior of the polytope P. As the objective function is bounded and the feasible region is compact (closed and bounded), the infimum is attained and there exists an optimum solution $p^*(\cdot)$. Furthermore, since the objective functions is strictly convex, this maximum entropy distribution $p^*(\cdot)$ is unique. Let $OPT_{(CP)}$ denote the optimum value of this convex program (CP).

The value $p^*(u)$ determines the probability of sampling any vertex u in the maximum entropy rounding scheme. We now want to show that, if we assume that z is in the relative interior of the polytope, then $p^*(u) > 0$ for every $u \in \mathcal{U}$ and $p^*(u)$ admits a simple exponential formula. Let us write the Lagrange dual to the convex program (CP). For every $e \in E$, we associate a Lagrange multiplier δ_e to the constraint corresponding to the marginal probability $z(e)$, and define the Lagrange function by

$$\begin{aligned} L(p,\delta,\theta) &= \sum_{u\in\mathcal{U}} p(u)\log p(u) - \sum_{e\in E}\delta_e\big(\sum_{u:e\in u} p(u) - z(e)\big) - \theta\big(\sum_{u\in\mathcal{U}} p(u) - 1\big), \\ &= \sum_{e\in E}\delta_e z(e) + \theta + \sum_{u\in\mathcal{U}}\big(p(u)\log p(u) - p(u)\sum_{e\in u}\delta_e - \theta p(u)\big). \end{aligned}$$

The Lagrange dual to CP is now

$$\sup_{\delta,\theta}\inf_{p\geq 0} L(p,\delta,\theta). \tag{5.17}$$

The inner infimum in this dual is easy to solve. As the contributions of the $p(u)'$s are separable, we have that, for every $u \in \mathcal{U}$, $p(u)$ must minimize the convex function $p(u)\log p(u) - p(u)\sum_{e\in u}\delta_e - \theta p(u)$. This minimum is given by $p(u) = \exp(\delta(u) + \theta - 1)$, where $\delta(u) = \sum_{e\in u}\delta_e$. Thus,

$$g(\delta,\theta) = \inf_{p\geq 0} L(p,\delta,\theta) = \sum_{e\in E}\delta_e z_e + \theta - \sum_{u\in\mathcal{U}}\exp(\delta(u)+\theta-1), \tag{5.18}$$

and the dual becomes to solve $\sup_{\delta,\theta} g(\delta,\theta)$. Optimizing $g(\delta,\theta)$ over θ, we get

$$1 - e^{\theta-1}\sum_{u\in\mathcal{U}}\exp(\delta(u)) = 0 \tag{5.19}$$

$$\Rightarrow e^{\theta-1} = 1/\sum_{u\in\mathcal{U}} \exp(\delta(u)). \tag{5.20}$$

Thus, the dual problem reduces to

$$\sup_{\delta} g(\delta) = \sup_{\delta} \sum_{e\in E} \delta_e z_e + \theta - 1 = \sup_{\delta} \sum_{e\in E} \delta_e z_e - \ln(\sum_{u\in\mathcal{U}} \exp(\delta(u))). \tag{5.21}$$

Since $z \in \text{relint}(P)$ (relative interior of P), the primal-dual programs satisfy Slater's condition and strong duality holds, implying the optimum value of CP and (5.21) are the same. Moreover, by the strict concavity of the entropy function, the optimum is unique. Hence, at optimality, $p^*(u) = \exp \delta^*(u)/\sum_{u\in\mathcal{U}} \exp \delta^*(u)$, where δ^* and p^* are optimal dual and primal solutions respectively. □

Product distributions thus allow us to maintain a distribution on (the exponentially sized) $\mathcal{U}$ by simply maintaining $\lambda \in \mathbb{R}^n_{>0}$. To be able to sample from these product distributions, or output the marginal point or in some applications to even compute the loss vector in round t (for instance when learning to find Nash-equilibria in two-player zero-sum games, as we will see in Chapter 6), we require access to a *generalized (approximate) counting oracle* $\mathbf{M}_\epsilon$ as defined in the introduction of this chapter (with conditions (5.1), (5.2)). For certain self-reducible structures[2] $\mathcal{U}$ [Schnorr, 1976] (such as spanning trees, matchings or Hamiltonian cycles), the generalized approximate counting oracle can be replaced by a fully polynomial approximate generator as shown by [Jerrum et al., 1986], i.e. being able to sample from product distributions is sufficient.

Next suppose that the generalized counting oracle is approximate, and it introduces errors in the marginal point corresponding to the product distribution. We show that the MWU algorithm is robust to such errors. Since we always maintain the true $\lambda^{(t)}$ in each round, the error due to the approximate counting oracle gets added to the regret bound of the MWU algorithm.

[2]Informally, self-reducibility means that there exists an inductive construction of the combinatorial object from a smaller instance of the same problem [Sinclair and Jerrum, 1989]. For example, conditioned on whether an edge is taken or not, the problem of finding a spanning tree (or a matching) on a given graph reduces to the problem of finding a spanning tree (or a matching) in a modified graph.

Corollary 3. *Given a polynomial approximate generalized counting oracle $\boldsymbol{M}_\epsilon$ such that $||x_\lambda - \tilde{x}_\lambda||_\infty \leq \epsilon$ and assuming that all the loss vectors satisfy $L^{(t)} \in [-1,1]^{\mathcal{U}}$, such that $L^{(t)}(u) = u^T l^{(t)}$ for some $l^{(t)} \in \mathbb{R}^n$, $\eta < 1$, then the* MWU *algorithm guarantees that after T rounds, we have*

$$\sum_{t=1}^{T} \tilde{x}^{(t)T} l^{(t)} - \min_{x \in P} \sum_{t=1}^{T} x^T l^{(t)} \leq \eta T + \frac{\ln \mathcal{U}}{\eta} + \epsilon \sum_{t=1}^{T} \|l^{(t)}\|_1. \tag{5.22}$$

Proof. Let the multipliers in each round t be $\lambda^{(t)}$, the corresponding true and approximate marginal points in round t be $x^{(t)}$ and $\tilde{x}^{(t)}$ respectively, such that $||x^{(t)} - \tilde{x}^{(t)}||_\infty \leq \epsilon_1$ (as in the definition of approximate generalized counting oracles in (5.2)). The loss vectors in each round are $l^{(t)}$, such that loss of any pure strategy $u \in \mathcal{U}$ is $u^T l^{(t)}$.

Even though we cannot compute $x^{(t)}$ exactly, we do maintain multipliers $\lambda^{(t)}$s that correspond to the true marginals. Using the proof for Theorem 16, we get the following regret bound with respect to the true marginals:

$$\frac{1}{t} \sum_{i=1}^{t} x^{(i)T} l^{(t)} \leq \frac{1}{t} \sum_{i=1}^{t} u^T l^{(i)} + T\eta + \frac{\ln |\mathcal{U}|}{\eta} \tag{5.23}$$

We do not have the value for $x^{(i)}$ for $i = 1, \ldots, t$, but only estimates $\tilde{x}^{(i)}$ for $i = 1, \ldots, t$ such that $||\tilde{x}^{(i)} - x^{(i)}||_\infty \leq \epsilon_1$. Since the losses we consider are bilinear, we can bound the loss of the estimated point in each iteration i as follows:

$$|\tilde{x}^{(i)T} l^{(i)} - x^{(i)T} l^{(i)}| \leq \epsilon_1 e^T l^{(i)} \leq \epsilon_1 \|l^{(i)}\|_1, \tag{5.24}$$

where $e = (1, \ldots, 1)^T$. Plugging in (5.24) in (6.8), we get the claim of the corollary. □

The framework we developed in this chapter is general and allows one to plug in results from the literature on generalized approximate counting (and sampling) of combinatorial structures to be able to simulate the multiplicative weights update algorithm over these strategies, efficiently. We next discuss some related work that can be viewed as a special case of this framework.

Applications and related work One of the earliest use of product distributions to simulate the MWU algorithm efficiently was to learn the best pruning of a binary decision diagram [Helmbold and Schapire, 1997]. Thereafter, Takimoto and Warmuth used a recursive algorithm to count weighted paths in a graph while allowing for cycles [Takimoto and Warmuth, 2003]. For learning over the spanning tree polytope, an exact generalized counting algorithm follows from Kirchoff's matrix theorem (attributed first to a result by Gustav Kirchhoff in 1847, and David Wilson gave an algorithm for generating spanning tree uniformly at random [Wilson, 1996], see also [Lyons and Peres, 2005]) that states that the value of $\sum_T \prod_{e\in T} \lambda_e$ is equal to the value of the determinant of any cofactor of the weighted Laplacian of the graph. One can use fast Laplacian solvers (see for e.g., [Koutis et al., 2010]) for obtaining an efficient approximate marginal oracle. Kirchhoff's determinantal formula also extends to (exact) counting of bases of regular matroids (see for e.g., [Welsh, 2009]). For learning over the bipartite matching polytope (i.e., rankings), one can use the randomized generalized approximate counting oracle from [Jerrum et al., 2004] for computing permanents to obtain a feasible marginal oracle. Note that the problem of counting the number of perfect matchings exactly in a bipartite graph is #P-complete as it is equivalent to computing the permanent of a $0/1$ matrix [Valiant, 1979]. The problem of approximately counting the number of perfect matchings in a general graph is however a long standing open problem. Other examples of polytopes that admit polynomial approximate counting oracles are the $0-1$ circulations in directed graphs or subgraphs with pre-specified degree sequences [Jerrum et al., 2004] and the cycle cover polytope (or $0-1$ circulations) over directed graphs [Singh and Vishnoi, 2014]. We summarize related work that used product distributions for simulating the MWU in polynomial time over certain combinatorial strategies.

5.2 Convex optimization

In this section, we show that any convex function $h : P \to \mathbb{R}$, defined over a $0/1$ polytope $P \subseteq \mathbb{R}^n$ can be minimized using the framework defined above for online linear optimization and efficient oracles for generalized counting over the vertex set $\mathcal{U}$ of P. We will use the mirror descent algorithm and its variants (for different convexity properties of $h(\cdot)$), and

Combinatorial Structure	Counting Oracle	Efficient MWU
bounded depth binary decision trees,	dynamic programming	[Helmbold and Schapire, 1997]
general s-t paths (that allow for cycles) in a graph, simple paths in a directed acylic graph	dynamic programming	[Takimoto and Warmuth, 2003]
Spanning trees	[Wilson, 1996]	[Koo et al., 2007]
Bipartite Matchings	[Jerrum et al., 2004]	[Koolen et al., 2010]
Bases of regular matroids	[Welsh, 2009]	this work
0-1 circulations in directed graphs with pre-specified degree sequences	[Jerrum et al., 2004]	this work
Cycle covers	[Jerrum et al., 2004], [Singh and Vishnoi, 2014]	this work

Table 5.1: List of known results for approximate counting over combinatorial strategies and efficient simulation of the MWU algorithm using product distributions.

refer the reader to Section 2.2.2 for background and useful references.

Consider a convex function $h : P \to \mathbb{R}$ that we would like to minimize over P. Note that each point $x \in P$ can be written as a convex combination of the vertices of P, i.e.,

$$P_u = \{x \mid x = \sum_{u \in \mathcal{U}} p(u)u, \sum_{u \in \mathcal{U}} p(u) = 1, p \geq 0\}. \tag{5.25}$$

Here, p is a probability distribution over the vertex set $\mathcal{U}$, i.e.

$$p \in \Delta_{\mathcal{U}} = \{p \in [0,1]^{|\mathcal{U}|} : \sum_{u} p(u) = 1, p \geq 0\}.$$

We have represented P by raising it to P_u that lies an exponentially larger dimension (see Figure 5-1 for an illustration). Note that $g(p) = h(\sum_{u \in \mathcal{U}} p(u)u)$ is also convex in $p \in \mathbb{R}^{|\mathcal{U}|}$ as convexity is invariant under affine maps.

An extended formulation of a polytope $P \subseteq \mathbb{R}^n$ is a polytope in a higher dimension $P_q \subseteq \mathbb{R}^{n+q}$ such that $P = \text{proj}_n(P_q) := \{x \in \mathbb{R}^n | \exists y \in \mathbb{R}^q, (x,y) \in P_q\}$. If the number of facets of P_q is polynomial in n, then the extended formulation is said to be *compact*. The number of facets of the smallest possible extended formulation of a polytope is called its *extension complexity*. Suppose a polytope P has a small extension complexity $xc(P_q)$, we can do linear optimization over it efficiently by using a formulation with a small number of constraints (and polynomial number of variables). The key idea behind the concept of extended formulations

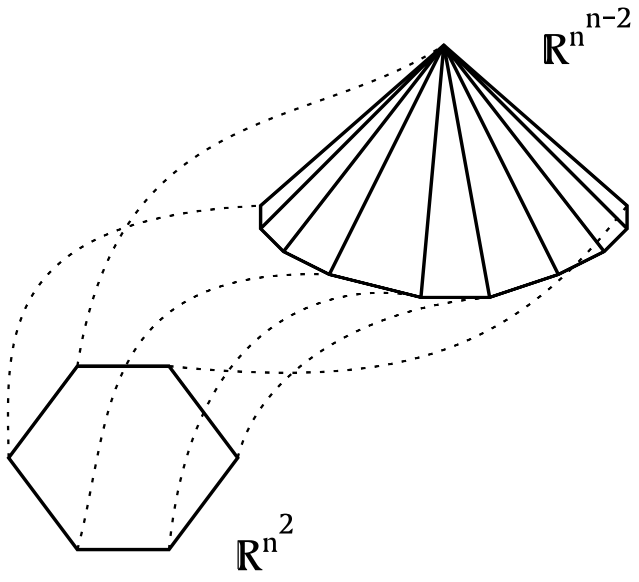


Figure 5-1: An intuitive illustration to show a polytope in $\mathbb{R}^{O(n^2)}$ raised to the simplex of its vertices, that lies in $\mathbb{R}^{O(n^{n-2})}$ space.

is to raise the polytope to a higher dimension so that linear optimization over it becomes easier. By representing P as P_u we have in fact raised P to a higher (exponential) dimension, and we will show that this has made convex optimization over P *easier*.

In Section 5.1, we showed that online linear optimization over combinatorial sets $\mathcal{U}$ can be done using the MWU algorithm as long as there exist efficient approximate counting oracles over the vertex set of P. Note that the gradient ∇g with respect to any vertex u is $(\nabla g(p))_u = \frac{\partial h(p)}{\partial p(u)} = \sum_{e \in u} (\nabla h(x))_e = \nabla h(x)^T u$ when $x = \sum_{u \in \mathcal{U}} p(u) u$. Therefore, one can use the framework for online linear optimization to optimize over $g(\cdot)$ as the losses in each round $l^{(t)} = \nabla h(x^{(t)})$ are linear over the vertex set. We give the complete description of MWU algorithm for convex minimization, with the help of a pseudocode in Algorithm 9. The constants input in the algorithm are the Lipschitz constant of $h(\cdot)$ with respect to L_1 norm, G_h; a scaling factor $\zeta = \max_{u \in \mathcal{U}} \|u\|_1$; the radius R of $\Delta_{\mathcal{U}}$ with respect to the entropic mirror map; and the desired approximation factor δ in the objective function value. Recall that the radius of $\Delta_{\mathcal{U}}$ with respect to any mirror map $\omega(\cdot)$ is defined as $R^2 = \max_{p \in \Delta_{\mathcal{U}}} \omega(p) - \min_{p \in \Delta_{\mathcal{U}}} \omega(p)$.

One can view Algorithm 9 as entropic online mirror descent over $\Delta_{\mathcal{U}}$, which is equivalent to the MWU algorithm over the set $\mathcal{U}$. We can simulate the latter efficiently by using

Algorithm 9: MWU for Convex Optimization

```
input  : $h(\cdot) : P \to \mathbb{R}$, counting oracle $M_\epsilon$, constants $G_h > 0$, $\zeta > 0$, $R > 0$, $\delta > 0$
output: $\tilde{x}$: $h(\tilde{x}) - \min_{x \in P} h(x) \leq \delta$
3  $\lambda^{(1)}(e) = 1$ for all $e \in E$; $T = \frac{2\ln(|\mathcal{U}|)\zeta^2 G_h^2}{\delta^2}$; $\eta = \frac{R}{\zeta G_h}\sqrt{\frac{2}{T}}$; $t = 1$; $\tilde{x} = M_\epsilon(\lambda^{(1)})$
4  repeat
5      $x^{(t)} = M_\epsilon(\lambda^{(t)})$
6      if $h(x^{(t)}) < h(\tilde{x})$, then $\tilde{x} = x^{(t)}$.
7      $l^{(t)} = \nabla h(x^{(t)})$
8      $\lambda^{(t+1)}(e) = \lambda^{(t)}(e)\exp(-\eta l^{(t)}(e)) \quad \forall e \in E$
9      t = t+1
   until $t + 1 \leq T$;
```

product distributions and updating corresponding multipliers λ that lie in a much smaller space.

Lemma 5.2. *The multiplicative weights update algorithm on n strategies is the same as the entropic online mirror descent on the simplex in $\mathbb{R}^n$.*

Proof. Consider the standard simplex $\Delta_n = \{x \in \mathbb{R}^n : \sum_i x_i = 1, x \geq 0\}$ and the unnormalized entropy mirror map $\omega(x) = \sum_i (x_i \ln x_i - x_i)$. It is well-known that ω is 1-strongly convex with respect to Δ_n (for a short proof, see [Nesterov, 2005]). The corresponding divergence is the KL-divergence, $D_\omega(x, y) = \sum_i x_i \ln \frac{x_i}{y_i} - \sum_i x_i + \sum_i y_i$. The ω-center of the standard simplex Δ_n is

$$p^{(1)} = \arg\min \omega(x) = \arg\min_{x \in \Delta_n} \sum_i x_i \ln x_i = \mathbf{1}_n / n, \tag{5.26}$$

where $\mathbf{1}_n$ is the vector of all ones. This corresponds to the uniform probability distribution over n strategies, which is also the starting probability distribution in the MWU algorithm. Next, the iterations of the entropic online mirror descent are (as discussed in Section 2.2.3),

$$y^{(t+1)} = (\nabla\omega)^{-1}(\nabla\omega(p^{(t)}) - \eta\nabla l^{(t)}(p^{(t)})) \tag{5.27}$$

$$= (p^{(t)}(1)e^{-\eta(\nabla l^{(t)}(p^{(t)}))_1}, \ldots, p^{(t)}(n)e^{-\eta(\nabla l^{(t)}(p^{(t)}))_n}) \tag{5.28}$$

$$p^{(t+1)} = \arg\min_{x \in \Delta_n} \sum_{i=1}^{n} x_i \ln(x_i / y_i^{(t+1)}). \tag{5.29}$$

Note that (5.28) corresponds to multiplicative updates to each strategy by exponentiating the gradient of the loss function, and (5.29) corresponds to simply re-normalizing the weights $y^{(t+1)}$ to obtain a probability distribution. Therefore, the MWU algorithm is exactly the online mirror descent with the unnormalized entropy mirror map. $\square$

We first show that the MWU algorithm as stated in Algorithm 9 converges for minimizing convex functions when the counting oracle M_ϵ is exact (i.e. $\epsilon = 0$) by simply using the analysis for the mirror descent algorithm (see Theorem 6), and the equivalence with the entropic case.

Theorem 17. *Consider* MWU *for minimizing a G_h-Lipschitz convex function $h(\cdot)$ over P, as stated in Algorithm 9. Let $\omega(x) = \sum_i (x_i \ln x_i - x_i)$, $\zeta = \max_{u\in\mathcal{U}} \|u\|_1$, $R^2 = \max_{p\in\Delta_\mathcal{U}} \omega(p) - \min_{p\in\Delta_\mathcal{U}} \omega(p)$ and $\eta = \frac{R}{\zeta G_h}\sqrt{\frac{2}{T}}$, then*

$$\min_{1\le t\le T} h(x^{(t)}) - \min_{x\in P} h(x) \le \zeta G_h R \sqrt{\frac{2}{T}}.$$

Proof. By the definition and convexity of g, we have

$$\begin{aligned}\min_{1\le t\le T} h(x^{(t)}) - \min_{x\in P} h(x) &= \min_{1\le t\le T} g(p^{(t)}) - \min_{p\in\Delta_\mathcal{U}} g(p)\\ &\le \frac{1}{T}\sum_{t=1}^{T} g(p^{(t)}) - \min_{p\in\Delta_\mathcal{U}} g(p)\\ &\le \frac{1}{T}\sum_{t=1}^{T} \nabla g(p^{(t)})(p^{(t)} - p^*), \end{aligned} \tag{5.30}$$

where p^* is a minimizer of $g(\cdot)$.

Note that $R^2 = \max_{p\in\Delta_\mathcal{U}} \omega(p) - \min_{p\in\Delta_\mathcal{U}} \omega(p) \le 0 - |\mathcal{U}| \sum_{u\in U} \frac{1}{|\mathcal{U}|} \ln(\frac{1}{|\mathcal{U}|}) \le \ln |\mathcal{U}|$. By the definition of g, we have $(\nabla g(p))_u = u^T \nabla h(x)$ for $\sum_{u\in\mathcal{U}} p(u)u = x \in P$. Thus, $\|\nabla g(p^{(i)})\|_\infty = \max_{u\in\mathcal{U}} u^T \nabla h(x^{(i)}) \le \zeta G_h = G_g$ (say). We claim that Algorithm 9 is simply the entropic mirror descent for minimizing $g(\cdot)$ over $p \in \Delta_\mathcal{U}$. We are maintaining a probability distribution $p^{(t)}$ over the vertex set $\mathcal{U}$ with the help of multipliers $\lambda^{(t)}$ in each round t. We start with a uniform probability distribution $p^{(1)}$, by setting the multipliers $\lambda^{(1)}(e) = 1$ for all $e \in E$. The losses $l^{(t)}$ in each round are $\nabla g(p)$. Thus, updating the multipliers $\lambda^{(t+1)}$ by $\lambda^{(t)}(e)\exp(-\eta l^{(t)}(e))$ $(e \in E)$ implicitly updates the probability distribution to weights proportional to $p^{(t+1)}$. The counting oracle M_ϵ helps compute the normalized probability distribution $p^{(t+1)}$ as well as the gradient corresponding to this distribution. Using Theorem 6 (from Chapter 2) and (5.30), we get the statement of the theorem:

$$\min_{1\le t\le T} h(x^{(t)}) - \min_{x\in P} h(x) \le \frac{1}{T}\sum_{t=1}^{T} \nabla g(p^{(t)})(p^{(t)} - p^*) \le RG_g\sqrt{2/T} = \zeta G_h \sqrt{\frac{2\ln|\mathcal{U}|}{T}}.$$

□

Note that the generalized counting oracle M_ϵ might be approximate, and this introduces some errors in the computation. The case of M_ϵ with $\epsilon > 0$ can be analyzed by invoking results about approximate projections in the mirror descent algorithm, however we do not reproduce those results here. One can also show better bounds on the rate of convergence of entropic mirror descent by considering the effect of the change of space from P to P_u on the convexity constants of $h(\cdot)$, which in turn affect the rate of convergence of the entropic mirror descent. We next show that if a function is β-smooth over P, then it is still smooth over Δ_u, and this can be exploited to obtain a faster rate of convergence.

Lemma 5.3. *Consider a convex function $h : P \to \mathbb{R}$ that is β-smooth with respect to the $\|\cdot\|_1$ norm, and the corresponding function $g : \Delta_{\mathcal{U}} \to \mathbb{R}$ such that $g(p) = h(x)$ when $\sum_{u\in\mathcal{U}} p(u)u = x$. Let $\zeta = \max_{u\in\mathcal{U}} \|u\|_1$. Then, g is $\zeta^2\beta$-smooth w.r.t. $\|\cdot\|_1$.*

Proof. h being β-smooth w.r.t. $\|\cdot\|_1$ means $\|\nabla h(x) - \nabla h(y)\|_\infty \le \beta\|x-y\|_1$. Let the probability distributions p and q correspond to the points x and $y \in P$ respectively. Then,

$$\begin{aligned}
\|\nabla g(p) - \nabla g(q)\|_\infty &= \|\big(u^T(\nabla h(x) - \nabla h(y))\big)_u\|_\infty \\
&= \max_{u\in\mathcal{U}} u^T(\nabla h(x) - \nabla h(y)) \\
&\le \|\nabla h(x) - \nabla h(y)\|_\infty \max_{u\in\mathcal{U}} \|u\|_1 \\
&\le \zeta\beta\|x-y\|_1 \\
&= \zeta\beta\sum_{e\in E} |\sum_{u:e\in u} p(u) - \sum_{u:e\in u} q(u)| \\
&\le \zeta\beta\sum_{e\in E}\sum_{u:e\in u} |p(u) - q(u)|
\end{aligned}$$

$$\leq \zeta^2\beta \sum_{u\in\mathcal{U}} |p(u) - q(u)| = \zeta^2\beta\|p-q\|_1.$$

Hence, g is $\zeta^2\beta$-smooth w.r.t. $\|\cdot\|_1$. $\square$

Using Lemma 5.3 and Theorem 3 (from Chapter 2), we now state the convergence rate of the MWU algorithm for β-smooth functions.

Lemma 5.4. *Consider* MWU *for minimizing a convex function $h(\cdot)$ over P, as stated in Algorithm 9. Let $h(\cdot)$ be β-smooth over P under the L_1 norm. Let $\omega(x) = \sum_i (x_i \ln x_i - x_i)$, $\zeta = \max_{u\in\mathcal{U}} \|u\|_1$, $R^2 = \max_{p\in\Delta_{\mathcal{U}}} \omega(p) - \min_{p\in\Delta_{\mathcal{U}}} \omega(p)$ and $\eta = \frac{1}{\zeta^2\beta}$, then*

$$\min_{1\leq t\leq T} h(x^{(t)}) - \min_{x\in P} h(x) \leq \frac{\zeta^2\beta R^2}{T} \leq \frac{\zeta^2\beta \ln|\mathcal{U}|}{T}.$$

Strong convexity We note that strong-convexity is not preserved when we move to the simplex of vertices. To illustrate this, consider two probability distributions p and q that correspond to the same marginal point x, i.e. $\sum_u p(u)u = x = \sum_u q(u)u$. Then, the convex function g has the same value on all points connecting the line segment joining p and q, and therefore, $g(\cdot)$ is no longer strongly-convex even though $h(\cdot)$ might be.

Interestingly, as a by-product of the MWU algorithm over the simplex of vertices, we implicitly obtain a decomposition of the approximate minimizer of $h(\cdot)$ as a product distribution. This also gives a process to obtain a decomposition of any point $x \in P$ over the vertices $\mathcal{U}$, as we state in the next corollary.

Corollary 4. *Consider an arbitrary vector $x^* \in P$, and let us use the* MWU *algorithm for minimizing $h(z) := (z - x^*)^2$ over the vertex set $\mathcal{U}$, with an exact marginal oracle M_0. Let $\zeta = \max_{u\in\mathcal{U}} \|u\|_1$. After $O(\zeta^2 \ln|\mathcal{U}|/\epsilon^2)$ iterations, the* MWU *algorithm returns an approximate minimizer $\tilde{x}$ such that $\|\tilde{x} - x^*\|_2^2 \leq \epsilon$ (or $\|\tilde{x} - x^*\|_\infty \leq O(\sqrt{\epsilon})$). Moreover, the multipliers $\tilde{\lambda}$ corresponding to $\tilde{x}$ satisfy $M_0(\tilde{\lambda}) = \tilde{x}$, thus resulting in an approximate decomposition of x^* into a product distribution.*

Proof. Let $\zeta = \max_{u\in\mathcal{U}} \|u\|_1$. We know that for the simplex of vertices, $R^2 \leq \ln|\mathcal{U}|$. Using Theorem 17, after $O(\zeta^2 \ln|\mathcal{U}|/\epsilon^2)$ iterations, the MWU algorithm returns $\tilde{x}$ such that $h(\tilde{x}) - h(x^*) \leq O(\epsilon)$, which in turn implies $\|\tilde{x} - x^*\|_2^2 \leq O(\epsilon)$. $\square$

Comparison with related work To the best of our knowledge, it was not observed before that it might make sense to do convex optimization over $0/1$ combinatorial polytopes using the MWU algorithm over the simplex of its vertices (which lies in a much larger dimension). What we point out in this chapter is that the MWU algorithm can be efficiently simulated in this large space as well, with the help of product distributions and approximate counting oracles. In Chapter 3, we considered the minimization of separable convex functions over submodular polytopes. We know that any N-dimensional simplex is submodular, and we showed that Card-Fix can be used to compute entropic projections over simplices in time $O(N \log N)$. In the case of $\Delta_{\mathcal{U}}$ however, those results are not meaningful as $|\mathcal{U}|$ is exponential in the input size of the problem. (One could still perform online mirror descent over the space of marginals, i.e. P, and use generalized projections over P). Further, note that the results of Chapter 3 apply to submodular polytopes only, however, in the current chapter we impose no such restriction.

Chapter 6

Nash-Equilibria in Two-player Games

"Nobody gets to live life backward. Look ahead, that is where your future lies."
- Ann Landers.

We have so far studied the minimization of separable convex functions over submodular polytopes, motivated by bottlenecks in projection-based first-order optimization methods in Chapter 3; parametric line searches in extended submodular polytopes, motivated by bottlenecks in INC-FIX and variants of the Frank-Wolfe method in Chapter 4; as well as approximate counting oracles over the vertex set of $0/1$ combinatorial polytopes to do online linear optimization over their vertex set and convex minimization in Chapter 5. In this chapter, we now view these results under the unified lens of computing optimal strategies (i.e. Nash-equilibria) for two-player games and compare their applicability and limitations. We also study the structure of Nash-equilibria for certain matroid games, without using any results from the previous chapters.

We consider here two-player zero-sum combinatorial games where both players play combinatorial objects, such as spanning trees, cuts, matchings, or paths in a given graph. The number of pure strategies of both players can then be exponential in a natural description of the problem.[1] For example, in a spanning tree game in which all the results of this thesis apply, pure strategies correspond to spanning trees T_1 and T_2 selected by the two players in a graph G (or two distinct graphs G_1 and G_2) and the payoff $\sum_{e\in T_1, f\in T_2} L_{ef}$ is a bilinear

[1]These are the *succinct* games, as discussed in the paper of Papadimitriou and Roughgarden on correlated equilibria [Papadimitriou and Roughgarden, 2008].

function. This allows for example to model classic network interdiction games (see for e.g., [Washburn and Wood, 1995]), design problems [Chakrabarty et al., 2006], and the interaction between algorithms for many problems such as ranking and compression as *bilinear duels* [Immorlica et al., 2011]. To formalize the games we are considering, assume that the pure strategies for player 1 (resp. player 2) correspond to the vertices u (resp. v) of a *strategy* polytope $P \subseteq \mathbb{R}^m$ (resp. $Q \subseteq \mathbb{R}^n$) and that the loss for player 1 is given by the bilinear function $u^T Lv$ where $L \in \mathbb{R}^{m \times n}$. A feature of bilinear loss functions is that the bilinearity extends to mixed strategies as well, and thus one can easily see that mixed Nash-equilibria correspond to solving the min-max problem:

$$\min_{x \in P} \max_{y \in Q} x^T Ly = \max_{y \in Q} \min_{x \in P} x^T Ly. \tag{6.1}$$

Nash-equilibria for two-player zero-sum games can be found by solving a linear program [von Neumann, 1928]. However, for succinct games in which the strategies of both players are exponential in a natural description of the game, the corresponding linear program has exponentially many variables and constraints, and as [Papadimitriou and Roughgarden, 2008] point out in their open questions section, "there are no standard techniques for linear programs that have both dimensions exponential." Under bilinear losses/payoffs however, the von Neumann linear program can be reformulated in terms of the strategy polytopes P and Q, and this reformulation can be solved using the equivalence between optimization and separation and the ellipsoid algorithm ([Grötschel et al., 1981], see also Section 6.1).

In this chapter, we first explore ways of solving efficiently the von Neumann linear program using online learning algorithms. It is well-known that if one of the players uses a (Hannan-consistent[2]) online learning algorithm and adapts his/her strategies according to the losses incurred so far (with respect to the most adversarial opponent strategy) then the average of the strategies played by the players in the process constitutes an approximate equilibrium ([Cesa-Bianchi and Lugosi, 2006], see also Lemma 6.2). The setting for online learning that we consider here is: in each round one of the players (i.e. the learner) chooses a mixed strategy $x^{(t)} \in P$; the second player (who acts as an adversary) then chooses a

[2]An online learning algorithm is called Hannan-consistent if its average regret vanishes as the number of time steps goes to infinity.

loss vector $l^{(t)} = Lv^{(t)}$ where $v^{(t)} \in Q$ and the loss incurred by the player is $x^{(t)T}l^{(t)}$. For simplicity we assume that the learner observes the full loss vector $l^{(t)}$. The goal of the learner is to minimize the regret, $R_t = \sum_{i=1}^{t} x^{(t)T}l^{(t)} - \min_{x\in P} x^T l^{(t)}$. If the learner uses an algorithm such that $\lim_{t\to\infty} \frac{R_t}{t} = 0$, then the average of the strategies played by the two players is an approximate equilibrium. In Section 6.2, we specifically compare the performance of two online learning algorithms over P: online mirror descent (using Inc-Fix for computing projections from Chapter 3) and the multiplicative weights update (using generalized approximate counting oracles, from Chapter 5) in the context of converging to approximate equilibria. In both cases, we assume that we have an (approximate) linear optimization oracle for Q, which allows to compute the (approximately) worst loss vector given a mixed strategy in P.

In Section 6.3, we combinatorially characterize the structure of symmetric Nash-equilibria (i.e. same mixed strategy is played by both the players) in a two-player game when both the players play bases of the same matroid. We give necessary and sufficient conditions for the existence of symmetric Nash-equilibria and show that they can be efficiently computed using any separable convex minimization algorithm (for e.g. using algorithm Inc-Fix from Chapter 3), without using learning.

6.1 Using the ellipsoid algorithm

In this section, we review the von Neumann linear program for a combinatorial game with strategy polytopes $P \subseteq \mathbb{R}^n$ and $Q \subseteq \mathbb{R}^m$ and a bilinear loss function. We show that this linear program has a polynomial (in n and m) vertex complexity, which in turn implies that we can use the machinery of the ellipsoid algorithm to find Nash-equilibria in polynomial time.

In a two-player zero-sum game with loss (or payoff) matrix $R \in \mathbb{R}^{M\times N}$, a mixed strategy x (resp. y) for the row player (resp. column player) trying to minimize (resp. maximize) his/her loss is an assignment $x \in \Delta_M$ (resp. $y \in \Delta_N$) where Δ_K is the simplex $\{x \in \mathbb{R}^K, \sum_{i=1}^{K} x_i = 1, x \geq 0\}$. A pair of mixed strategies (x^*, y^*) is called a Nash-equilibrium if $x^{*T}R\bar{y} \leq x^{*T}Ry^* \leq \bar{x}^T Ry^*$ for all $\bar{x} \in \Delta_M, \bar{y} \in \Delta_N$, i.e. there is no incentive for either player to switch from (x^*, y^*) given that the other player does not deviate. Similarly, a pair of strategies (x^*, y^*) is called an $\epsilon-$approximate Nash-equilibrium if $x^{*T} R\bar{y} - \epsilon \leq x^{*T} Ry^* \leq \bar{x}^T Ry^* + \epsilon$ for all $\bar{x} \in \Delta_M, \bar{y} \in \Delta_N$. Von Neumann showed that every two-player zero-sum game has a mixed Nash-equilibrium that can be found by solving the following dual pair of linear programs:

$$\begin{aligned} (LP1) : &\min \lambda & \qquad (LP2) : &\max \mu \\ &R^T x \leq \lambda e, & &Ry \geq \mu e, \\ &e^T x = 1, x \geq 0. & &e^T y = 1, y \geq 0. \end{aligned}$$

where e is a vector of all ones in the appropriate dimension.

In our two-player zero-sum combinatorial games, we let the strategies of the row player be $\mathcal{U} = \text{vert}(P)$, where $P = \{x \in \mathbb{R}^m, Ax \leq b\}$ is a polytope and $\text{vert}(P)$ is the set of vertices of P and those of the *column* player be $\mathcal{V} = \text{vert}(Q)$ where $Q = \{y \in \mathbb{R}^n, Cy \leq d\}$ is also a polytope. The numbers of pure strategies, $M = |\mathcal{U}|$, $N = |\mathcal{V}|$ will typically be exponential in m or n, and so may be the number of rows in the constraint matrices A and C. The linear programs $(LP1)$ and $(LP2)$ have thus exponentially many variables and constraints. We restrict our attention to bilinear loss functions that are represented as $R_{uv} = u^T Lv$ for some $m \times n$ matrix L.

An artifact of bilinear loss functions is that the bilinearity extends to mixed strategies as well. If $\lambda \in \Delta_{\mathcal{U}}$ and $\theta \in \Delta_{\mathcal{V}}$ are mixed strategies for the players then the expected loss is equal to $x^T Ly$ where $x = \sum_{u \in \mathcal{U}} \lambda_u u$ and $y = \sum_{v \in \mathcal{V}} \theta_v v$:

$$\mathbb{E}_{u,v}(R_{uv}) = \sum_{u \in \mathcal{U}} \sum_{v \in \mathcal{V}} \lambda_u \theta_v (u^T Lv) = (\sum_{u \in \mathcal{U}} \lambda_u u) L (\sum_{v \in \mathcal{V}} \theta_v v) = x^T Ly.$$

Thus, the loss incurred by mixed strategies only depend on the *marginals* of the distributions over the vertices of P and Q; distributions with the same marginals give the same expected loss. Therefore, the Nash-equilibrium problem for these games reduces to (6.1): $\min_{x \in P} \max_{y \in Q} x^T Ly = \max_{y \in Q} \min_{x \in P} x^T Ly$.

As an example of such a combinatorial game, consider a *spanning tree* game where the pure strategies of each player are the spanning trees of a given graph $G = (V, E)$ with m edges, and L is the $m \times m$ identity matrix. This corresponds to the game in which the row player would try to minimize the intersection of his/her spanning tree with that of the column player, whereas the column player would try to maximize the intersection. For a complete graph on n vertices, the number of pure strategies for each player is n^{n-2} by Cayley's theorem. For the graph G in Figure 6-1(a), the marginals of the unique[3] Nash-equilibrium for both players are given in 6-1(b) and (c), i.e. for the row player $p^* : p^*(e) = \begin{cases} 13/36 & e \in E(1,2,3,4,5) \setminus (1,3) \\ 3/4 & e \in E(1,6,7,8,3) \end{cases}$, and for the column player $q^* : q^*(e) = \begin{cases} 1/3 & e \in E(1,2,3,4,5) \\ 11/12 & e \in E(1,6,7,8,3) \setminus (1,3) \end{cases}$. The value of the game is $p^{*T}q^* = 4.0833$, this is also the cost of the minimum spanning tree under weights q^*, and the cost of the maximum spanning tree under weights p^*. We include more examples in Appendix B.

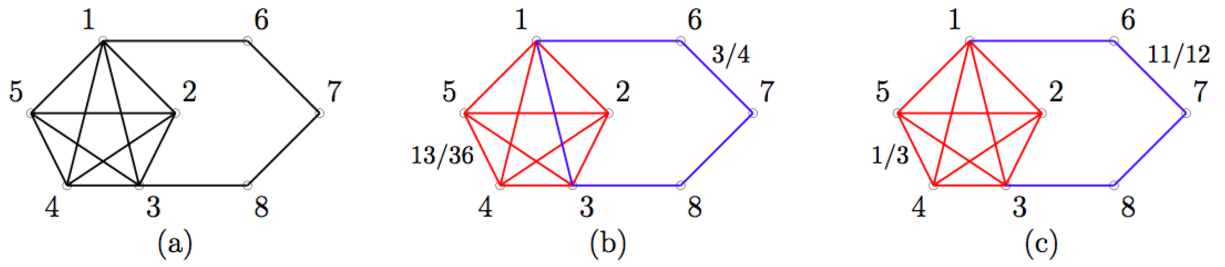


Figure 6-1: (a) $G = (V, E)$, (b) Optimal strategy for the row player minimizing the weight of the intersection of the two strategies, (c) Optimal strategy for the column player maximizing the weight of the intersection.

For combinatorial games with bilinear losses, the linear programs $(LP1)$ and $(LP2)$ can be reformulated over the space of marginals, and $(LP1)$ becomes

$$(LP1') : \min \lambda$$

$$x^T L v \leq \lambda \;\; \forall\, v \in \mathcal{V}, \tag{6.2}$$

$$x \in P \subseteq \mathbb{R}^m, \tag{6.3}$$

[3]We can show computationally that this Nash-equilibrium is unique.

and similarly for $(LP2)$: $\max\{\mu : u^T Ly \geq \mu \ \ \forall u \in \mathcal{U}, y \in Q\}$. This reformulation can be used to show that there exists a Nash-equilibrium with small (polynomial) encoding length. A polyhedron K is said to have *vertex-complexity* at most ν if there exist finite sets V, E of rational vectors such that $K = \text{conv}(V) + \text{cone}(E)$ and such that each of the vectors in V and E has encoding length at most ν. A polyhedron K is said to have *facet-complexity* at most ϕ if there exists a system of inequalities with rational coefficients that has solution set K such that the (binary) encoding length of each inequality of the system is at most ϕ. Let ν_P and ν_Q be the vertex complexities of polytopes P and Q respectively; if P and Q are 0/1 polytopes, we have $\nu_P \leq m$ and $\nu_Q \leq n$. This means that the facet complexity of P and Q are $O(m^2 \nu_P)$ and $O(n^2 \nu_Q)$ (see Lemma (6.2.4) in [Lovász et al., 1988]). Therefore the facet complexity of the polyhedron in $(LP1')$ can be seen to be $O(\max(m\langle L\rangle\nu_Q, n^2\nu_P))$, where $\langle L\rangle$ is the binary encoding length of L and the first term in the max corresponds to the inequalities (6.2) and the second to (6.3). From this, we can derive Lemma 6.1.

Lemma 6.1. *The vertex complexity of the linear program* $(LP1')$ *is* $O(m^2(m\langle L\rangle\nu_Q + n^2\nu_P))$ *where* ν_P *and* ν_Q *are the vertex complexities of* P *and* Q *and* $\langle L\rangle$ *is the binary encoding length of* L*. (If* P *and* Q *are* $0/1$ *polytopes then* $\nu_P \leq m$ *and* $\nu_Q \leq n$*.)*

This means that our polytope defining $(LP1')$ is well-described (à la Grötschel et al.). We can thus use the machinery of the ellipsoid algorithm [Grötschel et al., 1981] to find a Nash-equilibrium in polynomial time for these combinatorial games, provided we can optimize (or separate) over P and Q. Indeed, by the ellipsoid algorithm, we have the equivalence between strong separation and strong optimization for well-described polyhedra. The strong separation over (6.2) reduces to strong optimization over Q, while a strong separation algorithm over (6.3), i.e. over P, can be obtained from a strong separation over P by the ellipsoid algorithm.

We should also point out at this point that, if the polyhedra P and Q admit a compact extended formulation then $(LP1')$ can also be reformulated in a compact way (and solved using interior point methods, for example). A compact extended formulation for a polyhedron $P \subseteq \mathbb{R}^d$ is a polytope with polynomially many (in d) facets in a higher dimensional space that projects onto P. This allows to give a compact extended formulation for $(LP1')$

for the spanning tree game as a compact formulation is known for the spanning tree polytope [Martin, 1991] (and any other game where the two strategy polytopes can be described using polynomial number of inequalities). However, this would not work for a corresponding matching game since the extension complexity for the matching polytope is exponential [Rothvoß, 2014].

6.2 Bregman projections v/s approximate counting

As we mentioned in the introduction of this chapter, online learning algorithms can be used to find Nash-equilibria by simulating an iterative learning process, where one player acts as a learner and the other acts as an adversary to generate appropriate losses in each round[4]. The average strategy of the two players converges to approximate Nash-equilibria. We refer the reader to a survey by Arora, Hazan and Kale [Arora et al., 2012] for more details, and state a lemma (with a short proof) relating the regret of learning algorithms to the guarantee obtained in terms of approximate Nash-equilibria.

Lemma 6.2. *Consider a combinatorial game with strategy polytopes $P \subseteq \mathbb{R}^m$ and $Q \subseteq \mathbb{R}^n$, and let the loss function for the row player be given by $loss(x, y) = x^T Ly$ for $x \in P, y \in Q$. Suppose we simulate an online algorithm* A *such that in each round t the row player chooses decisions from $x^{(t)} \in P$, the column player reveals an adversarial loss vector $v^{(t)}$ such that $x^{(t)T} Lv^{(t)} \geq \max_{y \in Q} x^{(t)T} Ly - \delta$ and the row player subsequently incurs loss $x^{(t)T} Lv^{(t)}$ for round t. If the regret of the learner after T rounds goes down as $f(T)$, that is,*

$$R_T(A) = \sum_{i=1}^{T} x^{(i)T} Lv^{(i)} - \min_{x \in P} \sum_{i=1}^{t} x^T Lv^{(i)} \leq f(T) \tag{6.4}$$

then $(\frac{1}{T}\sum_{i=1}^{T} x^{(i)}, \frac{1}{T}\sum_{i=1}^{T} v^{(i)})$ is an $O(\frac{f(T)}{T} + \delta)$-approximate Nash-equilibrium for the game.

Proof. Let $\bar{x} = \frac{1}{T}\sum_{i=1}^{T} x^{(i)}$ and $\bar{v} = \frac{1}{T}\sum_{i=1}^{T} v^{(i)}$. By the von Neumann minimax theorem, we

[4] Another way to converge to approximate equilibria by letting both the players act as learners and observe the losses due to each other strategies in each round. The average of the strategies in this case also converges to approximate Nash equilibria.

know that the value of the game is $\lambda^* = \min_x \max_y x^T Ly = \max_y \min_x x^T Ly$. This gives,

$$\begin{aligned}
\min_x \max_y x^T Ly = \lambda^* &\le \max_y \bar{x}^T Ly = \max_y \frac{1}{T}\sum_{i=1}^{T} x^{(i)} Ly \le \frac{1}{T}\sum_{i=1}^{T} \max_y x^{(i)T} Ly && (6.5)\\
&\le \frac{1}{T}(\sum_{i=1}^{T} x^{(i)T} Lv^{(i)} + \delta) && (6.6)\\
&\le \min_{x\in P} \frac{1}{T}\sum_{i=1}^{T} x^T Lv^{(i)} + \frac{f(T)}{T} + \delta && (6.7)\\
&= \min_{x\in P} x^T L \frac{1}{T}\sum_{i=1}^{T} v^{(i)} + \frac{f(T)}{T} + \delta = \min_{x\in P} x^T L\bar{v} + \frac{f(T)}{T} + \delta\\
&\le \max_{y\in Q}\min_{x\in P} x^T Ly + \frac{f(T)}{T} + \delta = \lambda^* + \frac{f(T)}{T} + \delta.
\end{aligned}$$

where the last inequality in (6.5) follows from the convexity $\max_y x^T Ly$ in x, (6.6) follows from the error in the adversarial loss vector, and (6.7) follows from the given regret equation (6.4). Thus, we get $\bar{x}^T L\bar{v} \le \max_{y\in Q} \bar{x}^T Ly \le \lambda^* + \frac{f(T)}{T} + \delta$, and $\bar{x}^T L\bar{v} \ge \min_{x\in P} x^T L\bar{v} \ge \lambda^* - \frac{f(T)}{T} - \delta$. Hence, $(\bar{x}, \bar{v})$ is a $\left(\frac{2f(T)}{T} + 2\delta\right)$-approximate Nash-equilibrium for the game. □

We consider here two online learning algorithms for the purposes of finding Nash-equilibria: the online mirror descent and the multiplicative weights update method, and refer the reader to Sections 2.2.3 and Section 5.1 for background on these respectively. The regret of the online mirror descent scales as $O(RG/\sqrt{T})$ with the choice of a 1-strongly-convex mirror map $\omega(\cdot)$ (with respect to $\|\cdot\|$) such that the radius of the polytope P with respect to $\omega(\cdot)$ is R and the loss functions in each round are G-Lipschitz with respect to $\|\cdot\|$. Therefore, to converge to an ϵ-approximate Nash-equilibrium (assuming the worst-case loss vectors can be computed exactly) online mirror descent requires $O(R^2G^2/\epsilon^2)$ rounds of learning, each with the computation of a Bregman projection. On the other hand, the regret of the MWU algorithm over a decision set $\mathcal{U}$ scales as $O(\sqrt{\frac{\ln|\mathcal{U}|}{T}})$ for losses normalized to $[-1,1]$. Let $F = \max_{u\in P, v\in Q} |u^T Lv|$. Therefore, to converge to an ϵ-approximate Nash-equilibrium, the MWU algorithm requires $O(\ln|\mathcal{U}|F^2/\epsilon^2)$ rounds of learning, each with the computation of the (even if approximate) marginal strategy corresponding to the product distribution. The approximate marginal strategy can be used to compute the maximally adversarial loss vectors. We give the complete description of the MWU for computing Nash-equilibria in Algorithm 10. To converge to an ϵ-approximate Nash-equilibrium, the generalized approximate counting oracle can have an error of at most ϵ/F' for $F' = \max_{v\in Q} \|Lv\|_1$, as we show in the following lemma:

Algorithm 10: The MWU algorithm for computing Nash-equilibria

```
input : $\mathrm{M}_{\epsilon_1}\colon \mathbb{R}^m \to \mathbb{R}^m$, $L \in \mathbb{R}^{n\times m}$, $\epsilon > 0$.
output: $O(\epsilon + F\epsilon_1)$-approximate Nash equilibrium $(\bar{x}, \bar{y})$
$\lambda^{(1)} = \mathbf{1}, t = 1, F = \max_{x\in P, y\in Q} x^T Ly, \eta = \sqrt{\ln|\mathcal{U}|/T}$;
repeat
    $\tilde{x}^{(t)} = \mathrm{M}_{\epsilon_1}(\lambda^{(t)})$;
    $\tilde{v}^{(t)} \in \arg\max_{y\in Q} \tilde{x}^{(t)} Ly$;
    $\lambda^{(t+1)}(e) = \lambda^{(t)}(e) * \exp(-\eta L\tilde{v}^{(t)}(e)/F)\ \ \forall e \in E$;
    $t \leftarrow t+1$
until $t < \frac{F^2 \ln|U|}{\epsilon^2}$;
$(\bar{x}, \bar{y}) = (\frac{1}{t-1}\sum_{i=1}^{t-1} \tilde{x}^{(i)}, \frac{1}{t-1}\sum_{i=1}^{t-1} \tilde{v}^{(i)})$
```

Lemma 6.3. *Given a generalized approximate counting oracle M_{ϵ_1} that computes a marginal vector $\tilde{x}$ corresponding to multipliers λ with error $||M_0(\lambda) - \tilde{x}||_\infty \leq \epsilon_1$, a linear optimization oracle to compute the maximally adversarial strategy $\tilde{v}^{(i)} \in \arg\max_{v\in Q} x^T Lv$, $F = \max_{u\in P, v\in Q} |u^T Lv|$, $F' = \max_{v\in Q} \|Lv\|_1$, the* MWU *algorithm gives an $O(\epsilon + F'\epsilon_1)$-approximate Nash-equilibrium $(\frac{1}{t}\sum_{i=1}^{t} \tilde{x}^{(i)}, \frac{1}{t}\sum_{i=1}^{t} \tilde{v}^{(i)})$ in $O(\frac{F^2 \ln(|\mathcal{U}|)}{\epsilon^2})$ rounds and time polynomial in (n, m).*

Proof. Let the multipliers in each round t be $\lambda^{(t)}$, the corresponding true and approximate marginal points in round t be $x^{(t)}$ and $\tilde{x}^{(t)}$ respectively, such that $||x^{(t)} - \tilde{x}^{(t)}||_\infty \leq \epsilon_1$. Using the linear optimization oracle, we can compute adversarial loss vectors, $\tilde{v}^{(t)}$ in each round such that $\tilde{v}^{(t)} \in \arg\max_{y\in Q} \tilde{x}^{(t)} Ly$.

Even though we cannot compute $x^{(t)}$ exactly, we do maintain the corresponding $\lambda^{(t)}$s that correspond to these true marginals. Let us analyze first the MWU algorithm corresponding to the loss vectors $\tilde{v}^{(t)}$ using product distributions (that can be done efficiently) corresponding to the true marginal points. Using Corollary 2, we get the following regret bound with respect to the true marginals corresponding to $t = \frac{F^2 \ln(|\mathcal{U}|)}{\epsilon^2}$ rounds:

$$\frac{1}{t}\sum_{i=1}^{t} x^{(i)T} L\tilde{v}^{(i)} \leq \frac{1}{t}\sum_{i=1}^{t} \mathbf{x}^T L\tilde{v}^{(i)} + O(\epsilon) \tag{6.8}$$

We do not have the value for $x^{(i)}$ for $i = 1, \ldots, t$, but only estimates $\tilde{x}^{(i)}$ for $i = 1, \ldots, t$ such that $||\tilde{x}^{(i)} - x^{(i)}||_\infty \leq \epsilon_1$. Since the losses we consider are bilinear, we can bound the loss of the estimated point in each iteration i as follows:

$$|\tilde{x}^{(i)T} L \tilde{v}^{(i)} - x^{(i)T} L \tilde{v}^{(i)}| \leq \|\tilde{x}^{(i)} - x^{(i)}\|_\infty \|L\tilde{v}^{(i)}\|_1 \leq F' \epsilon_1. \tag{6.9}$$

Thus using (6.8), we get

$$\frac{1}{t}\sum_{i=1}^{t} \tilde{x}^{(i)T} L \tilde{v}^{(i)} \leq \frac{1}{t}\sum_{i=1}^{t} x^T L \tilde{v}^{(i)} + O(\epsilon + F'\epsilon_1) \ \ \forall x \in P. \tag{6.10}$$

Now, considering that we played points $\tilde{x}^{(i)}$ for each round i, and suffered maximally adversarial losses $\tilde{v}^{(i)}$, we have shown that the MWU algorithm achieves $O(\epsilon + F'\epsilon_1)$ regret on an average. Thus, using Lemma 6.2 we have that $(\frac{1}{t}\sum_{i=1}^{t} \tilde{x}^{(i)}, \frac{1}{t}\sum_{i=1}^{t} \tilde{v}^{(i)})$ is an $O(\epsilon + F'\epsilon_1)$-approximate Nash-equilibrium. $\square$

Both the learning approaches, online mirror descent and the multiplicative weights update, have different applicability and limitations. We know how to efficiently perform the Bregman projection only for polymatroids, and not for bipartite matchings for which the MWU algorithm with product distributions can be used. On the other hand, there exist matroids for which any generalized approximate counting algorithm requires an exponential number of calls to an independence oracle [Azar et al., 1994], while an independence oracle is all what we need to make the Bregman projection efficient in the online mirror descent approach. Further, the running time of the online mirror descent is dependent on the choice of the mirror map, as well as the choice of the norm. Our projection algorithm, Inc-Fix can be used to compute projections whenever the corresponding Bregman divergence is separable and one of the strategy polytopes of the game is submodular. The applicability of the online linear optimization framework for the MWU algorithm is crucially dependent on the existence of efficient (approximate) generalized counting oracles. [5]

We next consider a combinatorial games with the strategy polytope $P \subseteq \mathbb{R}^n$ being the

[5]One can also potentially use the MWU algorithm to minimize convex functions, and use that to approximately compute Bregman projections for projection-based first-order optimization methods. However, we do not explore this connection in this thesis.

spanning tree polytope (the number of edges in the underlying graph are assumed to be n, let the number of vertices be ν) and $Q \subseteq \mathbb{R}^m$ being an arbitrary $0/1$ polytope such that there exists a linear optimization oracle over Q. Consider a general loss matrix $L \in \mathbb{R}^{n\times m}$ with $\|L\|_\infty \leq 1$ (i.e. each entry of L is in $[-1, 1]$). We compare the running times of online mirror descent and the MWU algorithm in different settings. Recall that the online mirror descent algorithm starts with $x^{(0)}$ being the $\omega-$center of the combinatorial polytope, that can be obtained by projecting a vector of ones.

(i) Entropic mirror descent over P: The radius of the spanning tree polytope (we consider the one characterized by Edmonds) is $R^2 = \max_{x\in P} \omega(x) - \min_{x\in P} \omega(x)$, for $\omega(x) = \sum_e (x_e \ln x_e - x_e)$. Note that since $\omega(\cdot)$ is a convex function, the maximum of $\omega(\cdot)$ is obtained at the vertices. However, the vertices are $0/1$ vectors, therefore $\max_{x\in P} \omega(x) = -(\nu - 1)$. The minimum entropy point in the spanning tree polytope should be as uniform as possible. We can lower bound the entropy of any $x \in P$ by the entropy of $\frac{\nu-1}{n}\mathbf{1}$ (vector obtained by raising each edge of the graph to $(\nu - 1)/n$ such that the rank constraint on the ground set is satisfied): $\min_{x\in P} \omega(x) \geq n\frac{(\nu-1)}{n} \ln(\frac{\nu-1}{n}) - (\nu - 1)$. Therefore, $R^2 \leq \nu \ln \nu$. Next, we need to bound the gradient of the loss functions in the dual norm, i.e. $G = \|Lv\|_\infty$ for all $v \in Q$. Since $\|L\|_\infty \leq 1$, we can bound $G \leq \max_{v\in Q} \|v\|_1 = \bar{F}$ (say). The entropic mirror descent algorithm requires $O(R^2G^2/\epsilon^2)$ rounds of learning to converge to ϵ-approximate Nash-equilibria. Each round requires the computation of a Bregman projection. For the spanning tree polytopes, one can use $O(\nu)$ maximum flow computations (using Corollary 51.3a. from [Schrijver, 2003] and references therein) for finding the most violated submodular constraint (i.e. submodular function minimization) in $O(n^2\nu)$ time (using Orlin's $O(n\nu)$ algorithm for computing the maximum flows [Orlin, 2013]). In each projection, the Inc-Fix requires $O(\nu)$ minimizations (instead of $O(n)$ submodular function minimizations) as the chain of tight sets can only be $O(\nu)$ long. Therefore, for each projection the worst-case running time of Inc-Fix is $O(n^2\nu^2)$. Thus, the overall running time of the entropic mirror descent algorithm is $O(n^2\nu^3\bar{F}^2 \ln(\nu)/\epsilon^2)$.

(ii) Gradient descent over P (i.e. mirror descent with the squared L_2 norm and the Euclidean mirror map): Under the squared Euclidean distance, $R^2 = \max_{x \in P} \frac{1}{2}\|x\|_2^2 - \min_{x \in P} \frac{1}{2}\|x\|^2 \leq \frac{1}{2}(\nu - 1)$ as the maximum of the convex function is attained at a vertex. Using the squared Euclidean distance (as opposed to the entropic mirror map) even though we reduce the radius R^2, the Lipschitz constant might be greater with respect to the L_2-norm (as opposed to the L_1-norm). In this example, the loss functions are such that $G = \|\nabla l^{(i)}\|_2 = \|Lv^{(i)}\|_2 \leq \bar{F}\sqrt{n}$ and therefore, the online mirror descent algorithm converges to an ϵ-approximate strategy in $O(\nu \bar{F}^2 n/\epsilon^2)$ rounds of learning. Overall running time is $O(n^3 \nu^3 \bar{F}^2/\epsilon^2)$ by accounting for the time to compute projections over the spanning tree polytope.

(iii) Multiplicative weights update over $\Delta_{\mathcal{U}}$. In this case, we know that the radius $R^2 \leq \ln|\mathcal{U}| = O(\nu \ln \nu)$ in the case of the spanning tree polytope. Further, $G_h = \|Lv^{(i)}\|_\infty = \bar{F}$ and thus the Lipschitz constant in the space of the vertex set is $G_g \leq \max_{u \in \mathcal{U}} \|u\|_1 G_h = O(\nu \bar{F})$. To compute projections onto the $\Delta_{\mathcal{U}}$, we use an approximate counting oracle from [Koutis et al., 2010] that has worst-case running time $\tilde{O}(n^2)$. Therefore, using Theorem 17 the worst-case running time is $\tilde{O}(n^2 R^2 G_g^2/\epsilon^2) = \tilde{O}(n^2 \nu^3 \bar{F}^2 \ln(\nu)/\epsilon^2)$. One can also compute the worst-case running time to achieve an $O(\epsilon)$ approximate Nash-equilibrium by computing the scale factor: $F = \max_{x \in P, y \in Q} x^T L y = O(\nu \bar{F})$, and using the form $O(F^2 \ln(\mathcal{U})/\epsilon^2)$ from Lemma 6.3 which gives the same time complexity.

It is interesting to note that even though the radius of P under the entropic mirror map is larger than the radius under the Euclidean mirror map, the running time of the online mirror descent under the KL-divergence is better than the running time of gradient descent over P due to the choice of the norm. In spite of the fact that the MWU algorithm is operating in an exponential space with the help of product distributions, it achieves the same running time as the entropic mirror descent on the marginal space. We would also like to note that saddle point methods like saddle point mirror prox [Nemirovski, 2004] and optimistic mirror descent [Rakhlin and Sridharan, 2013] can be used for computing Nash-equilibria whenever projections and/or approximate counting can be done efficiently on both the strategy polytopes. This results in a better dependence on ϵ for the running time to converge to ϵ-approximate equilibria ($O(1/\epsilon)$ instead of $O(1/\epsilon^2)$), however we do not explore these results in this thesis. There has also been some recent work in developing a variant of the Frank-Wolfe algorithm for solving saddle-point problems [Gidel et al., 2016] that could potentially benefit from the line searches we explored in Chapter 4.

6.3 Combinatorial Structure of Nash-Equilibria

We now characterize the combinatorial structure of Nash-equilibria in matroid games that can be exploited to computationally find these without using learning algorithms. We show that if certain (symmetric) Nash-equilibria exist, they coincide with the solutions of $\min_{x \in B(M)} \sum_{e \in E} x_e^2/w_e$ for some positive weight vector $w \in \mathbb{R}^E_{>0}$. Since this separable convex function can be minimized using the INC-FIX algorithm, these results provide an alternate approach for finding Nash-equilibria. We refer the reader to [Schrijver, 2003] and [Oxley, 2006] for background on matroids.

We assume in this section that the strategy polytopes of both the players are the same. We study the structure of symmetric Nash-equilibria that are a set of optimal strategies such that both players play the exact same mixed strategy at equilibrium. We first give necessary and sufficient conditions for a symmetric Nash-equilibrium to exist in case of matroid games.

Theorem 18. *Consider a two-player zero-sum combinatorial game with respect to a matroid $M = (E, \mathcal{I})$ with an associated rank function $r : E \to \mathbb{R}_+$. Let L be the loss matrix for the row player such that it is symmetric, i.e. $L^T = L$. Let $x \in B(M) = \{x \in \mathbb{R}^{\mathrm{E}}_+ : x(S) \leq r(S) \; \forall \; S \subseteq E, x(E) = r(E)\}$. Suppose x partitions the elements of the ground set into $\{P_1, P_2, \ldots P_k\}$ such that $(Lx)(e) = c_i \; \forall e \in P_i$ and $c_1 < c_2 \ldots < c_k$. Then, the following are equivalent.*

(i). (x, x) is a symmetric Nash-equilibrium,

(ii). All bases of matroid M have the same cost with respect to weights Lx,

(iii). For all bases B of M, $|B \cap P_i| = r(P_i)$,

(iv). $x(P_i) = r(P_i)$ for all $i \in \{1, \ldots, k\}$,

(v). For all circuits C of M, $\exists i : C \subseteq P_i$.

Proof. Case (i) $\Leftrightarrow$ (ii). Assume first that (x, x) is a symmetric Nash-equilibrium. Then, the value of the game is $\max_{z \in B(M)} x^T Lz = \min_{z \in B(M)} z^T Lx = \min_{z \in B(M)} x^T L^T z$ which is in turn equal to $\min_{z \in B(M)} x^T Lz$ as $L^T = L$. This implies that every base of the matroid has the same cost under the weights Lx.
Conversely, if every base has the same cost with respect to weights Lx, then x belongs to both $\arg\max_{y \in B(M)} x^T Ly$ and $\arg\min_{y \in B(M)} x^T Ly$. Since no player has an incentive to deviate, this implies that (x, x) is a Nash-equilibrium.

Case (ii) $\Leftrightarrow$ (iii). Assume (ii) holds. Suppose there exists a base B such that $|B \cap P_i| < r(P_i)$ for some i. We know that there exists a base B' such that $|B' \cap P_i| = r(P_i)$. Since $B \cap P_i, B' \cap P_i \in \mathcal{I}$ and $|B' \cap P_i| > |B \cap P_i|$, $\exists e \in (B' \setminus B) \cap P_i$ such that $(B \cap P_i) + e \in \mathcal{I}$. Since $(B \cap P_i) + e \in \mathcal{I}$ and B is a base, $\exists f \in B \setminus P_i$ such that $B + e - f \in \mathcal{I}$. This gives a base of a different cost as e and f are in different members of the partition. Hence, we reach a contradiction. Thus, (ii) implies (iii).
Conversely, assume (iii) holds. Note that the cost of a base B is $c(B) = \sum_{i=1}^k c_i |P_i \cap B|$. Thus, (iii) implies that every base has the same cost $\sum_{i=1}^k c_i r(P_i)$.

Case (iii) $\Leftrightarrow$ (iv). Assume (iii) holds. Since $x \in B(M)$, x is a convex combination of the bases of the matroid, i.e. $x = \sum_B {}_B \chi(B)$ where $\chi(B)$ denotes the characteristic vector for the base B. Thus, (iii) implies that $x(P_i) = \sum_B \lambda_B |B \cap P_i| = \sum_B \lambda_B r(P_i) = r(P_i)$ for all $i \in \{1, \ldots, k\}$.
Conversely assume (iv) and consider any base B of the matroid. Then, $r(E) = |B| = \sum_{i=1}^k |B \cap P_i| \leq^{(1)} \sum_{i=1}^k r(P_i) \overset{(2)}{=} \sum_{i=1}^k x(P_i) = x(E) = r(E)$, where (1) follows from rank inequality and (2) follows from (iv) for each P_i. Thus, equality holds in (1) and we get that for each base B, $|B \cap P_i| = r(P_i)$.

Case (iii) $\Leftrightarrow$ (v). Assume (iii) and let C be a circuit. Let $e \in C$ and B be a base that contains $C - e$. Hence, the unique circuit in $B + e$ is C. Thus, for any element $f \in C - e$, $B - e + f \in \mathcal{I}$. Hence, (iii) implies that all the elements of $C - e$ must lie in the same member of the partition as e does. Hence, $\exists i : C \subseteq P_i$.

Conversely, assume (v). Consider any two bases B and B' such that $B \setminus B' = \{e\}$ and $B' - B = \{f\}$ for some $e, f \in E$. Let C be the unique circuit in $B' + e$ and hence $f \in C$. It follows from (v) that e, f are in the same member of the partition, and hence $|B \cap P_i| = |B' \cap P_i|$ for all $i \in \{1, \ldots, k\}$. Since we know there exists a base B_i such that $|B_i \cap P_i| = r(P_i)$ for each i, hence all bases must have the same intersection with each P_i and (iii) follows. $\square$

Corollary 5. *Consider a game where each player plays a base of the graphic matroid $M(G)$ on a graph G, and the loss matrix of the row player is the identity matrix $I \in \mathbb{R}^{\mathrm{E} \times \mathrm{E}}$. Then there exists a symmetric Nash-equilibrium* if and only if *every block of G is uniformly dense.*

Proof. Since the loss matrix is the identity matrix, $x(e) = c_i$ for all $e \in P_i$. Theorem 18 (v) implies that each P_i is a union of blocks of the graph. Further, as $x(P_i) = r(P_i) = c_i|P_i|$, each P_i (and hence each block contained in P_i) is uniformly dense. $\square$

Corollary 6. *Given any point $x \in \mathbb{R}^{\mathbb{E}}, x > 0$ in the base polytope of a matroid $M = (E, \mathcal{I})$, one can construct a matroid game for which (x, x) is the symmetric Nash-equilibrium.*

Proof. Let the loss matrix L be defined as $L_{e,e} = 1/x_e$ for $e \in E$ and 0 otherwise. Then, $Lx(e) = 1$ for all $e \in E$. Thus, all the bases have the same cost under Lx. It follows from Theorem 18 that (x, x) is a symmetric Nash-equilibrium of this game. $\square$

Uniqueness: Consider a symmetric loss matrix L such that $L_{e,f} = 1$ for all $e, f \in E$. Note that any feasible point in the base polytope $B(M)$ forms a symmetric Nash-equilibrium. Hence, we need a stronger condition for the symmetric Nash-equilibria to be unique. We note that for positive and negative-definite loss matrices, symmetric Nash-equilibria are unique, if they exist.

Theorem 19. *Consider the game with respect to a matroid $M = (E, \mathcal{I})$ with an associated rank function $r : E \to \mathbb{R}_+$. Let L be the loss matrix for the row player such that it is positive-definite, i.e. $x^T L x > 0$ for all $x \neq 0$. Then, if there exists a symmetric Nash-equilibrium of the game, it is unique.*

Proof. Suppose (x, x) and (y, y) are two symmetric Nash-equilibria such that $x \neq y$, then the value of the game is $x^T L x = y^T L y$. Then, $x^T L z \leq x^T L x \leq z^T L x \; \forall z \in B(M)$ implying that $x^T Lz \leq x^T Lx \leq x^T L^T z$ $\forall z \in B(M)$. Since L is symmetric, we get $x^T Lx = x^T Lz$ $\forall z \in B(M)$. Similarly, we get $y^T Ly = y^T Lz$ $\forall z \in B(M)$. Consider $z = \frac{x+y}{2}$. Then, $z^T Lz = \frac{(x+y)}{2}^T Lz = \frac{x}{2}^T Lz + \frac{y}{2}^T Lz = \frac{x}{2}^T Lx + \frac{y}{2}^T Ly$. This contradicts the strict convexity of the quadratic form of $x^T Lx$. Hence, $x = y$ and there exists a unique symmetric Nash-equilibrium. $\square$

Lexicographic optimality: We finally note that symmetric Nash-equilibria are closely related to the concept of being lexicographically optimal as studied by Fujishige in 1980 [Fujishige, 1980]. For a matroid $M = (E, \mathcal{I})$, $x \in B(M)$ is called lexicographically optimal with respect to a positive weight vector w if the $|E|$-tuple of numbers $x(e)/w(e)$ $(e \in E)$ arranged in the order of increasing magnitude is lexicographically maximum among all $|E|$-tuples of numbers $y(e)/w(e)$ $(e \in E)$ arranged in the same manner for all $y \in B(M)$. We evoke the following theorem from [Fujishige, 1980].

Theorem 20. *Let $x \in B(M)$ and w be a positive weight vector. Define $c(e) = x(e)/w(e)$ $(e \in E)$ and let the distinct numbers of $c(e)$ $(e \in E)$ be given by $c_1 < c_2 < \ldots < c_p$. Furthermore, define $S_i \subseteq E$ $(i = 1, 2, \ldots, p)$ by*

$$S_i = \{e | e \in E, c(e) \leq c_i\} \quad (i = 1, 2, \ldots, p).$$

Then the following are equivalent:

(i) x is the unique lexicographically optimal point in $B(M)$ with respect to the weight vector w;

(ii) $x(S_i) = r(S_i)$ $(i = 1, 2, \ldots, p)$.

The following corollary gives an algorithm for computing symmetric Nash-equilibria for matroid games under positive diagonal loss matrices.

Corollary 7. *Consider a matroid game with a diagonal loss matrix L such that $L_{e,e} > 0$ for all $e \in E$. If there exists a symmetric Nash-equilibrium for this game, then it is the unique lexicographically optimal point in $B(M)$ with respect to the weights $1/L_{e,e}$ $(e \in E)$.*

The proof follows from observing the partition of edges with respect to the weight vector Lx, and proving that symmetric Nash-equilibria satisfy the sufficient conditions for being a lexicographically optimal base. Lexicographically optimal bases can be computed efficiently using [Fujishige, 1980], [Nagano, 2007b], or by a single projection using the Inc-fix algorithm from Chapter 3. Hence, one could compute the lexicographically optimal base x for a weight vector defined as $w(e) = 1/L_{e,e}$ $(e \in E)$ for a positive diagonal loss matrix L, and check if that is a symmetric Nash-equilibrium. If it is, then it is also the unique symmetric Nash-equilibrium and if it is not then there cannot be any other symmetric Nash-equilibrium.

Chapter 7

Conclusions

"The best journeys answer questions that in the beginning you didn't even think to ask."
- 180 Degrees South.

We conclude this thesis with a summary of our contributions and a discussion of future research directions.

7.1 Summary

In this thesis, we considered three fundamental questions over combinatorial polytopes motivated by bottlenecks in various algorithms in online and convex optimization. We first developed in Chapter 3 an algorithm, Inc-Fix, for separable convex minimization over submodular base polytopes that increases the value of elements in the gradient space in a greedy manner. We show that the intermediate iterates can be rounded to the base polytope, thereby allowing the algorithm to allow for early termination. Inc-Fix requires access to only a submodular function evaluation oracle, and we show a worst-case running time of $O(n)$ submodular function minimizations or $O(n)$ parametric submodular function minimizations. However, we can obtain significantly faster running times ($O(n(\log n + d))$) when the submodular function is cardinality-based, with d distinct values ($d \leq n$). Next, in Chapter 4, we consider the line search problem in extended submodular polyhedra that computes the maximum feasible movement in a given direction starting with a point inside the polytope. We showed that the discrete Newton's algorithm converges in at most $O(n^2)$ iterations, where each iterations requires a submodular function minimization. The analysis required results on length of sequences of ring families as well as on the length of sequence of sets such that the submodular function value on these increases geometrically. In Chapter 5, we developed a general framework for simulating the multiplicative weights update algorithm over the vertex set of combinatorial polytopes, for online linear optimization. We showed that efficient approximate counting oracles can be used to get a polynomial worst-case running time, and that this framework can also be used to minimize convex functions over the space of the elements of the ground set. These results can be viewed as an extended formulation for convex optimization, and as a by product, we show that any point in a polytope can be decomposed efficiently into an approximate product distribution over the vertex set with the help of generalized approximate counting oracles. Finally, we viewed these results in the context of finding Nash-equilibria in two-player zero-sum games in Chapter 6 and compared their applicability and limitations. We augmented this comparison by proving structural results about symmetric Nash-equilibria for two-player matroid games with diagonal loss matrices, which can be in turn used to find (if they exist) these equilibria by minimizing a single separable convex function.

We believe that the research presented in this thesis opens up a lot of interesting research questions, and we survey some of these below, with respect to each chapter.

7.2 Open Problems

We present the open problems in the context of the techniques used and theorems we prove from different chapters.

7.2.1 Separable convex minimization

The Inc-Fix algorithm starts with $x^{(0)} = 0$ or $x^{(0)} = (\nabla h)^{-1}(c\mathbf{1})$ such that $x^{(0)} \in P(f)$. Our first open problem is whether this condition can be relaxed.

Open Problem 1. *Can one start* Inc-Fix *with an arbitrary point in the submodular polytope?*

The answer to the above question can potentially reduce the worst-case running time for projections, especially when used within an online or convex optimization method like online mirror descent. These methods maintain an iterate that is already close to the required projection and ideally one should be able to make use of this information.

Next, for implementing the Inc-Fix algorithm, we showed that $O(1)$ parametric submodular function minimizations can be used for computing a single increase in the gradient space. However, it would be interesting if one can compute all the gradient increases in a single parametric SFM.

Open Problem 2. *How can one perform* Inc-Fix *in a single parametric submodular function minimization?*

One approach for this might be to solve the line search problem in a single parametric SFM while always maintaining a feasible certificate in the submodular polytope. What we can do currently is to use the discrete Newton's method to find most violated constraints iteratively while converging to the maximum feasible movement along the line.

In Chapter 3, we only looked at separable convex minimization over submodular polyhedra. Separability helped us increase the value of elements while only having a local effect on the corresponding gradients. The natural question is whether we can minimize non-separable convex functions.

Open Problem 3. *How can one minimize arbitrary convex functions over the submodular polytopes exactly, assuming infinite precision arithmetic?*

Recall that for minimizing a convex function $h(x)$ over $x \in P$, one must construct a vector x^* such that the first-order approximation of h at x^* is minimized by x^* itself, i.e. $x^* \in \arg\min_{x \in P} \nabla h(x^*)^T x$. This is independent of the combinatorial set being submodular, and that leads to fourth open problem in this chapter.

Open Problem 4. *Develop an algorithm to minimize (separable) convex functions over other combinatorial polytopes where the linear optimization methods are well-understood, for instance, for the bipartite matching polytope.*

The minimum cost bipartite matching problem on a graph with n vertices can be solved using the well-known Hungarian algorithm that runs in $O(n^3)$ time [Kuhn, 1955]. To minimize convex functions over the bipartite matching polytope one can either use the machinery of general convex optimization, or the MWU algorithm from Chapter 5 (that uses an expensive approximate counting oracle). Bipartite matchings can also be represented as doubly stochastic matrices[1]. Bregman projections with respect to the entropic mirror map can then be interpreted as Sinkhorn balancing, which involves iteratively normalizing rows and columns of the matrix until convergence [Helmbold and Warmuth, 2009]. It would be interesting to develop a (faster, exact) combinatorial algorithm for minimizing convex functions over the bipartite matching polytope.

7.2.2 Parametric line search

In Chapter 4, we showed an $O(n^2)$ bound on the number of iterations of the discrete Newton's algorithm for the problem of finding $\max \delta : \min_S f(S) - \delta a(S) \geq 0$ for an arbitrary direction $a \in \mathbb{R}^n$. Even though we showed that certain parts of our analysis were tight, we do not know whether this bound is tight.

Open Problem 5. *What is the number of breakpoints of the piecewise linear function $g(\delta) = \min_S f(S) - \delta a(S)$, in the case of an arbitrary direction a?*

Our results do not imply anything on this number of breakpoints, and this number could still be quadratic, exponential or even linear. In the simpler, nonnegative setting $a \in \mathbb{R}^n_+$, it is not just that the discrete Newton's algorithm takes at most n iterations, but it is also the case that the number of breakpoints of the lower envelope is at most n (by the property of strong quotients). On the other hand, there exist instances of parametric minimum $s-t$ cut problems where the minimum cut value has an exponential number of breakpoints [Mulmuley, 1999]. However, this corresponds to the more general problem $\min_S f(S) - \delta a(S)$ where $f(\cdot)$ is submodular but the function $a(\cdot)$ is not modular (and not even supermodular or submodular as the slopes of the parametric capacities can be positive or negative).

7.2.3 Generalized approximate counting

In Chapter 5, we noted that generalized approximate counting gives an efficient way to compute projections onto the simplex of the vertex set when the probability distribution is

[1]A matrix is doubly stochastic if it has non-negative entries and each row and column sums to 1.

implicitly described using multipliers on the ground set of elements. It would be interesting to see if minimizing certain convex functions is equivalent to computing entropic projections over the simplex of vertices, thereby enabling efficient approximate counting.

Open Problem 6. *Can we do approximate counting by minimizing a convex function? When does optimization imply counting?*

We know that there exist matroids where approximate counting will necessarily involve exponential calls to the submodular evaluation oracle [Azar et al., 1994], however we can minimize separable strictly convex functions over any submodular base polytope in polynomial calls to the oracle. If indeed approximate counting can be performed using convex minimization (not just by minimizing separable convex functions), then this would solve an open problem of approximately counting the number of matchings in a graph (the problem of counting the exact number of perfect matchings in a graph is $\#P$-hard, [Valiant, 1979]).

7.2.4 Nash-equilibria in two-player games

In Chapter 6, we discussed the combinatorial structure of symmetric Nash-equilibria for matroid games with bilinear loss functions, however the structure of Nash-equilibria still remains unclear.

Open Problem 7. *What is the combinatorial structure of approximate Nash-equilibria of two-player zero-sum matroid games? Can we find these without using online learning or solving the von Neumann linear program?*

We include some examples of Nash-equilibria for the spanning tree game (each player plays spanning trees of a given graph) under the identity loss matrix, i.e. solutions of $\min_{x\in P(f)} \max_{y \in P(f)} x^T y$, in Appendix B. We hope that these can help the reader in testing or disproving possible conjectures on the structure of equilibria.

Appendix A

First-order optimization methods

In this chapter, we list the setting and iterations of the following derivatives of the mirror descent algorithm:

(i) mirror descent, lazy mirror descent and mirror prox used for minimizing a convex function $h(x)$ over a convex set X in Table A.1,

(ii) stochastic smooth mirror descent used for minimizing smooth convex functions under noisy gradient information, and saddle point mirror descent and saddle point mirror prox for finding saddle-points in Table A.2,

(iii) stochastic mirror descent for minimizing a convex function $h(x)$ under noisy gradient information, stochastic gradient descent which is a special case of the stochastic mirror descent, and online mirror descent for online convex optimization in Table A.3.

In each of these variants, one needs to compute a Bregman projection (refer to Section 2.2.2 for details) whenever X is a constrained set.

Algorithm	Iterations	Notes
Mirror descent	$x^{(1)} = \arg\min_{x\in X\cap\mathcal{D}} \omega(x)$ $\nabla\omega(y^{(t+1)}) = \nabla\omega(x^{(t)}) - \eta g_t$ $x^{(t+1)} = \arg\min_{x\in X\cap\mathcal{D}} D_\omega(x, y^{(t+1)})$	For $\min_{x\in X} h(x)$, h is convex, G-Lipschitz w.r.t. $\|\cdot\|$, $\eta = \frac{R}{G}\sqrt{\frac{2\kappa}{t}}$ then $h(\frac{1}{t}\sum_{s=1}^{t} x_s) - h(x^*) \leq RG\sqrt{\frac{2}{\kappa t}}$
Lazy mirror descent	$x_1 \in \arg\min_{x\in X\cap\mathcal{D}} \omega(x),$ $\nabla\omega(y^{(t)}) = \nabla\omega(y^{(t)}) - \eta g_t,$ $x^{(t)} = \operatorname{argmin}_{x\in X\cap\mathcal{D}} \eta \sum_{s=1}^{t-1} g_s^T x + \omega(x).$	For $\min_{x\in X} h(x)$, h is convex, G-Lipschitz, $g_t \in \partial h(x^{(t)})$, $\eta = \frac{R}{G}\sqrt{\frac{\kappa}{2t}}$ then $h(\frac{1}{t}\sum_{s=1}^{t} x_s) - h(x^*) \leq 2RG\sqrt{\frac{2}{\kappa t}}$
Mirror prox	$x^{(1)} \in \arg\min_{x\in X\cap\mathcal{D}} \omega(x),$ $\nabla\omega(y^{(t+1)\prime}) = \nabla\omega(x^{(t)}) - \eta\nabla h(x_t)$ $y^{(t+1)} \in \operatorname{argmin}_{x\in X\cap\mathcal{D}} D_\omega(x, y^{(t+1)\prime})$ $\nabla\omega(x^{(t+1)\prime}) = \nabla\omega(x^{(t)}) - \eta\nabla h(y^{(t+1)})$ $x^{(t+1)} \in \arg\min_{x\in X\cap\mathcal{D}} D_\omega(x, x^{(t+1)\prime})$	For $\min_{x\in X} h(x)$, h convex, β-smooth w.r.t. $\|\cdot\|$, $\eta = \frac{\kappa}{\beta}$ then $h(\frac{1}{t}\sum_{s=1}^{t} y_{s+1}) - h(x^*) \leq \frac{\beta R^2}{\kappa t}$

Table A.1: Mirror Descent and its variants. Here, the mirror map $\omega : X \cap \mathcal{D} \to \mathbb{R}$ is κ-strongly convex with respect to $\|\cdot\|$, $R^2 = \max_{x\in X}\omega(x) - \min_{x\in X}\omega(x)$, η is the learning rate. This table summarizes convergence rates as presented in [Bubeck, 2014].

Algorithm	Iterations	Notes
Smooth stochastic mirror descent	$x^{(1)} = \arg\min_{X\cap\mathcal{D}} \omega(x)$, $x^{(t+1)} = \arg\min_{x\in X\cap\mathcal{D}} \big(\eta\tilde{g}(x_t)^T x + D_\omega(x, x^{(t)})\big)$	For $\min_{x\in X} h(x)$, h is convex, β-smooth, under a stochastic oracle: given $x \in X$ and $h : X \to \mathbb{R}$ convex, returns $\tilde{g}(x)$ such that $\mathbb{E}(\tilde{g}(x)) \in \partial h(x)$, let $\mathbb{E}(\|\nabla h(x) - \tilde{g}(x)\|_*^2) \le \sigma^2$, with step-size $\frac{1}{\beta+1/\eta}$, $\eta = \frac{R}{B}\sqrt{\frac{2}{t}}$, then $\mathbb{E}(h(\frac{1}{t}\sum_{s=1}^{t} x_{s+1})) - \min_{x\in X} h(x) \le R\sigma\sqrt{\frac{2}{t}} + \frac{\beta R^2}{t}$
Saddle point mirror descent*	$z^{(1)} \in \arg\min_{z\in Z\cap\mathcal{D}} \omega(z)$, $z^{(t+1)} \in \arg\min_{z\in Z\cap\mathcal{D}} ng_t^T z + D_\omega(z, z^{(t)})$	For $\min_{x\in X}\max_{y\in Y} \phi(x,y)$, $a = G_X/R_X$, $b = G_Y/R_Y$, $\eta = \sqrt{2/t}$, $\tilde{x}(t) = \frac{1}{t}\sum_{s=1}^{t} x_s$, $\tilde{y}(t) = \frac{1}{t}\sum_{s=1}^{t} y_s$ then $\max_{y\in Y}\phi(\tilde{x}(t), y) - \min_{x\in X}\phi(x, \tilde{y}(t)) \le (R_X G_X + R_Y G_Y)\sqrt{2/t}$
Saddle point mirror prox*	$z^{(1)} \in \arg\min_{z\in Z\cap\mathcal{D}} \omega(z)$, $w^{(t+1)} = \arg\min_{z\in Z\cap\mathcal{D}} \big[D_\omega(z, z^{(t)}) + \eta(\nabla_x\phi(z^{(t)}) - \nabla_y\phi(z^{(t)}))^T z\big]$, $z^{(t+1)} = \operatorname{argmin}_{z\in Z\cap\mathcal{D}} \big[D_\omega(z, z^{(t)}) + \eta(\nabla_x\phi(w^{(t+1)}) - \nabla_y\phi(w^{(t+1)}))^T z\big]$	For $\min_{x\in X}\max_{y\in Y}\phi(x,y)$, $z^{(t)} = (x^{(t)}, y^{(t)})$, $w_t = (u^{(t)}, v^{(t)})$, ϕ is $(\beta_{11}, \beta_{12}, \beta_{22}, \beta_{21})$ smooth, $a = 1/R_X^2$, $b = 1/R_Y^2$, $\eta = \eta_{spmp}$, $\tilde{u}(t) = \frac{1}{t}\sum_{s=1}^{t} u^{(s+1)}$, $\tilde{v}(t) = \frac{1}{t}\sum_{s=1}^{t} v^{(s+1)}$ then $\max_{y\in Y}\phi(\tilde{u}(t), y) - \min_{x\in X}\phi(x, \tilde{v}(t)) \le \frac{2}{\eta t}$

Table A.2: Mirror Descent and its variants. Here, the mirror map $\omega : X \cap \mathcal{D} \to \mathbb{R}$ is κ-strongly convex with respect to $\|\cdot\|$, $R^2 = \max_{x\in X}\omega(x) - \min_{x\in X}\omega(x)$, η is the learning rate. For saddle point problems, $Z = X \times Y$, $\omega(z) = a\omega_X(x) + b\omega_Y(y)$, $g^{(t)} = (g_{X,t}, g_{Y,t})$, $g_{X,t} = \partial_x\phi(x_t, y_t), g_{Y,t} \in \partial_y(-\phi(x_t, y_t))$. $\eta_{spmp} = 1/(2\max(\beta_{11}R_X^2, \beta_{22}R_Y^2, \beta_{12}R_X R_Y, \beta_{21}R_X R_Y))$. This table summarizes convergence rates as presented in [Bubeck, 2014].

Algorithm	Iterations	Notes
Stochastic mirror descent	$x^{(1)} = \arg\min_{X\cap\mathcal{D}} \omega(x),$ $x^{(t+1)} = \arg\min_{x\in X\cap\mathcal{D}} \big(\eta \tilde{g}(x_t)^T x + D_\omega(x, x^{(t)})\big)$	For $\min_{x\in X} h(x)$, under a stochastic oracle: given $x \in X$ and $h : X \to \mathbb{R}$ convex, returns $\tilde{g}(x)$ such that $\mathbb{E}(\tilde{g}(x)) \in \partial h(x)$, let $\mathbb{E}(\\|\tilde{g}(x)\\|_*^2) \leq B^2$, $\eta = \frac{R}{B}\sqrt{\frac{2}{t}}$, then $\mathbb{E}(h(\frac{1}{t}\sum_{s=1}^{t} x_s)) - \min_{x\in X} h(x) \leq RB\sqrt{\frac{2}{t}}$
Stochastic gradient descent	$x^{(1)} = \arg\min_{X\cap\mathcal{D}} \\|x\\|^2,$ $x^{(t+1)} = \arg\min_{x\in X\cap\mathcal{D}} \\|x^{(t)} - \eta\tilde{g}(x^{(t)}) - x\\|^2$	For $\min_{x\in X} h(x)$, under a stochastic oracle: given $x \in X$ and $h : X \to \mathbb{R}$ convex, returns $\tilde{g}(x)$ such that $\mathbb{E}(\tilde{g}(x)) \in \partial h(x)$, let $\mathbb{E}(\\|\tilde{g}(x)\\|_*^2) \leq B^2$, $\eta = \frac{R}{B}\sqrt{\frac{2}{t}}$, then $\mathbb{E}(h(\frac{1}{t}\sum_{s=1}^{t} x^{(s)})) - \min_{x\in X} h(x) \leq RB\sqrt{\frac{2}{t}}$
Online mirror descent	$x^{(1)} = \arg\min_{X\cap\mathcal{D}} \omega(x),$ $\nabla\omega(y^{(t+1)}) = \nabla\omega(x^{(t)}) - \eta\nabla l^{(t)}(x^{(t)}),$ $x^{(t+1)} = \arg\min_{x\in X\cap\mathcal{D}} D_\omega(x, y^{(t+1)})$	For regret minimization: $R_t = \sum_{i=1}^{t} l^{(i)}(x^{(i)}) - \min_{x\in X}\sum_{i=1}^{t} l^{(i)}(x)$, under loss functions $l^{(i)}$ revealed in each round i, $l^{(i)} : X \to \mathbb{R}$ convex and $\\|\nabla l^{(i)}\\|_* \leq G \quad \forall i \in \{1,\ldots,t\}$, set $\eta = \frac{R}{G}\sqrt{\frac{2k}{t}}$ then: $\sum_{i=1}^{t} l^{(i)}(x^{(i)}) - \min_{x\in X}\sum_{i=1}^{t} l^{(i)}(x) \leq RG\sqrt{\frac{2t}{k}}$

Table A.3: Mirror Descent and relatives. Here, the mirror map $\omega : X \cap \mathcal{D} \to \mathbb{R}$ is κ-strongly convex with respect to $\|\cdot\|$, $R^2 = \max_{x\in X}\omega(x) - \min_{x\in X}\omega(x)$, η is the learning rate. This table summarizes convergence rates as presented in [Bubeck, 2014].

Appendix B

Examples of Nash-equilibria

In this chapter, we include some examples of Nash-equilibria of two-player zero-sum games when each player plays a spanning tree of the given graph, under an identity loss matrix (see Chapter 6 for background and details). More precisely, we give solutions to $\min_{x\in B(f)} \max_{y\in B(f)} x^T y$, where $B(f)$ is Edmonds' characterization of the spanning tree polytope with $f(\cdot)$ being the rank function of the graphic matroid.

(i) For the graph in Figure B-1(a), the marginals of the Nash-equilibrium (p^*, q^*) are given in Figures B-1(b) and (c). Here,

$$p^* : p^*(e) = \begin{cases} 4/9 & e \in E(1,2,3,4,5), \\ 3/4 & e \in E(1,6,7,8,3), \end{cases}$$

and for the column player

$$q^* : q^*(e) = \begin{cases} 1/3 & e \in E(1,2,3,4,5), \\ 1 & e \in E(1,6,7,8,3). \end{cases}$$

The value of the game is $p^{*T} q^* = 4.333$, this is also the cost of the minimum spanning tree under weights q^*, and the cost of the maximum spanning tree under weights p^*. Here the partition of the edge set is the same under p^* as well as q^*.

(ii) For the graph in Figure B-2(a), the marginals of the Nash-equilibrium (p^*, q^*) are

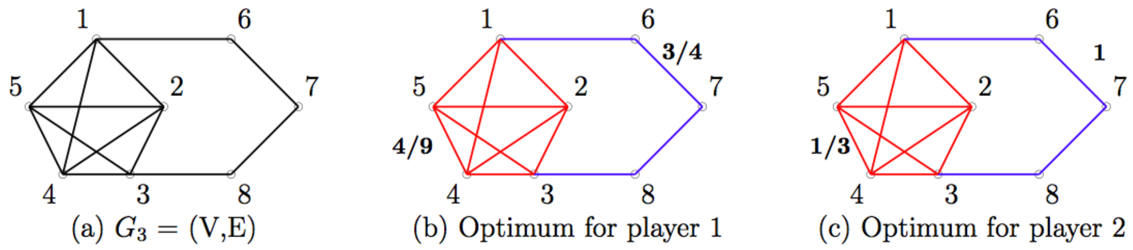


Figure B-1: (a) $G_3 = (V, E)$, (b) Optimal strategy for the row player (minimizer) (c) Optimal strategy for the column player (maximizer).

illustrated in Figures B-2(b) and (c). Here,

$$p^* : p^*(e) = \begin{cases} 3/4 & e \in E \setminus E(1,2,3), \\ 2/3 & e \in E(1,2,3), \end{cases}$$

and for the column player $q^* = 11/12\chi(E)$. It can be verified that the value of the game is $p^{*T}q^* = 33/4$. This example shows that the span of the set with edges of the maximum marginals for both the row and column player strategies contains the set of edges with the minimum marginals.

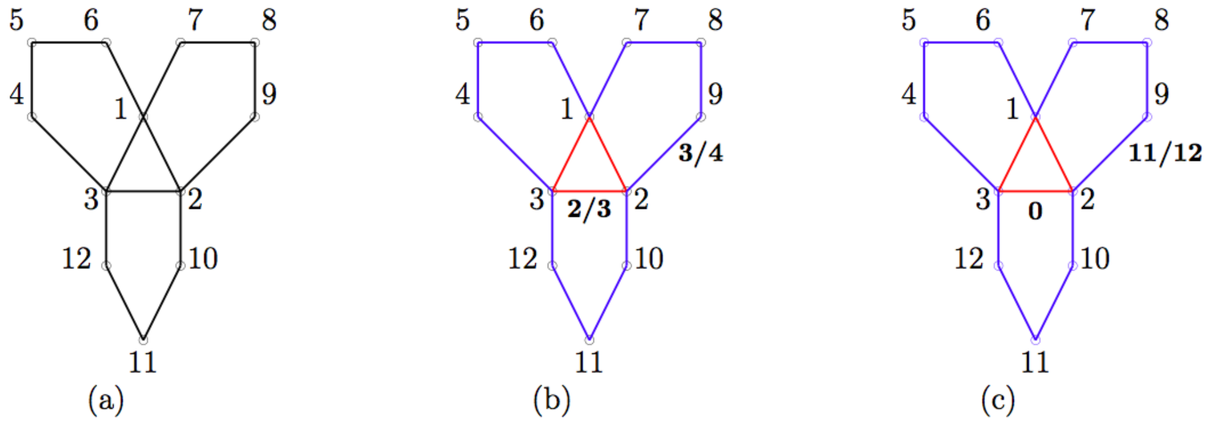


Figure B-2: (a) $G_4 = (V, E)$, (b) Optimal strategy for the row player (minimizer), (c) Optimal strategy for the column player (maximizer).

(iii) Finally, for the graph in Figure B-3(a), the marginals of the Nash-equilibrium (p^*, q^*), as illustrated in Figures B-3(b) and (c) are

$$p^* : p^*(e) = \begin{cases} 5/12 & e \in E \setminus E(7,8,9), \\ 7/12 & e \in E(7,8,9) \end{cases}$$

and for the column player

$$q^* : q^*(e) = \begin{cases} 1/3 & e \in E(1,2,3,4,5,6) \\ 2/3 & e \in E \setminus E(1,2,3,4,5,6). \end{cases}$$

It can be verified that the value of the game is $p^{*T} q^* = 11/3$.

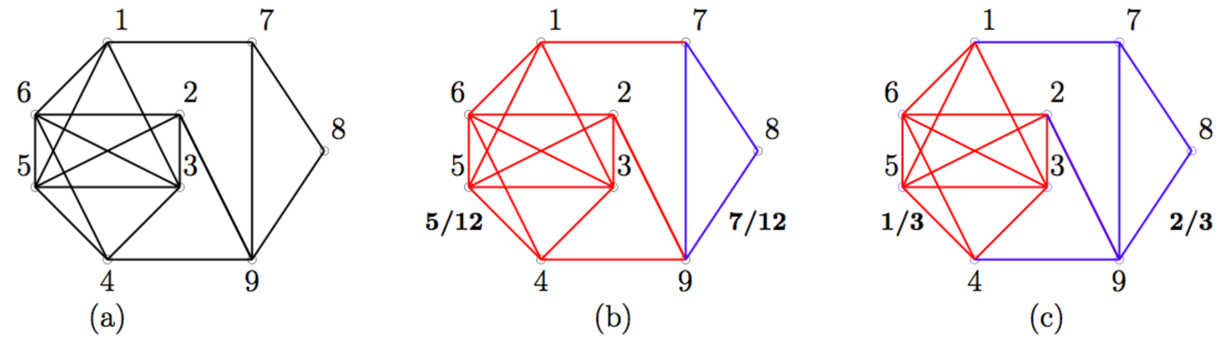


Figure B-3: (a) $G_5 = (V, E)$, (b) Optimal strategy for the row player (minimizer), (c) Optimal strategy for the column player (maximizer).

Bibliography

[Arora et al., 2012] Arora, S., Hazan, E., and Kale, S. (2012). The Multiplicative Weights Update Method: a Meta-Algorithm and Applications. *Theory of Computing*, 8:121–164. [Pages 25, 48, 108, and 131.]

[Asadpour et al., 2010] Asadpour, A., Goemans, M. X., Madry, A., Oveis Gharan, S., and Saberi, A. (2010). An O (log n/log log n)-approximation Algorithm for the Asymmetric Traveling Salesman Problem. *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. [Page 113.]

[Audibert et al., 2013] Audibert, J., Bubeck, S., and Lugosi, G. (2013). Regret in online combinatorial optimization. *Mathematics of Operations Research*, 39(1):31–45. [Page 46.]

[Azar et al., 1994] Azar, Y., Broder, A. Z., and Frieze, A. M. (1994). On the problem of approximating the number of bases of a matriod. *Information Processing Letters*, 50(1):9–11. [Pages 134 and 147.]

[Banerjee et al., 2005] Banerjee, A., Merugu, S., Dhillon, I. S., and Ghosh, J. (2005). Clustering with bregman divergences. *Journal of Machine Learning Research*, 6:1705–1749. [Pages 15 and 40.]

[Beck and Teboulle, 2003] Beck, A. and Teboulle, M. (2003). Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175. [Pages 26 and 46.]

[Ben-Tal and Nemirovski, 2001] Ben-Tal, A. and Nemirovski, A. (2001). *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. SIAM. [Page 43.]

[Bixby et al., 1985] Bixby, R. E., Cunningham, W. H., and Topkis, D. M. (1985). The partial order of a polymatroid extreme point. *Mathematics of Operations Research*, 10(3):367–378. [Page 76.]

[Blum et al., 2008] Blum, A., Hajiaghayi, M. T., Ligett, K., and Roth, A. (2008). Regret minimization and the price of total anarchy. *Proceedings of the fortieth annual ACM Symposium on Theory of Computing (STOC)*, pages 1–20. [Page 49.]

[Boyd and Vandenberghe, 2009] Boyd, S. and Vandenberghe, L. (2009). *Convex optimization*. Cambridge University Press. [Pages 40 and 43.]

[Bregman, 1967] Bregman, L. M. (1967). The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217. [Page 39.]

[Bubeck, 2011] Bubeck, S. (2011). Introduction to online optimization. *Lecture Notes, Princeton University*. [Page 46.]

[Bubeck, 2014] Bubeck, S. (2014). Theory of Convex Optimization for Machine Learning. *arXiv preprint arXiv:1405.4980*. [Pages 15, 16, 38, 40, 41, 43, 150, 151, and 152.]

[Cesa-Bianchi and Lugosi, 2006] Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge University Press. [Pages 46 and 126.]

[Chakrabarty et al., 2016] Chakrabarty, D., Lee, Y. T., Sidford, A., and Wong, S. C. (2016). Subquadratic submodular function minimization. *arXiv preprint arXiv:1610.09800*. [Pages 35 and 76.]

[Chakrabarty et al., 2006] Chakrabarty, D., Mehta, A., and Vazirani, V. V. (2006). Design is as easy as optimization. In *Automata, Languages and Programming*, pages 477–488. Springer. [Page 126.]

[Cunningham, 1985a] Cunningham, W. H. (1985a). On submodular function minimization. *Combinatorica*, 5(3):185–192. [Page 35.]

[Cunningham, 1985b] Cunningham, W. H. (1985b). Optimal attack and reinforcement of a network. *Journal of the ACM (JACM)*, 32(3):549–561. [Pages 23 and 88.]

[Edmonds, 1970] Edmonds, J. (1970). Submodular functions, matroids, and certain polyhedra. *Combinatorial Structures and their applications*, pages 69–87. [Pages 22, 33, 35, and 52.]

[Edmonds, 1971] Edmonds, J. (1971). Matroids and the greedy algorithm. *Mathematical Programming*, 1(1):127–136. [Page 60.]

[Fleischer and Iwata, 2003] Fleischer, L. and Iwata, S. (2003). A push-relabel framework for submodular function minimization and applications to parametric optimization. *Discrete Applied Mathematics*, 131(2):311–322. [Pages 35, 37, and 76.]

[Frank and Wolfe, 1956] Frank, M. and Wolfe, P. (1956). An algorithm for quadratic programming. *Naval Research Logistics quarterly*, 3(1-2):95–110. [Pages 41, 47, and 77.]

[Freund et al., 2015] Freund, R. M., Grigas, P., and Mazumder, R. (2015). An extended Frank-Wolfe method with "In-Face" directions, and its application to low-rank matrix completion. *arXiv preprint arXiv:1511.02204*. [Pages 23 and 88.]

[Fujishige, 1980] Fujishige, S. (1980). Lexicographically optimal base of a polymatroid with respect to a weight vector. *Mathematics of Operations Research*. [Pages 46, 49, 77, 140, and 141.]

[Fujishige, 2005] Fujishige, S. (2005). *Submodular functions and optimization*, volume 58. Elsevier. [Pages 37 and 67.]

[Gidel et al., 2016] Gidel, G., Jebara, T., and Lacoste-Julien, S. (2016). Frank-wolfe algorithms for saddle point problems. *arXiv preprint arXiv:1610.07797*. [Page 137.]

[Goemans et al., 2017] Goemans, M. X., Gupta, S., and Jaillet, P. (2017). Discrete Newton's algorithm for parametric submodular function minimization. *Proceedings of the nineteenth conference on Integer Programming and Combinatorial Optimization (IPCO)*. [Page 96.]

[Grigas, 2016] Grigas, P. P. E. (2016). *Methods for convex optimization and statistical learning*. PhD thesis, Massachusetts Institute of Technology. [Page 43.]

[Groenevelt, 1991] Groenevelt, H. (1991). Two algorithms for maximizing a separable concave function over a polymatroid feasible region. *European Journal of Operational Research*, 54(2):227–236. [Pages 46 and 77.]

[Grötschel et al., 1981] Grötschel, M., Lovász, L., and Schrijver, A. (1981). The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197. [Pages 35, 126, and 130.]

[Håstad, 1994] Håstad, J. (1994). On the size of weights for threshold gates. *SIAM Journal on Discrete Mathematics*, 7(3):484–492. [Page 96.]

[Hazan, 2012] Hazan, E. (2012). Survey: The convex optimization approach to regret minimization. *Optimization for Machine Learning*, page 287. [Page 46.]

[Hazan and Koren, 2015] Hazan, E. and Koren, T. (2015). The computational power of optimization in online learning. *arXiv preprint arXiv:1504.02089*. [Page 110.]

[Helmbold and Schapire, 1997] Helmbold, D. P. and Schapire, R. E. (1997). Predicting nearly as well as the best pruning of a decision tree. *Machine Learning*, 27(1):51–68. [Pages 49, 117, and 118.]

[Helmbold and Warmuth, 2009] Helmbold, D. P. and Warmuth, M. K. (2009). Learning permutations with exponential weights. *The Journal of Machine Learning Research*, 10:1705–1736. [Page 146.]

[Immorlica et al., 2011] Immorlica, N., Kalai, A. T., Lucier, B., Moitra, A., Postlewaite, A., and Tennenholtz, M. (2011). Dueling algorithms. In *Proceedings of the 43rd annual ACM Symposium on Theory of Computing*, pages 215–224. ACM. [Pages 28 and 126.]

[Itakura and Saito, 1968] Itakura, F. and Saito, S. (1968). Analysis synthesis telephony based on the maximum likelihood method. In *Proceedings of the 6th International Congress on Acoustics*, volume 17, pages C17–C20. pp. C17–C20. [Pages 15 and 40.]

[Iwata, 2008] Iwata, S. (2008). Submodular function minimization. *Mathematical Programming*, 112(1):45–64. [Pages 37 and 89.]

[Iwata et al., 1997] Iwata, S., Murota, K., and Shigeno, M. (1997). A fast parametric submodular intersection algorithm for strong map sequences. *Mathematics of Operations Research*, 22(4):803–813. [Pages 36 and 37.]

[Iwata and Orlin, 2009] Iwata, S. and Orlin, J. B. (2009). A simple combinatorial algorithm for submodular function minimization. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1230–1237. Society for Industrial and Applied Mathematics. [Pages 24, 76, and 90.]

[Jaggi, 2013] Jaggi, M. (2013). Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 427–435. [Pages 41 and 42.]

[Jerrum et al., 2004] Jerrum, M., Sinclair, A., and Vigoda, E. (2004). A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *ACM Symposium of Theory of Computing*, 51(4):671–697. [Pages 26, 117, and 118.]

[Jerrum et al., 1986] Jerrum, M. R., Valiant, L. G., and Vazirani, V. V. (1986). Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188. [Page 115.]

[Koo et al., 2007] Koo, T., Globerson, A., C. Pérez, X., and Collins, M. (2007). Structured prediction models via the matrix-tree theorem. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150. [Pages 26, 49, and 118.]

[Koolen et al., 2010] Koolen, W. M., Warmuth, M. K., and Kivinen, J. (2010). Hedging Structured Concepts. *Proceedings of the 23rd Annual Conference on Computational Learning Theory (COLT)*. [Pages 26 and 118.]

[Koutis et al., 2010] Koutis, I., Miller, G. L., and Peng, R. (2010). Approaching optimality for solving SDD linear systems. *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 235–244. [Pages 117 and 136.]

[Krichene et al., 2015] Krichene, W., Krichene, S., and Bayen, A. (2015). Efficient bregman projections onto the simplex. In *Proceedings of the 54th IEEE Conference on Decision and Control (CDC)*, pages 3291–3298. IEEE. [Pages 23 and 47.]

[Kuhn, 1955] Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics quarterly*, 2(1-2):83–97. [Page 145.]

[Lee et al., 2015] Lee, Y. T., Sidford, A., and Wong, S. C. (2015). A faster cutting plane method and its implications for combinatorial and convex optimization. In *Foundations of Computer Science (FOCS)*, pages 1049–1065. IEEE. [Pages 35, 70, 71, 76, and 90.]

[Lovász et al., 1988] Lovász, L., Grötschel, M., and Schrijver, A. (1988). Geometric algorithms and combinatorial optimization. *Berlin: Springer-Verlag*, 33:34. [Page 130.]

[Lyons and Peres, 2005] Lyons, R. and Peres, Y. (2005). *Probability on trees and networks.* [Page 117.]

[Martin, 1991] Martin, R. K. (1991). Using separation algorithms to generate mixed integer model reformulations. *Operations Research Letters*, 10(April):119–128. [Page 131.]

[McCormick and Ervolina, 1994] McCormick, S. T. and Ervolina, T. R. (1994). Computing maximum mean cuts. *Discrete Applied Mathematics*, 52(1):53–70. [Page 94.]

[Mulmuley, 1999] Mulmuley, K. (1999). Lower bounds in a parallel model without bit operations. *SIAM Journal on Computing*, 28(4):1460–1509. [Page 146.]

[Nagano, 2007a] Nagano, K. (2007a). A faster parametric submodular function minimization algorithm and applications. *Mathematical Engineering Technical Report.* [Pages 15, 37, 70, 71, and 76.]

[Nagano, 2007b] Nagano, K. (2007b). On convex minimization over base polytopes. *Integer Programming and Combinatorial Optimization.* [Pages 47, 48, 71, 77, 89, and 141.]

[Nagano, 2007c] Nagano, K. (2007c). A strongly polynomial algorithm for line search in submodular polyhedra. *Discrete Optimization*, 4(3):349–359. [Page 24.]

[Nagano and Aihara, 2012] Nagano, K. and Aihara, K. (2012). Equivalence of convex minimization problems over base polytopes. *Japan Journal of Industrial and Applied Mathematics*, pages 519–534. [Pages 64 and 77.]

[Nemirovski, 2004] Nemirovski, A. (2004). Prox-method with rate of convergence $o(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251. [Pages 41 and 136.]

[Nemirovski and Yudin, 1983] Nemirovski, A. S. and Yudin, D. B. (1983). Problem complexity and method efficiency in optimization. *Wiley-Interscience, New York.* [Pages 38, 42, and 45.]

[Nesterov, 2005] Nesterov, Y. (2005). Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152. [Page 120.]

[Nesterov, 2013] Nesterov, Y. (2013). *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media. [Pages 42 and 43.]

[Orlin, 2009] Orlin, J. B. (2009). A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming*, 118(2):237–251. [Pages 35 and 76.]

[Orlin, 2013] Orlin, J. B. (2013). Max flows in O(nm) time, or better. In *Proceedings of the forty-fifth annual ACM Symposium on Theory of Computing (STOC)*, pages 765–774. ACM. [Page 135.]

[Oxley, 2006] Oxley, J. G. (2006). *Matroid theory*, volume 3. Oxford University Press, USA. [Page 137.]

[Papadimitriou and Roughgarden, 2008] Papadimitriou, C. H. and Roughgarden, T. (2008). Computing correlated equilibria in multi-player games. *Journal of the ACM (JACM)*, 55(3):14. [Pages 125 and 126.]

[Radzik, 1998] Radzik, T. (1998). Fractional combinatorial optimization. In *Handbook of Combinatorial Optimization*, pages 429–478. Springer. [Pages 48, 90, 94, and 96.]

[Rakhlin and Sridharan, 2013] Rakhlin, A. and Sridharan, K. (2013). Optimization, learning, and games with predictable sequences. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3066–3074. [Page 136.]

[Rakhlin and Sridharan, 2014] Rakhlin, A. and Sridharan, K. (2014). Lecture Notes on Online Learning. *Draft*. [Page 45.]

[Robinson, 1951] Robinson, J. (1951). An iterative method of solving a game. *Annals of Mathematics*, pages 296–301. [Pages 49 and 110.]

[Rothvoß, 2014] Rothvoß, T. (2014). The matching polytope has exponential extension complexity. In *Proceedings of the 46th annual ACM Symposium on Theory of Computing (STOC)*, pages 263–272. ACM. [Page 131.]

[Schnorr, 1976] Schnorr, C. (1976). Optimal algorithms for self-reducible problems. In *ICALP*, volume 76, pages 322–337. [Page 115.]

[Schrijver, 2000] Schrijver, A. (2000). A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80(2):346–355. [Page 35.]

[Schrijver, 2003] Schrijver, A. (2003). *Combinatorial optimization: polyhedra and efficiency*. Springer. [Pages 34, 37, 64, 135, and 137.]

[Sinclair and Jerrum, 1989] Sinclair, A. and Jerrum, M. (1989). Approximate counting, uniform generation and rapidly mixing Markov chains. *Information and Computation*, 82(1):93–133. [Page 115.]

[Singh and Vishnoi, 2014] Singh, M. and Vishnoi, N. K. (2014). Entropy, optimization and counting. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 50–59. ACM. [Pages 113, 117, and 118.]

[Suehiro et al., 2012] Suehiro, D., Hatano, K., Kijima, S., Takimoto, E., and Nagano, K. (2012). Online prediction under submodular constraints. In *International Conference on Algorithmic Learning Theory*, pages 260–274. Springer. [Pages 23, 47, and 86.]

[Takimoto and Warmuth, 2003] Takimoto, E. and Warmuth, M. K. (2003). Path kernels and multiplicative updates. *The Journal of Machine Learning Research*, 4:773–818. [Pages 49, 117, and 118.]

[Topkis, 1978] Topkis, D. M. (1978). Minimizing a submodular function on a lattice. *Operations Research*, 26(2):305–321. [Pages 36, 48, and 89.]

[Valiant, 1979] Valiant, L. G. (1979). The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201. [Pages 117 and 147.]

[von Neumann, 1928] von Neumann, J. (1928). Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320. [Page 126.]

[Washburn and Wood, 1995] Washburn, A. and Wood, K. (1995). Two person zero-sum games for network interdiction. *Operations Research*. [Page 126.]

[Welsh, 2009] Welsh, D. (2009). Some problems on approximate counting in graphs and matroids. In *Research Trends in Combinatorial Optimization*, pages 523–544. Springer. [Pages 117 and 118.]

[Wilson, 1996] Wilson, D. B. (1996). Generating random spanning trees more quickly than the cover time. In *Proceedings of the twenty-eighth annual ACM Symposium on Theory of Computing (STOC)*, pages 296–303. ACM. [Pages 117 and 118.]

[Yasutake et al., 2011] Yasutake, S., Hatano, K., Kijima, S., Takimoto, E., and Takeda, M. (2011). Online linear optimization over permutations. In *International Symposium on Algorithms and Computation*, pages 534–543. Springer. [Pages 23, 47, and 86.]

[Zinkevich, 2003] Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. *Proceedings of the twentieth International Convergence on Machine Learning (ICML)*. [Page 45.]