TOPICS IN CLASSICAL AND QUANTUM OPTIMIZATION: COMPLEXITY AND ALGORITHMS

by
Phillip Kerger

A dissertation submitted to Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy

Baltimore, Maryland
March 2024
Abstract

This thesis presents advances revolving around optimization, complexity, and applications of quantum computing. In particular, part I studies information complexity, focusing on convex mixed-integer optimization through the contents of [10], primarily. With new results on lower bounds, the goal thereof is to better understand how difficult these problems are to solve in different settings. We refine the classical notion of first-order information complexity to allow for questions such as “how many bits are needed to solve a given optimization problem?”, provide results for a few settings of interest, and prove a powerful theorem to transfer lower bounds on information complexity across settings. Notably, lower-bounds on information complexity also imply lower-bounds on algorithmic complexity. With such lower-bounds in place for classical computational systems, with the recent emergence of quantum computing technologies, it is only natural to wonder whether existing lower-bounds on optimization problems for algorithmic complexity for classical machines can be “beaten” by using quantum computers. We conclude the chapter with some discussion comparing the oracles we consider for the information complexity lower bounds with quantum versions of those oracles and some related directions for future work. Part II then focuses on advancements in quantum computing for optimization problems. Chapter 5 presents a method for formulating the image denoising problem as a quadratic unconstrained binary problem through using Boltzmann Machines and quantum annealing, as in [65]. This is applied to real (downscaled) data using a state-of-the-art D-Wave quantum annealer. Some statistical guarantees for the method are proven, a robustness modification is proposed that performs well in practice, and promising numerical results are provided. Chapter 6 introduces quantum distributed computing models along with algorithms that are asymptotically
faster than any of their known counterparts in the classical model. Though these do not yet "beat" any known lower-bounds for the classical model, these results are the first step towards such an achievement. We also spend considerable time on computing the exact complexity of the algorithms discussed, revealing that the algorithms discussed, along with existing related algorithms from the literature, are extremely impractical for realistic problem sizes. Extensions of the work to the survivable network design problem are discussed afterwards. We conclude the thesis with a general discussion on optimization, information and algorithmic complexity, and different computing technologies in relation to the content presented.
Thesis Readers

Dr. Amitabh Basu (primary advisor)
Professor of Applied Mathematics and Statistics
Johns Hopkins University

Dr. David Bernal Neira
Assistant Professor of Chemical Engineering
Purdue University
Acknowledgements

First and foremost, I would like to thank my parents Kim and Peter for their constant support and belief in me going back as far as I can remember, and my siblings Eric and Lizzy who have enriched my life endlessly. Visiting home to spend time together during my breaks has always lifted me up again and again, and your love and support have always kept me going. I cannot give enough thanks to my wonderful partner Natalia who keeps me grounded, picks me up whenever I doubt myself, keeps me focused on what matters, and is simply a wonderful person to be with every day.

On the academic side, Amitabh has been the best advisor I could have hoped for on both the human and the academic level. I have always been met with nothing but kindness, understanding, support, and mathematical brilliance by Amitabh. Working on the challenging, technical parts of this thesis together has been a privilege, and your mentorship throughout has been priceless. I am grateful to have had the pleasure of also working together with Marco Molinaro and Hongyi Jiang on the contents of Chapters 2 and 3, with Ryoji Miyazaki on the contents of Chapter 5, and with David Bernal Neira, Eleanor Rieffel, and Zoe Gonzalez Izquierdo on the contents of Chapter 6. Your guidance, contributions, and insights have been priceless. Thank you also to Mario Micheli and Donniell Fishkind for their mentorship, who have particularly supported my teaching endeavors that I hope to continue pursuing moving forward.

Thank you also to all the friends that I have made along the way. My experience in grad school was completely transformed by my peers, so I am thankful to each and every one of you. Special thanks to Yashil and Joshua who have been there supporting me nonstop from my first day in the program til now – the big days we had will be cherished.
I am also grateful to the wonderful people I met during my undergraduate studies at Fordham who started me on this journey. Dave and Ishan, thanks for challenging me, pushing me and believing in me already well back then. Special thanks also to Kei Kobayashi, to whom I owe my decision of pursuing graduate school, who has been a great mentor, collaborator, and friend.

Thank you to my grandparents Sara and Harold, who never fail to shower me in compliments and support, and always offer a calming retreat from the work when I visit.

To my friends Marcus, Marc, and Christian in Germany, going all the way back to kindergarten: Danke, dass es sich jedes mal wenn ich zu Besuch komme so anfühlt, als wäre ich nie weg gewesen.

Finally, I dedicate this thesis to my late grandfather, Dr. rer. nat. Klaus Kerger, who was ever-dedicated to his family, admirably calm and logical, and a friend of nature and the mountains – may he rest in peace. He had always encouraged and developed my interest in science and technology, and is the reason I even saw research and getting my PhD as a possible path in life in the first place. There shall be a Dr. Kerger once more!
Dedicated to Dr. Klaus Kerger
Declaration

I confirm that the work contained in this PhD project report has been composed solely by myself and has not been accepted in any previous application for a degree. All sources of information have been specifically acknowledged and all verbatim extracts are distinguished by quotation marks.

Signed ............................................ Date ......................
Phillip Kerger
Preface

The modern world is filled with difficult computational challenges. Whether it be designing efficient transportation networks, scheduling employees or machines for certain tasks, finding the shortest path from one place to another, classifying the contents of digital images, determining predictive relationships in data, determining optimal medical treatment plans, and many more difficult challenges, the computing revolution of the twentieth century has allowed us to produce high quality solutions to many of these problems. Most commonly, these problems can be approached as follows: Some objective to be maximized or minimized is formulated, and some constraints on what solutions are allowed to be used are given. For example, consider the problem of finding the shortest route between some point A and point B. The objective to be minimized is the distance traveled, and the solution must be path from A to B. If one schedules medical personnel for different medical procedures at a hospital, the objective could be to maximize the number of procedures taking place with sufficient staff, and the main constraint would be that no personnel can be assigned to two different procedures at the same time. The process of formulating in precise ways such objectives and constraints, and designing algorithms to produce desired solutions, forms the core of the field of mathematical optimization. Good introductions to the field and common optimization techniques used across different settings can be found in [27, 88, 82].

To illustrate the enormous success mathematical optimization has seen in recent years in the types of problems described above, consider some of the recent nominees for the INFORMS Edelman Award [35]. This award celebrates successful implementations of optimization methods in the real world and highlights their impact on society. For
example, Deutsche Bahn AG, the largest train operator in the world by revenue and one of the nominees for the 2020 Edelman Award, implemented a highly sophisticated optimization framework for planning of the assignments of its wagons and locomotives for both freight and passenger trains, leading to an estimated savings of 34,000 tons of CO$_2$ emissions and 74€ million [17]. In 2022, the Government of Chile was awarded the prize for its use of analytics and optimization methods throughout the COVID-19 pandemic [7]. Conservative estimates on the impacts of this initiative suggest that at least 3000 lives and 300 million USD were saved. Inspired by and in awe of such success stories of optimization, this thesis humbly presents a few recent advances across the field of optimization, driven by algorithmic ideas contributing towards theoretical foundations of optimization and by the potential of leveraging emerging computational technologies through quantum computing.
Contents

Abstract ii
Thesis Readers iv
Acknowledgements v
Declaration viii
Preface ix
List of Tables xv
List of Figures xvi

About this Thesis 1
  0.1 Chapter List ....................................................... 1
  0.2 Table of Notation ................................................. 2

I Information Complexity 4

1 Introduction to Information Complexity 5
  1.1 Background and Elementary Definitions ......................... 8
  1.2 Existing Work and Results for Continuous Convex Optimization . 14
    1.2.1 Information Complexity of Continuous Convex Optimization under the Full-Information First-Order Oracle .............. 15
6.2.1 The CONGEST and CONGEST-CLIQUE Models of Distributed Computing .......................... 163
6.2.2 Quantum Versions of CONGEST and CONGEST-CLIQUE ......................................... 164
6.2.3 Notation and Problem Definitions ................................................................. 168

6.3 Contributions ................................................................. 168

6.4 APSP and Routing Tables ................................................................. 169
6.4.1 Distance Products and Routing Tables ......................................................... 169
6.4.2 Distance Products via Triangle Finding ...................................................... 170
6.4.3 Routing Tables via Efficient Computation of Witness Matrices ..................... 172
6.4.4 Triangle Finding ................................................................. 174
6.4.5 Distributed Quantum Searches .............................................................. 179
6.4.6 Final Steps of the Triangle Finding .......................................................... 180
6.4.7 Complexity ................................................................. 181

6.5 Algorithm for Approximate Steiner Tree ..................................................... 184
6.5.1 Algorithm Overview ................................................................. 184
6.5.2 Shortest Path Forest ................................................................. 185
6.5.3 Weight Modified MST and Pruning .......................................................... 187
6.5.4 Overall Complexity and Correctness .......................................................... 188

6.6 Directed Minimum Spanning Tree Algorithm ............................................... 188
6.6.1 Edmonds’ Centralized DMST Algorithm .................................................. 189
6.6.2 Lovasz’ Shrinking Iterations ....................................................................... 191
6.6.3 Complexity ................................................................. 195

6.7 Discussion and Future Work ................................................................. 195

6.8 Chapter Appendix ............................................................................. 198
6.8.1 Proof of claim 99 ...................................................................................... 198
6.8.2 The $\alpha > 0$ case in IdentifyClass .......................................................... 198
6.8.3 Expanding the DMST in the Distributed Setting ........................................... 202
6.8.4 Information access .............................................................................. 204

Conclusion ...................................................................................... 206

Bibliography ................................................................................... 208
List of Tables

1. Summary of Notation ......................................................... 3

2.1 Summary of results on the information complexity of mixed-integer convex optimization for the class of instances $\mathcal{I}_{n,d,R,\rho,M}$ that have $n$ integer variables, $d$ continuous variables, the feasible region lies in the box $[-R, R]^{n+d}$ and has a “$\rho$-deep feasible point” on the optimal fiber, and the objective function is $M$-Lipschitz with respect to $\ell_\infty$ (see Definition 4). The table presents simplified bounds showing only the main parameters. 29
List of Figures

1.1 An illustration of how $\nabla f(x)$ (arrow) at $x$ (blue) induces a halfspace (yellow) in which the optimal solution $x^*$ (red) must lie, with the ellipses representing the boundaries of level sets of $f$. ........................................ 16

1.2 A picture illustrating the construction of $f_s, I_s$ from $f'_s, I'_s$. .............. 23

2.1 (L) Illustration of two possible functions $f_0, f_1 \in F_{\text{cont}}$ (blue and red) of the continuous adversary $\text{Adv-Cont}^+$, for $d = 1$. (Right) Illustration of the function $\psi_{(f_0, f_1)}$ for the mixed-integer adversary $\text{Adv-MI}$ obtained by placing the functions $f_0$ and $f_1$ on the fibers $\{0\} \times \mathbb{R}$ and $\{1\} \times \mathbb{R}$ and interpolating appropriately between the fibers. ......................... 45

2.2 An example of a possible function from $f \in F_{\text{cont}}$ (left) together with an illustration of its truncated extension $\hat{f}_0(x, y)$ (right) as constructed in (2.1). ......................................................... 46

2.3 A render of $P^\epsilon(g, x)$ in $\mathbb{R}^3$ restricted to a sphere (upper image) compared to a stack of colloquial American breakfast pancakes (lower image) .... 72

3.1 An example where two approximate function values and subgradients do not have a convex extension. The true function $f$ is constant. The function values are reported with no error. The reported slopes are shown in red. However, these slopes decrease going from $x_1$ to $x_2$ thus eliminating the possibility of any convex function having these values and slopes at $x_1$ and $x_2$. ................................................................. 97
5.1 Examples of the denoising process using our method showing the true, noisy, and denoised images across different noise levels. . . . . . . . . . . 152

5.2 Proportion of pixels in denoised MNIST images that matched the original image, for different denoising methods with 95% CI error bars. . . . . . 154

5.3 Proportion of pixels in denoised BAS images that matched the original image, for different denoising methods with 95% CI error bars. . . . . . 154

5.4 Proportion of pixels in denoised MNIST images that matched the original image, for different denoising methods with 95% CI error bars. . . . . . 156

5.5 Proportion of pixels in denoised BAS images that matched the original image, for different denoising methods with 95% CI error bars. . . . . . 156

5.6 Proportion of pixels in denoised images that were correctly denoised, for different denoising methods on the MNIST dataset, with 95% confidence intervals shaded. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 158

5.7 An illustration of the Pegasus chip topology of the 5000 qubit D-Wave Advantage system. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 160
About this Thesis

This is the thesis of Phillip Kerger, submitted as part of the requirements for the degree of PhD in Applied Mathematics and Statistics at the Johns Hopkins University in Baltimore, Maryland, USA.

0.1 Chapter List

We provide here a list of the chapters and a brief summary of their contents.

Part I

Chapter 1 This chapter provides an introduction to and background in information complexity. It also provides an important contribution of definitions of a general class of oracles based on first-order information. Finally, to point out important existing work, the information complexity of unconstrained continuous convex optimization is derived.

Chapter 2 This chapter states and proves novel results in information complexity under different types of oracles. Highlights are a theorem that “transfers” lower bounds from the continuous to the mixed-integer setting and results for oracles that only give binary answers, including a new quadratic lower-bound on the number of bits needed to solve constrained continuous convex optimization problems.

Chapter 3 This chapter presents and proves a number of variations of a result to transfer guarantees for algorithms in the setting of exact first-order information to the setting of inexact first-order information. Consequences for information complexity are also discussed.

Part II

Chapter 5 This chapter builds a model for the image denoising problem via Restricted Boltzmann Machines that is particularly well-suited for implementing via quantum annealing (QA). The problem is formulated efficiently as a quadratic unconstrained binary objective for QA to be effective. Some statistical guarantees are proven, and the method is tested using a real quantum annealing machine.
Chapter 6 This chapter presents quantum distributed algorithms for the Steiner tree and directed minimum spanning tree (a.k.a. minimum weight arborescence) problems. The algorithms are asymptotically faster than any known classical algorithms. However, an investigation into their non-asymptotic complexities reveals that they (and related existing algorithms) are highly impractical for any realistically-sized problems.

Conclusion This chapter summarises the main outcomes and conclusions resulting from this thesis.

0.2 Table of Notation

We provide here a table of some of the notation that is most commonly used throughout this thesis.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning / Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{O}$</td>
<td>An oracle</td>
</tr>
<tr>
<td>$\mathcal{I}$</td>
<td>A class of optimization instances</td>
</tr>
<tr>
<td>$M$</td>
<td>Lipschitz constant for a function or class of functions</td>
</tr>
<tr>
<td>$C$</td>
<td>Feasible region for an optimization problem</td>
</tr>
<tr>
<td>$R$</td>
<td>Radius of box $[-R, R]$ in which the feasible region $C$ of an optimization problem is assumed to lie</td>
</tr>
<tr>
<td>$\rho$</td>
<td>Size of an $L_2$-ball that the feasible region $C$ of an optimization problem is assumed to contain</td>
</tr>
<tr>
<td>$\mathcal{I}_{n,d,\rho,M,R}$</td>
<td>A class of optimization instances with integer variables $x \in \mathbb{Z}^n$, continuous variables $y \in \mathbb{R}^d$, $M$-Lipschitz, $C$ containing a $\rho$-ball and $C$ being contained in $[-R, R]^d$</td>
</tr>
<tr>
<td>$\text{icomp}_\varepsilon(\mathcal{I}, \mathcal{O})$</td>
<td>The $\varepsilon$-information complexity for the family of instances $\mathcal{I}$ with access to oracle $\mathcal{O}$</td>
</tr>
<tr>
<td>$\log$</td>
<td>Logarithm with respect to base 2 (unless otherwise specified).</td>
</tr>
<tr>
<td>$\langle x, y \rangle$</td>
<td>Standard inner product between $x$ and $y$</td>
</tr>
<tr>
<td>$\partial f(x)$</td>
<td>The set of subgradients of $f$ at the point $x$</td>
</tr>
<tr>
<td>$\nabla f(x)$</td>
<td>An element of $\partial f(x)$, and thus the gradient of $f$ if $f$ is differentiable at $x$</td>
</tr>
<tr>
<td>$H^=(x, g)$</td>
<td>A hyperplane with normal vector $g$ with $x$ on its boundary</td>
</tr>
<tr>
<td>$H^=(x, g)$</td>
<td>A halfspace with outer normal vector $g$ with $x$ on its boundary</td>
</tr>
<tr>
<td>$G$</td>
<td>First-order chart for an oracle based on first-order information</td>
</tr>
<tr>
<td>$H$</td>
<td>Set of permissible queries for an oracle based on first-order information</td>
</tr>
<tr>
<td>$O, \Omega, \Theta$</td>
<td>$O$, $\Omega$, and $\Theta$ notation but dropping polylogarithmic terms</td>
</tr>
<tr>
<td>$\text{OPT}$</td>
<td>Minimum value of the objective function for an optimization problem</td>
</tr>
<tr>
<td>$B(x, \delta)$</td>
<td>An $L_2$-ball centered at $x$ with radius $\delta$</td>
</tr>
<tr>
<td>$A + B$ (for sets)</td>
<td>Minkowski sum of sets $A$ and $B$, i.e. ${x</td>
</tr>
<tr>
<td>$C_\delta$</td>
<td>The set $C$ blown up by $\delta$, i.e. $C + B(0, \delta)$</td>
</tr>
<tr>
<td>$C^{-\delta}$</td>
<td>The set of $\delta$-deep points in $C$, i.e. ${x \in C : B(x, \delta) \subseteq C}$</td>
</tr>
<tr>
<td>$</td>
<td>\psi\rangle$</td>
</tr>
<tr>
<td>$\langle \psi</td>
<td>$</td>
</tr>
<tr>
<td>$\langle x</td>
<td>y \rangle$</td>
</tr>
<tr>
<td>$U_f$</td>
<td>The unitary operator for computing the function $f$ in the quantum computational framework</td>
</tr>
<tr>
<td>$\mathbb{E}_P$</td>
<td>Expectation under the probability measure $P$</td>
</tr>
<tr>
<td>$I$</td>
<td>Indicator function</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>Noise level</td>
</tr>
<tr>
<td>$G = (V, E, W)$</td>
<td>A graph $G$ with nodes $V$, edges $E$, and edge weights $W$</td>
</tr>
<tr>
<td>$W_e$</td>
<td>The weight of an edge $e \in E$ for a graph $G$ with edges $E$</td>
</tr>
<tr>
<td>$\delta(v)$</td>
<td>The set of edges incident on a node $v$</td>
</tr>
<tr>
<td>$N_G(v)$</td>
<td>The neighborhood of a node $v$ in graph $G$, i.e. the set of nodes $u$ sharing an edge with $v$</td>
</tr>
<tr>
<td>$d_{G}(u,v)$</td>
<td>The shortest-path distance from node $u$ to $v$ in $G$</td>
</tr>
<tr>
<td>$\mathcal{P}_G(U,U')$</td>
<td>For subsets of nodes $U, U'$ of a graph, the set of edges between $U$ and $U'$</td>
</tr>
<tr>
<td>$A \star B$</td>
<td>The distance product between matrices $A$ and $B$</td>
</tr>
</tbody>
</table>

Table 1: Summary of Notation
Part I

Information Complexity
Chapter 1

Introduction to Information Complexity

A myriad of real-world challenges adhere to the following information-collection and decision-making pattern: One has a particular goal or objective and must make decisions in order to meet those goals in the best possible manner. To do this, one collects information to determine the possible outcomes of different decisions. For example, consider the simple problem of trying to find the best possible route to drive from New York City to Baltimore. What the best route is might be captured in different objectives: the fastest possible route, the route causing the fewest CO$_2$ emissions, the route with the fewest tolls, or some combination of those. To determine what route one should take, one might collect information about roads in the area, their speed limits, and information about traffic and congestion.

As another example, imagine a business that is deciding to expand and open new franchises. The usual business objective would be to do this in a way that maximizes future profits. Information about possible locations to build in, their prices, population and demand in surrounding areas, and operating costs would be collected in order to make the decision of how many franchises to establish and where to build them. See e.g. [27, 88] containing many interesting examples of optimization problems. To formulate such problems mathematically, we introduce the notion of an objective function $f$ and
decision variables, or simply variables, $x \in X$, where $X$ is a set of all possible decisions that could be made. The objective function models the goals, or objective, or the decision-maker, while the decision variables model the decisions to be made. Notably, $f$ is modeled as a function of $x$, so that $f : X \rightarrow \mathbb{R}$; the objective outcome, quantified as a real number that could represent cost, CO$_2$ emissions, time, or other objectives to be minimized or maximized, is determined by the decisions made. In the aforementioned example of finding the fastest route to drive from New York City to Baltimore, $X$ would be the set of all possible routes that lead from New York City to Baltimore. For some particular route $x \in X$, if the chosen objective were to minimize the time traveled, $f(x)$ would be the time it takes to drive from New York City to Baltimore by using route $x$. We can then write the optimization problem as

$$\min f(x) : x \in X,$$

which is the generic form of a mathematical optimization problem.

For the second example given above, let us provide a slightly more sophisticated formulation of the decision space and objective. Suppose that $k$ possible locations for building franchises have been identified. Building a franchise at location $i$ costs the business $c_i$ dollars, and is forecasted to bring $p_i$ dollars in profit. We can model the decision of building a franchise in location $i$ or not as a binary variable $x_i \in \{0, 1\}$. Suppose further that for each build franchise, the business can decide to make additional investments to improve profitability of that franchise, through e.g. marketing, quality upgrades, or higher quality facilities. One can let $y_i$ denote this additional optional investment, and might have forecasts $g(y_i)$ for how much additional profit is expected from investing $y_i$ additional dollars in a facility built in location $i$. Then the decisions that can be made are $(x_1, \ldots, x_n, y_1, \ldots, y_d) \in \{0, 1\}^n \times \mathbb{R}^d$, namely, whether a franchise is built in location $1, \ldots, k$ (the binary variables $x_i$), and how much additional investment beyond the minimum building costs is made at franchise $i$ if built (the $y_i$ real variables). Further, the business has a limited budget and cannot exceed a certain amount of total investment. Since $c_i$ is the cost for building a (basic) franchise at location $i$, and $y_i$ are possible additional investments at that location, the total costs for the business
are \( \text{cost}(x, y) := \sum x_i c_i + y_i \), the costs for each franchise built plus all the additional investments. It the budget of the business for its expansion is \( B \), the business has the constraint \( \text{cost}(x, y) \leq B \). The forecasted profits would be \( \text{profit}(x, y) := x_i (p_i + g(y_i)) \), the profit of building the franchise at location \( i \) plus additional profits from the additional investments \( y_i \), multiplied by 1 or 0 depending on whether a franchise is actually built at location \( i \). The optimization problem the business would want to solve is then

\[
\max \quad \text{profit}(x, y) - \text{cost}(x, y) : \\
(x, y) \in \{0, 1\}^n \times \mathbb{R}^d \\
\text{cost}(x, y) \leq B
\]

This type of optimization problem involving discrete variables (the \( x \) above) and continuous variables (the \( y \) variables) is called a mixed-integer optimization problem, defined in general later.

In light of such optimization problems, the subject of information complexity, at a high level, aims to study the following question: How much information is needed to solve a given optimization problem? In the example for finding the fastest route from New York City to Baltimore, how much information about the roads and traffic does one need to solve this? Certainly one doesn’t need the information and traffic data about every road in the world, or every road in the United States. In the business example, perhaps the \( \text{profit} \) and \( \text{cost} \) functions are not actually explicitly available, but might instead be calculated by some simulations. How many simulations to model different possible decisions would one have to run to determine an optimal, or near-optimal, decision for the business to make? In other words, how much information about the objective evaluated at different combinations do we have to collect? These are the types of questions that information complexity aims to formalize and study. In particular, we are interested in answering these questions in two ways:

1. Design strategies that guarantee provably optimal or near-optimal solutions while only collecting as little information as possible.
2. Proving that one must collect some at least minimal amount of information in order to reliably solve some optimization problem to an acceptable threshold of (near) optimality.

With these notions in mind, we now move on to a more mathematically formal treatment of these notions.

1.1 Background and Elementary Definitions

We consider the problem class of mixed-integer convex optimization:

$$\inf \{ f(x, y) : (x, y) \in C, (x, y) \in \mathbb{Z}^n \times \mathbb{R}^d \}$$  \hspace{1cm} (1.1)

where $f : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}$ is a convex (possibly nondifferentiable) function and $C \subseteq \mathbb{R}^n \times \mathbb{R}^d$ is a closed, convex set. Given $\varepsilon \geq 0$, we wish to report a point $(x, y) \in C \cap (\mathbb{Z}^n \times \mathbb{R}^d)$ such that $f(x, y) \leq f(x', y') + \varepsilon$ for all $(x', y') \in C \cap (\mathbb{Z}^n \times \mathbb{R}^d)$. Such a point will be called an $\varepsilon$-approximate solution and points in $C \cap (\mathbb{Z}^n \times \mathbb{R}^d)$ will be called feasible solutions.

We say that $x_1, \ldots, x_n$ are the integer-valued decision variables or simply the integer variables of the problem, and $y_1, \ldots, y_d$ are called the continuous variables.

The notion of information complexity (a.k.a. oracle complexity or analytical complexity) goes back to foundational work by Nemirovski and Yudin [82] on convex optimization (without integer variables) and is based on the following. An algorithm for reporting an $\varepsilon$-approximate solution to an instance $(f, C)$ must be “given” the instance somehow. Allowing only instances with explicit, algebraic descriptions (e.g., the case of linear programming) can be restrictive in some settings. To work with more general, nonlinear instances, the algorithm is allowed to make queries to an oracle to collect information about the instance. More formally, we have the following definition.

**Definition 1.** An oracle $O$ for an optimization problem class $\mathcal{I}$ is given by a family $\mathcal{Q}$ of possible queries along with a set $H$ of possible answers or responses. A query $q \in \mathcal{Q}$ is a function $q : \mathcal{I} \to H$. We say that $q(I) \in H$ is the answer or response to the query $q$ for the instance $I \in \mathcal{I}$. 
Any algorithm using such an oracle to find an \( \varepsilon \)-approximate solution for an instance makes queries about the instance in a sequence according to some strategy depending on the queries made and answers received, which we define formally as its query strategy.

**Definition 2.** A query strategy is a function \( D : (Q \times H)^* \to Q \), where \((Q \times H)^*\) denotes the set of all finite sequences over \( Q \times H \), including the empty sequence. The transcript \( \Pi(D, I) \) of a strategy \( D \) on an instance \( I = (f, C) \) is the sequence of query and response pairs \( (q_i, q_i(I)) \), \( i = 1, 2, \ldots \) obtained when one applies \( D \) on \( I \), i.e., \( q_1 = D(\emptyset) \) and \( q_i = D((q_1, q_1(I)), \ldots, (q_{i-1}, q_{i-1}(I))) \) for \( i \geq 2 \).

If different instances with no common \( \varepsilon \)-approximate solution produce the same transcript for the queries an algorithm has made, then the algorithm cannot tell them apart and will be unable to reliably report an \( \varepsilon \)-solution for those instances after those queries. The goal is to design a query strategy that can report an \( \varepsilon \)-approximate solution after making the smallest number of queries. This motivates the following definition of information complexity:

**Definition 3.** Given a family of instances \( I \) and access to an oracle \( O \), the \( \varepsilon \)-information complexity \( \text{icomp}_\varepsilon(D, I, O) \) of an instance \( I \) for a query strategy \( D \), is defined as the minimum natural number \( k \) such that the set of all instances in \( I \) for which \( O \) returns the same responses as the instance \( I \) to the first \( k \) queries of \( D \) have a common \( \varepsilon \)-approximate solution. The \( \varepsilon \)-information complexity of the problem class \( I \) with respect to the oracle \( O \), is defined as

\[
\text{icomp}_\varepsilon(I, O) := \inf_D \sup_{I \in I} \text{icomp}_\varepsilon(D, I, O)
\]

where the infimum is taken over all query strategies.

Thus, to prove an upper bound \( u \) on \( \text{icomp}_\varepsilon(I, O) \), it suffices to construct a query strategy that requires, in the worst case, at most \( u \) queries to narrow down to a collection of instances that all have a common \( \varepsilon \)-approximate solution. On the other hand, to establish a lower bound of \( \ell \) on \( \text{icomp}_\varepsilon(I, O) \), one needs to show that for any
query strategy $D$, there exists a collection of instances in $\mathcal{I}$ that give the same responses to the first $\ell$ queries of $D$ (on these instances), and there is no point in $\mathbb{R}^n \times \mathbb{R}^d$ that is a common $\varepsilon$-approximate solution to all these instances.

It is not hard to see that we need to restrict the set of possible instances $\mathcal{I}$ in order to have meaningful (finite) information complexity $\text{icomp}_\varepsilon(\mathcal{I}, \mathcal{O})$. We will focus on the following standard parameterization.

**Definition 4.** Define $\mathcal{I}_{n,d,R,\rho,M}$ to be the set of all instances of (1.1) such that:

(i) $C$ is a compact, convex set contained\(^1\) in the box $\{z \in \mathbb{R}^n \times \mathbb{R}^d : \|z\|_\infty \leq R\}$. The case $C = \{z \in \mathbb{R}^n \times \mathbb{R}^d : \|z\|_\infty \leq R\}$ will be called unconstrained.

(ii) There is an optimal solution $(x^*, y^*)$ of the instance such that $\hat{y} \in \mathbb{R}^d$ satisfying $\{(x^*, y) : \|y - \hat{y}\|_\infty \leq \rho\} \subseteq C$. In other words, there is a “strictly feasible” point $(x^*, \hat{y})$ in the same fiber as the optimum $(x^*, y^*)$.

(iii) $f$ is Lipschitz continuous with respect to the $\|\cdot\|_\infty$-norm with Lipschitz constant $M$ on $\{x\} \times [-R, R]^d$ for all $x \in [-R, R]^n \cap \mathbb{Z}^n$. In other words, for any $(x, y), (x, y') \in (\mathbb{Z}^n \times \mathbb{R}^d) \cap [-R, R]^{n+d}$ with $\|y - y'\|_\infty \leq R$, $|f(x, y) - f(x, y')| \leq M\|y - y'\|_\infty$.

While we introduce information complexity allowing for any general choice of oracle, the standard oracle that has been studied over the past several decades for convex optimization is the so-called *(full-information) first-order oracle*, which has two types of queries indexed by points in $\mathbb{R}^n \times \mathbb{R}^d$: i) a separation oracle query indexed by a point $z \in \mathbb{R}^{n+d}$ reports “YES” if $z \in C$ and otherwise reports a separating hyperplane for $z$ and $C$, ii) a subgradient oracle query indexed by a point $z \in \mathbb{R}^{n+d}$ reports $f(z)$ and a subgradient for $f$ at $z$. Tight lower and upper bounds (differing by only a small constant factor) on the number of queries required were obtained by Nemirovski and Yudin in their seminal work [82] for the case with no integer variables; roughly speaking,

\(^1\)This is a standard assumption in the analysis of optimization algorithms - if no such bound is assumed, then it can be shown that no algorithm can report a good solution within a guaranteed number of steps for every instance [85]. Alternatively, one may give the convergence rates in terms of the distance of the initial iterate of the algorithm and the optimal solution (one can think of $R$ as an upper bound on this distance). The results of this thesis can also be formulated in this language via simple modifications.
the bound is $\Theta\left(d \log\left(\frac{1}{\varepsilon}\right)\right)$. These insights were extended to the mixed-integer setting in [90, 9, 8], with the best known lower and upper bounds stated in [8].

Observe that the response to any separation/subgradient query is a vector in $\mathbb{R}^{n+d}$. Thus, each query reveals at least $n + d$ bits of information about the instance. A more careful accounting that measures the “amount of information” accrued would track the total number of bits of information obtained as opposed to just the total number of oracle queries made. A natural question, posed in [8], is whether the bounds from the classical analysis would change if one uses this new measure of the total number of bits, as opposed to the number of queries. The intuition, roughly, is that one should need a factor $(n + d) \log\left(\frac{1}{\varepsilon}\right)$ larger than the number of first-order queries, because one should need to probe at least $\log\left(\frac{1}{\varepsilon}\right)$ bits in $n + d$ coordinates to recover the full subgradient/separating hyperplane (up to desired approximations). We attempt to make some progress on this question in this paper.

The above discussion suggests that one should consider oracles that return a desired bit of a desired coordinate of the separating hyperplane vector or subgradient. However, one can imagine making other binary queries on the instance; for example, one can pick a direction and ask for the sign of the inner product of the subgradient and this direction. In fact, one can consider more general binary queries that have nothing to do with subgradients/separating hyperplanes. If one allows all possible binary queries, i.e., one can use any function from the space of instances to $\{0, 1\}$ as a query, then one can simply ask for the appropriate bits of the true minimizer and in $O((n + d) \log(1/\varepsilon))$ queries, one can get an $\varepsilon$-approximate solution. A matching lower bound follows from a fairly straightforward counting argument. Thus, allowing for all possible binary queries gives the same information complexity bound as the original Nemirovski-Yudin bound with subgradient queries in the $n = 0$ (no integer variables) case, but is an exponential improvement when $n \geq 1$ (see [8] and the discussion below). What this shows is that the bounds on information complexity can be quite different under different oracles. With all possible binary queries, while each query reveals only a single bit of information, the queries themselves are a much richer class and this compensates to give the same bound in the continuous case and exponentially better bounds in the presence of integer
variables. Thus, to get a better understanding of this trade-off, we restrict to queries that still extract information by only acting “locally”.

**Oracles based on first-order information.** Our first contribution is formalizing this notion of general “local” queries. While we focus on first-order information, our framework can be readily extended to consider, for example, information from higher-order derivatives.

**Definition 5.** An oracle using first-order information $O(G, H)$ consists of two parts:

1. For every $z \in [-R, R]^{n+d}$, there exist three maps $g_z^{\text{sep}} : \mathcal{I}_{n,d,R,\rho,M} \to \mathbb{R}^{n+d}$, $g_z^{\text{val}} : \mathcal{I}_{n,d,R,\rho,M} \to \mathbb{R}$, and $g_z^{\text{sub}} : \mathcal{I}_{n,d,R,\rho,M} \to \mathbb{R}^{n+d}$ such that for all $(f, C) \in \mathcal{I}_{n,d,R,\rho,M}$ the following properties hold.

   - (a) $C \subseteq \{ z' \in \mathbb{R}^{n+d} : \langle g_z^{\text{sep}}(f, C), z' \rangle < \langle g_z^{\text{sep}}(f, C), z \rangle \}$ if $z \notin C$ and $g_z^{\text{sep}}(f, C) = 0$ if $z \in C$. In other words, $g_z^{\text{sep}}(f, C)$ returns a (normal vector to a) separating hyperplane if $z \notin C$. We will assume that a nonzero response $g_z^{\text{sep}}(f, C)$ has norm 1, since scalings do not change the separation property.

   - (b) $g_z^{\text{val}}(f, C) = f(z)$. In other words, $g_z^{\text{val}}(f, C)$ returns the function value for $f$ at $z$.

   - (c) $g_z^{\text{sub}}(f, C) \in \partial f(z)$, where $\partial f(z)$ denotes the subdifferential (the set of all subgradients) of $f$ at $z$. In other words, $g_z^{\text{sub}}(f, C)$ returns a subgradient for $f$ at $z$.

   Such maps will be called first-order maps. A collection of first-order maps, one for every $z$, is called a first-order chart and will be denoted by $G$.

2. There are three sets of functions $H^{\text{sep}}$, $H^{\text{val}}$, and $H^{\text{sub}}$ and with domains $\mathbb{R}^{n+d}$, $\mathbb{R}$ and $\mathbb{R}^{n+d}$ respectively. We will use the notation $H = H^{\text{sep}} \cup H^{\text{val}} \cup H^{\text{sub}}$. $H$ will be called the collection of permissible queries of the oracle.

An algorithm for instances of (1.1) using $O(G, H)$ can, at any iteration, chooses a point $z$ and a function $h \in H$ and receive the response $h(g_z^{\text{sep}}(f, \hat{C})), h(g_z^{\text{val}}(f, \hat{C}))$
or \( h(g_{z}^{\text{sub}}(\hat{f}, \hat{C})) \), depending on whether \( h \in \mathcal{H}^{\text{sep}}, h \in \mathcal{H}^{\text{val}} \) or \( h \in \mathcal{H}^{\text{sub}} \), where \( \hat{f} \) and \( \hat{C} \) are the objective function and feasible region, respectively, of the unknown instance. Hence, queries to an oracle \( \mathcal{O}(\mathcal{G}, \mathcal{H}) \) using first-order information are indexed by \((z, h)\), \( z \in \mathbb{R}^n \times \mathbb{R}^d, h \in \mathcal{H} \). Since the goal of this paper is to provide bounds for different types of such oracles, i.e., with different permissible queries \( \mathcal{H} \), let us define some cases of interest.

**Definition 6** (Examples of oracles).

1. (Full-information first-order oracle) When \( \mathcal{H} \) consists only of the identity functions, i.e., \( h_{\text{sep}}(g_{z}^{\text{sep}}(\hat{f}, \hat{C})) = g_{z}^{\text{sep}}(\hat{f}, \hat{C}), h_{\text{val}}(g_{z}^{\text{val}}(\hat{f}, \hat{C})) = g_{z}^{\text{val}}(\hat{f}, \hat{C}) \) and \( h_{\text{sub}}(g_{z}^{\text{sub}}(\hat{f}, \hat{C})) = g_{z}^{\text{sub}}(\hat{f}, \hat{C}) \), we recover a full-information first-order oracle, which we will write as \( \mathcal{O}_{\text{full}} \).

2. (Bit oracle) Let \( \mathcal{H}_{\text{bit}}^{*} \) be the set of binary queries that return a desired bit (of a desired coordinate) of the binary representation of \( g_{z}^{\text{sep}}(\hat{f}, \hat{C}), g_{z}^{\text{val}}(\hat{f}, \hat{C}) \) or \( g_{z}^{\text{sub}}(\hat{f}, \hat{C}) \). Let \( \mathcal{H}_{\text{bit}}^{*} \) be the shifted bit oracle that additionally returns a desired bit of \( g_{z}^{\text{val}}(\hat{f}, \hat{C}) + u \), for any \( u \in \mathbb{R} \), i.e. \( \mathcal{H}_{\text{bit}}^{*} \) allows querying a bit of the function value shifted by some number.

3. (Inner product threshold queries) Let

\[
\mathcal{H}_{\text{dir}} := \{ h_{u,c}^{\text{sep}} : h_{u,c}^{\text{sep}}(g_{z}^{\text{sep}}(\hat{f}, \hat{C})) = \text{sgn}((u, g_{z}^{\text{sep}}(\hat{f}, \hat{C})) - c), u \in \mathbb{R}^{n+d}, c \in \mathbb{R} \} \\
\cup \{ h_{u,c}^{\text{val}} : h_{u,c}^{\text{val}}(g_{z}^{\text{val}}(\hat{f}, \hat{C})) = \text{sgn}(u \cdot g_{z}^{\text{val}}(\hat{f}, \hat{C}) - c), u \in \mathbb{R}, c \in \mathbb{R} \} \\
\cup \{ h_{u,c}^{\text{sub}} : h_{u,c}^{\text{sub}}(g_{z}^{\text{sub}}(\hat{f}, \hat{C})) = \text{sgn}((u, g_{z}^{\text{sub}}(\hat{f}, \hat{C})) - c), u \in \mathbb{R}^{n+d}, c \in \mathbb{R} \},
\]

where \( \text{sgn} \) denotes the sign function, be the set of binary queries that answers whether the inner product of the separating hyperplane, function value or subgradient, with a vector or a number of choice \( u \) or \( u \) in the appropriate space, is at least some value \( c \) or not. We write these as \( \mathcal{H}_{\text{dir}} \) since these queries allow for the choice of a “direction” \( u \), or a number \( u \) in the function value case, as part of the query.
4. When $\mathcal{H}$ is the set of all possible binary functions on $\mathbb{R}^n \times \mathbb{R}^d$ for the separating hyperplanes, $\mathbb{R}$ for the functions values, and $\mathbb{R}^n \times \mathbb{R}^d$ for the subgradients, we will call the resulting oracle the general binary oracle based on $\mathcal{G}$.

These now give us a variety of oracles using first-order information, that clearly provide very different information for each query depending on the choice of permissible queries $\mathcal{H}$. Note that different first-order charts will result in different oracles of each of these types that may give different answers at any point, depending on which separating hyperplane/subgradient the oracle’s first-order map selects at those points for that instance.

1.2 Existing Work and Results for Continuous Convex Optimization

In this subsection, we provide cover some notable existing work about the first-order information complexity of convex optimization in the presence of only continuous decision variables. As noted already, past work has focused on results given access to a full-information first-order oracle. Notably, seminal work by Nemirovski and Yudin, which can be found in [82], refined in [19] showed that the information complexity of unconstrained continuous convex optimization in this full-information first-order model, for $M$-Lipschitz continuous functions over $\mathbb{R}^d$ with minimizer in $B_\infty(0, R)$, is at least

$$\Omega\left(d \log\left(\frac{MR}{\epsilon}\right)\right).$$

Cutting plane schemes, well-described e.g. in [85, Section 3.2] using gradients as separating hyperplanes to the optimal solution for convex optimization, are able to achieve this complexity. This section will demonstrate the role that convex geometry plays in establishing these results, creating a strong link between convex geometry and optimization that will continue through the remainder of Part I of this thesis.
1.2.1 Information Complexity of Continuous Convex Optimization under the Full-Information First-Order Oracle

To familiarize the reader with some of the ideas and arguments used within the field of information complexity, we devote this subsection to proving that the information complexity of continuous unconstrained convex optimization is $\Theta(d \log(\frac{MR}{\varepsilon}))$, or more precisely that

$$icomp_\varepsilon(I_{n=0,d,R,\rho=R,M}, \mathcal{O}_{full}) = \Theta\left(d \log\left(\frac{MR}{\varepsilon}\right)\right).$$

(1.2)

As mentioned above, this consists of two parts – firstly, showing that there exists and algorithm achieving this complexity, and secondly proving that this is the best possible complexity.

Let us first establish the algorithm achieving the stated complexity, which we will call the center of gravity method. This algorithm will illustrate some interesting links between convex geometry and optimization.

Gradients of Convex Functions and Separating Hyperplanes to the Optima

Consider some convex function $f$. The first ingredient we need is the following result that allows the interpretation of a subgradient of a convex function as providing a (weakly) separating hyperplane to the global minimizers of $f$. For two vectors $x, g \in \mathbb{R}^d$, denote by $H^\leq(x, g)$ the halfspace with outward normal vector $g$ that has $x$ on its boundary (or is centered at $x$). More precisely,

$$H^\leq(x, g) := \{y \in \mathbb{R}^d : \langle y, g \rangle \leq \langle x, g \rangle\}.$$

The following lemma holds:

Lemma 7. Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function. For $x \in \mathbb{R}^d$, let $\nabla f(x)$ be a subgradient of $f$ at $x$. Then we have

$$\{y : f(y) \leq f(x)\} \subseteq H^\leq(x, \nabla f(x)).$$
Note that we are slightly abusing notation here by letting $\nabla f(x)$ denote any element in the subgradient $\partial f(x)$. In the forthcoming discussions, it won’t matter which element of $\partial f(x)$ is used, so simplifying notation in this way causes no issues.

In simple terms, for any $x \in \mathbb{R}^d$, all points $y$ achieving a function value of $f(x)$ or lower must be in the halfspace induced by $\nabla f(x)$ centered at $x$. In particular, the minimizer of $f$, call it $x^*$, must lie in that halfspace. We provide a visualization of the result Lemma 7 in Figure 1.1 and leave the proof to the interested reader as an exercise in convexity.

Figure 1.1: An illustration of how $\nabla f(x)$ (arrow) at $x$ (blue) induces a halfspace (yellow) in which the optimal solution $x^*$ (red) must lie, with the ellipses representing the boundaries of level sets of $f$.

Cutting plane schemes make use of this fact to locate $\varepsilon$-solutions for convex functions quickly by obtaining gradients at different points (iteratively) and intersecting the halfspaces that must contain the optimal solution. Once that intersection is small
enough, one reports a point in that intersection (typically the queried point with the lowest observed function value) and can often provide some guarantee for the function value at the point reported. In particular, such schemes maintain at each iteration \( i \) some polytope \( P_i \) that is guaranteed to contain a global minimizer of \( f \). With each iteration, the next such polytope \( P_{i+1} \) is typically updated by selecting some point \( x_{i+1} \), computing or querying \( \nabla f(x_{i+1}) \) and then setting \( P_{i+1} = P_i \cap H^\leq(x_{i+1}, \nabla f(x_{i+1})) \). See [85, Section 3] for a great exposition into these methods. The center of gravity method is one particular such method that chooses the next point \( x_{i+1} \) to be queried as the center of gravity of the polytope \( P_i \).

**Definition 8.** Let \( P \subset \mathbb{R}^n \) be a (bounded) polytope. The center of gravity of \( P \) is the point

\[
\text{COG}(P) := \mathbb{E}_{U_P}(X),
\]

where \( U_P \) denotes the uniform measure over \( P \).

If one considers \( P \) to be a physical object with uniform density, the point \( x \) is indeed the “center of gravity” of \( P \) in the physical sense. This is a generalization of the notion of a midpoint of a line, as is used in bisection search; indeed one can think of the center of gravity method as a generalization of bisection search into higher dimensions – in one dimension, setting \( x_{i+1} \) to be the center of gravity of \( P_i \) would simply be the midpoint of the interval \( P_i \) (noting that a polytope in \( \mathbb{R} \) is just an interval), and then updating \( P_{i+1} \) by intersecting it with the halfspace induced by \( \nabla f(x_{i+1}) \) yields exactly a bisection search. In the one dimensional case, it is obvious that the length of the interval shrinks by a factor of \( \frac{1}{2} \) each iteration. In higher dimension, however, it is unclear whether that should happen. However, a similar and only slightly weaker guarantee can indeed be given in arbitrary dimension \( d \), due to Grünbaum [53]:

**Lemma 9.** Let \( P \subset \mathbb{R}^d \) be a polytope with center of gravity \( x^* \). Then for any halfspace \( H \) with \( x^* \) on its boundary,

\[
\frac{\text{vol}(P \cap H)}{\text{vol}(P)} \leq 1 - \left( \frac{d}{d+1} \right)^d,
\]
or as a direct consequence for any $d$

\[
\frac{\text{vol}(P \cap H)}{\text{vol}(P)} \leq 1 - \frac{1}{e} < \frac{2}{3}
\]

This guarantees that the volume of $P_{i+1}$ is at most $\frac{e-1}{e} < \frac{2}{3}$ the volume of $P_i$ when using the center of gravity method.

Since we assume the optimal solution for $f$ is in the box $[-R, R]^d$, a standard and necessary assumption, the algorithm begins with $P_0 := [-R, R]^d$ and then sets $P_{i+1} := P_i \cap H^\leq(x_i^+, \nabla f(x_i^+))$, where $x_i^+$ is the center of gravity of $P_i$. We formally state the algorithm in Algorithm 1.

Algorithm 1 Center of Gravity Method

1. Initialize $P := [-R, R]^n$. WHILE $\text{vol}(P) > \left(\frac{\varepsilon}{2M}\right)^d$, do:
   (a) Let $x_t \in P_t$ be the center of gravity of $P_t$ as defined in 8.
   (b) Query the full-information first-order oracle at $x_t$, and store the answers $f(x_t)$ and $\nabla f(x_t)$.
   (c) Set $P_{t+1} := P_t \cap H^\leq(x_t, \nabla f(x_t))$, and update $P \leftarrow P_{t+1}$, $t \leftarrow t + 1$.
2. Report the point with the smallest observed function value, $\tilde{x}^* := \arg\min_i f(x_i^*)$.

Using Lemma 9, noting that $\text{vol}(P_0) = (2R)^d$, it follows that $\text{vol}(P_k) \leq \left(\frac{2}{3}\right)^k(2R)^d$. After $2d \log\left(\frac{4MR}{\varepsilon}\right)$ iterations, we thus have

\[
\text{vol}(P_k) \leq \left(\frac{e-1}{e}\right)2d\log\left(\frac{4MR}{\varepsilon}\right)2R^d = \left(\frac{e-1}{e}\right)2d(4MR)^{\log(e-1)}(\frac{4MR}{\varepsilon})^d2R
\]

\[
\left(\frac{(4MR)^{\log(\varepsilon)+1}}{(4MR)^{2d}2R}\right)^d = \left(\frac{(4MR)^{\log(\varepsilon)+1}}{(4MR)^{2d}2R}\right)^d
\]

\[
\leq \left(\frac{\varepsilon}{2M}\right)^d
\]

(1.3)

Finally, we need to show that this volume is small enough to guarantee that the queried point with the smallest observed function value, $\tilde{x}^* := \arg\min_i f(x_i^*)$ is indeed an $\varepsilon$-approximate solution to $f$.

Lemma 10. Suppose that $k$ iterations of the center of gravity method have been made,
with queried points $x_1, \ldots, x_k$, and the polytope maintained by the method has volume

$$\text{vol}(P_k) \leq \left(\frac{\varepsilon}{2M}\right)^d.$$ 

Then $\tilde{x}^* := \text{argmin}_i f(x_i^*)$ is an $\varepsilon$-approximate solution to $f$.

**Proof of Lemma 10.** Let $x^*$ denote the true global optimum of $f$. Note that $x^* \in P_k$. Consider the box $\{x^*\} + [-\left(\frac{\varepsilon}{2M}\right), \left(\frac{\varepsilon}{2M}\right)]$, where $+$ denotes the Minkowski sum. Since this box has volume $\left(\frac{\varepsilon}{2M}\right)^d$, some point in this box lies on the boundary of $P_k$. Take such a point $y$ on the boundary. Since $y$ is also in the aforementioned box, $\|y - x^*\|_{\infty} \leq \frac{\varepsilon}{2M}$. The assumed $M$-Lipschitz continuity of $f$ (with respect to the $\infty$-norm) then guarantees that $f(y) \leq f(x^*) + \varepsilon$. Since the boundary of $P_k$ consists of the boundaries of halfspaces induced by queried points and the gradients observed, there is some queried point $x_j$ such that $y$ lies on the boundary of $H \leq (x_i, \nabla f(x_i))$. Due to Lemma 7, one obtains

$$f(x_j) \leq f(y) \leq f(x^*) + \varepsilon,$$

and so the lemma is proven.

Finally, we can make a statement about guarantees for the center of gravity method (Algorithm 1 to get an upper bound on the information complexity claimed in (1.2).

**Theorem 11.** With $\mathcal{I}_{n=0,d,R,\rho} = R, M$ as defined previously,

$$\text{icomp}_{\varepsilon}(\mathcal{I}_{n=0,d,R,\rho} = R, M, \mathcal{O}_{\text{full}}) \leq 2d \log\left(\frac{4MR}{\varepsilon}\right).$$

The center of gravity method proves this theorem. Namely, equation (1.3) shows that the volume of the polytope maintained by the method has volume at most $\left(\frac{\varepsilon}{2M}\right)^d$ after $2d \log\left(\frac{4MR}{\varepsilon}\right)$ iterations, each of which makes one query to the full-information first-order oracle, and Lemma 10 guarantees that the solution reported in the end is indeed an $\varepsilon$-approximate solution. Hence, this algorithm certifies that indeed
\[ \text{icomp}_\varepsilon(\mathcal{I}_{n=0,d,R,\rho=R,M},\mathcal{O}_{\text{full}}) \leq 2d \log\left(\frac{4MR}{\varepsilon}\right). \]

1.2.2 Lower-Bounding the Information Complexity of Continuous Convex Optimization

To show the claimed result in (1.2), after having proved Theorem 11 what remains to be shown is the following matching lower-bound:

**Theorem 12.** With \( \mathcal{I}_{n=0,d,R,\rho=R,M} \) as defined previously,

\[ \text{icomp}_\varepsilon(\mathcal{I}_{n=0,d,R,\rho=R,M},\mathcal{O}_{\text{full}}) \leq \Omega\left(d \log\left(\frac{MR}{\varepsilon}\right)\right). \]

This subsection will be devoted to discussing this result.

We will establish this result using a particular type of full-information first-order oracle, as in [19]. Recall that a first-order oracle as from Definition 5 can give its answer using any subgradient of the objective function \( f \), but uses a particular first-order chart \( \mathcal{G} \) that determines, for each instance, which subgradient it reports at a given point \( x \); the oracle must have some mechanism to select a particular subgradient to use to provide its answers at points where \( f \) is nondifferentiable. In practice, this \( \mathcal{G} \) may be unknown, however assuming some particular \( \mathcal{G} \) for proving lower-bounds is reasonable, since the case where the oracle uses an unknown first-order is only more difficult for an algorithm to solve. We will build a particular family of nonsmooth functions to prove the lower bound in 12, and will fix a particular \( \mathcal{G} \) to obtain a specific full-information first order oracle, which we will call a single-coordinate oracle, following [19].

**Definition 13.** A single-coordinate oracle, denoted \( \mathcal{O} \), is a full-information first-order oracle such that for all \( x \in \mathbb{R}^d \) the subgradient \( g \in \partial f(x) \) it reports is, when possible, supported on the least possible coordinate axis; that is, if \( f \) is the objective function for instance \( I \), then \( \mathcal{G}(I, x) = \lambda e_i \) for the smallest \( 1 \leq i \leq d \) and some \( \lambda \in \mathbb{R} \), if there exists such a subgradient in \( \partial f(x) \). For all other \( x \), \( \mathcal{G} \) may report any \( g \in \partial f(x) \).

Put simply, a single-coordinate oracle is a full-information first-order oracle that, when possible, reports a scaled standard basis vector as the subgradient, and if there are
multiple options thereof, reports the one in the minimal coordinate. The main purpose of fixing this oracle is so that it is easy to reason about what responses the oracle will give, and so it will be easy to decide when two instances will have exactly the same answers to queries made and be indistinguishable.

We will prove Theorem 12 by explicitly building a family of functions $F$ such that when fewer than $c \cdot d \log \frac{MR}{\varepsilon}$ (adaptive) queries to $\widetilde{O}$ are made, there exist two instances in $F$ that give all the same answers to the queries, i.e., are indistinguishable, and have disjoint $\varepsilon$ solutions. Recalling the definition of information complexity (see 3), this will be sufficient to prove the desired lower bound.

This is formalized in the following theorem that will imply Theorem 12:

**Theorem 14.** Let $L, R, \varepsilon > 0$. There exists a finite family $F$ of Lipschitz convex functions on the $L^\infty$-ball $B_\infty(0, R)$ with Lipschitz constant $M$ in the $L^\infty$ norm, a single-coordinate local oracle $\widetilde{O}$ such that it satisfies the follows:

1. all the members in the family have the same minimum value $f^*$.

2. for $f_1, f_2 \in F$ and $f_1 \neq f_2$, $S((f_1, B_\infty(0, R)), \varepsilon) \cap S((f_2, B_\infty(0, R)), \varepsilon) = \emptyset$.

3. if we make a query sequence $D$ with length $\lceil \gamma d \log \frac{LR}{\varepsilon} \rceil$ based on $\widetilde{O}$ for some constant $\gamma > 0$, then there are two functions $f_1, f_2 \in F$ such that $\Pi_k(D, (f_1, B_\infty(0, R)))$ and $\Pi_k(D, (f_2, B_\infty(0, R)))$ are equal for any $k$ less than the length of $D$, but $S((f_1, B_\infty(0, R)), \varepsilon) \cap S((f_2, B_\infty(0, R)), \varepsilon) = \emptyset$. The minimum function value reported to the queries in $D$ is strictly larger than $f^* + \varepsilon$.

We build this function class $F$ as follows.

**Definition 15.** Let $M, R > 0$. We define a family of convex functions indexed by all finite binary strings (including the empty string) that are Lipschitz continuous with Lipschitz constant at most $M$ over the interval $[-R, R]$. The definition proceeds inductively on the length of the binary strings, starting with the empty string. We define the interval $I_{\text{empty}} = [-R, R]$ and define $f_{\text{empty}}(x) = M|x|$. Next, consider $s$ to be a binary string of length at least 1. We define $I_s$ and $f_s$ in the following way, using $|s|$ to denote the
length of the string.

If \( s = s'0 \), i.e., the last bit in \( s \) is 0 with prefix \( s' \), then set \( I_s = \left[ \frac{3\ell_{s'} + u_{s'}}{4}, \frac{\ell_{s'} + u_{s'}}{2} \right] \), where \( \ell_{s'} \leq u_{s'} \) are the endpoints of \( I_{s'} \). Define

\[
 f_s(x) := \max \left\{ f_{s'}(x), f_{s'} \left( \frac{3\ell_{s'} + u_{s'}}{4} \right) - \frac{M}{2 |s|} \left( x - \frac{3\ell_{s'} + u_{s'}}{4} \right), f_{s'}(u_{s'}) + \frac{M}{2 |s|} (x - u_{s'}) \right\}.
\]

If \( s = s'1 \), i.e., the last bit in \( s \) is 1 with prefix \( s' \), then set \( I_s = \left[ \frac{\ell_{s'} + u_{s'}}{2}, \frac{\ell_{s'} + 3u_{s'}}{4} \right] \), where \( \ell_{s'} \leq u_{s'} \) are the endpoints of \( I_{s'} \). Define

\[
 f_s(x) := \max \left\{ f_{s'}(x), f_{s'}(\ell_{s'}) - \frac{M}{2 |s|} (x - \ell_{s'}), f_{s'} \left( \frac{\ell_{s'} + 3u_{s'}}{4} \right) + \frac{M}{2 |s|} \left( x - \frac{\ell_{s'} + 3u_{s'}}{4} \right) \right\}.
\]

The definition is quite complicated, but the idea behind the construction is quite simple. One “level” of the iterative procedure of defining these functions is illustrated in Figure 1.2. The idea is to create functions that are indistinguishable outside of certain intervals. In the figure, notice that outside of \( I_{s'0} \) and \( I_{s'1} \), \( f_{s'0} \) and \( f_{s'1} \) take on exactly the same values and have the same subgradients. On \( I_{s'0} \) and \( I_{s'1} \), they respectively look like the absolute value function, and so we can iteratively define \( f_{s'00} \), \( f_{s'01} \) and \( f_{s'10} \), \( f_{s'11} \), which will again only differ on subintervals of \( I_{s'0} \) and \( I_{s'1} \), respectively. Note also that \( f_{s'0} \) and \( f_{s'1} \) alve disjoint \( \varepsilon \)-solutions for appropriate choice of \( \varepsilon \) (following the figure, setting \( \varepsilon \) to be less than the distance between the minimum of \( f_{s'0} \) and the value where \( f_{s'0} \) and \( f_{s'1} \) intersect in the middle is sufficient). The observations should help convey that indeed these are good candidates for building a class of functions that creates ambiguity between functions that have disjoint \( \varepsilon \)-solutions.

The given definition was only for \( d = 1 \). Next, we extend this definition to \( d \) dimensions by taking coordinate-wise maxima of these 1-dimensional we just defined. Given \( d \) binary strings \( s_1, \ldots, s_d \), define \( f_{s_1, s_2, \ldots, s_d} : \mathbb{R}^d \to \mathbb{R} \) and \( B_{s_1, \ldots, s_d} \subseteq \mathbb{R}^d \) as follows:
Figure 1.2: A picture illustrating the construction of $f_s, I_s$ from $f_{s'}, I_{s'}$. 
\[ f_{s_1, s_2, \ldots, s_d}(y) := \max \{ f_{s_1}(y_1), f_{s_2}(y_2), \ldots, f_{s_d}(y_d) \}, \quad B_{s_1, \ldots, s_d} := I_{s_1} \times \ldots \times I_{s_d}, \quad (1.4) \]

and we will let \( \mathcal{F} \) denote the family of these functions. We first provide an important lemma summarizing a few key properties of this class of functions.

**Lemma 16.** Given \( M, R > 0 \), consider the family of functions \( f_{s_1, \ldots, s_d} \) and sets \( B_{s_1, \ldots, s_d} \) defined in (1.4) and define, where \( s_1, \ldots, s_d \) are finite binary strings. The following properties hold:

1. Every function \( f_{s_1, \ldots, s_d} \) has Lipschitz constant with respect to the \( \ell_\infty \) norm bounded by \( M \).

2. For binary strings \( s_1, \ldots, s_d \), \( B_{s_1, \ldots, s_d} \) is a hypercuboid with length \( \frac{2R}{4^{|s_j|}} \) in coordinate \( j \).

3. The center of \( B_{s_1, \ldots, s_d} \) is the global minimizer of \( f_{s_1, \ldots, s_d} \) over \( \mathbb{R}^d \), where the center is defined as the cartesian product of the midpoints of \( I_{s_1}, \ldots, I_{s_d} \).

4. Let \( k \in \mathbb{Z}_+ \). All functions \( f_{s_1, \ldots, s_d} \) such that \( s_1, \ldots, s_d \) all have length \( k \) have the same minimum value. Moreover, for \( k_1 < k_2 \), the minimum corresponding to \( k_1 \) is strictly smaller than the minimum corresponding to \( k_2 \).

5. The difference between the maximum value and minimum value of \( f_{s_1, \ldots, s_d} \) over \( B_{s_1, \ldots, s_d} \) is at least \( \min \{ MR, \frac{MR}{8^{|s_i|}} \} \).

6. Let \( s_1, \ldots, s_d \) and \( t_1, \ldots, t_d \) be binary strings all of the same length such that there exists \( i \) such that \( s_i \neq t_i \). Then \( \text{int}(B_{s_1, \ldots, s_d}) \cap \text{int}(B_{t_1, \ldots, t_d}) = \emptyset \).

These properties are all either immediate or straightforward to verify, and so we forego proving them here. Importantly, any function in the class of functions we consider is fully defined by its associated list of binary strings \( s_1, \ldots, s_d \). The next lemma provides some important connections of these binary strings with the intervals and functions they define.
Lemma 17. Given \( M, R > 0 \), family of functions \( f_{s_1, \ldots, s_d} \) and sets \( B_{s_1, \ldots, s_d} \) defined in (1.4). For any binary string \( s \) and \( y \in [-R, R] \), define depth\((s; y) := \max\{ m \in \mathbb{N} : y \in I_{s|m}\} \), where \( s|m \) denotes the prefix of \( s \) of length \( m \). For any binary strings \( s_1, \ldots, s_d \) and \( y \in [-R, R]^d \), define depth\((s_1, \ldots, s_d; y) := \text{argmin}\{ \text{depth}(s_j; y_j) : j \in \{1, \ldots, d\} \} \).

The following properties hold:

1. For any binary strings \( s \) and \( t \) such that \( s \) is a prefix of \( t \), and any \( y \in [-R, R] \),
   \[ \text{depth}(s; y) \leq \text{depth}(t; y). \]

2. Let \( s_1, \ldots, s_d \) and \( t_1, \ldots, t_d \) be binary strings such that \( s_i \) is a prefix of \( t_i \) for all \( i = 1, \ldots, d \). Let \( y \in [-R, R] \). Suppose \( j \in \text{depth}(s_1, \ldots, s_d; y) \) and depth\((s_j; y_j) = \text{depth}(t_j; y_j) \). Then \( j \in \text{depth}(s_1, \ldots, s_d; y) \).

3. The following are equivalent:
   
   (a) \( f_{s_1, \ldots, s_d}(y) = f_{s_j}(y_j) \)
   
   (b) \( \gamma e_j \in \partial f_{s_1, \ldots, s_d}(y) \), where \( \gamma \) is the derivative of \( f_{s_j} \) at \( y_j \).
   
   (c) \( j \in \text{depth}(s_1, \ldots, s_d; y) \).

4. Let \( s_1, \ldots, s_d \) be binary strings. Let \( y \in [-R, R]^d \). Let \( j^* \in \{1, \ldots, d\} \) such that
   \[ f_{s_1, \ldots, s_d}(y) = f_{s_{j^*}, (y_{j^*})}. \]
   If \( y_{j^*} \in I_{s_{j^*}} \), we set \( b := 0 \) if \( y_{j^*} \) is greater than or equal to the midpoint of \( I_{s_{j^*}} \); otherwise we set \( b = 1 \). Define \( s_{j^*}' = s_j \) for all \( j \neq j^* \) and define \( s_{j^*}' = s_{j^*}b \), i.e., \( s_{j^*}' \) is obtained by appending the bit \( b \) to \( s_{j^*} \). Then for any binary strings \( t_1, \ldots, t_d \) such that \( s_{j^*}' \) is a prefix of \( t_i \),
   \[ f_{t_1, \ldots, t_d}(y) = f_{s_{j^*}', \ldots, s_d'}(y) = f_{s_{j^*}', (y_{j^*})} \] and \( e_{j^*} \in \partial f_{t_1, \ldots, t_d}(y) \).

Again, these properties are straightforward to verify starting from the definition of the objects involved given in Definition 15 and (1.4). In particular item 3b) follows from defining the functions as coordinate-wise maxima in (1.4), and provides the connection to the single-coordinate oracle that we will use – each query will essentially only reveal information about one of the coordinates of the function, or equivalently, about one of the \( d \) binary strings defining the function. Property 4 importantly shows that functions that share prefixes in their binary strings are indistinguishable outside of specific intervals.
This can also be seen in Figure 1.2: observe that \( f_{s'_0} \) and \( f_{s'_1} \) will only differ on \( I_{s'} \), no matter how many levels into the iterative definition they are (i.e., no matter the length of \( s \)).

**Proof sketch for lower-bound:** The key idea for establishing the desired lower-bound is the following: Item 4 in Lemma 17 above essentially demonstrates that in the worst case, each adaptive query reveals only one bit of one of the \( d \) strings that determine the true function \( f \). Hence, making \( k \) queries can be viewed as learning \( k \) bits worth of prefixes of the strings defining the true \( f \). Items 3, 5, and 6 of Lemma 16 will then imply that unless sufficiently many bits of the prefix of each of the \( d \) strings defining \( f \) are known, there exist \( f_1, f_2 \) giving the same responses (or equivalently, having the learned prefixes in the strings defining them) but having disjoint \( \varepsilon \)-solutions.

Given this sketch of the main idea to construct these two functions, we leave the details of this construction to the interested reader since all of the needed ingredients are ready in Lemmas 17 and 16. We also refer to [19] and [82] which contain the details for proving this result.
Chapter 2

Information Complexity of Mixed-Integer Optimization

Chapter Abstract

This chapter presents new results in the information complexity of mixed-integer convex optimization under different types of oracles. In particular, we prove a fundamental “transfer” result that shows how lower bounds on information complexity of continuous convex optimization under different oracles can be transferred to the mixed-integer setting in a black-box manner. This gives multiple lower-bounds for mixed-integer convex optimization as a consequence. In particular, applying this theorem to the lower-bound of the standard (full-information) lower bound for continuous convex optimization improves upon the previous best known lower bound for the mixed-integer setting. This leaves only a lower order linear term (in the dimension) as the gap between the lower and upper bounds.

Further, this chapter proves results on information complexity under general oracles based on first-order information (Definition 5), i.e. oracles that may only reveal partial first-order information. These could, for instance, be oracles where one can only make a binary query over the function value or subgradient at a given point. We
give algorithms for (mixed-integer) convex optimization that work under these less informative oracles. We also give lower bounds showing that, for some of these oracles, every algorithm requires more iterations to achieve a target error compared to when complete first-order information is available. That is, these oracles are provably less informative than full first-order oracles for the purpose of optimization. A number of the contributions presented in this chapter are also available in [10].

2.1 Results

Central to many of our results is the general notion of oracles based on first-order information, as from Definition 5. Some novel results are also obtained for specific instances of these, such as the bit oracle, inner product oracle, and general binary oracle from Definition 6. In fact, we consider the language provided by the careful definition of these oracles based on first-order information in their generality to itself be a meaningful contribution of this line of work. With these in hand, we are ready to state our quantitative results for lower and upper bounds on the information complexity of mixed-integer convex optimization under different oracles; see Table 2.1 for a summary.

The results of this chapter are focused to largely revolve around Theorem 18, which transfers lower-bounds on information complexity of continuous problems to the mixed-integer setting. We defer to the next chapter a different type of “transfer” result for a slightly broader setting than (1.1) that we frame more as an algorithmic result than an information theoretic one (even though these go hand-in-hand), which transfers guarantees for algorithms under the full-information setting to the approximate information setting.

**Lower bounds.** Our first result is a “transfer” theorem that will be a powerful tool for obtaining concrete mixed-integer lower bounds under different oracles. This theorem lifts lower bounds for unconstrained optimization from the continuous to the mixed-integer setting. In particular, if one has a lower bound \( \ell \) with respect to an oracle using first-order information (Definition 5) for the information complexity for some family of
Table 2.1: Summary of results on the information complexity of mixed-integer convex optimization for the class of instances $I_{n,d,R,\rho,M}$ that have $n$ integer variables, $d$ continuous variables, the feasible region lies in the box $[-R, R]^{n+d}$ and has a $\rho$-deep feasible point on the optimal fiber, and the objective function is $M$-Lipschitz with respect to $\ell_\infty$ (see Definition 4). The table presents simplified bounds showing only the main parameters.

<table>
<thead>
<tr>
<th>Type of first-order oracle $\mathcal{O}(G, \mathcal{H})$</th>
<th>Variables</th>
<th>Lower bound</th>
<th>Upper bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{H}$ is hereditary</td>
<td>Mixed</td>
<td>$\Omega(2^n \ell)$, where $\ell \leq \text{icomp}<em>{\mathcal{H}}(I</em>{0,d,R,\rho,M}, \mathcal{O}(G, \mathcal{H}))$ (Theorem 18)</td>
<td></td>
</tr>
<tr>
<td>Full-information first-order oracle</td>
<td>Mixed</td>
<td>$\Omega \left(2^n d \log \left(\frac{MR_{\min(I)}}{\min(1/\epsilon)}\right)\right)$ (Corollary 19)</td>
<td>$O \left(2^n d(n + d) \log \left(\frac{MR_{\min(I)}}{\min(1/\epsilon)}\right)\right)$ (Oertel [90], Basu-Oertel [9])</td>
</tr>
<tr>
<td>$\mathcal{H}^{bit}$, $\mathcal{H}^{bit^*}$, $\mathcal{H}^{dir}$, or General Binary Queries</td>
<td>Mixed</td>
<td>$\tilde{\Omega} \left(2^n \max \left{d^2, d \log \left(\frac{MR_{\min(I)}}{\min(1/\epsilon)}\right)\right}\right)$ (Theorem 20)</td>
<td>$O \left(2^n d(n + d)^2 \log^2 \left(\frac{(n + d)MR_{\min(I)}}{\min(1/\epsilon)}\right)\right)$ (Theorem 23)</td>
</tr>
<tr>
<td>$\mathcal{H}^{bit}$, $\mathcal{H}^{bit^*}$</td>
<td>Continuous</td>
<td>$\tilde{\Omega} \left(\max \left{d^2, d \log \left(\frac{MR_{\min(I)}}{\min(1/\epsilon)}\right)\right}\right)$ (Theorem 20)</td>
<td>$O \left(d^2 \log^2 \left(\frac{dMR_{\min(I)}}{\min(1/\epsilon)}\right)\right)$ (Theorem 24)</td>
</tr>
<tr>
<td>General Binary mixed</td>
<td>Continuous</td>
<td>$\tilde{\Omega} \left(2^n \max \left{d^2, d \log \left(\frac{MR_{\min(I)}}{\min(1/\epsilon)}\right)\right}\right)$ (Theorem 20)</td>
<td>$O \left(\log</td>
</tr>
<tr>
<td>Continuous</td>
<td>Continuous</td>
<td>$\tilde{\Omega} \left(\max \left{d^2, d \log \left(\frac{MR_{\min(I)}}{\min(1/\epsilon)}\right)\right}\right)$ (Theorem 20)</td>
<td>$O \left(\log</td>
</tr>
</tbody>
</table>

purely continuous instances, then one can “transfer” that lower bound to the mixed-integer case as $\Omega(2^n \ell)$ with access to the “same” oracle in the $n + d$ dimensional space.

For this notion, we require the set of permissible queries $\mathcal{H}$ to be hereditary. Roughly speaking, this means that the set of queries has the same richness on a purely continuous space as it has in a mixed-integer space. We formally define hereditary queries in Section 2.2, and note that all of the types of permissible queries discussed in Definition 6 satisfy this property, except for $\mathcal{H}^{bit}$ (the slightly enhanced $\mathcal{H}^{bit^*}$ queries are hereditary).

**Theorem 18.** Let $\mathcal{H}_{n,d}$ be any class of hereditary permissible queries, and assume $\mathcal{H}_{0,d}$ contains function threshold queries $h_c$ that answer $h_c(g_{\text{val}}(\hat{f}, \hat{C})) := \text{sgn}(g_{\text{val}}(\hat{f}, \hat{C}) + c)$ for any $c \in \mathbb{R}$. Let $\epsilon \geq 0$. Suppose, for some $d \geq 1$, there exists a class $\mathcal{I} \subseteq \mathcal{I}_{0,d,R,\rho,M}$ of continuous convex (unconstrained) optimization problems in $\mathbb{R}^d$, and a first order chart
for $I$ such that $\text{icomp}_e(I, \mathcal{O}(G_0, H_{0,d})) \geq \ell$. Suppose further that all instances in $I$ have the same optimal value. Then, for any number of integer variables $n \geq 1$, there is a first order chart $G_n$ such that $\text{icomp}_e(I_{n,d,R,\rho,M}, \mathcal{O}(G_n, H_{n,d})) \geq 2^{n-1}\ell$.

As a first consequence of this transfer theorem we obtain a sharpened lower bound for the standard full-information first-order oracle case for mixed-integer problems. For this setting, Basu [8] proved the lower bound of $\Omega(2^n \cdot d \log \left( \frac{MR}{\rho} \right))$. However, this bound is independent of the Lipschitz constant $M$ of the objective function, and thus does not capture the hardness of the problem as $M$ increases. By applying Theorem 18 to the classical lower bound of $\Omega(d \log \left( \frac{MR}{\varepsilon} \right))$ for continuous convex optimization with the standard first-order oracle by Nemirovski and Yudin [82], and combining the result with the existing mixed-integer lower bound, we obtain the following improved bound.

**Corollary 19.** There exists a first-order chart $G$ such that for the full-information first-order oracle based on $G$ (i.e., $H$ consists of the identity functions) we have

$$\text{icomp}_e(I_{n,d,R,\rho,M}, \mathcal{O}(G, H)) = \Omega \left( 2^n \left( 1 + d \log \left( \frac{MR}{\varepsilon \min\{\rho, 1\}} \right) \right) \right).$$

Moving on to “non-standard” oracles, we consider mixed-integer convex optimization under the general binary oracle. Recall from Definition 6 that this means that the algorithm can make any binary query on subgradients/separating hyperplanes. Despite the power of these queries, we prove a separation between the information complexity under the standard full-information first-order oracle and the general binary oracle, i.e., the latter provides quantitatively less information for solving the problem. For example, in the pure continuous setting, $O(d)$ queries suffice (ignoring the logarithmic dependence on other parameters) under the full-information first-order oracle. However, we show that $\Omega(d^{8/7})$ queries are needed under the general binary oracle. More precisely, we show the following lower bound.

**Theorem 20.** For every $n \geq 0$, there exists a first-order chart $G$ such that for the
general binary oracle based on $G$ we have

$$\text{icomp}_\epsilon(I_{n,d,R,\rho,M}, \mathcal{O}(G, \mathcal{H})) = \tilde{\Omega} \left( 2^n \left( 1 + \max \left\{ d^2, d \log \left( \frac{MR}{\min\{\rho,1\}^{2\epsilon}} \right) \right\} \right) \right),$$

where $\tilde{\Omega}$ hides polylogarithmic factors in $d$.

We note that, since $\mathcal{H}^{\text{bit}}$, $\mathcal{H}^{\text{bit}^*}$ and $\mathcal{H}^{\text{dir}}$ are more restrictive than the general binary oracle, this lower bound applies to oracles with those permissible queries as well. The proof of this result relies on a connection between information complexity and memory constrained algorithms for convex optimization, and the recent lower bound for the latter from [79] (in addition to Theorem 18 for lifting the result to the mixed-integer case).

Next, we present a stronger version of this theorem for the bit oracle; As just noted, $\mathcal{H}^{\text{bit}}$ is more restrictive than the general binary oracle, and we are able to prove a lower-bound that is quadratic in the continuous dimension under $\mathcal{H}^{\text{bit}}$.

**Theorem 21.** Suppose that $\rho \leq \frac{R}{d}$. Then there exists a first-order chart $G$ such that for the Bit Oracle based on $G$ we have

$$\text{icomp}_\epsilon(I_{n=0,d,R,\rho,M}, \mathcal{O}(G, \mathcal{H}^{\text{bit}})) = \Omega \left( d^2 \right).$$

In particular, together with the transfer Theorem 18 and Theorem 20, we obtain the following corollary serving as strengthened version of Theorem 20 under the shifted bit oracle

**Corollary 22.** Suppose $\rho \leq \frac{R}{d}$. For every $n \geq 0$, there exists a first-order chart $G$ such that for the shifted bit oracle based on $G$ we have

$$\text{icomp}_\epsilon(I_{n,d,R,\rho,M}, \mathcal{O}(G, \mathcal{H}^{\text{bit}^*})) = \tilde{\Omega} \left( 2^n \left( 1 + \max \left\{ d^2, d \log \left( \frac{MR}{\min\{\rho,1\}^{2\epsilon}} \right) \right\} \right) \right),$$

where $\tilde{\Omega}$ hides polylogarithmic factors in $d$.

Note that these this result assumes $\rho \leq \frac{R}{d}$. This is because we prove the result by
making the feasibility portion of the problem difficult. If \( \rho \) is large, the feasibility portion of the problem becomes easy, and \( \rho \leq \frac{R}{d} \) is enough for the feasibility to be critically difficult. We do not currently have a matching result for unconstrained optimization, though we suspect it to be possible and encourage future work in this direction.

**Upper bounds.** We now present upper bound results that illustrate the connection between information complexity based on full-information first-order oracles and information complexity based on binary queries on separating hyperplanes and subgradients. We first formalize the intuition that by making roughly \( O((n + d) \log \left( \frac{1}{\varepsilon} \right)) \) bit or inner product sign queries on a separating hyperplane or subgradient, one should have enough information to solve the problem as with full information (Theorems 23 and 24). Next, in Theorem 25 and Corollary 26, we show how this natural bound can be improved in certain settings.

**Theorem 23.** Assume \( d \geq 1 \). For \( U > 0 \), consider the subclass of instances of \( \mathcal{I}_{n,d,R,\rho,M} \) whose objective function values lie in \([-U,U]\), and the fiber over the optimal solution contains a \( z \) such that the \((n + d)\)-dim \( \rho \)-radius \( \ell_\infty \) ball centered at \( z \) is contained in \( C \). There exists a query strategy for this subclass that reports an \( \varepsilon \)-approximate solution by making at most

\[
O \left( 2^d (n + d) \log \left( \frac{MR}{\min\{\rho, 1\} \varepsilon} \right) \right) \cdot \left( n + d \right) \log \left( \frac{(n + d)MR}{\rho \varepsilon} \right) + \log \frac{U}{\varepsilon}
\]

queries to an oracle \( \mathcal{O}(\mathcal{G}, \mathcal{H}) \), where \( \mathcal{G} \) is any first-order chart and \( \mathcal{H} \) is either \( \mathcal{H}^{\text{bit}} \) or \( \mathcal{H}^{\text{dir}} \).

Prescribing an *a priori* range for objective function values is not a serious restriction for two reasons: i) The difference between the maximum and the minimum values of an objective function in \( \mathcal{I}_{n,d,R,\rho,M} \) is at most \( 2MR \), and ii) All optimization problems whose objective functions differ by a constant are equivalent. We also comment that while we assume \( d \geq 1 \) in Theorem 23, similar bounds can be established for the \( d = 0 \) (pure integer) case. We omit this here because a unified expression for the \( d = 0 \) and \( d \geq 1 \) cases becomes unwieldy and difficult to parse.
The main idea behind Theorem 23 is to show that existing methods with the best known information complexity for mixed-integer convex optimization that use full-information first-order oracles can also work with approximate separation and subgradient oracles that return desired approximations of the true vectors (with no loss in the information complexity). Then one shows that one can produce these approximations with roughly $O\left((n + d) \log\left(\frac{1}{\varepsilon}\right)\right)$ bit or inner product sign queries on a separating hyperplane or subgradient. With bit queries, this is just a matter of probing enough bits of each coordinate of the vector. The case with inner product sign queries is a bit more involved and our main tool is a result that shows how to approximate any vector up to desired accuracy with such queries (Lemma 43).

Subsequently, using similar techniques we present an enhanced upper bound for the scenario where $n = 0$ (pure continuous case).

Theorem 24. For $U > 0$, consider the subclass of instances of $\mathcal{I}_{n,d,\rho,M}$ where $n = 0$ (pure continuous case) and the objective function values lie in $[-U, U]$. There exists a query strategy for this subclass that reports an $\varepsilon$-approximate solution by making at most

$$O\left(d \log\left(\frac{MR}{\min\{\rho, 1\} \varepsilon}\right)\right) \cdot \left(d \log\left(\frac{dMR}{\rho \varepsilon}\right) + \log \frac{U}{\varepsilon}\right)$$

queries to an oracle $O(\mathcal{G}, \mathcal{H})$, where $\mathcal{G}$ is any first-order chart and $\mathcal{H}$ is either $\mathcal{H}^{\text{bit}}$ or $\mathcal{H}^{\text{dir}}$.

Notably, this is nearly optimal in $d$ in light of the lower bound presented in Theorem 21, only differing in a $\log(d)$ term.

Finally, we provide a kind of transfer result that allows one to transfer algorithms designed for full-information first-order oracles to the (harder) setting of a general binary oracle.

Theorem 25. Suppose there exists an algorithm that reports an $\varepsilon$-approximate solution for instances in $\mathcal{I}_{n,d,\rho,M,R}$ with at most $u$ queries to the full-information first-order oracle based on a first-order chart $\mathcal{G}$. Then, for any subclass of finitely many instances $\mathcal{I} \subset \mathcal{I}_{n,d,\rho,M,R}$, there exists a query strategy for this subclass using the general binary oracle.
oracle based on \( G \) that reports an \( \varepsilon \)-approximate solution by making at most

\[
O(\log |I| + u)
\]

queries.

Using the centerpoint-based algorithm from [90, 9], we obtain the following corollary:

**Corollary 26.** Given any subclass of finitely many instances \( \mathcal{I} \subset \mathcal{T}_{n,d,R,\rho,M} \) and any first-order chart \( G \), there exists a query strategy for this subclass using the general binary oracle based on \( G \) that reports an \( \varepsilon \)-approximate solution by making at most

\[
O \left( \log |I| + 2^n d(n + d) \log \left( \frac{dMR}{\min\{\rho, 1\} \varepsilon} \right) \right)
\]

queries. In the (pure continuous) case of \( n = 0 \), \( O \left( \log |I| + d \log \left( \frac{MR}{\min\{\rho, 1\} \varepsilon} \right) \right) \) queries suffice.

In particular, when the number of instances under consideration is \( |I| = O(2^{2^d(d(n + d))}) \), Corollary 26 gives a strictly better upper bound than Theorem 23. Similarly, for \( n = 0 \), in the case when \( |I| = O(2^d) \), we get a better upper bound compared to Theorem 24; in fact, we beat the lower bound provided by Theorem 20. This demonstrates that even with exponentially many instances under consideration, the case of finite instances yields a lower information complexity than the case of infinitely many instances. We point out that the first-order chart \( G \) must be known to implement the query strategy in Theorem 25. In contrast, the algorithms in Theorems 23 and 24 are oblivious of the first order chart, i.e., they work with any first order chart.

### 2.1.1 Discussion and future avenues

The concept of information complexity in continuous convex optimization and its study go back several decades, and it is considered a fundamental question in convex optimization. In comparison, much less work on information complexity has been carried out in the presence of integer constrained variables. Nevertheless, we believe there are
important and challenging questions that come up in that domain that are worth studying. Further, even within the context of continuous convex optimization, the notion of information complexity has almost exclusively focused on the number of full-information first-order queries. As we hope to illustrate with the results of this thesis, considering other kinds of oracles leads to very interesting questions at the intersection of mathematical optimization and information theory. In particular, the study of binary oracles promises to give a more refined understanding of the fundamental question “How much information about an optimization instance do we need to be able to solve it with provable guarantees?”. For instance, establishing any superlinear (in the dimension) lower bound for the continuous problem with general binary oracles, like the one in Theorem 20, seems to be nontrivial. In fact, the results from [79], on which Theorem 20 is based, were considered a breakthrough in establishing superlinear lower bounds on space complexity of convex optimization. Even so, the right bound is conjectured to be quadratic in the dimension (see Theorem 24). We see Theorem 21 as a significant contribution in this direction, achieving this quadratic lower bound in the dimension for the bit oracle case, especially since standard bit representations are most commonly used when doing actual computations. In that regard, these oracles have a practical motivation. Obtaining exact first-order information may be difficult or impossible in many practical situations, and one has to work with approximations of separating hyperplanes and subgradients. The binary oracles can be viewed as providing these approximations and information complexity under these oracles becomes important from a practical standpoint.

We thus view the results of this chapter as expanding our understanding of information complexity of optimization in two different dimensions: what role does the presence of integer variables play and what role does the nature of the oracle play (with or without integer variables)? For the role of integer variables, in the pure optimization case Theorem 18 provides a lifting of lower bound from the continuous case. Allowing for constraints, Corollary 19 brings the lower bound closer to the best known upper bound on information complexity based on the classical subgradient oracle. The remaining gap is now simply a factor linear in the dimension. A conjecture in convex geometry first
articulated in [90, Conjecture 4.1.20] and elaborated upon in [9, 8] would resolve this and would show that the right bound is essentially equal to the lower bound we prove in this thesis.

Beyond the contributions discussed above, our work also opens up new future directions for study. We believe the following additional conjectures to be good catalysts for future research, especially in regard to understanding the interplay of integer variables and other oracles.

The first conjecture is a generalization of our Theorem 18 to incorporate constraints as well. This would make this “transfer” tool more powerful, and would, for example, give Corollary 19 as a special case without appealing to [8] for the feasibility lower bound.

**Conjecture 1.** If there exist continuous, **constrained** convex optimization instances such that \( \ell \) is a lower bound for this family on the information complexity with respect to an oracle, then for every \( n \geq 1 \), there exist mixed-integer instances with \( n \) integer variables such that the information complexity of these mixed-integer instances is lower bounded by \( \Omega(2^n \cdot \ell) \) for the same oracle.

Another consequence of resolving this conjecture is that if future research on the information complexity of continuous convex optimization results in better/different lower bounds based on feasibility, these would immediately imply new lower bounds for the mixed-integer case. For instance, we believe the following conjecture, which can be viewed as a strengthening of Corollary 22, to be true for the mixed-integer convex optimization problem.

**Conjecture 2.** There exists a first-order chart \( \mathcal{G} \) such that the general binary oracle based on \( \mathcal{G} \) has information complexity \( \Omega \left( 2^n \left( 1 + d^2 \log \left( \frac{MR}{\rho \varepsilon} \right) \right) \right) \).

A version of Conjecture 2 is also stated in the language of “memory-constrained” algorithms in [108, 79] for the continuous case (see Section 2.3 below); the way we have stated the conjecture here presents its transfer to the mixed-integer case.
Analogously, it would be nice to have “transfer” theorems for upper bounds as well. In the spirit of Theorems 23, 24 and 25, we believe a useful result would be a theorem that takes upper bound results proved in the full-information first-order oracle setting and obtains upper bound results in the general binary oracle setting. A use case of such a result would be the following: if the upper bound for the general mixed-integer problem with full-information first-order oracles is improved by resolving the convex geometry conjecture mentioned above (and we believe the lower bound is correct and the upper bound is indeed loose), then this would also give better upper bounds for the general binary oracle setting. Thus, we make the following conjecture.

**Conjecture 3.** If there exists a query strategy with worst case information complexity 
\[ u(n, d, R, \rho, M, G) \] under the full-information first-order oracle based on a first-order chart \( G \), then there exists a query strategy with worst case information complexity bounded by 
\[ u(n, d, R, \rho, M, G) \cdot O \left( (n + d) \log \left( \frac{MR}{\rho \epsilon} \right) \right) \]
under the general binary oracle based on \( G \).

We focus on oracles that use first order information in this thesis (Definitions 5 and 6). Oracles that use “zero-order information” have also been studied in the literature, beginning with the seminal work of Yudin and Nemirovski [82]; see [51] for an exposition of how those ideas can be used in the mixed-integer setting and [28] for an exposition in the nonconvex setting. Such oracles report function values only for the objective, with no subgradient information, and only report membership for the constraints, with no separating hyperplanes. A related oracle is the “value comparison” oracle that has found many applications. These oracles comprise of questions of the form “Is \( f(z) \leq f(z') \)”, with no access to the subgradients of \( f \). Such algorithms are particularly useful in learning from users’ behaviors, since while a user typically cannot accurately report its (dis)utility value \( f(z) \) for an option \( z \), it can more reliably compare the values \( f(z) \) and \( f(z') \) of two options; see [62, 92] and references therein for discussions and algorithms in the continuous convex case. The mixed-integer setting under the value comparison oracle has been extensively studied in recent work [25, 49, 50, 105].
The ideas in this thesis can also be adapted to give algorithms for mixed-integer convex optimization using the comparison oracle, but we do not undertake a deeper study here. There seems to be scope for future research in this direction, especially in tying together these different strands of ideas for “zero order information”.

The remainder of this chapter is dedicated to the formal proofs of our main results discussed above.

2.2 Proof of Theorem 18

The high-level idea for the proof of Theorem 7 is to construct difficult mixed-integer instances by taking hard instances of the continuous case, “placing” one of them on each fiber \( x \times \mathbb{R}^d, x \in \{0, 1\}^n \), and interpolating between fibers appropriately. We do this in a way such that effectively one needs to solve the continuous problems obtained by restricting to each fiber, which leads the \( \Omega(2^n \ell) \) lower bound from an \( \ell \) lower bound on the continuous problems – namely, there will be one difficult function from the continuous case placed on each of the \( 2^n \) fibers, so if one can’t do better than solving each of them separately, one ends up with an \( \Omega(2^n \ell) \) lower bound. To make this idea work, the interpolation needs to be done in a way that no query in the full \([0, 1]^n \times \mathbb{R}^d\) space reveals information about two (or more) of the continuous functions placed on different fibers, or reveals significantly more information about a function on a fiber than a query on that fiber would. For example, we need to ensure that a single query at the point \( (\frac{1}{2}, \ldots, \frac{1}{2}, y) \) for \( y \in \mathbb{R}^d \) does not reveal information about multiple functions on different fibers.

2.2.1 Game-theoretic perspective

So far we have described the information complexity of optimization using an oracle \( O \) over a family of instances \( I \) based on having an optimization algorithm that in each round \( t \) makes a query \( q_t \) to \( O \) and receives as answer the result \( q_t(\hat{I}) \) for the unknown instance \( \hat{I} \) it is trying to optimize. However, for obtaining lower bounds on the information complexity, it is more helpful to consider the algorithm as interacting with an adversary
for the family of instances under $O$, instead of the unknown instance $\hat{I}$. More precisely, at round $t$, the adversary receives the query $q_t$ of the algorithm and produces, possibly based on all the previous queries $q_1, \ldots, q_{t-1}$, a response $r_t$. The only requirement is that there must always exist at least one instance $\bar{I} \in I$ that is consistent with all of its responses, namely $r_t = q_t(\bar{I})$ for all $t$, under the oracle $O$ being considered. With each such response, the set of instances that are consistent with all responses given may change, motivating the following definition:

**Definition 27.** Given a class of instances $I$, an oracle $O$, and a transcript of query-response pairs $(q_1, r_1), \ldots, (q_t, r_t)$, the set of surviving instances for $(q_1, r_1), \ldots, (q_t, r_t)$ under $O$ is

$$\{I \in I : q_j(I) = r_j \forall j \in [t]\},$$

i.e., the set of instances consistent with the responses in the transcript under the oracle $O$. When all instances in $I$ are unconstrained, let the set of surviving functions be the set of functions corresponding to the surviving instances.

We say that an adversary $\text{Adv}$ is $\varepsilon$-hard for $\ell$ rounds if for any algorithm $\text{Alg}$, after $\ell$ rounds there are surviving instances in $I$ that do not have a common $\varepsilon$-approximate solution, i.e., if $q_1, \ldots, q_\ell$ and $r_1, \ldots, r_\ell$ are $\text{Alg}$’s queries and $\text{Adv}$’s responses, respectively, then there is a collection of instances $\mathcal{J} \subset I$ that have no common $\varepsilon$-approximate solution but such that $r_t = q_t(I)$ for all $I \in \mathcal{J}$ and $t = 1, \ldots, \ell$. Since the sets of $\varepsilon$-approximate solutions of instances in $I_{n,d,R,\rho,M}$ are compact convex sets, this collection $\mathcal{J}$ of instances may always be taken to be finite. Intuitively, the existence of such an adversary should imply that no algorithm can reliably report an $\varepsilon$-approximate solution within $\ell$ iterations, that is, $\text{icomp}_\varepsilon(O, I) > \ell$. The next result shows that this adversary-based perspective is indeed equivalent to information complexity, and may be of independent interest (for a proof see Appendix 2.8).

**Lemma 28.** Consider a class of instances $I$ and an oracle $O$. Then $\text{icomp}_\varepsilon(I, O) > \ell$ if and only if there exists an adversary under $O$ using $I$ that is $\varepsilon$-hard for $\ell$ rounds.

---

If a collection of compact sets has empty intersection, then there exists a finite subcollection that already has empty intersection.
2.2.2 Proof for the full-information first-order oracle

The full proof of Theorem 18 is a bit technical and requires a few conceptual connections. For a better exposition, we first prove the theorem in the case that the oracle is the full-information first-order oracle. As $H$ thus consists of the identity maps, throughout this subsection we will write oracles using first-order information as $O(G)$, where $G$ is the corresponding first order chart.

Given the assumption of the theorem and the equivalent adversarial perspective from Lemma 28, assume there is a family of continuous, unconstrained instances $I_{cont} \subseteq I_{0,d,R,\rho,M}$, all with the same optimal value $OPT$, and a full-information first-order adversary $Adv-Cont$ for $I$ that is $\varepsilon$-hard for $\ell - 1$ rounds. Let us use $F_{cont}$ to denote the objective functions of the instances $I_{cont}$. In the full-information first-order case, queries of an optimization algorithm consist of points $y_1, y_2, \ldots \in \mathbb{R}^d$, and either query the function value or the subgradient. For simplicity, let us allow the algorithm to query both the function value and subgradient in a single query, so that the queries become simply $y_1, y_2, \ldots \in \mathbb{R}^d$ and the responses of an adversary consist of a sequence of consistent function values and subgradients, namely a sequence $(v_1, g_1), (v_2, g_2), \ldots \in \mathbb{R} \times \mathbb{R}^d$ such that there is some $f \in F_{cont}$ satisfying $v_t = f(y_t)$ and $g_t \in \partial f(y_t)$ for all rounds $t$.

To prove the theorem, we will construct a full-information first-order adversary $Adv-MI$ for a family of mixed-integer instances over $\{0, 1\}^n \times \mathbb{R}^d$ that is $\varepsilon$-hard for $2^n \ell - 1$ rounds. As alluded to before, the very high-level is to place a copy of the continuous adversary $Adv-Cont$ on each of the continuous fibers $x \times \mathbb{R}^d$ for $x \in \{0, 1\}^n$. In fact, we will work with a slightly modified version of the continuous adversary that is constructed next.

Modifying the continuous adversary $Adv-Cont$

For the mixed-integer adversary $Adv-MI$, it will be important to render a fiber “useless” for the optimization algorithm after it queries (close to) this fiber too many times, so as to intuitively force it to query (close to) other fibers, or gain no new information otherwise. This will be done by modifying the continuous adversary $Adv-Cont$ such that whenever it is probed $\ell$ or more times, it commits to answering all future queries consistently with
a single function that has optimal value $> \text{OPT} + \varepsilon$; since our mixed-integer instances will be constructed to have optimal value \text{OPT}, gathering more information about the function on such fibers will not help the algorithm solve the mixed-integer problem. To do this, the modified continuous adversary will also keep track of the set $S$ of surviving functions (Definition 27) given its responses. More precisely, here are its main properties.

**Lemma 29.** There is a family of convex functions $\overline{F}_\text{cont}$ corresponding to instances $\mathcal{I}_{0,d,R,p,M}$ of the purely continuous case, a first-order chart $\mathcal{G}$ and a full-information first-order adversary $\text{Adv-Cont}^+$ that, for any algorithm $\text{Alg}$, maintains a set of functions $S_t \subseteq \overline{F}_\text{cont}$ for every query-response round $t$ with the following properties:

1. In every round $t \geq 1$, all functions in $S_t$ are consistent with the responses returned by $\text{Adv-Cont}^+$ thus far, under some oracle using first-order information $O(\mathcal{G})$.

2. In the first $t \leq \ell - 1$ rounds, there is a finite collection of functions in $S_t \cap F_\text{cont}$ that do not share an $\varepsilon$-approximate solution. In particular, $\text{Adv-Cont}^+$ is still $\varepsilon$-hard for $\ell - 1$ rounds.

3. For all rounds $t \geq 1$, $S_t$ is closed under taking maxima of finitely many of its elements, and also contains a function that has minimum value $> \text{OPT} + \varepsilon$.

4. For rounds $t \geq \ell$, $S_t$ contains a single function with minimum value $> \text{OPT} + \varepsilon$.

Item 1. means that for rounds $t < \ell$, there exists a full-information first-order oracle $O(\mathcal{G})$ such that $S_t$ is exactly the set of surviving functions under $O(\mathcal{G})$ given the responses produced by $\text{Adv-Cont}^+$ up to round $t$. Hence, we will refer to this $S_t$ as the set of surviving functions maintained by $\text{Adv-Cont}^+$ at round $t$. We now make precise our modification to the continuous adversary $\text{Adv-Cont}$ and prove Lemma 29. As a preliminary, let $\overline{F}_\text{cont}$ denote the closure of $F_\text{cont}$ under taking maxima of finitely many functions, i.e. for any finite collection $\mathcal{J} \subset \overline{F}_\text{cont}$, $\max_{f \in \mathcal{J}} \{f\} \in \overline{F}_\text{cont}$. Notice these functions are still convex. The following lemma highlights the key property of $\overline{F}_\text{cont}$ we will make use:

**Lemma 30.** Let $\mathcal{J} \subset \mathcal{F}$ be a finite set. If $f \in \mathcal{J}$ do not have a common $\varepsilon$-solution, then the pointwise maximum function $\max_{f \in \mathcal{J}} \{f\}$ has minimum value greater than $\text{OPT} + \varepsilon$.  

41
Proof. Suppose for sake of contradiction that there exists a point $z$ such that \( \max_{f \in \mathcal{J}} \{ f(z) \} \leq \text{OPT} + \varepsilon \). Then \( f(z) \leq \text{OPT} + \varepsilon \) for all \( f \in \mathcal{J} \), which means \( z \) is an \( \varepsilon \)-solution for all \( f \in \mathcal{J} \), which contradicts the assumption that they do not share an \( \varepsilon \)-solution. \qed

We now formally describe \textbf{Adv-Cont}+, and then prove that it satisfies the invariants of Lemma 29.

**Procedure 1. Adv-Cont+**

Initialize set of surviving functions \( S_0 = \mathcal{F}_{\text{cont}} \)

For each round \( t = 1, 2, \ldots \):

1. Receive query point \( y_t \in \mathbb{R}^d \) from the optimization algorithm

2. If \( t \leq \ell - 1 \): Send \( y_t \) to the adversary \textbf{Adv-Cont}, receiving back a value \( v_t \) and subgradient \( g_t \). Obtain \( S_t \) by removing from \( S_{t-1} \) the functions \( f \) that are not consistent with this response for any first-order chart \( \mathcal{G} \), namely where \( f(y_t) \neq v_t \) or \( g_t \notin \partial f(y_t) \).

   Send the response \((v_t, g_t)\) to the optimization algorithm.

3. If \( t = \ell \): Since \textbf{Adv-Cont} if \( \varepsilon \)-hard for \( \ell - 1 \) rounds, there is a finite collection of functions \( \{f_1, \ldots, f_k\} \subset S_{t-1} \cap \mathcal{F}_{\text{cont}} \) that do not share an \( \varepsilon \)-solution. Define their pointwise maxima \( f_{\text{max}} = \max \{f_1, \ldots, f_k\} \) and set \( S_{t+k} = \{f_{\text{max}}\} \), for all \( k = 0, 1, 2, \ldots \).

   Set the value \( v_t \) to be \( f_{\text{max}}(y_t) \) and set \( g_t \) to be a subgradient in \( \partial f_{\text{max}}(y_t) \) (consistent with what the first order chart \( \mathcal{G}_0 \) gives for \( f_1, \ldots, f_k \) at \( y_t \), if \( y_t \) has been queried in an earlier round), and send the response \((v_t, g_t)\) to the optimization algorithm.

4. If \( t > \ell \): Let \( f_{\text{max}} \) be the only function in \( S_{t-1} \). If \( y_t \) was queried in an earlier round \( k \), answer \((v_k, g_k)\). Otherwise, set the value \( v_t \) to be \( f_{\text{max}}(y_t) \) and set \( g_t \) to be any subgradient in \( \partial f_{\text{max}}(y_t) \), and send the response \((v_t, g_t)\) to the
Proof of Lemma 29. We will proceed by induction on the number of rounds \( t \). The lemma clearly holds for \( S_0 \), so suppose it holds for \( S_{t-1} \).

If \( t \leq \ell - 1 \), then \( S_t \) satisfies Item 1 due to the update rule in the procedure for obtaining \( S_t \), since all functions that are not consistent with the given response are removed. More precisely, since the responses given are those produced by \textbf{Adv-Cont}, these functions in \( S_t \) are consistent with the responses under exactly the oracle \( \mathcal{O}(G_0) \) that \textbf{Adv-Cont} is hard under. \( S_t \) also satisfies Item 2 because \textbf{Adv-Cont} is assumed to be \( \varepsilon \)-hard for \( \ell - 1 \) rounds, so there exists a finite collection of functions \( \{f_1, \ldots, f_k\} \subset \mathcal{F}_{\text{cont}} \) with no common \( \varepsilon \)-solution that are consistent with all responses given to \textbf{Adv-Cont+} by \textbf{Adv-Cont}; thus \( S_t \) contains them. For Item 3, to show the closure under taking maxima, we need to argue that if functions \( f_1, \ldots, f_k \) were not removed from \( S \), then neither was \( \max(f_1, \ldots, f_k) \). Since \( f_1, \ldots, f_k \) are convex, then \( \partial f_j(y) \subset \partial \max\{f_1, \ldots, f_k\}(y) \) for any \( j \) such that \( f_j(y) = \max\{f_1(y), \ldots, f_k(y)\} \). Hence, if \( f_1, \ldots, f_k \) all have function value \( v_t \) and subgradient \( g_t \) at \( y \), then so does \( \max\{f_1, \ldots, f_k\} \), so \( \max\{f_1, \ldots, f_k\} \) was not removed from \( S \), as desired. Furthermore, if \( f_1, \ldots, f_k \) are taken to be the functions guaranteed by Item 2, Lemma 30 implies that \( \max\{f_1, \ldots, f_k\} \) has optimal value greater than \( \text{OPT} + \varepsilon \), so since we just showed \( \max\{f_1, \ldots, f_k\} \in S_t \), the remainder of item 3 follows.

If \( t = \ell \), \( S_t \) contains the single function \( f_{\text{max}} \), which has optimal value greater than \( \text{OPT} + \varepsilon \) by its construction as a consequence of Lemma 30. Hence, Item 3 follows. To prove item 1, we will use that \( S_{t-1} \) satisfies Item 1, and by Item 3 applied to \( S_{t-1} \), \( f_{\text{max}} \) is consistent with the responses returned by the procedure up to round \( t - 1 \). For round \( t \) itself, consistency follows from the definition of \( v_t \) and \( g_t \), and so \( f_{\text{max}} \) is consistent with all responses given. Item 2 does not apply in this case and Item 4 is immediate by the construction of \( S_t := \{f_{\text{max}}\} \).

If \( t > \ell \), then \( S'_t = S'_{t-1} = \{f_{\text{max}}\} \) and it suffices to check that the response \((v_t, g_t)\) is compatible with \( f_{\text{max}} \), which follows immediately from the definition of the response.
Hence, Items 1-4 of the lemma follow. It remains to show that \texttt{Adv-Cont}+ is indeed a well-defined adversary under a full-information first-order oracle. For rounds \( t \leq \ell - 1 \), this is inherited from \texttt{Adv-Cont}, while for \( t \geq \ell \), this is ensured because if the queried point \( y_t \) is the same as \( y_{t'} \) for some round \( t' < t \), \texttt{Adv-Cont}+ provides the same response in round \( t \) as in round \( t' \). Thus, there is indeed a first-order chart \( \mathcal{G} \) (derived from the first order map \( \mathcal{G}_0 \) for \texttt{Adv-Cont}) such that \texttt{Adv-Cont}+ is an adversary under the corresponding full-information first-order oracle \( \mathcal{O}(\mathcal{G}) \). \hfill \Box

**Constructing the mixed-integer adversary \texttt{Adv-MI}**

We now construct the family \( \mathcal{F}_{MI} \) of functions over \( \mathbb{R}^n \times \mathbb{R}^d \) used to transfer the lower bound to the mixed-integer setting, along with the adversary \texttt{Adv-MI} for that family. We call functions over \( \mathbb{R}^n \times \mathbb{R}^d \) \emph{full-dimensional} to distinguish them from the functions over \( \mathbb{R}^d \), the continuous part of the problem. As indicated previously, these full-dimensional functions \( \psi \) in \( \mathcal{F}_{MI} \) will be obtained by considering combinations of selecting one function \( f_{\bar{x}} \) from \( \mathcal{F}_{cont} \) for each of the mixed-integer fibers \( \bar{x} \times \mathbb{R}^d, \bar{x} \in \{0, 1\}^n \), letting \( \psi \) equal the appropriate function selected over each corresponding fiber, and applying an interpolation scheme between the fibers. This interpolation is illustrated in Figure 2.1 and described in detail later in this section.

For the behavior of \texttt{Adv-MI}, we instantiate a copy of the modified continuous adversary \texttt{Adv-Cont}+ on each fiber. Whenever the optimization algorithm queries a point \((\bar{x}, y)\) on a fiber, we send \( y \) to the continuous adversary on the fiber and report back the response \((v, g)\) received, although \( g \) needs to be appropriately lifted to the full \( \mathbb{R}^n \times \mathbb{R}^d \) space to be consistent with the way we interpolate the functions between the fibers. If the optimization algorithm only probes on these fibers, then it is intuitive that such an adversary would be \( \varepsilon \)-hard for \( 2^n \ell - 1 \) rounds: informally, up to this round, at least one of the \( 2^n \) fibers that has received no more than \( \ell - 1 \) queries, so using the hardness of \texttt{Adv-Cont}+ (Item 2 of Lemma 29) we can obtain full-dimensional functions that do not share an \( \varepsilon \)-approximate solution, which confirm the desired \( \varepsilon \)-hardness of the mixed integer adversary \texttt{Adv-MI}.  

44
The crucial element is how to deal with queries on points outside of the mixed-integer fibers. If such queries provide the algorithm with more information about the full-dimensional functions $\mathcal{F}_{MI}$ than queries on the fibers do, then we may not have full-dimensional functions with no common $\varepsilon$-approximate solution surviving for $2^n\ell - 1$ rounds. To handle this issue, the interpolation used to define the full-dimensional functions $\psi$ guarantees that its behavior on a fractional point $(\tilde{x}, y) \notin \{0,1\}^n \times \mathbb{R}^d$ is completely determined by the value of the function $f_{\tilde{x}}(y)$ from $\mathcal{F}_{cont}$ selected for the fiber $\tilde{x} \times \mathbb{R}^d$, where $\tilde{x} \in \{0,1\}^n$ is the closest 0/1 point to $\tilde{x}$. Thus, $\text{Adv-MI}$ can also answer such a query at a fractional point by making a query to the appropriate continuous adversary $\text{Adv-Cont}+$ on $\{\tilde{x}\} \times \mathbb{R}^d$, and the hardness of the latter can still be leveraged.

We now formally define the functions $\mathcal{F}_{MI}$ and the adversary $\text{Adv-MI}$.

![Figure 2.1](image-url)

Figure 2.1: (L) Illustration of two possible functions $f_0, f_1 \in \mathcal{F}_{cont}$ (blue and red) of the continuous adversary $\text{Adv-Cont}+$, for $d = 1$. (Right) Illustration of the function $\psi_{(f_0, f_1)}$ for the mixed-integer adversary $\text{Adv-MI}$ obtained by placing the functions $f_0$ and $f_1$ on the fibers $\{0\} \times \mathbb{R}$ and $\{1\} \times \mathbb{R}$ and interpolating appropriately between the fibers.

**Construction of the functions $\mathcal{F}_{MI}$.** For a 0/1 point $\tilde{x} \in \{0,1\}^n$ and a function $f \in \mathcal{F}_{cont}$ in $\mathbb{R}^d$, we first define its (convex) extension to the full-dimensional space $\mathbb{R}^{n+d}$ as

$$\hat{f}_{\tilde{x}}(x, y) = \max \left\{ f(y) + \langle M_{\tilde{x}}, x - \tilde{x} \rangle, \text{OPT} \right\}, \quad (2.1)$$
with $M_x := 3MR \cdot \text{sgn}(\bar{x} - 0.5 \cdot 1)$, where $\text{sgn}$ denotes the sign function. This construction effectively places $f$ along the $y$ space at the fiber $\bar{x}$ and extends it in each of the $x$ variables via a linear function with slope $\pm 3MR$, in a way that it decreases the value as it moves into the unit cube, or equivalently, away from $\bar{x}$; it then truncates the final value to being at least $\text{OPT}$; see Figure 2.2 for an illustration. We note for later use that wherever the extension is not truncated by $\text{OPT}$, a subgradient is given by appending the vector $M_x$ to a subgradient of $f$, and otherwise the all zeroes vector is a subgradient. More precisely, we have

$$\partial \hat{f}_\bar{x}(x, y) \supseteq \begin{cases} \{M_x\} \times \partial f(y), & \text{if } \hat{f}_\bar{x}(x, y) > \text{OPT} \\ \{0\}, & \text{otherwise.} \end{cases}$$

(2.2)

Figure 2.2: An example of a possible function from $f \in \mathcal{F}_{\text{cont}}$ (left) together with an illustration of its truncated extension $\hat{f}_0(x, y)$ (right) as constructed in (2.1).

Given a collection $\mathcal{F} = (f_\bar{x})_{\bar{x}}$ with one function $f_\bar{x} \in \mathcal{F}_{\text{cont}}$ for each 0/1 point $\bar{x}$, we combine them into the convex function

$$\psi_F(x, y) := \max_{\bar{x} \in \{0, 1\}^n} \hat{f}_\bar{x}(x, y),$$

(2.3)

where we abuse notation slightly and write the convex extension $(\hat{f}_\bar{x})_{\bar{x}}$ as simply $\hat{f}_\bar{x}$ to simplify the notation. As mentioned above, a crucial property of these functions is that their behavior between the fibers is determined by the behavior on the closest fiber. Intuitively, the slope $\pm 3MR$ guarantees that as the $x$ argument moves away from the base fiber $\bar{x}$ of each extended function $\hat{f}_\bar{x}$, $\hat{f}_\bar{x}$ decreases rapidly enough so
that the maximum in (2.3) is always achieved by the extended function at the closest fiber to $x$. Figure 2.1 illustrates this, where one can see that both functions placed on the fibers get fully truncated in between the fibers. To make this precise, let $r(x) : [0,1]^n \to \{0,1\}^n$ map any $x$ in the box to its closest 0/1 point in $\ell_\infty$-norm, that is $r(x) := \text{argmin}_{x'} \{ \|x - x'\|_\infty : x' \in \{0,1\}^n \}$.

**Lemma 31.** For every collection $F = (f_{\bar{x}})_x$, for every point $(x,y) \in [0,1]^n \times [-R,R]^d$ we have

$$\psi_F(x,y) = \tilde{f}_{r(x)}(x,y), \quad \text{and} \quad \partial \psi_F(x,y) = \partial \tilde{f}_{r(x)}(x,y)$$

**Proof.** Define $B_{\bar{x}} := \{ x' \in \mathbb{R}^n : \|x' - \bar{x}\|_\infty \leq \frac{1}{3} \}$ for every $\bar{x} \in \{0,1\}^n$. Consider an arbitrary $(x,y) \in [0,1]^n \times [-R,R]^d$.

**Case 1:** $x \notin B_{\bar{x}}$ for any $\bar{x} \in \{0,1\}^n$. This implies that for every $\bar{x} \in \{0,1\}^n$, we have

$$f_{\bar{x}}(y) + \langle M_{\bar{x}}, x - \bar{x} \rangle = f_{\bar{x}}(y) + \sum_{j \in [n]} 3MR \cdot \text{sgn}(\bar{x}_j - 0.5) \cdot (x_j - \bar{x}_j) \leq f_{\bar{x}}(y) - 3MR \cdot \max_{j \in [n]} |x_j - \bar{x}_j| < f_{\bar{x}}(y) - MR \leq \text{OPT},$$

Thus, $\tilde{f}_{\bar{x}}(x,y) = \text{OPT}$ for all $\bar{x} \in \{0,1\}^n$. As a result, $\psi_F(x,y) = \text{OPT} = \tilde{f}_{r(x)}(x,y)$.

Moreover, since $f_{\bar{x}}(y) + \langle M_{\bar{x}}, x - \bar{x} \rangle < \text{OPT}$ for all $\bar{x} \in \{0,1\}^n$, there exists a neighborhood of $(x,y)$ such that for any point $(x',y')$ in the neighborhood, it holds that $f_{\bar{x}}(y') + \langle M_{\bar{x}}, x' - \bar{x} \rangle < \text{OPT}$, and $\tilde{f}_{\bar{x}}(x',y') = \text{OPT}$ for all $\bar{x} \in \{0,1\}^n$. As a result, $\partial \tilde{f}_{\bar{x}}(x,y) = \{0\}$ for all $\bar{x} \in \{0,1\}^n$, and $\partial \psi_F(x,y) = \{0\}$.

**Case 2:** $x \in B_{\bar{x}}$ for some $\bar{x} \in \{0,1\}^n$. In this case, $r(x) = \bar{x}$. This is because for any
\( \hat{x} \in \{0, 1\}^n \backslash \{\bar{x}\} \), we have that
\[
\|x - \hat{x}\|_\infty \geq \frac{2}{3} > \frac{1}{3} \geq \|x - \bar{x}\|_\infty. \tag{2.4}
\]

It is also true that \( \hat{f}_\hat{x}(x, y) \geq \text{OPT} = \hat{f}_\bar{x}(x, y) \) for any \( \hat{x} \neq \bar{x} \), which holds due to the result from Case 1.

Moreover, the arguments from Case 1 and ((2.4)) imply that there exists a neighborhood of \((x, y)\) such that for any point \((x', y')\) in the neighborhood, it holds that \( r(x') = r(x) = \bar{x} \), and \( \hat{f}_\hat{x}(x', y') \geq \text{OPT} = \hat{f}_\bar{x}(x', y') \) for all \( \hat{x} \neq \bar{x} \). As a result, \( \psi_F(x', y') = \hat{f}_\hat{x}(x', y') = \hat{f}_r(x'(x', y') = \hat{f}_r(x)(x', y') \) and \( \partial \psi_F(x, y) = \partial \hat{f}_\bar{x}(x, y) = \partial \hat{f}_r(x)(x, y) \).

**Construction of the mixed-integer adversary Adv-MI.** We finally describe Adv-MI in Procedure 2. Its main property is captured in the following invariant.

**Procedure 2. Adv-MI**

Instantiate a copy of Adv-Cont+ on each fiber \( x \in \{0, 1\}^n \), and let \( S(x) \) denote the set of surviving functions maintained in every round by this copy, initialized to \( \mathcal{F}_{\text{cont}} \).

For each round \( t = 1, 2, \ldots \):

1. Adv-MI receives the query \((x_t, y_t)\) from the algorithm. Send \( y_t \) to the adversary Adv-Cont+ associated with the closest fiber \( r(x_t) \), which then returns a value \( v \) and subgradient \( g \), and updates its maintained set of surviving functions \( S(r(x_t)) \) of its fiber \( r(x_t) \).

2. Adv-MI returns as its response to the query \((x_t, y_t)\) the value
\[
\tilde{v}_t = \max \left\{ v + \langle M_{r(x_t)}, x_t - r(x_t) \rangle, \text{OPT} \right\},
\]
and as subgradient returns either \( \tilde{g}_t = (M_{\bar{x}}, g) \) or \( \tilde{g}_t = 0 \) depending whether \( \tilde{v}_t > \text{OPT} \) or not (i.e., whether \( \hat{f}_r(x) \) was truncated at \((x_t, y_t)\) or not), respectively.
Invariant 1. There exists a first order chart $G$ (derived from $G_0$) such that, for any algorithm, the sets $S(x), x \in \{0,1\}^n$ maintained by $Adv-MI$ satisfy the following property.

In every round, for every collection $F = (f_x)_{x \in \{0,1\}^n}$ of current surviving functions $f_x \in S(x)$ for $x \in \{0,1\}^n$, the function $\psi_F$ is consistent with the response returned by $Adv-MI$ under the full-information first-order oracle $O(G)$, i.e., $\psi_F(x_t, y_t) = \tilde{v}_t$ and $\tilde{g}_t \in \partial \psi_F(x_t, y_t)$.

Notice that Invariant 1 is indeed maintained after each response in Step 2 of Procedure 2: For every collection $F = (f_{\bar{x}})_{\bar{x}}$ of still surviving functions $f_{\bar{x}} \in S(\bar{x})$, by the consistency guarantee of $Adv-Cont^+$ (Item 1 of Lemma 29) the function $f_{r(x_t)}$ selected for the fiber $r(x_t)$ has value $v$ and subgradient $g$ at $y_t$; thus, Lemma 31 combined with (2.1) implies that the function $\psi_F$ has value

$$\psi_F(x_t, y_t) = \hat{f}_{r(x_t)}(x_t, y_t) = \max \left\{ f_{r(x_t)}(y_t) + \langle M_{r(x_t)}, x_t - r(x_t) \rangle, \text{OPT} \right\} = \tilde{v}_t,$$

and similarly from (2.2) we see that $\tilde{g}$ is a subgradient in $\partial \psi_F(x_t, y_t) = \partial \hat{f}_{r(x_t)}(x_t, y_t)$, as desired.

We now prove that $Adv-MI$ is $\varepsilon$-hard for $2^n \ell - 1$ rounds; using Lemma 28, this implies Theorem 18 for the case of full-information first-order oracle. Suppose the optimization algorithm runs for fewer than $2^n \cdot \ell$ iterations. Then there is a fiber $x^* \in \{0,1\}$ where $Adv-MI$ sent at most $\ell - 1$ queries to the adversary $Adv-Cont^+$ of the fiber $x^*$. Thus, by the guarantee of the latter (Item 2 of Lemma 29), the surviving set $S(x^*)$ has some finite collection of functions $f^1_{x^*}, \ldots, f^k_{x^*}$ with no common $\varepsilon$-approximate solution. Consider the collections $F^1, \ldots, F^k$ of surviving functions that have $f^1_{x^*}, \ldots, f^k_{x^*}$, respectively, for the fiber $x^*$ and any function $f_{\bar{x}} \in S(\bar{x})$ with optimal value $> \text{OPT} + \varepsilon$ for each of the other fibers $\bar{x} \neq x^*$, which exist on each of the other fibers by Item 3 of Lemma 29. By Invariant 1, all functions $\psi_{F^1}, \ldots, \psi_{F^k}$ are compatible with the responses returned by $Adv-MI$. The desired $\varepsilon$-hardness of $Adv-MI$ then follows from the following claim, which then concludes the proof.
Claim 32. The functions $\psi_{F_1}, ..., \psi_{F_k}$ share no common $\varepsilon$-approximate solution.

Proof. From the construction above, we have that $F^\dagger := F^1 \backslash \{f^1_{x^*}\} = F^2 \backslash \{f^2_{x^*}\} = ... = F^k \backslash \{f^k_{x^*}\}$. Due to ((2.3)) and the definitions of $F^1, ..., F^k$, for any fiber $\bar{x} \neq x^*$ and $f_{\bar{x}} \in F^1$, it follows that $\psi_{F_1}(\bar{x}, y) = ... = \psi_{F_k}(\bar{x}, y) = f_{\bar{x}}(y) > \text{OPT} + \varepsilon$. Thus, the $\varepsilon$-approximate solutions for the functions $\psi_{F_1}, ..., \psi_{F_k}$ only exist within the fiber $x^*$.

Given that the $\varepsilon$-approximate solutions of $f^1_{x^*}, ..., f^k_{x^*}$ are disjoint, and considering that $\psi_{F_j}(x^*, y) = f^j_{x^*}(y)$ for all $j \in [k]$, we can conclude our proof.

2.2.3 Proof of Theorem 18 for general oracles

We now prove Theorem 18 in full generality. We will do this using the exact same family of difficult functions $\psi_{F}$ from (2.3), and also with the same idea of constructing a mixed-integer adversary that produces its answers by making queries to an adversary for the continuous problems on the fibers. Since the mixed-integer adversary will need to answer queries made in the full $\mathbb{R}^n \times \mathbb{R}^d$ space by making queries in the continuous space $\mathbb{R}^d$ on each fiber, we will require that the set of permissible queries that can be made to the continuous adversary is, in some sense, as rich as the queries allowed in the full space. For example, if one allows full-information queries to be made in $\mathbb{R}^n \times \mathbb{R}^d$, but only binary queries to be made in $\mathbb{R}^d$ to the continuous adversary, one would struggle to determine how the mixed-integer adversary should answer those full-information queries by making only binary queries to the adversaries for the continuous subproblems. Specifically, for a query at $(x, y) \in \mathbb{R}^n \times \mathbb{R}^d$, knowing how $x$ affects the function values and subgradients of $\psi_{F}$, the mixed-integer adversary needs to be able to determine what response to give by making a suitably chosen query about $f_{r(x)}$ to the continuous adversary. We formalize this requirement of having the same richness of queries for the continuous subproblems as for the full $\mathbb{R}^n \times \mathbb{R}^d$ space with the concept of hereditary queries.

Hereditary queries. For simplicity, we define the notion of hereditary queries for unconstrained problems (i.e., only for value/subgradient queries), but we remark that the same idea can be applied to separation queries as well.
Definition 33. Let \( \mathcal{H}_{n,d}^{\text{val}} \) and \( \mathcal{H}_{n,d}^{\text{sub}} \) be classes of permissible function value and subgradient queries, respectively, with response sets (codomains) \( \mathcal{H}_{n,d}^{\text{val}} \) and \( \mathcal{H}_{n,d}^{\text{sub}} \). \( \mathcal{H}_{n,d}^{\text{val}} \) and \( \mathcal{H}_{n,d}^{\text{sub}} \) are said to be hereditary if the following holds for all \( n, d \in \mathbb{N} \) and functions \( M : \{0,1\}^n \to \mathbb{R}^n \). For any \( x \in \{0,1\}^n \), \( \delta \in \mathbb{R} \), \( h^{\text{val}} \in \mathcal{H}_{n,d}^{\text{val}} \), and \( h^{\text{sub}} \in \mathcal{H}_{n,d}^{\text{sub}} \), there exists \( h^{\text{val}}_\ast \in \mathcal{H}_{0,d}^{\text{val}} \), \( h^{\text{sub}}_\ast \in \mathcal{H}_{0,d}^{\text{sub}} \) and functions \( B^{\text{val}} : \mathcal{H}_{0,d}^{\text{val}} \to \mathcal{H}_{n,d}^{\text{val}} \), \( B^{\text{sub}} : \mathcal{H}_{0,d}^{\text{sub}} \to \mathcal{H}_{n,d}^{\text{sub}} \) such that

\[
B^{\text{val}}(h^{\text{val}}_\ast(v)) = h^{\text{val}}(v + \delta) \quad \forall v \in \mathbb{R},
\]

\[
B^{\text{sub}}(h^{\text{sub}}_\ast(v, g)) = h^{\text{sub}}(M(x), g) \quad \forall g \in \mathbb{R}^d
\]

Intuitively, a class of queries being hereditary has the consequence that if for a point \((x, y) \in \mathbb{R}^n \times \mathbb{R}^d\), one knows exactly the \( x \) component \( \mathcal{M}(x) \) of the subgradient, then one can simulate a query in the \( \mathbb{R}^n \times \mathbb{R}^d \) space by only making a query on the \( \mathbb{R}^d \) space, and similarly that there are queries rich enough to consider shifted function values \( v + \delta \), where the interpretation is that \( \delta \) is the effect \( x \) has on the overall function value – see (2.1).

Example 34. We show that natural permissible queries, as from Definition 6, are hereditary. Let \( \mathcal{M}(x), \delta \) be as in Definition 33.

1. (Full-information first-order oracle) If \( \mathcal{H}_{n,d}^{\text{val}} \) and \( \mathcal{H}_{n,d}^{\text{sub}} \) are simply the identity functions, then we can take \( B^{\text{val}} \) to be \( B^{\text{val}}(v) = v + \delta \) and take \( B^{\text{sub}} \) to be the “lifting/rotation” map \( B^{\text{sub}}(g) = (\mathcal{M}(x), g) \), noting that \( h^{\text{sub}}_\ast \), \( h^{\text{val}}_\ast \), \( h^{\text{sub}} \), and \( h^{\text{val}} \) are all the identity functions.

2. (General binary oracle) For the general binary oracle based on a first-order chart \( \mathcal{G} \), \( B^{\text{val}} \) and \( B^{\text{sub}} \) can be taken to be the identity map from \( \{0,1\} \) to \( \{0,1\} \), and one can take \( h^{\text{val}}_\ast(v) = h(v + \delta) \), \( h^{\text{sub}}_\ast = h(\mathcal{M}(x), g) \), which are permissible queries since all binary queries are permissible.

3. (Shifted bit oracle) If \( \mathcal{H}_{n,d}^{\text{val}} \) and \( \mathcal{H}_{n,d}^{\text{sub}} \) are from a shifted bit oracle \( \mathcal{H}^{\text{bit}} \), then \( B^{\text{val}} \) can be taken to be the identity map from \( \{0,1\} \) to \( \{0,1\} \). For a query on the function value, if \( h^{\text{val}} \) reports some bit of \( v + \delta \), then the appropriate hereditary
query is exactly the query \( h^\text{val}_u \) such that \( h^\text{val}_u(v) = h(v+\delta) \), i.e. using the shift \( u = \delta \) in the notation of Definition 6. A subgradient bit query \( h^\text{sub}(\mathcal{M}(x), g) \) returns a bit of either \( \mathcal{M}(x) \) or \( g \), so there are two cases.

i) \( h^\text{sub} \) returns the \( j \)th bit of the \( k \)th entry of \( \mathcal{M}(x) \). Set \( B^\text{sub}(\cdot) \) to return exactly that bit of \( \mathcal{M}(x) \), no matter the input to \( B^\text{sub} \), so \( h^\text{sub} \) may be chosen arbitrarily.

ii) \( h^\text{sub} \) returns the \( j \)th bit of the \( k \)th entry of \( g \). Set \( B^\text{sub} \) to be the identity map from \( \{0,1\} \) to \( \{0,1\} \), and set \( h^\text{sub} \) to return the desired bit of \( g \).

4. (Inner product threshold queries) If \( H^\text{val}_{n,d} \) and \( H^\text{sub}_{n,d} \) consist of the inner product threshold queries, take both \( B^\text{val} \) and \( B^\text{sub} \) to be the identity maps. For function value queries \( h^\text{val}_{u,c}(v) = \text{sgn}(u \cdot (v+\delta) - c) \), use

\[
h^\text{val}_u(v) := h^\text{val}_{u,c-u\delta}(v) = \text{sgn}(uv - (c - u\delta)),
\]

since then \( h^\text{val}_u(v) = \text{sgn}(uv - (c - u\delta)) = \text{sgn}(u \cdot (v+\delta) - c) = h^\text{val}_{u,c}(v) \) as desired. For subgradient queries \( h^\text{sub}_{u}(\mathcal{M}(x), g) = \text{sgn}(\langle u, \mathcal{M}(x), g \rangle - c) \), with \( u \in \mathbb{R}^{n+d} \), denote by \( u_n \) the vector of the first \( n \) entries of \( u \), and by \( u_d \) the vector of the last \( d \) entries of \( u \). One may use

\[
h^\text{sub}_u(g) := h^\text{sub}_{u_d,c-(u_n,\mathcal{M}(x))}(g) = \text{sgn}\left(\langle u_d, g \rangle - (c - \langle u_n, \mathcal{M}(x) \rangle)\right),
\]

since then we similarly have

\[
h^\text{sub}_u(g) = \text{sgn}\left(\langle u_d, g \rangle - (c - \langle u_n, \mathcal{M}(x) \rangle)\right) = \text{sgn}(\langle u, (\mathcal{M}(x), g) \rangle - c) = h^\text{sub}_{u}(\mathcal{M}(x), g)
\]

as desired. For these hereditary queries, note that \( h^\text{val}_{u,c-u\delta} \) and \( h^\text{sub}_{u_d,c-(u_n,\mathcal{M}(x))} \) are indeed in \( H^\text{val}_{0,d} \) and \( H^\text{sub}_{0,d} \), respectively, since \( u \in \mathbb{R}, u_n \in \mathbb{R}^n \), and \( u_d \in \mathbb{R}^d \).

We remark here that \( H^{\text{bit}} \), without the permitted “shifts” allowed in \( H^{\text{bit}^*} \) is not hereditary, as it may not satisfy condition (2.5) for the function values for all \( \delta \); however, any lower bounds obtained with \( H^{\text{bit}^*} \) must also hold for \( H^{\text{bit}} \), since the former is a richer
class of queries.

**Definition of the adversary.** We will define Adv-Cont+ analogously as in the full-information case, now receiving queries in \( \mathcal{H} \) according to the oracle setting considered. Adv-Cont+ will be \( \varepsilon \)-hard for \( \ell \) rounds answering queries from \( \mathcal{H} \), and after \( \ell - 1 \) rounds it commits to a single surviving function with optimal value \( > \text{OPT} + \varepsilon \). As queries for general oracles using first-order information consist of a point \( z = (x, y) \in \mathbb{R}^n \times \mathbb{R}^d \) and a permissible query \( h \in \mathcal{H} \), let us write \( (x, y, h) \) for notational simplicity to denote such a query.

We describe here the behavior of Adv-Cont+ in the general oracle case, and such that it satisfies the same invariant of Lemma 29 as in the full-information case, i.e. it is \( \varepsilon \)-hard for \( \ell \) rounds and only keeps a single surviving function with optimal value at least \( \text{OPT} + \varepsilon \) after \( \ell \) queries have been made.

**Procedure 3. Adv-Cont+**

Initialize set of survived functions \( S_0 = \mathcal{F}_{\text{cont}} \)

For each round \( t = 1, 2, \ldots \):

1. Receive query point \( (y_t, h_t) \in \mathbb{R}^d \times \mathcal{H} \) from the optimization algorithm.

2. If \( t \leq \ell - 1 \): Send \( (y_t, h_t) \) to the adversary Adv-Cont, receiving back the answer \( \alpha \). Obtain \( S_t \) by removing from \( S_{t-1} \) the functions \( f \) that are not consistent with this response under any first-order chart \( \mathcal{G} \), namely \( f \) for which \( h_t(f(y_t)) \neq \alpha \) if \( h_t \in \mathcal{H}_{\text{val}} \), or for which there does not exist a \( g_t \in \partial f(y_t) \) such that \( h_t(f(y_t), g_t) = \alpha \) if \( h_t \in \mathcal{H}_{\text{sub}} \).

   Send the response \( \alpha \) to the optimization algorithm.

3. If \( t = \ell \): Since Adv-Cont is \( \varepsilon \)-hard for \( \ell - 1 \) rounds, there is a finite collection of functions \( \{f_1, \ldots, f_k\} \subset S_{t-1} \cap \mathcal{F}_{\text{cont}} \) that do not share an \( \varepsilon \)-solution. Define their pointwise maxima \( f_{\text{max}} = \max\{f_1, \ldots, f_k\} \) and set \( S_t + k = \{f_{\text{max}}\} \), for all \( k = 0, 1, 2, \ldots \).
Set the value $v_t$ to be $f_{\text{max}}(y_t)$ and set $g_t$ to be a subgradient in $\partial f_{\text{max}}(y_t)$ (consistent with what the first order chart $G_0$ gives for $f_1, \ldots, f_k$ at $y_t$, if $y_t$ has been queried in an earlier round), and send the response $h_t(v_t)$ or $h_t(g_t)$ to the optimization algorithm, according to whether $h_t \in \mathcal{H}^\text{val}$ or $h_t \in \mathcal{H}^\text{sub}$ respectively.

4. If $t > \ell$: Let $f_{\text{max}}$ be the only function in $S_{t-1}$. If $y_t$ was queried in an earlier round $k$, answer $(v_k, g_k)$. Otherwise, as in the step above, set the value $v_t$ to be $f_{\text{max}}(y_t)$ and set $g_t$ to be any subgradient in $\partial f_{\text{max}}(y_t)$, and again send the appropriate response $h_t(v_t)$ or $h_t(g_t)$ to the optimization algorithm.

Proving that this $\text{Adv-Cont}^+$ satisfies the invariant from Lemma 29 follows exactly the same steps as in the full-information case. Hence, using this $\text{Adv-Cont}^+$ and the same family of functions $\psi_F$ from (2.3), we will be able to construct $\text{Adv-MI}$ for this general oracle case to satisfy a version of Invariant 1, slightly modified for this general case to refine what we mean by functions being consistent with responses given to the more general queries. To achieve this, let $\text{Adv-MI}$ operate according to the following procedure.

**Procedure 4. Adv-MI**

Instantiate a copy of $\text{Adv-Cont}^+$ on each fiber $x \in \{0,1\}^n$, and let $S(x)$ denote the set of surviving functions maintained by this copy, initialized to $\mathcal{F}^\text{cont}$.

Set $M_x := 3MR \cdot sgn(x - 0.5 \cdot 1)$, for any $x \in \{0,1\}^n$, as in (2.1). For each round $t = 1, 2, \ldots$:

1. $\text{Adv-MI}$ receives the query $(x_t, y_t, h_t)$ from the algorithm. Set $\delta := \langle M_{r(x_t)}, x_t - r(x_t) \rangle$.

2. Send the function value threshold query that answers “is $f_{r(x_t)}(y_t) \leq OPT + \delta$?” to the adversary $\text{Adv-Cont}^+$ of the closest fiber $r(x_t)$, which responds and updates its set of surviving functions $S(r(x_t))$. 

54
If the answer is yes, **Adv-MI** responds \( h_t(OPT) \) or \( h_t(0) \) to the original query, according to whether \( h_t \in \mathcal{H}^{val} \) or \( h_t \in \mathcal{H}^{sub} \), respectively.

Else, determine an appropriate \( B \) and hereditary query \( h^* \) as from Definition 33 using \( r(x_t), M_r(x_t), \delta \) and \( h_t \), and send the query \((y_t, h^*)\) to the adversary **Adv-Cont+** of the closest fiber \( r(x_t) \), which then returns some answer \( \alpha \) and updates its set of surviving functions \( S(r(x_t)) \). **Adv-MI** returns \( B(\alpha) \) as its response to the original query.

**Invariant 2.** There exists a first order chart \( \mathcal{G} \) (derived from \( \mathcal{G}_0 \)) such that, for any algorithm, the sets \( S(x), x \in \{0,1\}^n \) maintained by **Adv-MI** satisfy the following property.

In every round, for every collection \( F = (f_x)_{x \in \{0,1\}^n} \) of current surviving functions \( f_x \in S(x) \) for \( x \in \{0,1\}^n \), the function \( \psi_F \) is consistent with the response returned by **Adv-MI** under the oracle \( \mathcal{O}(\mathcal{H}, \mathcal{G}) \), i.e., the response **Adv-MI** gives is equal to \( h_t(\bar{v}_t, \bar{g}_t) \) for \( \bar{v}_t = \psi_F(x_t, y_t) \) and some \( \bar{g}_t \in \partial \psi_F(x_t, y_t) \).

We claim that the Invariant 2 is maintained after each response in Step 2 of Procedure 4.

If **Adv-Cont+** answers yes, then \( \psi_F(x_t, y_t) = OPT \) and \( 0 \in \partial \psi_F(x_t, y_t) \) for any collection of surviving functions \( F = (f_x)_{x} \), since all the extensions \( \hat{f}_x \) as in (2.1) that are consistent with this affirmative response in Step 2 must be truncated at \((x_t, y_t)\). To see this, note that due to the choice of \( \delta \), the query “is \( f_{r(x_t)}(y_t) \leq OPT + \delta ? \)” is equivalent to asking whether \( \hat{f}_{r(x_t)} \) has value \( OPT \) at \((x_t, y_t)\) (i.e., was truncated), and Lemma 31 guarantees that we only need to consider \( \hat{f}_{r(x_t)} \) for \( \psi_F(x_t, y_t) \).

If the answer is no, then Lemma 31 combined with (2.1) implies that for every choice \( F \) of surviving functions from each fiber, the function \( \psi_F \) has value

\[
\psi_F(x_t, y_t) = \hat{f}_{r(x_t)}(x_t, y_t) = \max \left\{ f_{r(x_t)}(y_t) + \langle M_{r(x_t)}, x_t - r(x_t) \rangle , \quad OPT \right\} = f_{r(x_t)}(y_t) + \delta,
\]

(2.7)
and for its subgradient we have

\[ g \in \partial \hat{f}(x_t) \implies (M_r(x_t), g) \in \partial \psi_F(x_t, y_t), \]  

(2.8)

by Lemma 31 and (2.2).

Suppose first that \( h_t \in H_{val} \) was a function value query, and denote by \( B_{val} \) and \( h^*_{val} \) the transformation and hereditary query that \( \text{Adv-MI} \) uses, according to definition 33, giving \( B_{val}(h^*_{val}(v)) = h_t(v + \delta) \) for all \( v \in \mathbb{R} \). For every collection \( F = (f_x)_x \) of surviving functions \( f_x \in S(\bar{x}) \), by the consistency guarantee of \( \text{Adv-Cont}^+ \) (Item 1 of Lemma 29), the function \( f_r(x_t) \) selected for the fiber \( r(x_t) \) has response \( h^*_{val}(f_r(x_t)(y_t)) = \alpha \). Then, from the definition of hereditary queries and (2.7), we have \( B_{val}(h^*_{val}(f_r(x_t)(y_t))) = h_t(f_r(x_t)(y_t) + \delta) = h_t(\psi_F(x_t, y_t)) \), and so \( \psi_F \) is indeed consistent with the response \( B_{val}(\alpha) \) provided by \( \text{Adv-MI} \).

If instead \( h \in H_{sub} \) was a subgradient query, again denote \( B_{sub} \) and \( h^*_{sub} \) as the appropriate transformation and hereditary query, with \( B_{sub}(h^*_{sub}(g)) = h(M_r(x), g) \) for all \( g \in \mathbb{R}^d \). Again, for every choice \( F \) of surviving functions on the fibers, the function \( f_r(x_t) \) on \( r(x_t) \) has \( h^*_{sub}(g) = \alpha \), for some \( g \in \partial f_r(x_t)(y_t) \). Then, from the definition of hereditary queries and (2.8), \( B_{sub}(h^*_{sub}(g)) = h_t(M_r(x_t), g) = h_t(g_\psi) \), with \( g_\psi \in \partial \psi_F(x_t, y_t) \). Hence, whether \( h_t \) is a function value or subgradient query, all functions \( \psi_F \) for choices \( F \) of the surviving functions on the fibers are consistent with the responses given by \( \text{Adv-MI} \), for the oracle \( O(G, \mathcal{H}) \) with permissible queries \( \mathcal{H} \) and the first-order chart \( G \) from Invariant 2.

We now prove that \( \text{Adv-MI} \) is \( \varepsilon \)-hard for \( 2^{n-1} \ell - 1 \) rounds, thus proving Theorem 18 in the general case. Since \( \text{Adv-MI} \) makes at most 2 queries to \( \text{Adv-Cont}^+ \) in every round (Step 2) of Procedure 4, if the optimization algorithm runs for fewer than \( 2^{n-1} \cdot \ell \) iterations, there is a fiber \( x^* \in \{0, 1\} \) where \( \text{Adv-MI} \) sent at most \( \ell - 1 \) hereditary queries to the adversary \( \text{Adv-Cont}^+ \) of the fiber \( x^* \). Thus, by the guarantee of the latter (Item 2 of Lemma 29), the surviving set \( S(x^*) \) has some finite collection of functions \( f^1_{x^*}, ..., f^k_{x^*} \) with no common \( \varepsilon \)-approximate solution, and the remainder of the proof follows as in the full-information case, by considering \( \psi_{F^T_1}, ..., \psi_{F^T_k} \) that have \( f^1_{x^*}, ..., f^k_{x^*} \) on the fiber.
2.3 Proof of Theorem 20

To demonstrate Theorem 20, we need to introduce the idea of information memory of any query strategy/algorithm.

**Definition 35.** A first-order query strategy with information memory comprises three functions:

1. $\phi_{\text{query}} : \{0, 1\}^* \to [-R, R]^n \times [-R, R]^d$

2. $\phi_{\text{sep update}} : (\mathbb{R}^n \times \mathbb{R}^d) \times \{0, 1\}^* \to \{0, 1\}^*$

3. $\phi_{\text{val update}} : \mathbb{R} \times \{0, 1\}^* \to \{0, 1\}^*$

4. $\phi_{\text{sub update}} : (\mathbb{R}^n \times \mathbb{R}^d) \times \{0, 1\}^* \to \{0, 1\}^*$,

where $\{0, 1\}^*$ denotes the set of all binary strings (finite sequences over $\{0, 1\}$), including the empty string.

Given access to a first-order chart $G$, the query strategy maintains an information memory $r_k$ at every iteration $k \geq 0$, which is a finite length binary string in $\{0, 1\}^*$, with $r_0$ initialized as the empty string. At every iteration $k = 1, 2, \ldots$, the query strategy computes $z_k := \phi_{\text{query}}(r_{k-1})$ and updates its memory using either $r_k = \phi_{\text{sep update}}(g_{\text{sep}}(\hat{f}, \hat{C}), r_{k-1})$, $r_k = \phi_{\text{val update}}(g_{\text{val}}(\hat{f}, \hat{C}), r_{k-1})$ or $r_k = \phi_{\text{sub update}}(g_{\text{sub}}(\hat{f}, \hat{C}), r_{k-1})$, where $(\hat{f}, \hat{C})$ is the unknown true instance. After finitely many iterations, the query strategy does a final computation based on its information memory and reports an $\varepsilon$-approximate solution, i.e., there is a final function $\phi_{\text{fin}} : \{0, 1\}^* \to \mathbb{Z}^n \times \mathbb{R}^d$.

The information memory complexity of an algorithm for an instance is the maximum length of its information memory $r_k$ over all iterations $k$ during the processing of this instance.

The following proposition allows us to relate the information memory complexity...
of first-order algorithms with information complexity under access to a general binary oracle using first-order information.

**Proposition 36.** Let \( \mathcal{G} \) be a first-order chart. For any first-order query strategy \( \mathcal{A} \) with information memory that uses \( \mathcal{G} \), there exists a query strategy \( \mathcal{A}' \) using the general binary oracle based on \( \mathcal{G} \), such that for any instance \((f, C)\), if \( \mathcal{A} \) stops after \( T \) iterations with information memory complexity \( Q \), \( \mathcal{A}' \) stops after making at most \( Q \cdot T \) oracle queries.

Conversely, for any query strategy \( \mathcal{A}' \) using the general binary oracle based on \( \mathcal{G} \), there exists a first-order query strategy \( \mathcal{A} \) with information memory such that for any instance \((f, C)\), if \( \mathcal{A}' \) stops after \( T \) iterations, \( \mathcal{A} \) stops after making at most \( T \) iterations with information memory complexity at most \( T \).

**Proof.** Let \( \mathcal{A} \) be a first-order query strategy with information memory. We can simulate \( \mathcal{A} \) by the query strategy whose queries are precisely the bits of the information memory state \( r_k \) at each iteration \( k \) of \( \mathcal{A} \). More formally, the query is \( z = \phi_{\text{query}}(r_{k-1}) \) and \( h(\cdot) = (\phi_{\text{update}}(\cdot, r_{k-1}))_i, h(\cdot) = (\phi_{\text{val}}(\cdot, r_{k-1}))_i, \) or \( h(\cdot) = (\phi_{\text{sub}}(\cdot, r_{k-1}))_i, \) depending on which type of query was made, where \( i \) indexes different bits of the corresponding binary string.

Conversely, given a query strategy \( \mathcal{A}' \) based on the general binary oracle, we can simulate it with a first-order query strategy with information memory where in each iteration, we simply append the new bit queried by \( \mathcal{A}' \) to the current state of the memory.

We need the following result derived from Marsden et al. [79] on information memory complexity.

**Theorem 37.** [79, Theorem 1] For every \( \delta \in [0, 1/4] \), there is a class of instances \( \mathcal{I} \subset \mathcal{I}_{n,d,R,p,M} \), where \( n = 0 \), and a first-order chart \( \mathcal{G} \) such that any first-order query strategy with information memory must have either \( d^{1.25-\delta} \) information memory complexity (in the worst case) or make at least \( \tilde{\Omega}(d^{1+\frac{4}{3}\delta}) \) iterations (in the worst case).
Proof of Theorem 20. In the case when \( n = 0 \), we can set \( \delta = \frac{3}{28} \) in Theorem 37 to obtain that any first-order query strategy uses either \( d^{8/7} \) information memory or makes at least \( \Omega(d^{8/7}) \) iterations. Using the second part of Proposition 36, we obtain the lower bound of \( \tilde{\Omega}(d^{8/7}) \) on the number of queries made by any query strategy using the general binary oracle based on \( G \).

Applying Theorem 18 enables us to extend the bound to the mixed-integer scenario \((n > 0)\). Further, by integrating this with Corollary 19, we can obtain the desired bound. \( \square \)

2.4 Proof of Theorems 23 and 24

We will use \( B_\infty(p, \delta) \) to denote the \( \ell_\infty \) ball of radius \( \delta \) centered at \( p \in \mathbb{R}^n \times \mathbb{R}^d \), i.e.,

\[
B_\infty(p, \delta) = \{ z \in \mathbb{R}^n \times \mathbb{R}^d : \| z - p \|_\infty \leq \delta \}.
\]

Recall that we consider the subclass of instances \((f, C) \in \mathcal{I}_{n,d,R,\rho,M} \) such that the fiber containing the optimal solution also contains a point that is \( \rho \)-deep in \( C \), that is: if \((x^*, y^*) \in \mathbb{Z}^n \times \mathbb{R}^d \) is an optimal solution for this instance, then there is a point \((x^*, \bar{y}) \) such that the full-dimensional ball \( B_\infty((x^*, \bar{y}), \rho) \) is contained in \( C \). We use \( \mathcal{I}_{n,d,R,\rho,M} \) to denote this subclass of instances.

We will use \( C_{-\rho} := \{ z \in C : B_\infty(z, \rho) \subseteq C \} \) to denote the set of all \( \rho \)-deep points in \( C \).

Our strategy for proving Theorems 23 and 24 is to: 1) solve the the problems using approximate subgradients/separating hyperplanes; 2) use bit queries/inner product sign queries to construct such approximations.

For the first item, we use an algorithm designed by Oertel [90] (see also [9]) based on the concept of a centerpoint: this is a point in the convex set where every halfspace supported on it cuts off a significant (mixed-integer) volume of the set. The algorithm maintains an outer relaxation \( P \) of the feasible region \( C \) in every iteration, and repeatedly applies separation or subgradient-based cuts through the centerpoint of \( P \). The assumption that the feasible region contains a ball (in the optimal fiber) establishes a volume lower bound that essentially limits the number of iterations of the algorithm. While the original algorithm in [90, 9] uses exact separation/subgradient oracles, we show, not surprisingly, that approximate ones suffice. To prove Theorem 24, we employ
a similar approach. However, due to the continuous nature of the setting, we can obtain a better upper bound compared to Theorem 23 by applying a stronger bound on the centerpoints from Grünbaum [53].

The next item is to construct approximate separation/subgradient oracles by making only a limited number of binary queries on the separating hyperplanes and/or subgradients. In case of bit queries $\mathcal{H}^{\text{bit}}$ this can be easily done by querying enough bits of the latter. The case of inner product sign queries $\mathcal{H}^{\text{dir}}$, where we can pick a direction $a$ and ask “Is $\langle a, g \rangle \geq 0$?” for the subgradient or separating hyperplane $g$, is more interesting. It boils down to approximating the vector $g$ (subgradient/separating hyperplane) using few such queries.²

To formalize the first item, we begin by defining three approximate oracles as follows.

**Definition 38.** We have the following:

- An $\varepsilon$-approximate separation oracle $\hat{g}^{\text{sep}}$ is such that $\hat{g}^{\text{sep}}_\varepsilon(f, C) = 0$ iff $\bar{z}$ belongs to $C$, and otherwise the cut $\langle \hat{g}^{\text{sep}}_\varepsilon(f, C), z \rangle \leq \langle \hat{g}^{\text{sep}}_\varepsilon(f, C), \bar{z} \rangle$ is valid for all $\varepsilon$-deep points $z \in C_{-\varepsilon}$.

- An $\varepsilon$-approximate value cut oracle $\hat{g}^{\text{sub}}$ is such that for every $z$ such that $\langle \hat{g}^{\text{sub}}_\varepsilon(f, C), z \rangle \geq \langle \hat{g}^{\text{sub}}_\varepsilon(f, C), \bar{z} \rangle$, we have $f(z) \geq f(\bar{z}) - \varepsilon$.

- An $\varepsilon$-approximate value comparison oracle is such that for every function $f : [-R, R]^{n+d} \to [-U, U]$ and every pair of points $z, z'$ we obtain the answer to the query “Is $f(z) \leq f(z') + \varepsilon$?”.

Then the first item can be formalized as the following.

**Theorem 39.** There exists an algorithm that, for any $M, R > 0$, $0 < \varepsilon \leq MR$ and $\rho > 0$, can report an $\varepsilon$-approximate solution for every instance in $\mathcal{T}^{\text{deep}}_{n,d,R,\rho,M}$, using at

²This is related to (actively) learning the linear classifier whose normal is given by $g$ [4]. These methods can perhaps be adapted to our setting, but we present a different and self-contained statement and proof. See the discussion at the end of Section 2.4.2.
most

\[ O\left(2^n(n + d)d \log\left(\frac{MR}{\min\{\rho, 1\} \varepsilon}\right)\right) \]

oracle calls, given access to any \( \rho' \)-approximate separation oracle, \( \varepsilon' \)-approximate value cut oracle, and \( \varepsilon' \)-approximate value comparison oracle with \( \rho' = \frac{\varepsilon' \rho}{MR} \) and \( \varepsilon' = \frac{\varepsilon}{6} \).

For the continuous setting with \( n = 0 \), the bound can be improved to

\[ O\left(d \log\left(\frac{MR}{\min\{\rho, 1\} \varepsilon}\right)\right). \]

We postpone the proof of Theorem 39 to Section 2.4.1.

The next lemma shows that one can implement the approximate oracles from Definition 38 using bit queries and inner product sign queries.

**Lemma 40.** Consider a first-order chart \( G \). Let \( f : \mathbb{R}^{n+d} \to \mathbb{R} \) be a convex \( M \)-Lipschitz function taking values in \([-U, U]\), and \( C \subseteq [-R, R]^{n+d} \) a convex set.

Then for every pair of points \( \bar{z}, \bar{z}' \in [-R, R]^{n+d} \), we can obtain an \( \varepsilon \)-approximate separation oracle vector \( \hat{g}_{\text{sep}}(f, C) \), an \( \varepsilon \)-approximate value cut vector \( \hat{g}_{\text{sub}}(f, C) \), and an \( \varepsilon \)-approximate value comparison between \( \bar{z} \) and \( \bar{z}' \) using either a sequence of bit queries from \( \mathcal{H}_{\text{bit}} \), or a sequence of inner product sign queries from \( \mathcal{H}_{\text{dir}} \), on the separating hyperplane \( g_{\text{sep}}(f, C) \), the subgradient \( g_{\text{sub}}(f, C) \) and the function value \( g_{\text{val}}(f, C) \). The number of required queries to implement the approximate oracles is \( O\left((n + d) \log\frac{(n+d)R}{\varepsilon}\right) \), \( O\left((n + d) \log\frac{(n+d)MR}{\varepsilon}\right) \) and \( O\left(\log\frac{U}{\varepsilon}\right) \) respectively, for both \( \mathcal{H}_{\text{bit}} \) and \( \mathcal{H}_{\text{dir}} \).

The proof of Lemma 40 is deferred to Section 2.4.2.

**Proof.** Theorems 23 and 24 follow from Theorem 39 and Lemma 40.

2.4.1 Proof of Theorem 39

We first describe the centerpoint algorithm for convex optimization due to Oertel [90] (see also [9]). Let the mixed-integer volume of a (Borel) set \( U \in \mathbb{R}^{n+d} \) be \( \mu(U) := \)
\[ \sum_{x \in \mathbb{Z}^n} \text{vol}_d(U \cap (\{x\} \times \mathbb{R}^d)) \], where \( \text{vol}_d \) is the \( d \)-dimensional Lebesgue measure. The following notion is the main element of the algorithm.

**Theorem 41** (Mixed-integer centerpoint [90, 9]). For any compact convex set \( C \subseteq \mathbb{R}^{n+d} \), there is a point \( z \in C \cap (\mathbb{Z}^n \times \mathbb{R}^d) \) (called a mixed-integer centerpoint) such that for every halfspace \( H \) with \( z \) on its boundary, we have \( \mu(C \cap H) \geq \frac{1}{2^{n(d+1)}} \mu(C) \).

Algorithm 2 below is the centerpoint-based algorithm for solving mixed-integer convex optimization problems from [90], restated in terms of approximate separation oracle, namely set \((\tilde{g}_z^\text{sep}(f, C)), \text{value cut} (\tilde{g}_z^\text{sub}(f, C))\) and value comparison oracles.

**Algorithm 2 Centerpoints**

1. Initialize the version set \( P_0 := [-R, R]^{n+d} \) and the collection of feasible points \( F = \emptyset \).

   For iterations \( t = 0, \ldots, T - 1 \):
   
   (a) Let \( z_t \in P_t \cap (\mathbb{Z}^n \times \mathbb{R}^d) \) be a mixed-integer centerpoint of \( P_t \) given by Lemma 41.
   
   (b) If the \( \rho \)-approximate separation oracle says that \( z_t \) is infeasible for \( C \), add the cut \( \langle \tilde{g}_z^\text{sep}(f, C), z \rangle \leq \langle \tilde{g}_z^\text{sep}(f, C), z_t \rangle \) to \( P_t \), namely set \( P_{t+1} = P_t \cap \{ z : \langle \tilde{g}_z^\text{sep}(f, C), z \rangle \leq \langle \tilde{g}_z^\text{sep}(f, C), z_t \rangle \} \).
   
   (c) Else, add \( z_t \) to the set of feasible solutions \( F \), and add the cut from the \( \varepsilon' \)-approximate value cut oracle, namely set \( P_{t+1} = P_t \cap \{ z : \langle \tilde{g}_z^\text{sub}(f, C), z \rangle \leq \langle \tilde{g}_z^\text{sub}(f, C), z_t \rangle \} \).

2. Finally, return a point \( \hat{z} \) from \( F \) that has approximately the minimum value among all solutions in \( F \), namely such that \( f(\hat{z}) \leq \min_{z \in F} f(z) + \varepsilon' \). This can be accomplished by asking \(|F| - 1\) queries to the \( \varepsilon' \)-approximate value comparison oracle.

To analyze this algorithm we need the following technical lemma regarding deep points in instances in \( I_{n,d,R,\rho,M}^{\text{deep}} \).

**Lemma 42.** For every instance \((f, C)\) in \( I_{n,d,R,\rho,M}^{\text{deep}} \), and for every \( 0 < \varepsilon < 2MR \), there is an \( \varepsilon \)-approximate solution \( z \) such that the ball \( B_\infty(z, \frac{\varepsilon}{2MR}) \) is contained in \( C \).

**Proof.** Let \( z^* = (x^*, y^*) \) be an optimal solution for the instance, and let \( \tilde{z} = (x^*, \tilde{y}) \) be such that \( B_\infty(\tilde{z}, \rho) \) is contained in \( C \). For \( \alpha = \frac{\varepsilon}{2MR} \), we claim that the point \( z := (1 - \alpha)z^* + \alpha \tilde{z} \) has the desired properties. First, by convexity of \( C \) we have that the desired ball \( B_\infty(z, \frac{\varepsilon \rho}{2MR}) = (1 - \alpha)z^* + \alpha B_\infty(\tilde{z}, \rho) \) is contained in \( C \). In addition, since \( z - z^* = \alpha \cdot (0, \tilde{y} - y^*) \) and \( f \) is \( M \)-Lipschitz over the integer fibers, we have

\[
 f(z) \leq f(z^*) + \alpha M \cdot \|y^* - \tilde{y}\|_\infty \leq f(z^*) + \varepsilon,
\]
where the last inequality uses that $y^*, \bar{y} \in [-R, R]^d$ and the definition of $\alpha$; so $z$ is an $\varepsilon$-approximate solution. This concludes the proof.

\textit{Proof of Theorem 39.} We show that Algorithm 2 with number of iterations set as

$$T = 2^n(n + d)(d + 1) \ln \left( \frac{2R}{\min\{\rho', 1\}} \right) \in O \left( 2^n(n + d)d \ln \left( \frac{MR}{\min\{\rho, 1\} \varepsilon} \right) \right)$$

has the desired properties. First, regarding the number of oracle queries performed: in each iteration it performs at most 2 approximate separation/value cut queries, and in Step 2 it performs $|F| - 1 \leq T$ approximate value comparison queries. In total, the algorithm performs at most $3T$ queries, giving the desired complexity.

Now we show that the algorithm returns an $\varepsilon$-optimal solution. For that, it suffices to show that for this value of $T$, the set of feasible solutions $F$ contains an $\varepsilon/2$-optimal solution.

Using Lemma 42, let $\bar{z}$ be an $\varepsilon'$-approximate solution such that the ball $B_\infty(\bar{z}, \frac{\rho'}{2MR}) = B_\infty(\bar{z}, 2\rho')$ is contained in $C$. Thus, the ball $B_\infty(\bar{z}, \rho')$ is contained in $C_{-\rho'}$. Since the cut added to $P_t$, whether in Step (b) or (c), goes through the centerpoint $z_t$, the mixed-integer volume of $P_t$ is reduced by a factor of at least $(1 - \frac{1}{2^n(d+1)})$ in each iteration (to simplify the notation let $\alpha := \frac{1}{2^n(d+1)}$). The definition of $T$ shows that the last set $P_T$ has mixed-integer volume at most

$$\left(1 - \alpha\right)^T \mu(P_0) = \left(1 - \alpha\right)^T (2R)^{n+d} \leq e^{-T\alpha} (2R)^{n+d} \leq \left( \min\{\rho', 1\} \right)^{n+d} \leq \left( \min\{\rho', 1\} \right)^d.$$

(2.9)

Let $X$ be the intersection of $B_\infty(\bar{z}, \rho')$ with the mixed-integer fiber containing $\bar{z}$. $X$ has the same structure as an $\ell_\infty$ ball of radius $\rho'$ in $\mathbb{R}^d$, and thus has volume at least $(2\rho')^d$, which is strictly bigger than the right-hand side of (2.9). This means that some mixed-integer point from $X$ is cut off by one of the hyperplanes applied by the algorithm. However, such a hyperplane cannot be one added in Step (b), since $B_\infty(\bar{z}, \rho') \subseteq C_{-\rho'}$ and the cuts in that step are valid for $C_{-\rho'}$. Thus, there is an iteration $t$ that added a Step (c) approximate value cut that cut off a point $\tilde{z} \in X$. Thus, $\langle \bar{g}^{\text{sub}}_z(f, C), \tilde{z} \rangle >$
$\langle \hat{g}_{z_t}^{\text{sub}}(f, C), z_t \rangle$. Since this is an $\varepsilon'$-approximate value cut, we get that $f(\tilde{z}) \geq f(z_t) - \varepsilon'$.

Since $f$ is $M$-Lipschitz on the fiber containing $\bar{z}$ and $\tilde{z}$, and the $\ell_{\infty}$ distance between $\bar{z}$ and $\tilde{z}$ is at most $\rho'$, we get

$$f(z_t) \leq f(\tilde{z}) + \varepsilon' \leq f(\tilde{z}) + \rho' M + \varepsilon' \leq \text{OPT} + 2\varepsilon' + \rho' M \leq \text{OPT} + \frac{\varepsilon}{2},$$

where the last inequality uses that $\rho' = \frac{\varepsilon' \rho}{4MR} \leq \frac{\varepsilon'}{M}$ and that $\varepsilon' = \frac{\varepsilon}{6}$.

This shows that the set of feasible solutions $F$ contains an $\frac{\varepsilon}{2}$-approximate solution, namely the above $z_t$, as desired. This concludes the proof for the mixed-integer setting.

For the continuous setting with $n = 0$, the improved bound follows from using an improved bound on centerpoints due to Grünbaum [53]. Specifically, $\alpha$ in the left hand side of ((2.9)) can be taken to be $\frac{1}{e}$, where $e$ is Euler’s constant.

### 2.4.2 Proof of Lemma 40

The proof will be divided into two parts: first, we show how to obtain approximate oracles using bit queries, after which we show how to do the same using inner product sign queries.

**Obtaining approximate oracles using bit queries.** Since $f$ is $M$-Lipschitz with respect to the $\ell_{\infty}$ norm, any subgradient has $\ell_{\infty}$ norm at most $M$. Thus, letting $\varepsilon' := \frac{\varepsilon}{2(n + d) R}$, we will query the sign and the bits indexed by the integers $[\log M], [\log M] - 1, \ldots, -[\log \frac{1}{\varepsilon'}]$ of each coordinate of $g_{\bar{z}}^{\text{sub}}(f, C)$ (nonnegative integers index the bits before the decimal, and negative integers index the bits after the decimal in the binary representation). This can be done by querying the bits of $g_{\bar{z}}^{\text{sub}}(f, C)$ for a total of $(n + d)(\log \frac{M}{\varepsilon'} + 2)$ queries – for each coordinate, one queries $\log M + \log(\frac{1}{\varepsilon'}) + 1$ bits for the desired precision and one additional bit for the sign. This gives a vector $\hat{g}_{\bar{z}}^{\text{sub}}(f, C)$ such that $\|g_{\bar{z}}^{\text{sub}}(f, C) - \hat{g}_{\bar{z}}^{\text{sub}}(f, C)\|_{\infty} \leq \sum_{i > \log \frac{1}{\varepsilon'}} \frac{1}{2^i} \leq \varepsilon'$.

Then $\hat{g}_{\bar{z}}^{\text{sub}}(f, C)$ is an $\varepsilon$-approximate value cut. Let $g := g_{\bar{z}}^{\text{sub}}(f, C)$ and $\hat{g} := \hat{g}_{\bar{z}}^{\text{sub}}(f, C)$ to simplify notation. For every $z \in [-R, R]^{n+d}$ such that $\langle \hat{g}, z \rangle \geq \langle \hat{g}, \tilde{z} \rangle$ we
have by convexity of $f$

$$f(z) - f(\tilde{z}) \geq \langle g, z - \tilde{z} \rangle = \langle \hat{g}, z - \tilde{z} \rangle + \langle g - \hat{g}, z - \tilde{z} \rangle \geq 0$$

$$\geq -\|g - \hat{g}\|_\infty \cdot \|z - \tilde{z}\|_1 \geq -2\varepsilon'(n + d)R = -\varepsilon,$$  \hspace{1cm} (2.10)

where the second inequality follows from Hölder’s inequality, and so $\hat{g}$ has the desired property.

For $\hat{g}_{\tilde{z}}^{\text{sep}}(f, C)$, recall that by assumption the separating vector $g_{\tilde{z}}^{\text{sep}}(f, C)$ has unit length, and hence $\|g_{\tilde{z}}^{\text{sep}}(f, C)\|_\infty \leq 1$. Then querying the sign and the bits indexed by $0, -1, \ldots, -\log \frac{1}{\varepsilon'}$ of each coordinate of $g_{\tilde{z}}^{\text{sep}}(f, C)$ we obtain a vector $\hat{g}_{\tilde{z}}^{\text{sep}}(f, C)$ such that $\|g_{\tilde{z}}^{\text{sep}}(f, C) - \hat{g}_{\tilde{z}}^{\text{sep}}(f, C)\|_\infty \leq \varepsilon'$.

We claim that $\hat{g}_{\tilde{z}}^{\text{sep}}(f, C)$ is an $\varepsilon$-approximate separation oracle, namely the inequality $\langle \hat{g}_{\tilde{z}}^{\text{sep}}(f, C), z \rangle \leq \langle \hat{g}_{\tilde{z}}^{\text{sep}}(f, C), \tilde{z} \rangle$ holds for all $z \in C_{-\varepsilon}$. As before, to simplify the notation we use $g := g_{\tilde{z}}^{\text{sep}}(f, C)$ and $\hat{g} := \hat{g}_{\tilde{z}}^{\text{sep}}(f, C)$. For every $z \in [-R, R]^{n+d}$ we have

$$\langle \hat{g}, z \rangle = \langle g, z \rangle + \langle \hat{g} - g, z \rangle \leq \langle g, z \rangle + \|\hat{g} - g\|_\infty \cdot \|z\|_1 \leq \langle g, z \rangle + \varepsilon'R(n + d) = \langle g, z \rangle + \frac{\varepsilon}{2}. \hspace{1cm} (2.11)$$

Now we claim that for every point $z \in C_{-\varepsilon}$ we have $\langle g, z \rangle \leq \langle g, \tilde{z} \rangle - \varepsilon$: since the inequality $\langle g, x \rangle \leq \langle g, \tilde{z} \rangle$ is valid for the ball $B(z, \varepsilon) \subseteq C$, we have

$$\langle g, z \rangle \geq \max_{w \in B(0,\varepsilon)} \langle g, z + w \rangle = \langle g, z \rangle + \varepsilon, \hspace{1cm} (2.12)$$

proving the claim. Finally, we claim that $\langle g, \tilde{z} \rangle \leq \langle \hat{g}, \tilde{z} \rangle + \frac{\varepsilon}{2}$:

$$\langle g, \tilde{z} \rangle - \langle \hat{g}, \tilde{z} \rangle = \langle g - \hat{g}, \tilde{z} \rangle \leq \|g - \hat{g}\|_\infty \cdot \|\tilde{z}\|_1 \leq \varepsilon'R(n + d) = \frac{\varepsilon}{2}. \hspace{1cm} (2.13)$$

Combining inequalities (2.11)-(2.13) proves that the cut $\langle \hat{g}, z \rangle \leq \langle \hat{g}, \tilde{z} \rangle$ is valid for $C_{-\varepsilon}$.

To obtain the $\varepsilon$-approximate value comparison oracle, since $f$ takes values in
[−U, U], it suffices to probe the sign plus \( \log \frac{2U}{\varepsilon} \) bits of \( f(\bar{z}) \) and \( f(\bar{z}') \) to approximate each of the values within ±\( \frac{\varepsilon}{2} \), in which case we can decide whether \( f(\bar{z}) \leq f(\bar{z}') + \varepsilon \) or not. This concludes the proof when using bit queries.

**Obtaining approximate oracles using inner product sign queries.** This proof largely follows the same steps as for the first part, except for the need for the following result, which may be of independent interest.

**Lemma 43.** For any \( \varepsilon \in (0, 1) \) and any vector \( g \in \mathbb{R}^d \), using \( O(d \log \frac{d}{\varepsilon}) \) inner product sign queries one can obtain a unit-length vector \( \hat{g} \in \mathbb{R}^d \) such that \( \|\hat{g} - \frac{g}{\|g\|}\| \leq \varepsilon \).

**Proof.** We prove by induction on the dimension \( d \) that for every \( \delta \in (0, 2) \), with \( d \log \frac{8}{\delta} \) inner product sign queries we can obtain a vector \( \hat{g} \) such that \( \|\hat{g} - \frac{g}{\|g\|}\| \leq 2d\delta \); the lemma then follows by setting \( \delta = \frac{\varepsilon}{2d} \).

Just one query suffices when \( d = 1 \), so consider the base case \( d = 2 \). Perform a binary search as follows: Start with the cone \( K_0 = \mathbb{R}^2 \), with corresponding angle 2\( \pi \).

In iteration \( t \), we maintain a cone \( K_t \) containing \( g \) whose angle is half that of \( K_{t-1} \) as follows. For each iteration, find a line \( \{ x : \langle a, x \rangle = 0 \} \) that cuts \( K_t \) into two cones \( K^L_t = K_t \cap \{ x : \langle a, x \rangle \leq 0 \} \) and \( K^R_t = K_t \cap \{ x : \langle a, x \rangle \geq 0 \} \) each with half the angle of \( K_t \), i.e. bisecting \( K_t \). Ask the query “Is \( \langle a, g \rangle \geq 0? \)”, and if so set \( K_{t+1} = K^R_t \), otherwise set to \( K_{t+1} = K^L_t \), and repeat the procedure. By construction all the cones \( K_t \) contain \( g \), and after \( \log \frac{8}{\delta} \) iterations we obtain a cone \( K \) with angle \( \frac{\delta \pi}{4} \). Let \( \hat{g} \) be any vector in this cone with unit \( \ell_2 \)-norm. For any other \( x \in K \) also of unit norm, we have

\[
\|\hat{g} - x\|_2^2 = 2 - 2 \langle \hat{g}, x \rangle \leq 2 - 2 \cos(\delta \pi/4) \leq (\delta \pi/4)^2 \leq \delta^2,
\]

where the second inequality uses that fact that \( \cos(\theta) \geq 1 - \frac{\theta^2}{2} \) for all \( \theta \in (0, \pi/2) \). So \( \|\hat{g} - x\|_2 \leq \delta \) for all unit-norm vectors in \( K \), and in particular \( \hat{g} \) gives the desired approximation of \( \frac{g}{\|g\|} \), proving the desired result when \( d = 2 \).

Now consider the general case \( d > 2 \). Consider any 2-dimensional subspace \( A \) of \( \mathbb{R}^d \), and let \( \Pi_A \) denote the projection onto this subspace. Using the 2-dimensional case
on the subspace $A$, we see that by using $\log \frac{8}{\delta}$ queries of the form “Is $\langle a, \Pi_A g \rangle \geq 0$?”, we can obtain a unit length vector $\tilde{g} \in A$ such that $\|\lambda_A \cdot \tilde{g} - \Pi_A g\| \leq \delta \|\Pi_A g\|$, where $\lambda_A := \|\Pi_A g\|$. We note that since $\langle a, \Pi_A g \rangle = \langle \Pi_A^* a, g \rangle$, the required queries can be obtained by inner product sign queries (here $\Pi_A^*$ denotes the adjoint linear operator for the projection operator $\Pi_A$, whose matrix representation is given by the transpose of the matrix representing the projection $\Pi_A$).

Now consider the $(d - 1)$-dimensional subspace $B := \text{span}\{\tilde{g}, A^\perp\}$, and notice that $\sim (g, B) \leq \delta \|g\|$: the vector $b := \lambda_A \cdot \tilde{g} + (g - \Pi_A g)$ belongs to $B$ and $\|g - b\| = \|\lambda_A \cdot \tilde{g} - \Pi_A g\| \leq \delta \|\Pi_A g\| \leq \delta \|g\|$. Since $g$ is close to this subspace, we project it there and recurse on dimension. More precisely, consider the projection $\Pi_B g$ of $g$ onto $B$, and inductively obtain a vector $\hat{g} \in B$ such that $\|\lambda_B \cdot \hat{g} - \Pi_B g\|_2 \leq 2(d - 1)\delta \cdot \|\Pi_B g\|$ (letting $\lambda_B := \|\Pi_B g\|$), by using additional $(d - 1) \log \frac{8}{\delta}$ queries (for a total of $d \log \frac{8}{\delta}$ queries).

We claim that $\hat{g}$ is the desired approximation of $g$, namely $\|g - \|g\| \cdot \hat{g}\|_2 \leq 2d\delta$. To see this, from triangle inequality we have

$$\|g - \|g\| \cdot \hat{g}\| \leq \|g - \Pi_B g\| + \|\Pi_B g - \lambda_B \cdot \hat{g}\| + \|\lambda_B \cdot \hat{g} - \|g\| \cdot \hat{g}\|. \quad (2.14)$$

The first term of the right-hand side equals $\sim (g, B)$, which is at most $\delta \|g\|$ as argued above. For the second term, by induction we have

$$\|\Pi_B g - \lambda_B \cdot \hat{g}\|_2 \leq 2(d - 1)\delta \cdot \|\Pi_B g\| \leq 2(d - 1)\delta \|g\|.$$ 

Finally, we claim that the last term of (2.14) is at most $\delta \|g\|$: since $\hat{g}$ has unit norm, it equals $|\lambda_B - \|g\|| = \|\Pi_B g\| - \|g\| = \|g\| - \|\Pi_B g\|$, and by triangle inequality we have

$$\|g\| \leq \|\Pi_B g\| + \|g - \Pi_B g\| \leq \|\Pi_B g\| + \delta \|g\|,$$

giving the claim.

Applying all these bounds to (2.14), we get that $\|g - \|g\| \cdot \hat{g}\| \leq 2d\delta \|g\|$, as desired. This concludes the proof of the lemma.
Now we are ready to finish the proof. To obtain an $\varepsilon$-approximate value comparison oracle using $H_{\text{dir}}$, we can do a binary search on the function values using the queries $h_{u,c}^{\text{val}}$ with $u = 1$ and different values of $c$ (as the midpoint of the interval in the binary search) – see Definition 6. Thus, with $O(\log \frac{U}{\varepsilon})$ queries, we can implement an $\varepsilon$-approximate value comparison oracle.

For the $\varepsilon$-approximate separation, we apply Lemma 43 above to the separation oracle $g_{\varepsilon}^{\text{sep}}(f, C)$, with $O((n + d) \log \frac{(n + d)R}{\varepsilon})$ inner product sign queries to obtain a vector $\hat{g}_{\varepsilon}^{\text{sep}}(f, C)$ such that $\|g_{\varepsilon}^{\text{sep}}(f, C) - \hat{g}_{\varepsilon}^{\text{sep}}(f, C)\| \leq \frac{\varepsilon}{2R\sqrt{n + d}}$ (recall the non-zero separation oracles are assumed to have unit length). Using the same arguments as in inequalities (2.11)-(2.13), we see that $\hat{g}_{\varepsilon}^{\text{sep}}(f, C)$ gives a cut valid for $C - \varepsilon$, and hence is an $\varepsilon$-approximate separation oracle.

For the $\varepsilon$-approximate value cut oracle, we do the same thing, but apply Lemma 43 to $g_{\varepsilon}^{\text{sub}}(f, C)$, with $O((n + d) \log \frac{(n + d)MR}{\varepsilon})$ oracle calls to obtain an approximation $\|g_{\varepsilon}^{\text{sub}}(f, C) - \hat{g}_{\varepsilon}^{\text{sub}}(f, C)\| \leq \frac{\varepsilon}{2M\sqrt{n + d}}$ and then use the argument from (2.10).

**Remark 44.** Notice that the main ingredient for implementing approximate oracles using inner product sign queries is to use such queries for approximating a given vector $g$ (Lemma 43). We remark that this task is related to (actively) learning a linear classifier, namely that whose normal is given by $g$. While there are existing procedures for doing this, they only guarantee the desired approximation with high probability (albeit on a slightly weaker query model), instead of with probability 1 as we want here; see for example [4]. While it is possible that these methods can be adapted to our setting, we present a different, self-contained statement and proof.

### 2.5 Proofs of Theorem 25 and Corollary 26

**Proof of Theorem 25.** Suppose we have an algorithm $A$ that reports an $\varepsilon$-solution to any instance in $I_{n,d,R,\rho,M}$ after $u$ queries to a full-information first-order oracle based on the first-order chart $G$, a finite set of instances $I \subseteq I_{n,d,R,\rho,M}$, and a true (unknown) instance $I \in I$. Our goal is to report a feasible $\varepsilon$-solution using few queries to $O(G, H)$, where $H$ contains all binary queries. For this, we design a procedure that maintains a
family $\mathcal{U} \subseteq \mathcal{I}$ of the instances, which always includes the true instance $I$, and possibly determines exact information to pass to $\mathcal{A}$. We will show that we can always either reduce $|\mathcal{U}|$ by a constant factor, or determine exact information to use with $\mathcal{A}$.

Denote by $D$ the query strategy of $\mathcal{A}$. Initialize $\mathcal{U} = \mathcal{I}$, and (ordered) lists $Q = \emptyset, H = \emptyset$, which will serve as query-response pairs for the algorithm $\mathcal{A}$. In particular, $H$ will contain full first-order information about the true instance, and $Q$ will be the sequence of queries $\mathcal{A}$ makes. While $|\mathcal{U}| > 1$ and $|Q| \leq u$, do the following:

- **Case 1:** For every $v \in V_z$, at most half of the instances $I' \in \mathcal{U}$ give $g_z(I') = v$. Then there exists a set $A \subseteq V_z$ such that the number of instances $I' \in \mathcal{U}$ with $g_z(I') \in A$ is between $\frac{1}{4}|\mathcal{U}|$ and $\frac{3}{4}|\mathcal{U}|$. Let $\mathcal{U}_0 := \{I \in \mathcal{U} : g_z(I') \notin A\}$, $\mathcal{U}_1 := \{I \in \mathcal{U} : g_z(I') \in A\}$; thus, $|\mathcal{U}_i| \leq \frac{i}{4}|\mathcal{U}|$ for $i = 0, 1$. Query whether the true instance $I$ has $g_z(I) \in A$, using the binary query $h_A : h_A(v) = 1$ iff $v \in A$, so that $h_A(g_z(I)) = 0$ if $I \in \mathcal{U}_0$ and $h_A(g_z(I)) = 1$ if $I \in \mathcal{U}_1$. Update $\mathcal{U} \leftarrow \mathcal{U}_q$, where $q$ is the answer to the query $(z, h_A)$ given by the oracle.

- **Case 2:** There exists $\bar{v} \in V_z$ such that more than half of the instances $I' \in \mathcal{U}$ have $g_z(I') = \bar{v}$. Query whether the true instance $I$ has $g_z(I) = \bar{v}$, using the binary query $h : h(\bar{v}) = 1$ and $h(x) = 0$ for all other inputs $x \neq \bar{v}$, so that $h(g_z(I)) = 1$ iff $g_z(I) = \bar{v}$. If $g_z(I) \neq \bar{v}$, then update $\mathcal{U}$ by removing from it all instances $I'$ such that $g_z(I') = \bar{v}$, reducing the size of $\mathcal{U}$ by at least half. Otherwise, if $z$ was infeasible, we then know the exact separating hyperplane (or function value or subgradient, in the case $z$ is feasible) for the true instance $I$ and the first-order chart $\mathcal{G}$, namely $g_z^{\text{sep}}(I) = \bar{v}$, and so employ it to update $Q$ and $H$ by appending $z$ and $v$ to them, respectively, which will serve as information for the algorithm $\mathcal{A}$. 69
In each step, either the size of $\mathcal{U}$ decreases by at least $1/4$, or full (exact) first-order information at the query point determined by the query strategy of $\mathcal{A}$ is obtained and $Q, H$ are updated. The former can only happen $O(\log |I|)$ times until $\mathcal{U}$ becomes a singleton, in which case we know the true instance and can report its optimal solution, while if the latter happens $u$ times, one can run the algorithm $\mathcal{A}$ with the information $(Q, H)$ to report an $\varepsilon$-approximate solution to the true instance $I$, noting that since the points in $Q$ were determined according to the query strategy of the algorithm, the information in $(Q, H)$ is indeed sufficient to run the $\mathcal{A}$ on for $u$ iterations. Hence, after at most $\log |I| + u$ queries to the general binary oracle, one can report an $\varepsilon$-solution to the true instance.

Corollary 26 follows immediately when one uses the centerpoint-based algorithm of [90, 9] as $\mathcal{A}$, which is the exact oracle version of Algorithm 2 above and needs at most

$$O\left(2^n d(n + d) \log \left(\frac{dMR}{\min\{\rho, 1\}\varepsilon}\right)\right)$$

queries in the mixed-integer case, or

$$O\left(d \log \left(\frac{MR}{\min\{\rho, 1\}\varepsilon}\right)\right)$$

queries in the continuous ($n = 0$) case to produce an $\varepsilon$-approximate solution to any instance in $\mathcal{I}_{n,d,p,M,R}$.

### 2.6 Proof of Theorem 21: A Quadratic Lower-Bound for Constrained Convex Optimization under First-Order Bit-oracles

Next, we prove a lower-bound for the Bit Oracle using first-order information that is quadratic in the continuous dimension of the problem. This lower-bound will tell us that when one has access to a bit-representation of the function values, subgradients, and separating hyperplanes for the instance, one needs to query at least $O(d^2)$ of those bits to solve the problem. In particular, this is much stronger than the result proven above
in Theorem 20 for the specific case of the bit-oracle, and is a significant step towards proving Conjecture 2. This case is also of particular practical interest, since in a typical modern computational model the function values, gradients, and hyperplanes will be available only in finite-bit representations.

Recall that Theorem 21 states the following. Suppose that $\rho \leq \frac{R}{d}$. Then there exists a first-order chart $G$ such that for the Bit Oracle based on $G$ we have

$$\text{icomp}_\varepsilon(I_{n=0,d,R,\rho,M}, O(G, H^{bit})) = \Omega(d^2),$$

that is, $\Omega(d^2)$ bits worth of first-order information need to be queried to solve the problem. Practically speaking, Theorem 21 reveals that $\Omega(d^2)$ bits of first-order information are required to solve constrained convex optimization problems if one uses standard bit-encodings of the information. Recall also that Theorem 24 reveals that it is possible to solve the problem within $O(d^2 \log(d))$ queries, and so one has almost matching upper- and lower bounds with respect to the number of bit queries needed to solve constrained convex optimization problems in the regime $\rho \leq \frac{R}{d}$.

The proof will use an argument recursing on dimension. We will make use of the following definition for this purpose:

**Definition 45.** Let $H^= (g, x) := \{ y \in \mathbb{R}^d : \langle y, g \rangle = \langle x, g \rangle \}$ be a hyperplane with normal vector $g$ containing $x$. An $\varepsilon$-pancake $P^\varepsilon (g, x)$ is the set $H^= (g, x) + B(0, \varepsilon)$, where $+$ denotes the Minkowski sum.

In general, for any affine space

$$H^= (g_1, ..., g_k, x) := \{ y \in \mathbb{R}^d : \langle y, g_k \rangle = \langle x, g_k \rangle, ..., \langle y, g_k \rangle = \langle x, g \rangle \},$$

we call

$$P^\varepsilon (g_1, ..., g_k, x) := H^= (g_1, ..., g_k, x) + B(0, \varepsilon)$$

a $d - k$ dimensional pancake (assuming that $g_1, ..., g_k$ are linearly independent).
In other words, \( P^\varepsilon(g, \delta) \) is a, \( \varepsilon \) “fattening” of the hyperplane \( H^\varepsilon(g, x) \). We illustrate the namesake pancake in Figure 2.3, where a 3D plot of a pancake \( P^\varepsilon(g, x) \), restricted to a sphere, is placed in juxtaposition to American breakfast pancakes.

![3D plot of a pancake and American breakfast pancakes](image)

Figure 2.3: A render of \( P^\varepsilon(g, x) \) in \( \mathbb{R}^3 \) restricted to a sphere (upper image) compared to a stack of colloquial American breakfast pancakes\(^3\)(lower image).

As usual, we will restrict to the box \([-R, R]\). We make the following claim to lower-bound how many queries to a Bit Oracle are needed to report a point in an unknown pancake \( P^\varepsilon(g, x) \) in this box. More precisely, letting \( \mathcal{P}^\varepsilon \) be the family of \( \varepsilon \)-pancakes in the box \([-R, R]\), we want to lower-bound the number of queries needed such that there are no two disjoint \( P_1, P_2 \in \mathcal{P}^\varepsilon \) giving all the same answers to the queries made. The following claim gives a lower-bound on the number of queries needed to find a \( k - 1 \) dimensional pancake within a \( k \) dimensional pancake. This will be used to prove Theorem 21 using an argument recursing on dimension.

\(^3\)Render of colloquial American pancakes obtained from Pixabay copyright-free images, https://pixabay.com/illustrations/pancakes-breakfast-meal-food-7437690/
Claim 46. Let

\[ P^\varepsilon(e_1^*, ..., e_k^*, x) := H^\varepsilon(e_1^*, ..., e_k^*, x) + B(0, \varepsilon) \]

be a \( d - k \) dimensional pancake in \([-R, R]^d\) with \( \rho < \frac{R}{d} \), with \( e_1^*, ..., e_k^* \) being linearly independent standard basis vectors. If fewer than \( d - k - 1 \) queries to a Bit Oracle using first-order information are made by an algorithm, the queries can be answered adversarially in an online fashion such there exist two disjoint \( d - k - 1 \) dimensional \( \rho \)-pancakes within \( P^\varepsilon(e_1^*, ..., e_k^*, x) \) that are consistent with the answers given.

Where \( e_i^* \) simply signifies any standard basis vector (but not necessarily the standard basis vector \( e_i \)). This claim tells us that identifying a \( \rho \)-pancake in \([-R, R]^d\) requires at least \( d - 1 \) queries to a first-order Bit Oracle. To prove Theorem 21, we will then argue that one needs \( d - 1 \) bits to identify a \( d \) dimensional pancake, \( d - 2 \) queries to identify a \( d - 1 \) dimensional pancake within that \( d \) dimensional pancake, and so on. Noting that a ball of size \( \rho \) a 0-dimensional pancake, one requires at least \( \sum_{i=1}^{d-1} i = O(d^2) \) queries to find a \( \rho \)-ball in the \([-R, R]^d\) box.

To illustrate the main idea behind the proof of Claim 46, consider finding a \( d - 1 \) dimensional pancake in the box \([-R, R]^d\). The algorithm makes \( d - 1 \) bit queries at \( d - 1 \) possibly different points. The oracle answers that they are all infeasible and says that the queried bit for the normal defining a separating hyperplane is 0. At most \( d - 1 \) different coordinates were queried, so there remains a coordinate \( j \) in which no bit was queried. Hence, the separating hyperplanes with normal vector \( \pm e_j \) are consistent with all the answers given. If \( \rho \leq \frac{R}{d} \), one can pack \( 2d - 1 \) disjoint pancakes with normal vector \( e_j \) into the \([-R, R]^d\) box, so there exist two disjoint such pancakes that don’t contain any of the queried points \( x_1, ..., x_k \). Hence, they are both consistent with all the answers the algorithm has received; they cannot be distinguished. We now provide a formal proof of Claim 46 before stating the formal proof of Theorem 21.

Proof of Claim 46. Suppose that \( \rho < \frac{R}{d} \) and only \( d - k - 1 \) queries to a Bit Oracle using first order information have been made at points \( x_1, ..., x_{d-k-1} \). Without loss of generality, suppose that we have \( P^\varepsilon(e_1, ..., e_k, x) \), so that \( e_i^* = e_i \). The adversary answers the queries as follows. For any \( x_i \) outside of \( P^\varepsilon(e_1, ..., e_k, x) \), the oracle simply returns an
answer consistent with all of $P^x(e_1, ..., e_k, x)$, ie. it answers $\text{INFEASIBLE}$ and $q(s)$, where $s$ is the normal for any separating hyperplane for $P^x(e_1, ..., e_k, x)$. Otherwise, if $x_i \in P^x(e_1, ..., e_k, x)$, answer $\text{INFEASIBLE}$ and return 0 no matter what bit for the separating hyperplane the algorithm queried. Since $d-k-1$ bit queries were made, bits in at most $d - k - 1$ different coordinates of the separating hyperplanes were queried. Hence, there exists some coordinate $j \in d-k, ..., d$ such that no bit in that coordinate of the normal of a separating hyperplane was queried, at any of the $x_{1}, ..., x_{d-k-1}$. Consider now $P^\rho(e_1, ..., e_k, e_j, y)$ for different possible values of $y$. We claim that there exist $y_1, y_2 \in H^\rho(e_1, ..., e_k, x)$ such that $P^\rho(e_1, ..., e_k, e_j, y_1)$ and $P^\rho(e_1, ..., e_k, e_j, y_2)$ are disjoint and neither of them contain any of the queried points $x_1, ..., x_k$. With $\rho \leq \frac{R}{d}$, one can pack $2d-1$ disjoint translates of $P^\rho(e_1, ..., e_k, e_j, y)$ into $P^\rho(e_1, ..., e_k, x)$. This is because $P^\rho(e_1, ..., e_k, x) \cap [-R, R]^d$ has width $2R$ in direction $e_j$, noting that $e_j \perp e_1, ..., e_k$.

Since there are $2d-1$ disjoint translates of $P^\rho(e_1, ..., e_k, e_j, y)$ in $P^\rho(e_1, ..., e_k, y)$, but only $d-k-1$ queries were made, there exist the two desired points $y_1, y_2$. For each queried point $x_i$, either $e_j$ or $-e_j$ is normal vector for a valid separating hyperplane to $P^\rho(e_1, ..., e_k, e_j, y_1)$ or $P^\rho(e_1, ..., e_k, e_j, y_2)$. Hence, the responses the oracle gave for the queries at $x_1, ..., x_k$, namely $\text{INFEASIBLE}$ and bits 0 for all coordinates other than $e_j$, are consistent with both pancakes $P^\rho(e_1, ..., e_k, e_j, y_1)$ or $P^\rho(e_1, ..., e_k, e_j, y_2)$. So, after $d-k-1$ queries, when the adversary uses the above strategy for providing its answers, there indeed exist two disjoint $\rho$-pancakes in $P^x(e^*1, ..., e^*_k, x)$ that are consistent with the answers given, as desired.

With this result at hand, we are ready to prove Theorem 21.

**Proof of Theorem 21.** Let $\rho \leq \frac{R}{d}$ as assumed for the theorem. We will again use an adversarial oracle, which will consider the set of all $\rho$-balls as the feasible regions for the instances, and maintains a set $U$ as a set that is still consistent with all answers the adversary has provided. We will use Claim 46 recursively on dimension. For the first $d-1$ queries that were made, Claim 46 tells us that after $d-1$ queries, the adversary can update $U \leftarrow P_{d-1}$, where $P_{d-1}$ is one of the $d-1$ dimensional pancakes guaranteed
by the claim that is consistent with all the answers given.

⋆ Now suppose that $U$ updated to be some $d - k$ dimensional $\rho$-pancake. By Claim 46, the adversary can answer the next $d - k - 1$ queries of the algorithm such that two disjoint $d - k - 1$ dimensional $\rho$-pancakes, $P_1, P_2 \subset U$ are consistent with all the answers given (by using the strategy used in the proof of the same claim). The adversary updates $U \leftarrow P_1$. Recursing from ⋆ until $d = 2$, after the algorithm makes

$$\sum_{k=1}^{d - 2d - k = O(d^2)}$$

the adversary updates $U$ to be a 1-dimensional $\rho$-pancake $P$, or $P \cap [-R, R]^d$ to be precise, which contains disjoint $\rho$-balls as long as $\rho < 2R$. Since $U$ is consistent with all answers provided by the oracle, the theorem follows. $\square$
2.7 Chapter Appendix

2.8 Information Games

In this section we define the game-theoretic perspective for information complexity and prove Lemma 28, which we restate for convenience.

**Lemma 28.** Consider a class of instances $\mathcal{I}$ and an oracle $\mathcal{O}$. Then $\text{icomp}_\varepsilon(\mathcal{I}, \mathcal{O}) > \ell$ if and only if there exists an adversary under $\mathcal{O}$ using $\mathcal{I}$ that is $\varepsilon$-hard for $\ell$ rounds.

Let $\mathcal{I}$ be a family of optimization instances in $\mathbb{R}^n \times \mathbb{R}^d$. Let $\mathcal{O}$ be an oracle for $\mathcal{I}$. Let $\varepsilon > 0$. We define the information game for $(\mathcal{I}, \mathcal{O}, \varepsilon)$ as a two player game between Alg and Adv. The players make moves alternatingly with Alg moving first. For Alg, a move is a choice of query $q$ from the oracle $\mathcal{O}$. For Adv, a move is a choice of $r \in H$, where $H$ is the response set of $q$ from the immediately preceding move by Alg. The only constraint on Adv is that there should exist at least one instance in $\mathcal{I}$ that will give the same responses as the moves made by Adv in the game so far to queries corresponding to the moves made by Alg so far. More precisely, at any stage of the game, there must exist an instance $I$ such that for every response $r$ given by Adv at any round of the game played so far to an immediately preceding query $q$ made by Alg, we have $r = q(I)$.

Notice that a game strategy for Alg is equivalent to a query strategy from Definition 2. Also, any instance $\bar{I} \in \mathcal{I}$ gives a game strategy for Adv, by simply reporting the response $q(\bar{I})$ for all queries $q$ made by Alg.

In the game tree defined by the above game, a node is said to be $\varepsilon$-unambiguous if the instances that are consistent with Adv’s moves all have a common $\varepsilon$-approximate solution; otherwise, the node is $\varepsilon$-ambiguous. The game stops when the game reaches an $\varepsilon$-unambiguous node; thus, all leaf nodes of the game tree are $\varepsilon$-unambiguous. When the game stops at an $\varepsilon$-unambiguous node $L$, the loss (or payoff) $\mathcal{L}_\varepsilon(L)$ is defined as the number of moves made by Alg to arrive at $L$ in the game tree, i.e., it is exactly half of the depth of $L$ in the game tree.

A subtree $T$ of the game tree is said to be a full subgame tree if it is precisely
the set of all descendants of a node in the game tree (including itself). A subtree $T$ is said to have *finite horizon* if it has bounded depth, i.e., there exists $D > 0$ such that all nodes of $T$ have depth (in $T$) bounded by $D$. Otherwise, $T$ is said to have *infinite horizon* (note this includes the case where $T$ has an infinite length path and the case where all paths are finite length, but there is no upper bound on the lengths).

We define the value $v(T)$ of a finite horizon subtree $T$ by induction on the depth of $T$, which we call the *value of the subgame defined by $T$*.

- If $T$ consists of a single node $N$, $v(T) := L_\varepsilon(N)$ if $N$ is an $\varepsilon$-unambiguous node, else $v(T) := \infty$.
- If $T$ is rooted at a node $N$ corresponding to Alg, then $v(T) := \inf\{v(T') : T' \text{ child subtree in } T \text{ at } N\}$.
- If $T$ is rooted at a node $N$ corresponding to Adv, then $v(T) := \sup\{v(T') : T' \text{ child subtree in } T \text{ at } N\}$.

Let $T$ and $T'$ be subtrees of the game tree. $T'$ is said to be a *(finite horizon)* truncation of $T$ if there exists $D > 0$ such that $T'$ consists of all nodes of $T$ with depth bounded by $D$. Observe that if $T, T'$ are finite horizon subtrees and $T'$ is a truncation of $T$, then $v(T') \geq v(T)$. We can thus naturally extend the notion of value to an infinite horizon subtree $T$, as a limit of its finite horizon truncations:

$$v(T) := \inf\{v(T') : T' \text{ finite horizon truncation of } T\}.$$

A *game strategy* for Alg (resp. Adv) is a choice of a specific move at every non leaf node corresponding to Alg (resp. Adv). Thus, any game strategy $Q$ for Alg (resp. game strategy $A$ for Adv) corresponds to a subtree of the entire game tree, where we select a single outgoing edge at every node corresponding to Alg (resp. Adv). Such a subtree is called a *decision tree* for Alg (resp. Adv). With a slight abuse of notation, we will use $v(Q)$ (resp. $v(A)$) to denote the values of these decision trees. A simultaneous choice of strategies $Q, A$ yields a single path in the original game tree; $v(Q, A)$ will denote the
value of this path (subtree).

We note that we can express the notion of an $\varepsilon$-hard adversary for $\ell$ using this notation; it follows directly from the definition of the value function $v(\cdot)$.

**Observation 47.** An adversary $A$ is $\varepsilon$-hard for $\ell$-rounds iff $\inf_Q v(Q, A) > \ell$, where the infimum is taken over all strategies for $\operatorname{Alg}$.

We now focus on proving Lemma 28, which requires a few preparatory observations. The first is the following, which is a useful extension of the recursive nature of the value of a subtree to infinite horizon trees.

**Lemma 48.** Let $T$ be any subtree of the game tree (not necessarily of finite horizon). Then,

- If $T$ consists of a single node $N$, $v(T) = L_\varepsilon(N)$ if $N$ is an $\varepsilon$-unambiguous node, else $v(N) = \infty$.

- If $T$ is rooted at a node $N$ corresponding to $\operatorname{Alg}$, then $v(T) = \inf\{v(T') : T'$ child subtree in $T$ at $N}\}$.

- If $T$ is rooted at a node $N$ corresponding to $\operatorname{Adv}$, then $v(T) = \sup\{v(T') : T'$ child subtree in $T$ at $N}\}$.

**Proof.** The result follows from the fact that for any finite horizon truncation $T'$ of $T$, the subtrees of $T'$ rooted at the children of the root of $T'$ (and $T$) are finite horizon truncations of the subtrees of $T$ rooted at the children of the root. \hfill \Box

The following technical property will be used several times in the sequel.

**Lemma 49.** The value of the full game is finite if and only if every full subgame has finite value.

**Proof.** Since the full game is a subgame of itself, if every subgame has finite value so does the full game. For the other direction, suppose now that the full game tree has finite value. Then, there is a finite horizon truncation $T$ with finite value. The next
Claim shows that for every full subgame tree, its value can be upper bounded using the value of $T$, so in particular is finite as desired.

**Claim 50.** Let $S$ be a full subgame tree, and let $N$ denote its root node. Then $v(S) \leq v(T) + \text{depth}(N) + 1$.

**Proof.** Let $D$ be the depth of $T$. Consider the truncation $S'$ of $S$ which has depth $D$ if $N$ is a node corresponding to Alg, and depth $D + 1$ if $N$ is a node corresponding to Adv. Since $S'$ is a truncation of $S$, it follows that $v(S) \leq v(S')$, and so we only need to show $v(S') \leq v(T) + \text{depth}(N) + 1$.

To prove this bound, consider first the case when $N$ is a node corresponding to Alg. In this case, $T$ is isomorphic to a subtree of $S'$, i.e., each node of $T$ corresponds to a node in $S'$ in the natural way: the root of $T$ corresponds to the root of $S'$, the children of the root of $T$ corresponds to (some of) the children of the root of $S'$, etc. Observe that the nodes corresponding to Alg have the same possibilities for moves in both trees $S'$ and $T$ (in fact, every node of Alg in the full game tree has the same choice of moves by definition). However, nodes corresponding to Adv have a smaller set of choices in $S'$, i.e., children, compared to $T$, because they are deeper in the game tree. Thus, in the inductive definition of $v(S')$, a supremum computed at a node in $S'$ for Adv is over a smaller set of choices compared to the corresponding supremum when computing $v(T)$. Finally, the depth (in the full game tree) of any node in $S$ is precisely $\text{depth}(N)$ more than the depth of the corresponding node in $T$. Thus, $v(S') \leq v(T) + \text{depth}(N) \leq v(T) + \text{depth}(N) + 1$.

Next, consider the case when $N$ is a node corresponding to Adv. All the subtrees of $S'$ rooted at the children of $N$ are again in correspondence with $T$, since $S$ is of depth $D + 1$. The previous argument applies to these subtrees and thus, their values are all bounded by $v(T) + \text{depth}(N) + 1$. Taking a supremum at node $N$ shows that $v(S') \leq v(T) + \text{depth}(N) + 1$. 

\qed
We first prove the existence of a saddle-point in this game (in fact, the same argument gives that there is a subgame perfect equilibrium, which essentially means that this saddle point property holds for every subtree).

**Lemma 51.** Suppose the value of the full game is finite (in other words, the optimization problem is solvable with finitely many queries using the given oracles). Then, there exists a saddle-point \( Q^*, A^* \) for Alg and Adv respectively, i.e., strategies \( Q^* \) and \( A^* \) satisfying

\[
\inf_Q v(Q, A^*) = v(Q^*, A^*) = \sup_A v(Q^*, A) = v(\text{full game}),
\]

where full game denotes the full game tree, the infimum on the left-hand side is over all possible strategies for Alg, and the supremum on the right-hand side is over all strategies for Adv.

**Proof.** Since we assume the full game has finite value, by Lemma 49 we know that the value \( v(S) \) of any full subgame tree \( S \) is also finite. Combined with the fact that \( v(S) \) is integer valued, the supremum or infimum in Lemma 48 is attained. A saddle-point \( Q^* \) and \( A^* \) can now be defined as follows. At any node \( N \) corresponding to a move by Alg, select the move that attains the infimum giving the value of the full subgame tree rooted at \( N \) by Lemma 48. At any node \( N \) corresponding to a move by Adv, select the move that attains the supremum giving the value of the full subgame tree rooted at \( N \) by Lemma 48.

We now prove that \( Q^*, A^* \) satisfy the desired properties. When the full tree is finite, this follows directly from the definition of \( Q^*, A^* \); the case where it is not finite is that requires more subtle arguments based on finite horizon truncations. Let \( T \) denote the full game tree, and for a node \( u \) let \( T_u \) denote its full subtree rooted at \( u \).

We start by proving that \( v(Q^*, A^*) = v(T) \). Let \( p_1, p_2, \ldots \) be the nodes in the path \( P^* \) induced by \( Q^*, A^* \) (\( p_1 \) being the root of the full game tree). By the second item of Lemma 48, we have that \( v(T_{p_1}) \) is the smallest value of the subtrees rooted at the children of \( p_1 \); since the strategy \( Q^* \) chooses precisely such child \( p_2 \) of smallest value, we get \( v(T_{p_1}) = v(T_{p_2}) \). Similarly, using the same lemma and the definition of
the adversarial strategy $A^*$, we get $v(T_{p_2}) = v(T_{p_3})$. Thus, subtrees $T_{p_i}$'s have the same value, equal to $v(T_{p_1}) = v(T)$ (which is finite). Moreover, the value of a subtree $T_u$ is at least half of the depth of $u$: $v(T_u)$ is either $\infty$ or is the (finite) payoff of a node $w$ in $T_u$, which is at least half the depth of $w$, and so at least half the depth of $u$. Therefore, all nodes in the path $P^*$ are at depth at most $2 \cdot v(T)$; in particular, $P^*$ has finite length. Now consider a finite horizon truncation $T'$ of $T$ with value $v(T') = v(T)$; we can further increase the depth of this truncation and assume that the path $P^*$ is contained in $T'$. Since $T'$ is finite, its value $v(T')$ is defined by taking the child that has subtree of smallest/largest value on the nodes of Alg/Adv, until a leaf (of finite value) is reached. But this is by construction the choices that the path $P^*$ makes, namely they reach the same leaf (or leaves of the same value). The values of $v(T')$ and $v(P^*)$ are then the value of this leaf/leaves, and hence $v(T') = v(P^*)$, or equivalently $v(Q^*, A^*) = v(T)$ as desired.

We now consider the first desired equation, namely $\inf_Q v(Q, A^*) = v(Q^*, A^*)$, and notice that it suffices to show

$$\inf_Q v(Q, A^*) \geq v(Q^*, A^*)$$

(2.16)

(the equality follows by taking $Q = Q^*$). Assume that the paths induced by $Q, A^*$ and $Q^*, A^*$ are different, else there is nothing to show. Let $u$ be the first node of Alg in these paths where $Q$ and $Q^*$ make different decisions; let $w$ and $w^*$ be the children of $u$ on the paths induced by $Q, A^*$ and $Q^*, A^*$ respectively. Notice that by construction of the choice $Q^*$ makes at $u$, we have $v(T_{w^*}) \leq v(T_w)$.

Now we claim that $v(Q, A^*) \geq v(T_w)$. If the path $P$ induced by $Q, A^*$ has infinite length, then all of its nodes are $\varepsilon$-ambiguous and so by definition $v(Q, A^*) = \infty$ and the inequality holds. So suppose $P$ has finite length. Since the value $v(T_w)$ is finite, again due to our assumption that $v(T)$ is finite, there is a finite horizon truncation $T'$ of this subtree with the same value; again, we can further increase the depth of this truncation and assume that the path $P \cap T_w$ (the suffix of $P$ starting at $w$) is contained in $T'$ (and hence $P \cap T_w = P \cap T'$). By definition, the nodes of Adv in the path $P \cap T'$ always
select the action that lead to the child whose subtree has largest value (being based on $A^*$), while the nodes of Alg may not necessarily select the children with lowest value subtrees (being based on $Q$). Since in the definition of the value $v(T')$, both the nodes of Adv and Alg make the optimal decisions, we see that $v(T') \leq v(P \cap T')$. Since the value of the path $v(P)$ and of its suffix $v(P \cap T')$ are the same (they are the value of the leaf of the path), we obtain $v(T_w) = v(T') \leq v(P \cap T') = v(P) = v(Q, A^*)$, as claimed.

Next, we claim that $v(T_w^*) = v(Q^*, A^*)$. As proved above, the path $P^*$ induced by $Q^*, A^*$ has finite length. Since the value $v(T_w^*)$ is finite, as above, consider a truncation $T'$ of $T_w^*$ with same value $v(T') = v(T_w^*)$ and that contains the suffix $P^* \cap T_w^*$. By the same argument as in the previous paragraph, but now using that both $Q^*$ and $A^*$ make optimal decisions, we have $v(T') = v(P^* \cap T_w^*)$, which is also equal to $v(P^*)$: this implies $v(T_w^*) = v(P^*) = v(Q^*, A^*)$ as desired.

Putting together the bounds from the three previous paragraphs yields $v(Q^*, A^*) = v(T_w^*) \leq v(T_w) \leq v(Q, A^*)$ for every strategy $Q$ for the algorithm. Taking an infimum over all such $Q$’s finally proves (2.16), and hence that $\inf_Q v(Q, A^*) = v(Q^*, A^*)$.

The proof that $\sup_A v(Q^*, A) = v(Q^*, A^*)$ uses the exact same arguments, but exchanging the roles of the players Alg and Adv.

By standard arguments, the existence of a saddle-point implies a minimax result, which is the first element required to prove Lemma 28.

**Lemma 52.** Suppose the value of the full game is finite, and let $Q^*, A^*$ be a subgame perfect equilibrium guaranteed to exist by Lemma 51. Then the following minimax holds;

$$\sup_A \inf_Q v(Q, A) = v(Q^*, A^*) = \inf_Q \sup_A v(Q, A),$$

where the infima are over all possible strategies for Alg, and the suprema are over all strategies for Adv.
Proof. From the guarantees of \( Q^*, A^* \), for all strategies \( Q, A \) for Alg and Adv respectively, we have
\[
v(Q^*, A^*) \leq \inf_Q v(Q, A^*) \leq \sup_A \inf_Q v(Q, A)
\]
and
\[
\inf_Q \sup_A v(Q, A) \leq \sup_A v(Q^*, A) \leq v(Q^*, A^*).
\]
Since we always have \( \sup_A \inf_Q v(Q, A) \leq \inf_Q \sup_A v(Q, A) \), we get equalities throughout.

The last element required for proving Lemma 28 is the equivalence between the value of this game and the information complexity of the underlying optimization problem.

Lemma 53. The value of the full game \( v(\text{full game}) \) equals \( \text{icomp}_\varepsilon(I, O) \).

Proof. Combining Lemmas 51 and 52, \( v(\text{full game}) = \inf_Q \sup_A v(Q, A) \), so it suffices to show
\[
\inf_Q \sup_A v(Q, A) = \inf_Q \sup_{I \in \mathcal{I}} \text{icomp}_\varepsilon(Q, I, O).
\]
To see the “\( \geq \)” direction: Consider any strategy \( Q \) for Alg and instance \( I \in \mathcal{I} \). Consider the adversary \( A \) based on the instance \( I \), namely that reports \( q(I) \) whenever \( Q \) asks a query \( q \). Notice that \( v(Q, A) = \text{icomp}_\varepsilon(Q, I, O) \) (if \( \text{icomp}_\varepsilon(Q, I, O) \) is finite, after exactly \( \text{icomp}_\varepsilon(Q, I, O) \) many moves of the algorithm \( Q \) we reach an \( \varepsilon \)-unambiguous node in the path induced by \( Q, A \), so \( v(Q, A) = \text{icomp}_\varepsilon(Q, I, O) \); if \( \text{icomp}_\varepsilon(Q, I, O) \) is infinite, every finite horizon truncation of the path induced by \( Q, A \) must end on an \( \varepsilon \)-ambiguious node, so \( v(Q, A) \) is also infinite). That is, for every \( I \) we can find a strategy \( A \) for the adversary with the “same value”, which then implies \( \sup_A v(Q, A) \geq \sup_{I \in \mathcal{I}} \text{icomp}_\varepsilon(Q, I, O) \). Taking an infimum over all \( Q \)'s on both sides gives the desired inequality \( \inf_Q \sup_A v(Q, A) \geq \inf_Q \inf_{I \in \mathcal{I}} \text{icomp}_\varepsilon(Q, I, O) \).

The proof for the “\( \leq \)” direction is analogous: Consider any strategy \( Q \) for Alg and strategy \( A \) for Adv. If \( v(Q, A) = \infty \), then for every finite horizon truncation
of length $N$ (i.e. with $\frac{N}{2}$ queries by the algorithm) of the path induced by $Q, A$ ends on an $\varepsilon$-ambiguous node; let $I \in \mathcal{I}$ be an instance consistent with $A$ given up to this node. Then we see that $\text{icomp}_\varepsilon(Q, I, \mathcal{O}) \geq \frac{N}{2}$. Since this holds for every even number $N$, we get $\sup_{I \in \mathcal{I}} \text{icomp}_\varepsilon(Q, I, \mathcal{O}) = \infty$. Similarly, if $v(Q, A)$ is finite, then there is a node $L$ in the path induced by $Q, A$ with $\mathcal{L}_\varepsilon(L) = v(Q, A)$; any instance $I$ associated to this node $L$ (i.e., compatible with the answers given by $A$ until reaching $L$) then has $\text{icomp}_\varepsilon(Q, I, \mathcal{O}) \geq \mathcal{L}_\varepsilon(L) = v(Q, A)$. These observations imply $\sup_{I \in \mathcal{I}} \text{icomp}_\varepsilon(Q, I, \mathcal{O}) \geq \sup_A v(Q, A)$. Taking an infimum over $Q$ gives $\inf_Q \sup_{I \in \mathcal{I}} \text{icomp}_\varepsilon(Q, I, \mathcal{O}) \geq \inf_Q \sup_A v(Q, A)$ as desired.

This concludes the proof of the lemma. \hfill \Box

Now we can finally prove Lemma 28, stated in the beginning of the section.

Proof of Lemma 28. From Observation 47, there is an adversary $A$ that is $\varepsilon$-hard for $\ell$-rounds iff $\sup_A \inf_Q v(Q, A) > \ell$. Combining Lemmas 51, 52, and 53, the latter is equivalent to $\text{icomp}_\varepsilon(\mathcal{I}, \mathcal{O}) > \ell$, which concludes the proof. \hfill \Box
Chapter 3

Transfers from Approximate to Exact Information

Chapter Abstract

Given any algorithm for convex optimization that uses exact first-order information (i.e., function values and subgradients), we show in this chapter how to use such an algorithm to solve the problem with access to inexact first-order information. This is done in a “black-box” manner without knowledge of the internal workings of the algorithm. Other existing work, such as work done by Devolder-Glineur-Nesterov and Schmidt-Le Roux-Bach, considers the performance of specific algorithms like (accelerated) gradient descent with inexact information. In comparison, the results presented here apply to a wider range of algorithms beyond variants of gradient descent, such as projection-free methods, cutting-plane methods, or any other first-order methods formulated in the future. Further, they also apply to algorithms that handle structured nonconvexities like mixed-integer decision variables. While we present these as transfers of the guarantees of an algorithm from exact to inexact information, there are also implications for upper bounds on the information complexity of convex optimization under approximate first-order oracles; namely, any upper bound for the exact-information case can be transferred appropriately to the approximate-information case.
3.1 Introduction

Algorithms for any sufficiently general class of relevant optimization problems in settings such as (1.1) need to collect information about the particular instance by making (adaptive) queries about the objective before they can report a good solution. In this chapter, we focus on the following important class of optimization problems over a fixed ground set $X \subseteq \mathbb{R}^d$

$$\min\{f(x) : x \in X\},$$

where $f : \mathbb{R}^d \to \mathbb{R}$ is a (possibly nonsmooth) convex function. The setup is extremely similar to that of (1.1), only that $X$ is allowed to be any type of set. In fact, when the underlying ground set $X$ is all of $\mathbb{R}^d$ or some fixed convex subset, (3.1) is the classical convex optimization problem of (1.1) (without integer variables). In this chapter, we allow $X$ to be more general and to be used to model some known nonconvexity, e.g. integrality constraints by setting $X = C \cap (\mathbb{Z}^{d_1} \times \mathbb{R}^{d_2})$ with $d_1 + d_2 = d$, where $C \subseteq \mathbb{R}^d$ is a fixed convex set. From an algorithmic perspective, the setup is that the algorithm has complete knowledge of what $X$ is, but does not a priori know $f$ and must collect information via queries. As discussed in Chapter 1, a standard model for accessing the function $f$ is through so-called first-order oracles. At any point during its execution, the algorithm can request the function value and the (sub)gradient of $f$ at any point $\bar{x} \in \mathbb{R}^d$.

Given access to a first-order oracle, a first-order algorithm makes adaptive queries to this oracle and, after it judges that it has collected enough information about $f$, it reports a solution with certain guarantees. A long line of research has gone into understanding exactly how many queries are needed to solve different classes of problems (with different sets of assumptions on $f$ and $X$), with tight upper and lower bounds on the information complexity known in the literature, along with the novel results we have already presented in Chapter 2; see e.g. [85, 21, 81, 8, 11] for an overview of the literature.

In this chapter, we are particularly interested in what happens if the response of the oracle is not exact, but approximate (with possibly desired accuracy); oracles
returning approximations of first-order information can also be seen as oracles based on first-order information (Definition 5), and have practical motivations. For example, the response of the oracle might be itself a solution to another computational problem which is solved only approximately, which happens when using function smoothing [84], and in minimax problems [107]. Stochastic first-order oracles, modeling applications where only some estimate of the gradient is used, may also be viewed as inexact oracles whose accuracy is a random variable at each iteration. Thus, researchers have also investigated what one can say about algorithms that have access to inexact oracle responses (with possibly known guarantees on the inexactness). Early work on this topic appears in Shor [98] and Polyak [91], and more recent progress can be found in [34, 96, 73, 57, 68, 30] and references therein. To the best of our knowledge, all previous work on inexact first-order oracles has focused either on how specific algorithms like (accelerated) gradient methods perform with inexact (sub)gradients with no essential change to the algorithm, or on how to adapt a particular class of algorithms to perform well with inexact information.

In this chapter, we provide a different approach to the problem of inexact information. We provide a way to take any first-order algorithm that solves (3.1) with exact first-order information and, with absolutely no knowledge of its inner workings, show how to make the same algorithm work with inexact oracle information. Thus, in contrast to earlier work, our result is not the analysis of specific algorithms under inexact information or the adaptation of specific algorithms to use inexact information. It is in this sense that we believe our results to be universal because they apply to a much wider class of algorithms than previous work, including gradient descent, cutting plane methods, bundle methods, projection-free methods etc., and also to any first-order method that is invented in the future for optimization problems of the form (3.1).

3.2 Formal statement of results and discussion

We re-state some notation here to keep the chapter largely self-contained. We use $\| \cdot \|$ to denote the standard Euclidean ($L_2$) norm and $B(c, r)$ to denote the Euclidean ball of radius $r$ centered at $c \in \mathbb{R}^d$, and simply write $B(r)$ if the center is the origin. A function
A $\eta$-approximate first-order oracle for a convex function $f : \mathbb{R}^d \to \mathbb{R}$ takes as input a query point $\bar{x} \in \mathbb{R}^d$ and returns a first-order pair $(\tilde{f}, \tilde{g})$ satisfying $|\tilde{f} - f(\bar{x})| \leq \eta$ and $\|\tilde{g} - g\| \leq \frac{\eta}{2R}$ for some subgradient $g \in \partial f(\bar{x})$.

Note that this is indeed an oracle based on first-order information as in Definition 5. The first-order chart $G$ can still assign to each $x$ some $g \in \partial f(x)$ for each instance, but the answer the oracle gives is then $q(x) = g + u$, where $\|u\| \leq \eta$.

We now state this chapter’s main results. Let $\mathcal{F}^0(M, R)$ denote the standard family of instances of the optimization problem (3.1) consisting of all $M$-Lipschitz (possibly nondifferentiable) convex functions $f$, with respect to the $L^2$-norm, such that the minimizer $x^* \in X$ is contained in the ball $B(R)$ $^1$. We remind the reader that the underlying set $X$ need not be $\mathbb{R}^d$ and may be nonconvex; below, when we talk about a first order algorithm for (3.1) we mean an algorithm that can solve (3.1) with access to first-order oracles for $f$. We use $\text{OPT}(f)$ to denote the optimal value of the instance $f$.

**Theorem 55.** Consider an algorithm for (3.1) such that for any instance $f \in \mathcal{F}^0(M, R)$, with access to function values and subgradients of $f$, after $T$ iterations the algorithm reports a feasible solution $x \in X$ with error at most $\text{err}(T, M, R)$, i.e., $f(x) \leq \text{OPT}(f) + \text{err}(T, M, R)$.

Then there is an algorithm that, with access to an $\eta$-approximate first-order oracle for $f$ for any $\eta \geq 0$, after $T$ iterations the algorithm returns a feasible solution $\bar{x} \in X$ with value

$$f(\bar{x}) \leq \text{OPT}(f) + \text{err}(T, M', R) + 4\eta T,$$

where $M' = M + \frac{\eta}{2R}$.

$^1$Note that in Chapter 2 we worked with $x^*$ being in $B_{\infty}(R)$ (the $R$-ball with respect to $\|\cdot\|_\infty$), and also assumed Lipschitzness with respect to $\|\cdot\|_\infty$. Here, however, we are using the $L^2$-norm. Both are commonly seen across the optimization literature, though Euclidean norm tends to be more common in continuous optimization, while $\infty$-norm is more commonly seen for (mixed-)integer settings.
Though we state this theorem as an existence result, our proof is constructive and exactly formulates the desired algorithm via Procedures 5 and 6. Let us illustrate what this theorem says when applied to two classical algorithms for convex optimization (i.e., $X = \mathbb{R}^d$): subgradient methods and cutting-plane methods [85]. When using exact first-order information, the subgradient method produces after $T$ iterations a solution with error at most $O\left(\frac{MR}{\sqrt{T}}\right)$. Applying the procedures mentioned from Theorem 55 to this algorithm, one obtains an algorithm that uses only $\eta$-approximate first-order information and after $T$ iterations produces a solution whose error is at most $O\left(\frac{MR\eta}{\sqrt{T}}\right) + 2T\eta$. If one can choose the accuracy of the inexact oracle, setting $\eta = \frac{\varepsilon^3}{M^2R^2}$ and $T = \left\lceil \frac{M^2R^2}{\varepsilon^2} \right\rceil$ gives a solution with error at most $O(\varepsilon)$. Note that this does not involve knowing anything about the original algorithm; it simply illustrates the tradeoff between the oracle accuracy and final solution accuracy.

Similarly, for classical cutting-plane methods (e.g., center-of-gravity, ellipsoid, Vaidya) the error after $T$ iterations is at most $\left(\frac{MR^2}{\rho} \right) \exp\left(\frac{-T}{\text{poly}(d)}\right)$. Thus, with access to $\eta$-approximate first-order oracles, we can use our result to produce a solution with error at most $\left(\frac{(M+\frac{R}{\rho})R^2}{\rho} \right) \exp\left(\frac{-T}{\text{poly}(d)}\right) + 4\eta T$. With the desired accuracy of $\eta = \frac{\varepsilon}{\text{poly}(d) \log\left(\frac{MR}{\rho\varepsilon}\right)}$, and $T = 2\text{poly}(d) \log\left(\frac{MR}{\rho\varepsilon}\right)$, it gives a solution with error at most $O(\varepsilon)$.

We next consider the family of $\alpha$-smooth functions, i.e., the family $\mathcal{F}^1(M, \alpha, R)$ of $M$-Lipschitz convex functions that are differentiable with $\alpha$-Lipschitz gradient maps, whose minimizers are contained in $B(R)$. This is a classical family of objective functions in convex optimization that admits the celebrated accelerated method of [83] (see [31] for a survey). We give the following universal transfer theorem for algorithms for smooth objective functions.

**Theorem 56.** Consider an algorithm for (3.1) such that for any instance in $f \in \mathcal{F}^1(M, \alpha, R)$, with access to function values and subgradients of $f$, after $T$ iterations the algorithm reports a feasible solution $x \in X$ with error at most $\text{err}(T, M, \alpha, R)$, i.e., $f(x) \leq \text{OPT}(f) + \text{err}(T, M, \alpha, R)$.

Then there is an algorithm that, with access to an $\eta$-approximate first-order oracle
for \( f \), after \( T \) iterations the algorithm returns a feasible solution \( \bar{x} \in X \) with value

\[
f(\bar{x}) \leq \text{OPT}(f) + \text{err}(T, M', \alpha', R) + 5\eta \cdot (T + 2),
\]

where \( M' = M + \frac{\eta}{2R} \), \( \alpha' = \alpha \cdot \sqrt{d} \cdot \left( 4\sqrt{5(T + 1)} + 3 \right) \).

As an illustration, we apply this transfer theorem to the accelerated algorithm of [83] for continuous optimization (\( X = \mathbb{R}^d \)): Under perfect first-order information, it obtains error \( O(\frac{\alpha^2 R}{T^2}) \) after \( T \) iterations. Using our transfer theorem as a wrapper gives an algorithm that, using only \( \eta \)-approximate first-order information, obtains error \( O(\frac{\sqrt{d} \alpha}{\sqrt{T}} + \eta T) \); if the accuracy of the oracle is set to \( \eta = O(\frac{\sqrt{d} \alpha}{\sqrt{T}}) \), this gives an algorithm with error \( O(\frac{\sqrt{d} \alpha}{\sqrt{T}}) \). While this does not recover in full the acceleration of Nesterov’s method, the key take away is that a significant amount of acceleration (i.e., error rates better than those possible for non-smooth functions) can be preserved under inexact oracles in a universal way, for any accelerated algorithm requiring exact information.

**Remark 57.** For the sake of exposition, we have assumed that the accuracy \( \eta \) of the oracle is fixed and the additional error is \( O(\eta T) \). However, one can allow different oracle accuracies \( \eta_t \) at each query point \( x_t \) and the additional error is \( O(\sum_t \eta_t) \) (and the parameter \( M' = M + \frac{1}{2R} \max_t \eta_t \)).

### 3.2.1 Allowing inexactness in the constraint set

So far we have assumed that the algorithm has complete knowledge of the constraints \( X \). Now, we extend our results to include algorithms that can work with larger classes of constraints that are not fully known up front. In other words, just like the algorithm needs to collect information about \( f \), it also needs to collect information about \( X \), via another oracle, to be able to solve the problem. To capture the most general algorithms of this type, we formalize this setting by assuming \( X \) is of the form \( C \cap Z \), where \( C \) belongs to a class of closed, convex sets and \( Z \) is possibly nonconvex but completely known (e.g., \( Z = Z^{d_1} \times \mathbb{R}^{d_2} \) with \( d_1 + d_2 = d \)).

\[
\min \{ f(x) : x \in C \cap Z \}. \tag{3.2}
\]
The algorithm then must collect information about $C$, for which we use the common model of allowing the algorithm access to a separation oracle, as discussed in Chapter 1. Recall that upon receiving a query point $x$, a separation oracle either reports correctly that $x$ is inside $C$ or otherwise returns a separating hyperplane that separates $x$ from $C$. We note that a separation oracle for $C$ is in some sense comparable to a first-order oracle for a convex function $f$: since the pair $(f(x), \nabla f(x))$ can be viewed as providing a supporting hyperplane for the epigraph of $f$ at $x$, using an oracle that returns separating hyperplanes for $C$ provides a comparable way of collecting information about the constraints. Let us first precisely define the inexact version of a separation oracle.

**Definition 58.** For a closed, convex set $C \subseteq B(R)$ and a query point $\bar{x} \in B(R)$, an $\eta$-approximate separation oracle reports a separation response $(\text{flag}, \tilde{g}) \in \{\text{Feasible}, \text{Infeasible}\} \times \mathbb{R}^d$ such that if $\bar{x} \in C$ then flag = Feasible (with no requirement on $\tilde{g}$), and otherwise flag = Infeasible and $\tilde{g}$ is a unit vector such that there exists some unit vector $g$ satisfying $\langle g, x \rangle \leq \langle g, \bar{x} \rangle$ for all $x \in C$ and $\|\tilde{g} - g\|_2 \leq \frac{\eta}{2R}$. Given such a $\tilde{g}$ (for $\bar{x} \notin C$), we call the hyperplane through $\bar{x}$ induced by this normal vector an $\eta$-approximate separating hyperplane for $\bar{x}$.

We now state our results for algorithms that work with separation oracles. Note that for this, instances of (3.1) have to specify both $f$ and $C$, as opposed to just $f$, since only $Z$ is known but not $C$. In this chapter we will use $\mathcal{I}_{R,\rho,M}$ as shorthand for $\mathcal{I}_{R,\rho,M}$ from the previous chapters, i.e. $\mathcal{I}_{R,\rho,M}$ denotes the set of all instances $(f, C)$ where $f : \mathbb{R}^d \to \mathbb{R}$ is an $M$-Lipschitz convex function and $C$ is a compact, convex set that contains a ball of radius $\rho$ and is contained in $B(R)$. We opt to suppress $d$ here because our results are independent thereof. We use $\text{OPT}(f, C)$ to denote the minimum value of (3.2). The first result we state is for pure convex problems, i.e., $Z = \mathbb{R}^d$.

**Theorem 59.** Consider an algorithm for (3.2) with $Z = \mathbb{R}^d$, such that for any instance in $(f, C) \in \mathcal{I}_{R,\rho,M}$, with access to function values and subgradients of $f$ and separating hyperplanes for $C$, after $T$ iterations the algorithm reports a feasible solution $x \in C$ with error at most $\text{err}(T, M, R, \rho)$, i.e., $f(x) \leq \text{OPT}(f, C) + \text{err}(T, M, R, \rho)$. 

91
Then there is an algorithm that, with access to an $\eta_f$-approximate first-order oracle for $f$ and an $\eta_C$-approximate separation oracle for $C$ for any $\eta_f \geq 0$ and $0 \leq \eta_C \leq \rho$, after $T$ iterations the algorithm returns a feasible solution $\bar{x} \in C$ with value

$$f(\bar{x}) \leq \text{OPT}(f, C) + \text{err}(T, M', R, \rho') + 4\eta_f T + \frac{2\eta_CM\rho}{\rho},$$

where $M' = M + \frac{\eta_f}{2\rho}$ and $\rho' = \rho - \eta_C$.

We can handle more general, nonconvex $Z$ with separation oracles under a slightly stronger “strict feasibility” assumption on $C$: let $\mathcal{I}_{R,\rho,M}^\star$ denote the subclass of instances from $\mathcal{I}_{R,\rho,M}$ where the minimizer $x^*$ of (3.2) is $\rho$-deep inside $C$, i.e., $B(x^*, \rho) \subseteq C$.

**Theorem 60.** Consider an algorithm for (3.2), such that for any instance in $(f, C) \in \mathcal{I}_{R,\rho,M}^\star$, with access to function values and subgradients of $f$ and separating hyperplanes for $C$, after $T$ iterations the algorithm reports a feasible solution $x \in C \cap Z$ with error at most $\text{err}(T, M, R, \rho)$, i.e., $f(x) \leq \text{OPT}(f, C) + \text{err}(T, M, R, \rho)$.

Then there is an algorithm that, with access to an $\eta_f$-approximate first-order oracle for $f$ and an $\eta_C$-approximate separation oracle for $C$ for any $\eta_f \geq 0$ and $0 \leq \eta_C \leq \rho$, after $T$ iterations the algorithm returns a feasible solution $\bar{x} \in C \cap Z$ with value

$$f(\bar{x}) \leq \text{OPT}(f, C) + \text{err}(T, M', R, \rho') + 4\eta_f T,$$

where $M' = M + \frac{\eta_f}{2\rho}$ and $\rho' = \rho - \eta_C$.

**Remark 61.** The objective functions in the above results were allowed to be any $M$-Lipschitz, possibly nondifferentiable, convex function. One can state versions of these results for algorithms that work for the smaller class of $\alpha$-smooth functions (e.g., accelerated projected gradient methods), just as Theorem 56 is a version of Theorem 55 for $\alpha$-smooth objectives. The reason is that the analysis for handling constraints is independent of the arguments needed to handle the objective using inexact oracles. Additionally, one can prove versions of all our theorems for strongly convex objective functions; we forego including the details thereof here and opt to present the main message of these contributions more concisely.
3.2.2 Consequences for Information Complexity

Each of the theorems listed above does the following: Given some algorithm that guarantees to report an $\varepsilon_T$-solution after $T$ queries to a full-information first order oracle, it shows the existence of another algorithm that in $T$ queries to an approximate first-order oracle reports an $\varepsilon'_T$-solution. Therefore, given some upper-bound on the information complexity for finding an $\varepsilon_T$ solution under full first-order information, this immediately implies an upper-bound on the information complexity for finding an $\varepsilon'_T$ solution under approximate information. We state here the information complexity equivalent of Theorem 60 to draw the connection to results presented in Chapter 2. Denote by $O_{\eta_f, \eta_c}$ an oracle based on first-order information (see Definition 5) that behaves like an $\eta_f$-approximate first-order oracle for (function / subgradient queries at) feasible points and a $\eta_c$-approximate oracle for (separation queries at) infeasible points.

Theorem 62. Suppose that

$$icomp_{\varepsilon}(I_{n,d,M,R,\rho}, O_{full}) \leq T,$$

with an algorithm giving error $err(n, d, T, M, R, \rho)$ certifying this, i.e. after $T$ iterations the algorithm gives error $err(n, d, T, M, R, \rho) \leq \varepsilon$. Then with $\bar{\rho} = \rho + \eta_c$, $\bar{M} = M - \frac{\eta_f^2}{2R}$ and $\varepsilon' = err(n, d, T, M, R, \rho) + 4\eta_f T$ and we have

$$icomp_{\varepsilon'}(I_{n,d,\bar{M},R,\bar{\rho}}, O_{\eta_f,\eta_c}) \leq T.$$

In particular, when $\eta_f \leq \frac{\varepsilon}{T}$ and $\eta_c \leq \frac{\rho}{2}$, we have

$$\varepsilon' = err(T, M + \frac{\varepsilon}{2TR}, R, \frac{\rho}{2}) + 4\varepsilon. \quad (3.3)$$

We point out (3.3) because for many cases one could expect this $\varepsilon'$ to simply be some constant multiple of $\varepsilon$. For example, consider the center of gravity method of Algorithm 1 for continuous unconstrained optimization. The algorithm needs $T = 2d\log\left(\frac{AMR}{\varepsilon}\right)$ iterations to report an $\varepsilon$ solution, or equivalently (solving for $\varepsilon$), after $T$
iterations it reports in the worst case an $\varepsilon$-solution with

$$\varepsilon = 4MR \cdot e^{-\frac{T}{2M}},$$

i.e. we have $\text{err}(d, M, R) = 4MR \cdot e^{-\frac{T}{2M}}$. Then with $\eta_f \leq \frac{\varepsilon}{T}$ and $\eta_C \leq \frac{\varepsilon}{T}$ as in Theorem 62, we have

$$\varepsilon' = 4\left(M + \frac{\varepsilon}{2TR}\right)R \cdot e^{-\frac{T}{2M}} + 4\varepsilon$$

$$\leq 8MR \cdot e^{-\frac{T}{2M}} + 4\varepsilon \leq 6\varepsilon,$$

assuming $\frac{\varepsilon}{2TR} \leq M$ in the second line (if this is not the case, the problem is trivial). Hence, here $\varepsilon' \leq 6\varepsilon$, meaning that up to this constant for $\varepsilon$, the information complexities under exact versus inexact information are the same when $\eta_f \leq \frac{\varepsilon}{T}$ and $\eta_C \leq \frac{\varepsilon}{T}$. One could interpret this as saying that for this case, any information obtained that provides more accuracy in the oracles than these values for $\eta_f$ and $\eta_C$ is superfluous (up to the constant factor difference).

### 3.2.3 Relation to existing work

Previous work on inexact first-order information focused on how certain known algorithms perform or can be made to perform under inexact information, most recently on (accelerated) proximal-gradient methods. For instance, [34] analyze the performance of (accelerated) gradient descent in the presence of inexact oracles, with no change to algorithm. They show that simple gradient descent (for unconstrained problems) will return a solution with additional error $O(\eta)$ and accelerated gradient descent incurs an additional error of $O(\eta T)$ (similar to our guarantees). Similarly, [96] does an analysis for (accelerated) proximal gradient methods, with more complicated forms of the additional error, depending on how well the proximal problems are solved.

In contrast, our result does not assume any knowledge of the internal logic of the algorithm. We must, therefore, use the algorithm in a “black-box” manner. We are able to do this by using the inexact oracles to construct a modified instance whose optimal
solution is similar in quality to that of the true instance, and where this inexact information from the true instance can be interpreted as exact information for the modified instance. Thus, we can effectively run the algorithm as a black-box on this modified instance and leverage its error guarantee. Constructing this modified instance in an online fashion requires technical ideas that are new, to the best of our knowledge, in this literature. For instance, it is not even true that given approximate function values and subgradients of a convex function, we can find another convex function that has these as exact function values and subgradients; see Figure 3.1. Thus, one cannot directly use the inexact information as is (contrary to what is done in many of the papers dealing with inexact information for specific algorithms), in the general case we consider. The key is to modify the inexact information so that the information the algorithm receives admits an extension into a convex function/set that is still close to the original instance.

When dealing with $\alpha$-smooth objectives, the arguments are especially technically challenging since we have to report approximate function and gradient values that allow for a smooth extension that also approximates the unknown objective well. This involves careful use of new, localized smoothing techniques and maximal couplings of probability distributions. Such smoothing guarantees based on the proximity to the class of smooth functions may be of independent interest (see Theorem 67).

New applications: Since our results apply to algorithms for any ground set $X$, we are able to handle mixed-integer convex optimization, i.e., $X = C \cap (\mathbb{Z}^{d_1} \times \mathbb{R}^{d_2})$, with inexact oracles. Recently, there have been several applications of such optimization problems in machine learning and statistics [15, 80, 5, 33, 36, 56, 55]. General algorithms for mixed-integer convex optimization, as well as specialized ones designed for specific applications in the above papers, all involve a sophisticated combination of techniques like branch-and-bound, cutting planes and other heuristics. To the best of our knowledge, the performance of these algorithms has never been analyzed under the presence of inexact oracles which can cause issues for all of these components of the algorithm. Our results apply immediately to all these algorithms, precisely because the internal workings of the algorithm are abstracted away in our analysis. This yields the first ever versions of these methods that can work with inexact oracles. Moreover, $X$
can be used to model other types of structured nonconvexities (e.g., complementarity constraints [29]) and our results show how to adapt algorithms in those settings to work with inexact oracles. Note that this holds for the cases where the convex set $C$ is explicitly known a priori (Theorems 55 and 56), or must be accessed via separation oracles (Theorems 59 and 60).

The remainder of this paper is dedicated to the proof sketches of Theorems 55 and 56. The missing details and proofs of Theorems 59 and 60 can be found in the Appendix.

3.3 Universal transfer for Lipschitz functions

In this section we prove our transfer result stated in Theorem 55. The proof relies on the following key concept: Given a set of points $x_1, \ldots, x_T \in B(R)$ (e.g., queries made by an optimization algorithm), we say that the sequence of first-order pairs\(^2\) $(f_1, g_1), \ldots, (f_T, g_T) \in \mathbb{R} \times \mathbb{R}^d$ has an $M$-Lipschitz convex extension, or simply $M$-extension, if there is a function $F$ that is convex, $M$-Lipschitz, and such that $f_t = F(x_t)$ and $g_t \in \partial F(x_t)$ for all $t$, i.e., the first-order information of $F$ at the queried points is exactly $\{(f_t, g_t)\}_t$.

As mentioned in the introduction, the main idea is to feed to the convex optimization algorithm $\mathcal{A}$ a sequence of pairs $(f_t, g_t)$'s that have an $M$-Lipschitz extension $F$ that is close to the original function $f$. Since the information is consistent with what the algorithm expects when interacting exactly with the function $F$, it will approximately optimize the latter which will then give an approximately optimal solution to the neighboring function $f$.

Unfortunately, it is easy to see approximate first-order information from $f$ for the queried points $x_t$'s does not necessarily have a Lipschitz convex extension (see Figure 3.1). Thus, the main subroutine of our algorithm Approximate-to-Exact given below is that given an approximate first-order oracle for $f$, it constructs first-order pairs $(\tilde{f}_t, \tilde{g}_t)$'s in an online fashion (i.e. $(\tilde{f}_t, \tilde{g}_t)$ only depends on $x_1, \ldots, x_t$) with the desired extension

\(^2\)We use first-order pair as just a more “visual” name for a pair in $\mathbb{R} \times \mathbb{R}^d$. 
properties.

\[ f(x) \]

\[ f_1 = f(x_1) \quad \text{slope } g_1 = 0.25 \]

\[ f_2 = f(x_2) \quad \text{slope } g_2 = -0.2 \]

\[ x_1 \quad x_2 \]

Figure 3.1: An example where two approximate function values and subgradients do not have a convex extension. The true function \( f \) is constant. The function values are reported with no error. The reported slopes are shown in red. However, these slopes decrease going from \( x_1 \) to \( x_2 \) thus eliminating the possibility of any convex function having these values and slopes at \( x_1 \) and \( x_2 \).

**Theorem 63** (Online first-order Lipschitz-extensibility). Consider an \( M \)-Lipschitz convex function \( f : B(R) \to \mathbb{R} \), and a sequence of points \( x_1, \ldots, x_T \in B(R) \). There is an online procedure that, given \( \eta \)-approximate first-order oracle access to \( f \), produces first-order pairs \((\hat{f}_1, \hat{g}_1), \ldots, (\hat{f}_T, \hat{g}_T)\) that have a \((M + \frac{\eta}{2T})\)-extension \( F : B(R) \to \mathbb{R} \) satisfying \( \|F - f\|_\infty \leq 2\eta T \). (Moreover, the procedure only probes the approximate oracle at the given points \( x_1, \ldots, x_T \).)

With this at hand, given any first-order algorithm \( A \) we can run it using only approximate first-order information in the following natural way:

**Procedure 5. Approximate-to-Exact(\( A, T \))**

For each timestep \( t = 1, \ldots, T \):

1. Receive query point \( x_t \in B(R) \) from \( A \).

2. Send point \( x_t \) to the \( \eta \)-approximate oracle for \( f \) and receive the information \((\hat{f}_t, \hat{g}_t)\).

3. Use the online procedure from Theorem 63 to construct the first-order pair \((\hat{f}_t, \hat{g}_t)\).

4. Send \((\hat{f}_t, \hat{g}_t)\) to the algorithm \( A \).

Return the point in \( X \) returned by \( A \).
Proof of Theorem 55. Consider a first-order algorithm $A$ that, for any $M$-Lipschitz convex function, after $T$ iterations returns a point $\bar{x} \in X$ such that $f(\bar{x}) \leq \text{OPT}(f) + \text{err}(T, M, R)$. We show that running Procedure 5 with $A$ as input, which only uses an $\eta$-approximate oracle for $f$, returns a point $\bar{x} \in X$ such that $f(\bar{x}) \leq \text{OPT}(f) + \text{err}(T, M', R) + 4\eta T$ with $M' = M + \frac{\eta}{2\eta}$. 

To see this, let $F$ be an $M'$-extension for the first-order pairs $(\hat{f}_t, \hat{g}_t)$ sent to the algorithm $A$ in Procedure 5 with $\|F - f\|_{\infty} \leq 2\eta T$, guaranteed by Theorem 63. This means that the execution of the first-order algorithm $A$ during our procedure is exactly the same as executing $A$ directly on the convex function $F$. Thus, by the error guarantee of $A$, the point $\bar{x} \in X$ returned by $A$ after $T$ iterations (which is the same point returned by our procedure) is almost optimal for $F$, i.e., $F(\bar{x}) \leq \text{OPT}(F) + \text{err}(T, M', R)$. Since $F$ and $f$ are pointwise within $\pm 2\eta T$ of each other, the value of the solution $\bar{x}$ with respect to the original function $f$ satisfies 

\[
f(\bar{x}) \leq F(\bar{x}) + 2\eta T \leq \text{OPT}(F) + \text{err}(T, M', R) + 2\eta T \leq \text{OPT}(f) + \text{err}(T, M', R) + 4\eta T,
\]

which proves the desired result. 

3.3.1 Computing Lipschitz-extensible first-order pairs

In this section we describe the procedure that constructs the first-order pairs with a Lipschitz convex extension $F$ that satisfies $\|F - f\|_{\infty} \leq 2\eta T$, proving Theorem 63. Before getting into the heart of the matter, we show that the latter property can be significantly weakened: instead of requiring both $f(x) \geq F(x) - 2\eta T$ and $f(x) \leq F(x) + 2\eta T$ for all $x \in B(R)$, we can relax the latter to only hold for the queried points $x_1, \ldots, x_T$.

**Lemma 64.** Consider a sequence of points $x_1, \ldots, x_T \in B(R)$, and a sequence of first-order pairs $(\hat{f}_1, \hat{g}_1), \ldots, (\hat{f}_T, \hat{g}_T)$. Consider $\delta > 0$ and $M' \geq M$, and suppose that there
is an $M'$-extension $F$ of these first-order pairs that satisfies:

\[
\begin{align*}
\text{approx. under-approximation} & \quad f(x) \geq F(x) - \delta, \quad \forall x \in B(R) \quad (3.6) \\
\text{approx. queried values} & \quad f(x_t) \leq F(x_t) + \delta, \quad \forall t = \{1, \ldots, T\}. \quad (3.7)
\end{align*}
\]

Then the first-order pairs have an $M'$-extension $F'$ such that $\|F' - f\|_\infty \leq \delta$.

We sketch the main idea of the proof here and defer the complete proof to Appendix 3.6.

**Proof sketch:** Since $F(x)$ only (approximately) under-approximates $f(x)$, other than at the queried points $F(x)$ may be much smaller than $f(x)$ for some $x$’s. We fix this by taking the maximum of the function $F$ and the function obtained by a “downward shift” of the original function $f$ by $-\delta$ (which preserves Lipschitzness and convexity). The shift, together with property (3.7) guarantees that the function $F$ is still “exposed” at the queried points $x_t$, and so it is still an extension of the first-order pairs $(\hat{f}_t, \hat{g}_t)$’s.

Given Lemma 64, to prove Theorem 63 it suffices to do the following. Consider a sequence of points $x_1, \ldots, x_T \in B(R)$. Using an $\eta$-approximate first-order oracle to access the function $f$ (at the points $x_1, \ldots, x_T$), we need to produce a sequence of first-order pairs $(\hat{f}_1, \hat{g}_1), \ldots, (\hat{f}_T, \hat{g}_T)$ in an online fashion that have an $M$-extension $F$ achieving the approximations (3.6) and (3.7). We do this as follows.

At iteration $t$ we maintain the function $F_t(x) := \max\{\hat{f}_\tau + \langle \hat{g}_\tau, x - x_\tau \rangle : \tau \leq t\}$, that is, the maximum of the linear functions induced by the first-order pairs $(\hat{f}_\tau, \hat{g}_\tau)$ constructed up to this point. We would like to define the pairs $(\hat{f}_\tau, \hat{g}_\tau)$ to guarantee that for all $t$, $F_t$ is an $M$-extension for these pairs, and satisfies (3.6) and (3.7) for $x_1, \ldots, x_t$. In this case, $F = F_T$ gives the desired function.

For that, suppose the above holds for $t - 1$; we will show how to define $(\hat{f}_t, \hat{g}_t)$ to maintain this invariant for $t$. We should think of constructing $F_t$ by taking the maximum of $F_{t-1}$ and a new linear function $\hat{f}_t + \langle \hat{g}_t, x_t - x_t \rangle$. To ensure that $F_t$ is an extension
of the first-order pairs thus far, we need to make sure that:

1. \( \hat{f}_t \geq F_{t-1}(x_t) \). This is necessary to ensure that \( F_t(x_t) = \hat{f}_t \), and also guarantees \( \hat{g}_t \in \partial F_t(x_t) \).

2. \( \hat{f}_t + \langle \hat{g}_t, x_{\tau} - x_t \rangle \leq F_{t-1}(x_\tau), \quad \forall \tau \leq t - 1 \). This is necessary to ensure that \( F_t(x_\tau) = F_{t-1}(x_\tau) = \hat{f}_\tau \), and also guarantees \( \partial F_t(x_\tau) \supseteq \partial F_{t-1}(x_\tau) \ni \hat{g}_\tau \).

To construct \((\hat{f}_t, \hat{g}_t)\) with these properties, we probe the approximate first-order oracle for \( f \) at \( x_t \), and receive an answer \((\tilde{f}_t, \tilde{g}_t)\). If setting \((\hat{f}_t, \hat{g}_t) = (\tilde{f}_t, \tilde{g}_t)\) violates the first item above, we simply use the first-order information of \( F_{t-1} \) at \( x_t \), i.e., we set \( \hat{f}_t = F_{t-1}(x_t) \) and \( \hat{g}_t \in \partial F_{t-1}(x_t) \).

If the second item above is violated instead, we shift the value \( \tilde{f}_t \) down as little as possible to ensure the desired property, i.e., we set \( \hat{f}_t = \tilde{f}_t - s^* \) for appropriate \( s^* > 0 \). With this shifted value, the first item may now be violated, in which case we again just use the current first-order information of \( F_{t-1} \).

These steps are formalized in the following procedure.

**Procedure 6.**

For each \( t = 1, \ldots, T \):

1. Query the \( \eta \)-approximate oracle for \( f \) at \( x_t \), receiving the first-order pair \((\tilde{f}_t, \tilde{g}_t)\).

2. Let \( s^* := \min\{s \geq 0 : \tilde{f}_t - s + \langle \tilde{g}_t, x_\tau - x_t \rangle \leq F_{t-1}(x_\tau), \forall \tau \leq t - 1\} \)

3. If \( \tilde{f}_t - s^* \geq F_{t-1}(x_t) \), then set \((\hat{f}_t, \hat{g}_t) = (\tilde{f}_t - s^*, \tilde{g}_t)\).

4. Else, set \( \hat{f}_t = F_{t-1}(x_t) \) and let \( \hat{g}_t \) be subgradient in \( \partial F_{t-1}(x_t) \).

5. Set \( F_t(x) = \max\{F_{t-1}(x), \hat{f}_t + \langle \hat{g}_t, x - x_t \rangle\} \).

We remark that this requires storing historical values of \( \tilde{f}_t \) and \( \tilde{g}_t \) (this seems unavoidable to ensure convexity of \( F_t \)). We now prove that the functions \( F_t \)'s have the desired properties.

**Lemma 65.** For every \( t = 1, \ldots, T \), the function \( F_t \) is an \( (M + \frac{\eta}{2M}) \)-extension of the first-order information pairs \((\hat{f}_1, \hat{g}_1), \ldots, (\hat{f}_t, \hat{g}_t)\).
Proof. Since $F_t$ is the maximum over affine functions, it is convex. Moreover, all of its subgradients come from the set $\{\tilde{g}_\tau\}_{\tau}$, and by the approximation guarantee of the oracle we have that for some subgradient $g \in \partial f(x_\tau)$, $\|\tilde{g}\|_2 \leq \|\tilde{g} - g\|_2 + \|g\|_2 \leq \frac{\eta}{\sqrt{R}} + M$, where we used that fact that $f$ is $M$-Lipschitz; thus, $F_t$ is $(M + \frac{\eta}{\sqrt{R}})$-Lipschitz.

We prove by induction on $t$ that $F_t$ is an extension of the desired pairs (the base case $t = 1$ can be readily verified). Recall $F_t(x_t) = \max\{F_{t-1}(x_t), H(x_t)\}$, where $H(x) := \hat{f}_t + \langle \hat{g}, x - x_t \rangle$. By the definition of $s^*$, for all $x = x_1, \ldots, x_{t-1}$, this maximum is achieved by the function $F_{t-1}$, giving, by induction, that for all $\tau \leq t - 1$, $F_t(x_\tau) = F_{t-1}(x_\tau) = \hat{f}_\tau$; this also implies that for such $\tau$’s, $\partial F_t(x_\tau) \supseteq \partial F_{t-1}(x_\tau) \ni \hat{g}_\tau$, the last inclusion again following by induction. These give the extension property for the pairs $(\hat{f}_\tau, \hat{g}_\tau)$ with $\tau \leq t - 1$.

It remains to verify that this also holds for $\tau = t$. Now the maximum in the definition of $F_t(x_t)$ is achieved by the function $H$: if $\hat{f}_t - s^* \geq F_{t-1}(x_t)$, the procedure sets $\hat{f}_t = \hat{f}_t - s^*$ and we have $H(x_t) = \hat{f}_t \geq F_{t-1}(x_t)$; otherwise the procedure sets $\hat{f}_t = F_{t-1}(x_t)$ and so $H(x_t) = F_{t-1}(x_t)$. Again this implies that $\partial F_t(x_t) \supseteq \partial H(x_t) = \{\hat{g}_t\}$. This concludes the proof of the lemma.

Finally, we show that the functions $F_t$ approximate $f$ according to (3.6) and (3.7).

Lemma 66. For every $t = 1, \ldots, T$, the $F_t$ satisfies inequalities (3.6) and (3.7) with $\delta = 2\eta t$.

Proof. Again we prove this by induction on $t$. Fix $t$. Let $\Delta := \hat{f}_t - f(x_t)$ be the error the inexact oracle makes on the function value. We claim that the shift $s^*$ used in iteration $t$ of Procedure 6 satisfies $s^* \leq \max\{0, \Delta + 2\eta t\}$. To see this, the $\eta$-approximation of the oracle guarantees that there is a subgradient $g \in \partial f(x_t)$ such that $\|\hat{g}_t - g\|_2 \leq \frac{\eta}{\sqrt{R}}$, and
so for every $\tau \leq t - 1$

$$\tilde{f}_t + \langle \tilde{g}_t, x_\tau - x_t \rangle$$

$$= \Delta + \underbrace{f(x_t) + \langle g, x_\tau - x_t \rangle}_{\leq f(x_\tau)} + \underbrace{\langle \tilde{g}_t - g, x_\tau - x_t \rangle}_{\leq \|\tilde{g}_t - g\|\|x_\tau - x_t\|\leq \eta}$$

$$\leq F_{t-1}(x_\tau) + \Delta + 2t\eta; \quad (3.8)$$

the first underbrace following since $g$ is a subgradient of $f$, and the last inequality following from the induction hypothesis $F_{t-1}(x_\tau) \geq f(x_\tau) - 2(t - 1)\eta$ (inequality (3.7)); the optimality of $s^*$ then guarantees that it is at most $\max\{0, \Delta + 2\eta t\}$, proving the claim.

Now we show that $F_t$ satisfies the desired bounds, namely $F_t(x_\tau) \geq f(x_\tau) - 2\eta t$ for all $\tau \leq t$, and $F_t(x) \leq f(x) + 2\eta t$ for all $x \in B(R)$. From the inductive hypothesis, for $\tau \leq t - 1$ we have $F_t(x_\tau) \geq F_{t-1}(x_\tau) \geq f(x_\tau) - 2\eta(t - 1)$, giving the first bound for these $x_\tau$. For $x_t$, notice that $F_t(x_t) \geq \hat{f}_t - s^*$: if Line 3 of the procedure is executed we have $F_t(x_t) \geq \hat{f}_t \geq \tilde{f}_t - s^*$, and if Line 4 is executed then it means $\hat{f}_t - s^* < F_{t-1}(x_t)$ and so $F_t(x_t) = F_{t-1}(x_t) > \tilde{f}_t - s^*$. Then

$$F_t(x_t) \geq \tilde{f}_t - s^* \geq \hat{f}_t - \max\{0, \Delta - 2\eta t\}$$

$$\geq \max\{f(x_t) - \eta, f(x_t) - 2\eta t\} = f(x_t) - 2\eta t,$$

where in the second inequality we used the upper bound on the shift $s^* \leq \max\{0, \Delta + 2\eta t\}$, and in the next inequality we used the guarantee $|\hat{f}_t - f(x_t)| \leq \eta$ from the approximate oracle.

For the upper bound $F_t(x) \leq f(x) + 2\eta t$, when Line 3 of the procedure is executed we have $F_t(x) = F_{t-1}(x) \leq f(x) + 2\eta(t - 1)$, the last inequality from the inductive hypothesis. When Line 4 is executed instead, we have $\hat{f}_t = \tilde{f}_t - s^*$, and the same
development as in (3.8) reveals that
\[
\hat{f}_t + \langle \hat{g}_t, x - x_t \rangle \leq \hat{f}_t + \langle \tilde{g}_t, x - x_t \rangle \leq f(x) + \Delta + \eta \leq f(x) + 2\eta,
\]
where the last inequality again uses that \( \Delta = \hat{f}_t - f(x_t) \leq \eta \) due to the guarantee of
the approximate oracle. Thus, \( F_t(x) \leq \max\{f(x) + 2\eta(t-1), f(x) + 2\eta\} \leq f(x) + 2\eta t \),
giving the desired bound. This concludes the proof of the lemma.

Combining Lemmas 64, 65, and 66 shows that the first-order pairs produced by Procedure 6 satisfies the properties stated in Theorem 63, finally concluding its proof.

### 3.4 Universal transfer for smooth functions

In this section we prove our transfer theorem for smooth functions stated in Theorem 56. Recall that a function \( f \) is \( \alpha \)-smooth if it has \( \alpha \)-Lipschitz gradients:
\[
\forall x, y \in \mathbb{R}^d, \quad \|\nabla f(x) - \nabla f(y)\| \leq \alpha \|x - y\|.
\]

As in the proof of the previous transfer theorem, the core element is the following:
Given the sequence of iterates \( x_1, \ldots, x_t \) of a black-box optimization algorithm and access to an approximate first-order oracle to the smooth objective function \( f \), construct in an online fashion first-order pairs \( (\hat{f}_t, \hat{g}_t) \) and, implicitly, a smooth function \( S \) close to the original \( f \) such that \( (\hat{f}_t, \hat{g}_t) \) provide exactly the value and gradient of \( S \) at \( x_t \).

**Theorem 67** (Online first-order smooth-extensibility). Consider an \( \alpha \)-smooth, \( M \)-Lipschitz convex function \( f : B(R) \to \mathbb{R} \), and a sequence of points \( x_1, \ldots, x_T \in B(R) \). Then there is an online procedure that given \( \eta \)-approximate first-order oracle access to \( f \), produces first-order pairs \( (\hat{f}_1, \hat{g}_1), \ldots, (\hat{f}_T, \hat{g}_T) \) that have an \( \alpha' \)-smooth \( (M + \frac{\eta R}{\sqrt{T}}) \)-extension \( S : B(R) \to \mathbb{R} \) satisfying \( \|S - f\|_\infty \leq 5\eta(T + 2) \), where \( \alpha' = \alpha \sqrt{d} \left( 4\sqrt{5} \cdot (T + 1) + 3 \right) \). Moreover, the procedure only probes the approximate oracle at the given points \( x_1, \ldots, x_T \).
In the previous section, the extension was created by adding a new linear function at every iteration; this produced the piecewise linear (non-smooth) functions $F_t$ in the previous section. Having to construct a smooth extension creates a challenge. Our approach is to apply a smoothing procedure to these piecewise linear functions, in an online manner. One issue is that most standard smoothing procedures (e.g., via inf-convolution [12] or Gaussian smoothing [84]) may use the values of the non-smooth base function over the whole domain; in our online construction, at a given point in time we have determined the value of the function only in a neighborhood of the previous iterates, and the updated functions can change at points outside these small neighborhoods. Thus, we employ a localized smoothing procedure. Moreover, we need the procedure to leverage the fact that the non-smooth base function is close to a smooth one, and produce stronger smoothing guarantees by making use thereof. We start by describing this smoothing technique and its properties, and then describe the full procedure that gives Theorem 67.

**Randomized smoothing of almost smooth functions.** Given a function $h : \mathbb{R}^d \to \mathbb{R}$ and a radius $r > 0$, we define the smoothed function $h_r$ by $h_r(x) := \mathbb{E} h(x + rU)$, where $U$ is uniformly distributed on the unit ball $B(0,1)$. It is well-known that when $h$ is convex and $M$-Lipschitz, then $h_r$ is differentiable, also $M$-Lipschitz, and, most importantly, is $\frac{Mr^2}{\sqrt{d}}$-smooth [110]. However, we show that the smoothing parameter can be significantly improved when the function $h$ is already close to a smooth function. The proof is deferred to Appendix 3.5.1.

**Lemma 68.** Let $h : \mathbb{R}^d \to \mathbb{R}$ be a convex function such that there exists an $\alpha$-smooth convex function $f$ with $\|h - f\|_\infty \leq \varepsilon$. Then the smoothed function $h_r$ satisfies:

1. $h_r$ is $\left(\frac{4\sqrt{\alpha d}}{\varepsilon} + 3\alpha \sqrt{d}\right)$-smooth

2. $\|h_r - f\|_\infty \leq \varepsilon + \frac{\alpha r^2}{2}$

**Construction of the smooth-extension.** As mentioned, in each iteration $t$ we will maintain a piecewise linear function $F_t$ constructed very similarly to the proof of Theorem 63. Now we will also maintain the smoothened version $(F_t)_r$ of this function that uses the randomized smoothing discussed above (for a particular value of $r$). Our transfer
procedure then returns the first-order information $\hat{f}_t := (F_t)_r(x_t)$ and $\hat{g}_t := \nabla(F_t)_r(x_t)$ of the latter. The final smooth function $S : B(R) \rightarrow \mathbb{R}$ compatible with the first-order information returned by the procedure will be given, as in Lemma 64, by taking the maximum between the final $F_T$ and a shifted version of the original function $f$.

The main difference in how the functions $F_t$’s are constructed, compared to the proof of Theorem 63, is the following. Previously, in order to ensure that $F_T$ (and so the final extension) was compatible with the first-order pairs output in earlier iterations, we needed to “protect” the points $x_t$ and ensure that the function values and gradients at these points did not change over time, e.g., we needed $F_T(x_t) = F_t(x_t)$. But now the first-order pair output for the query point $x_t$ depends not only on the value of $F_t$ at $x_t$, but also on the values on the whole ball $B(x_t, r)$ that are used to determine the smoothed function $(F_t)_r$ at $x_t$. Thus, we will now need to “protect” these balls and ensure that the function values over them do not change in later iterations.

We now formalize the construction of the functions $F_t$, the first-order information returned, and the final extension $S$ in Procedure 7.

**Procedure 7.**

Set $r = \sqrt{\eta/\alpha}$. For each $t = 1, \ldots, T$:

1. Query the $\eta$-approximate oracle for $f$ at $x_t$, receiving the first-order pair $(\hat{f}_t, \hat{g}_t)$

2. Define the function $F_t$ by setting $F_t(x) = \max\{F_{t-1}(x), \hat{f}_t + \langle \hat{g}_t, x - x_t \rangle - (4\eta t + \alpha r^2 t + 2\eta)\}$ for all $x$

3. Output the first-order information of the randomly smoothed function $(F_t)_r$:

   $\hat{f}_t := (F_t)_r(x_t)$ and $\hat{g}_t := \nabla(F_t)_r(x_t)$

Define the function $S : B(R) \rightarrow \mathbb{R}$ by $S = (\max\{F_T, f - 4\eta(T + 1) + \alpha r^2(T + 1)\})_r$, where max denotes pointwise maximum.

The proof that this procedure indeed yields Theorem 67 is presented in Appendix 3.5.2.
### 3.5 Universal transfer for smooth functions

In this section we present the missing proofs for our transfer theorem for smooth functions from Section 3.4. We start by recalling the definition of a smooth function.

**Definition 69.** A function $f : \mathbb{R}^d \to \mathbb{R}$ is said to be $\alpha$-smooth if it is differentiable and its gradient is Lipschitz continuous with a Lipschitz constant $\alpha$, namely

$$\forall x, y \in \mathbb{R}^d, \quad \|\nabla f(x) - \nabla f(y)\| \leq \alpha \|x - y\|.$$  

An $\alpha$-smooth function possesses the following useful upper bounding property: for $x, y \in \mathbb{R}^n$:

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\alpha}{2} \|y - x\|^2. \quad (3.9)$$

#### 3.5.1 Proof of Lemma 68

Let $h$ and $f$ be the convex functions satisfying the statement of the lemma, i.e., $\|h - f\|_\infty \leq \varepsilon$ and $f$ is $\alpha$-smooth. Recall that the smoothed function $h_r$ is defined as $h_r(x) = \mathbb{E}h(x + rU)$, where $U$ is a random vector uniformly distributed on the unit ball $B(0, 1)$.

The first observation is that since $h$ is close to $f$ and the latter is smooth, their (sub)gradients are close to each other; the same also holds between $h_r$ and $f$.

**Lemma 70.** For every $x$:

1. $\|\partial h(x) - \nabla f(x)\| \leq 2\sqrt{\alpha\varepsilon}$ for every subgradient $\partial h(x)$

2. $\|\nabla h_r(x) - \nabla f(x)\| \leq 2\sqrt{\alpha\varepsilon} + \alpha r$.

**Proof.** To prove the first item, consider any point $y \in \mathbb{R}^d$. Using $\alpha$-smoothness of $f$,
\[ \| h - f \|_{\infty} \leq \varepsilon, \] and convexity of \( h \), we have

\[
f(x) + \langle \nabla f(x), y - x \rangle + \frac{\alpha}{2} \| x - y \|^2 \geq f(y) \geq h(y) - \varepsilon
\]

\[
\geq h(x) + \langle \nabla h(x), y - x \rangle - \varepsilon,
\]

and so

\[
\langle \nabla h(x) - \nabla f(x), y - x \rangle \leq f(x) - h(x) + \varepsilon + \frac{\alpha}{2} \| x - y \|^2
\]

\[
\leq 2\varepsilon + \frac{\alpha}{2} \| x - y \|^2.
\]

Then setting \( y \) so that \( y - x = \frac{1}{\alpha} (\nabla h(x) - \nabla f(x)) \) gives

\[
\frac{1}{2\alpha} \| \nabla h(x) - \nabla f(x) \|^2 \leq 2\varepsilon,
\]

which gives the first item of the lemma.

For the second item, again let \( U \) be uniformly distributed in \( B(0, 1) \). This random variable is sufficiently regular that gradients and expectations commute, namely \( \nabla h_r(x) = \nabla \langle E h(x + rU) \rangle = E \partial h(x + rU) \), were \( \partial h \) denotes any subgradient of \( h \) [14]. Then applying Jensen’s inequality we get

\[
\| \nabla h_r(x) - \nabla f(x) \| = \| E \partial h(x + rU) - \nabla f(x) \|
\]

\[
\leq E \| \partial h(x + rU) - \nabla f(x) \|.
\]

Also, for any unit-norm vector \( u \) we have

\[
\| \partial h(x + ru) - \nabla f(x) \| \leq \| \partial h(x + ru) - \nabla f(x + ru) \|
\]

\[
+ \| \nabla f(x + ru) - \nabla f(x) \|
\]

\[
\leq 2\sqrt{\alpha \varepsilon} + \alpha r,
\]

where the last inequality follows from Item 1 of the lemma and \( \alpha \)-smoothness of \( f \) (which is equivalent to \( \| \nabla f(z) - \nabla f(z') \| \leq \alpha \| z - z' \| \) [86]). This concludes the proof. \( \square \)
The second element that we will need is a bound on the total variation between the uniform distributions on the two same-radius balls with different centers (see for example Lemma 8 of \cite{110} for a proof). We state it more conveniently in the language of coupling.

**Lemma 71.** Let \( X \in \mathbb{R}^d \) be the uniformly distributed on \( B(x, r) \) and \( Y \in \mathbb{R}^d \) be uniformly distributed on \( B(y, r) \). Then there is a random variable \((X', Y') \in \mathbb{R}^{2d}\) where \( X' \) has the same distribution as \( X \) and \( Y' \) the same distribution as \( Y \), and where \( \Pr(X' \neq Y') \leq \frac{\|x - y\|\sqrt{d}}{r} \).

We are now ready to prove Lemma 68.

**Proof of Lemma 68.** Item 1: We prove that \( \|\nabla h_r(x) - \nabla h_r(y)\| \leq \left( \frac{4\sqrt{\alpha \varepsilon}}{r} + 3\alpha \sqrt{d} \right)\|x - y\| \) for all \( x, y \). In fact, it suffices to prove this for \( x, y \) where \( \|x - y\| \leq r \), since the inequality can then be chained to obtain the result for any pair of points.

Then fix \( x, y \) with \( \|x - y\| \leq r \). Using the notation from Lemma 71, \( \nabla h_r(x) = \mathbb{E} \partial h(X') \) and \( \nabla h_r(y) = \mathbb{E} \partial h(Y') \) and \( \Pr(X' \neq Y') \leq \frac{\|x - y\|\sqrt{d}}{r} \). Applying Jensen’s inequality,

\[
\|\nabla h_r(x) - \nabla h_r(y)\| \leq \frac{\|x - y\|\sqrt{d}}{r} \cdot \max_{x', y' \in B(x, r) \cup B(y, r)} \|\partial h(x') - \partial h(y')\|
\]

We upper bound the last term by applying triangle inequality and then Lemma 70:

\[
\|\partial h(x') - \nabla h(y')\| \leq \|\partial h(x') - \nabla f(x')\| + \|\nabla f(x') - \nabla f(y')\| + \|\nabla f(y') - \partial h(y')\| \leq 4\sqrt{\alpha \varepsilon} + \alpha \|x' - y'\| \leq 4\sqrt{\alpha \varepsilon} + 3\alpha r;
\]

where the second inequality uses that \( f \) is \( \alpha \)-smooth, and the last inequality uses the
assumption \( \|x - y\| \leq r \). Plugging this into the previous inequality gives

\[
\|\nabla h_r(x) - \nabla h_r(y)\| \leq \left( \frac{4\sqrt{\alpha d}}{r} + 3\alpha \sqrt{d} \right) \cdot \|x - y\|
\]

as desired.

Second item: We now show that \( \|h_r - f\|_\infty \leq \varepsilon + \frac{\alpha r^2}{2} \). Fix \( x \in \mathbb{R}^d \), and again let \( U \) be uniformly distributed in the unit ball. Using the assumption \( \|h - f\|_\infty \leq \varepsilon \) and convexity of \( f \), we have

\[
h(x + rU) \geq f(x + rU) - \varepsilon \geq f(x) + \langle \nabla f(x), rU \rangle - \varepsilon.
\]

Since \( U \) has mean zero, taking expectations gives \( h_r(x) \geq f(x) - \varepsilon \). Similarly, since \( f \) is \( \alpha \)-smooth

\[
h(x + rU) \leq f(x + rU) + \varepsilon \leq f(x) + \langle \nabla f(x), rU \rangle + \frac{\alpha}{2} \|rU\|^2 + \varepsilon,
\]

and taking expectations gives \( h_r(x) \leq f(x) + \frac{\alpha r^2}{2} + \varepsilon \). Together, these yield \( |h_r(x) - f(x)| \leq \varepsilon + \frac{\alpha r^2}{2} \), thus proving the result. This concludes the proof of the theorem.

3.5.2 Proof of Theorem 67

Throughout this section, fix an \( \alpha \)-smooth \( M \)-Lipschitz function \( f : B(R) \to \mathbb{R} \). Recall that we have a sequence of queried points \( x_1, \ldots, x_T \in B(R) \) and access to an \( \eta \)-approximate first-order oracle for \( f \). Our goal is to produce, in an online fashion, a sequence of first-order pairs \((\hat{f}_1, \hat{g}_1), \ldots, (\hat{f}_T, \hat{g}_T)\) for the queried points and a function \( S \) that is smooth, Lipschitz, and compatible with these first-order pairs (i.e., \( S(x_t) = \hat{f}_t \) and \( \nabla S(x_t) = \hat{g}_t \)).

As mentioned, in each iteration \( t \) we will keep a piecewise linear function \( F_t \) and their smoothened version \( (F_t)_r \) (by using the randomized smoothing from the previous section for a specific value of \( r \)). Our transfer procedure then returns the first-order information \( \hat{f}_t := (F_t)_r(x_t) \) and \( \hat{g}_t := \nabla (F_t)_r(x_t) \) of the latter. The final smooth function

109
$S : B(R) \rightarrow \mathbb{R}$ compatible with the first-order information output by the procedure will be given, as in Lemma 64, by using the maximum between the final $F_T$ and a shifted version of the original function $f$. Also recall that in order to ensure the compatibility of $S$ with the first-order information $(\hat{f}_t, \hat{g}_t)$ output throughout the process, we need to “protect” the points $x_t$ and ensure that the function values and gradients at these points did not change across iterations, i.e. $(F_T)_r(x_t) = (F_t)_r(x_t)$ and $\nabla(F_T)_r(x_t) = \nabla(F_t)_r(x_t)$. Since $(F'_r)(x_t)$ depends on the values of $F'_r$ at the ball $B(x_t, r)$ around $x_t$, we need to “protect” $F'_r$ on these balls, namely to have $F_T(x) = F_t(x)$ for all $x \in B(x_t, r)$.

For convenience, we recall the exact construction of the functions $F_t$, the first-order information returned, and the final extension $S$. In hindsight, set $r := \sqrt{\eta/\alpha}$, and for every $t$ define the shift $s_t := 4\eta t + \alpha r^2 t$.

**Procedure 7.**

For each $t = 1, \ldots, T$:

1. Query the $\eta$-approximate oracle for $f$ at $x_t$, receiving the first-order pair $(\hat{f}_t, \hat{g}_t)$
2. Define the function $F_t$ by setting $F_t(x) = \max\{F_{t-1}(x), \hat{f}_t + \langle \hat{g}_t, x - x_t \rangle - (s_t + 2\eta)\}$ for all $x$
3. Output the first-order information of the randomly smoothed function $(F_T)_r$: $\hat{f}_t := (F_T)_r(x_t)$ and $\hat{g}_t := \nabla(F_T)_r(x_t)$

Define the function $S : B(R) \rightarrow \mathbb{R}$ by $S = (\max\{F_T, f - s_{T+1}\})_r$, where max denotes pointwise maximum.

We now prove the main properties of the functions $F_t$, formulated in the following lemma. The first two are similar to (3.6) and (3.7) used in our non-smooth transfer result and guarantee, loosely speaking, that $F_t$ is close to the original function $f$. The third property is precisely the “ball protection” idea discussed above.

**Lemma 72.** For all $t$, the function $F_t$ satisfies the following:

1. $F_t(x) \leq f(x)$ for every $x \in B(R)$
2. For every $t' \leq t$, we have $F_t(x) \geq f(x) - s_{t+1}$ for all $x \in B(x_t, \sqrt{2}r)$.

3. For every $t' \leq t$, we have $F_t(x) = F_{t'}(x)$ for every $x \in B(x_t, \sqrt{2}r)$. In particular $(F_t)_r(x_{t'}) = (F_{t'})_r(x_{t'})$ and $\nabla (F_t)_r(x_{t'}) = \nabla (F_{t'})_r(x_{t'})$.

Proof. We prove these properties by induction on $t$.

First item: Since the property holds by induction for $F_{t-1}$ and $F_t(x) = \max \{ F_{t-1}(x), \tilde{f}_t + \langle \tilde{g}_t, x - x_t \rangle - (s_t + 2\eta) \}$, it suffices to show that

$$\tilde{f}_t + \langle \tilde{g}_t, x - x_t \rangle - (s_t + 2\eta) \leq f(x)$$

(3.10)

for all $x \in B(R)$. For that, since $(\tilde{f}_t, \tilde{g}_t)$ comes from an $\eta$-approximate first-order oracle, by definition $|\tilde{f}_t - f(x_t)| \leq \eta$ and $\|\tilde{g}_t - \nabla f(x_t)\| \leq \frac{\eta}{2R}$; in particular, $|\langle \tilde{g}_t - \nabla f(x_t), x - x_t \rangle| \leq \|\tilde{g}_t - \nabla f(x_t)\||x - x_t| \leq \eta$ for every $x \in B(R)$ (since also $x_t \in B(R)$, by assumption). Then using convexity of $f$ we get

$$f(x) \geq f(x_t) + \langle \nabla f(x_t), x - x_t \rangle$$

$$\geq \tilde{f}_t + \langle \tilde{g}_t, x - x_t \rangle - 2\eta,$$

(3.11)

which implies (3.10) as desired.

Second item: Again since this property holds by induction for $F_{t-1}$, it suffices to show

$$\tilde{f}_t + \langle \tilde{g}_t, x - x_t \rangle - (s_t + 2\eta) \geq f(x) - s_{t+1}$$

(3.12)

for all $x \in B(x_t, \sqrt{2}r)$. Since $f$ is $\alpha$-smooth, for every such $x$ we have

$$f(x) \leq f(x_t) + \langle \nabla f(x_t), x - x_t \rangle + \frac{\alpha}{2} \|x - x_t\|^2$$

$$\leq \tilde{f}_t + \langle \tilde{g}_t, x - x_t \rangle + 2\eta + \alpha r^2.$$  

(3.13)

Since $s_{t+1} = s_t + 4\eta + \alpha r^2$, reorganizing the terms gives (3.12) as desired.
Third item: To show that for every \( t' \leq t \), we have \( F_t(x) = F_{t'}(x) \) for every \( x \in B(x_t, \sqrt{2}r) \), it suffices to show that for every \( t' < t \)

\[
\tilde{f}_t + \langle \tilde{g}_t, x - x_t \rangle - (s_t + 2\eta) \leq F_{t-1}(x) \tag{3.14}
\]

for all \( x \in B(x_t, \sqrt{2}r) \). For that, first notice that for all \( t' < t \) we have \( F_{t-1} \geq F_{t'} \), and the latter can be lower bounded by the affine term added during iteration \( t' \). Combining this with (3.13), applied to iteration \( t' \), we get for all \( x \in B(x_t, \sqrt{2}r) \)

\[
F_{t-1}(x) \geq \tilde{f}_{t'} + \langle \tilde{g}_{t'}, x - x_{t'} \rangle - (s_{t'} + 2\eta) \\
\geq f(x) - (s_{t'} + 4\eta + \alpha r^2) \\
\geq \tilde{f}_t + \langle \tilde{g}_t, x - x_t \rangle - (s_{t'} + 6\eta + \alpha r^2),
\]

where the last inequality uses (3.11). Since \( s_t \geq s_{t'} + 4\eta + \alpha r^2 \), this implies (3.14) as desired.

To conclude the proof of this item, notice that \((F_t)_r(x_{t'})\) (respectively \((F_{t'})_r(x_{t'})\)) only depends on the values of \( F_t \) (resp. \( F_{t'} \)) on the ball \( B(x_{t'}, r) \). Since we just showed the value of \( F_t \) and \( F_{t'} \) agree on this ball, we get \((F_t)_r(x_{t'}) = (F_{t'})_r(x_{t'})\). Similarly, the gradient \( \nabla (F_t)_r(x_{t'}) \) only depends on the values of \( F_t \) on an arbitrarily small open neighborhood of the ball \( B(x_{t'}, r) \), and the same holds for \( \nabla (F_{t'})_r(x_{t'}) \). Since the bigger ball \( B(x_{t'}, \sqrt{2}r) \) contains such a neighborhood, we again obtain \( \nabla (F_t)_r(x_{t'}) = \nabla (F_{t'})_r(x_{t'}) \).

This concludes the proof of the lemma.

We are now ready to prove Theorem 67.

**Proof of Theorem 67.** We need to prove that the function \( S : B(R) \to \mathbb{R} \) defined in Procedure 7 satisfies:

1. \( \|S - f\|_\infty \leq s_{T+1} + \frac{\alpha r^2}{2} \).

2. \( S \) is \( \left( \frac{4\sqrt{\alpha d} + s_{T+1}}{r} + 3\alpha \sqrt{d} \right) \)-smooth

3. \( S \) is \( (M + \frac{\eta}{2R}) \)-Lipschitz
4. $S$ is an extension for the first-order pairs $(\hat{f}_t, \hat{g}_t)$ output by the procedure

First item: Define the function $\bar{S} := \max\{F_t(x), f(x) - s_{T+1}\}$, so $S = \bar{S}_t$. Using Item 1 of Lemma 72, we see that $\bar{S}(x) \leq f(x)$ for all $x \in B(R)$, and by definition we have $\bar{S}(x) \geq f(x) - s_{T+1}$, thus $\|\bar{S} - f\|_\infty \leq s_{T+1}$. Then using Item 2 of Lemma 68 we get $\|S - f\|_\infty \leq s_{T+1} + \frac{\eta^2}{2\alpha}$.

Second item: This follows directly from the above fact that $\|\bar{S} - f\|_\infty \leq s_{T+1}$ and the guarantees for randomized smoothing given in Lemma 68.

Third item: The subgradients of $F_T$ are (a convex combination of a subset of the) vectors $\hat{g}_t$, and so $F_T$ is $(\max_t \|\hat{g}_t\|)$-Lipschitz. Since the vectors came from an $\eta$-approximate oracle for $f$, we have $\|\hat{g}_t - \nabla f(x_t)\| \leq \frac{\eta}{2R}$, and since $f$ is $M$-Lipschitz we get $\|\hat{g}_t\| \leq M + \frac{\eta}{2R}$; it follows that $F_T$ is $(M + \frac{\eta}{2R})$-Lipschitz. Next, the subgradients of $\bar{S}$ come either from subgradients of $F_T$ or gradients of $f$ (or a convex combination thereof), and so $\bar{S}$ is $\max\{M + \frac{\eta}{2R}, M\} = M + \frac{\eta}{2R}$ Lipschitz. Finally, for every $x$ we have ($U$ being uniformly distributed in the unit ball again)

$$\|\nabla S(x)\| = \|E \partial \bar{S}(x + rU)\| \leq E \|\partial \bar{S}(x + rU)\| \leq M + \frac{\eta}{2R},$$

where $\partial \bar{S}(x + rU)$ denotes any subgradient at $x + rU$ and the first inequality follows from Jensen’s inequality. This proves that $S$ is $(M + \frac{\eta}{2R})$-Lipschitz.

Fourth item: We need to show that for all $t$, $\hat{f}_t = S(x_t)$ and $\hat{g}_t = \nabla S(x_t)$. By definition, $\hat{f}_t = (F_t)_r(x_t)$ and $\hat{g}_t = \nabla (F_t)_r(x_t)$. Moreover, by Item 3 of Lemma 72, using $F_T$ instead of $F_t$ gives the same quantities, namely $\hat{f}_t = (F_T)_r(x_t)$ and $\hat{g}_t = \nabla (F_T)_r(x_t)$. We claim that for every $t$, $F_T$ and $\bar{S}$ are equal inside the ball $B(x_t, \sqrt{2}r)$, which then implies that $\hat{f}_t = (F_T)_r(x_t) = \bar{S}_r(x_t) = S(x_t)$ and $\hat{g}_t = \nabla (F_T)_r(x_t) = \nabla \bar{S}_r(x_t) = \nabla S(x_t)$, as desired. To show the equality in the ball $B(x_t, \sqrt{2}r)$, it suffices that the other term in the max defining $\bar{S}$ does not “cut off” $F_T$, namely that $f(x) - s_{T+1} \leq F_T(x)$ for every $x \in B(x_t, \sqrt{2}r)$. But this follows from Item 2 of Lemma 72.

Substituting the value $r = \sqrt{\eta/\alpha}$ and $s_t = 4\eta t + \alpha r^2 t$ in the items above concludes the proof of Theorem 67.
3.6 Proof of Lemma 64

Lemma 64: Consider a sequence of points $x_1, \ldots, x_T \in B(R)$, and a sequence of first-order pairs $(\hat{f}_1, \hat{g}_1), \ldots, (\hat{f}_T, \hat{g}_T)$. Consider $\delta > 0$ and $M' \geq M$, and suppose that there is an $M'$-Lipschitz convex extension $F$ of these first-order pairs that satisfies:

\[
\begin{align*}
  f(x) &\geq F(x) - \delta, \quad \forall x \in B(R) \quad (3.15) \\
  f(x_t) &\leq F(x_t) + \delta, \quad \forall t = \{1, \ldots, T\}. \quad (3.16)
\end{align*}
\]

Then the first-order pairs have an $M'$-Lipschitz convex extension $F'$ such that $\|F' - f\|_\infty \leq \delta$.

Proof. Define the function $F'$ as $F'(x) := \max\{f(x) - \delta, F(x)\}$ by taking the maximum between $F$ and a downward-shifted $f$. We will show that this function is the desired convex extension of the first-order pairs $(\hat{f}_1, \hat{g}_1), \ldots, (\hat{f}_T, \hat{g}_T)$.

First, to show $\|F' - f\|_\infty \leq \delta$, by the definition of $F'$ one has $F'(x) \geq f(x) - \delta$ for all $x$. Furthermore, because of the guarantee that $F(x) \leq f(x) + \delta$, we also have $F'(x) \leq f(x) + \delta$ for all $x$; together these imply that $\|F' - f\|_\infty \leq \delta$. Since $f$ and $F$ are $M'$-Lipschitz convex functions, so is $F'$.

It remains to be shown that $F'$ is an extension of the first-order pairs, that is, to show $\hat{f}_t = F'(x_t)$ and $\hat{g}_t \in \partial F'(x_t)$ for all $t = 1, \ldots, T$. Given property (3.16), we have $F(x_t) \geq f(x_t) - \delta$, and so $F'(x_t) = F(x_t) = \hat{f}_t$. The fact that $F'(x_t) = F(x_t)$ also implies that every vector in $\partial F(x_t)$ is a subgradient of $F'$ at $x_t$, namely $\partial F'(x_t) \supseteq \partial F(x_t) \ni \hat{g}_t$. To see this, recall that since $F$ is convex, for $\hat{g}_t \in \partial F(x_t)$ we have $F(x_t) + \langle \hat{g}_t, x_t - x \rangle \leq F(x)$. Using the fact that $F'(x_t) = F(x_t)$, we thus have $F'(x_t) + \langle \hat{g}_t, x_t - x \rangle \leq F'(x) \leq F'(x)$ for all $x$, and so any $\hat{g}_t \in \partial F(x_t)$ is also a subgradient for $F'$ at $x_t$, as desired to conclude the proof. □
3.7 Separation oracles: proofs of Theorems 59 and 60

We now consider the original constrained problem \( \min \{ f(x) : x \in C \cap X \} \), and show how to run any first-order algorithm \( A \) using only approximate first-order information about \( f \) and approximate separation information from \( C \), proving Theorems 59 and 60. The main additional element is to convert the approximate separation information for \( C \) into an exact information for a related set \( K \approx C \) so it can be used in a black-box fashion by \( A \), as the previous section did for the first-order information of \( f \). For simplicity, we assume throughout that the algorithm \( A \) only queries points in \( B(R) \) (the ball containing the feasible set \( C \)), since points outside it can be separated exactly.

Given a set of points \( x_1, ..., x_T \in B(R) \), we say a sequence of separation responses \( r_1, ..., r_T \in \{ \text{Feasible, Infeasible} \} \times \mathbb{R}^d \) has a convex extension if there is a convex set \( K \neq \emptyset \) such that there exists an exact (i.e., 0-approximate) separation oracle for \( K \) giving responses \( r_1, ..., r_T \) for the query points \( x_1, ..., x_T \). We will also refer to such responses as consistent with \( K \). As in the previous section, responses from an \( \eta \)-approximate separation oracle may not by themselves admit a convex extension, and need to be modified in order to allow a consistent, convex extension; for example, approximate separating hyperplanes may not be consistent with a convex set, or may ”cut off” points previously reported as feasible. When we say a point \( y \) is cut off by a separating hyperplane through \( x \) with normal vector \( g \), we mean that \( \langle g, y \rangle > \langle g, x \rangle \), i.e. that \( y \) is not in the induced halfspace. Note that when given an exact separating hyperplane for some \( x \notin C \), no point in \( C \) is cut off by it, whereas approximate separating hyperplanes have no such guarantee. With this in mind, we now give a theorem serving as a feasibility analogue to Theorem 63.

**Definition 73.** For any convex set \( C \subseteq \mathbb{R}^d \) and any \( \delta > 0 \), we define \( C_{-\delta} := \{ x \in C : B(x, \delta) \subseteq C \} \), which will be called \( \delta \)-deep points of \( C \).

**Theorem 74** (Online Convex Extensibility). Consider a convex set \( C \subseteq B(R) \) and a sequence of points \( x_1, ..., x_T \in B(R) \). There is an online procedure that, given access to an \( \eta \)-approximate separation oracle for \( C \), produces separation responses \( \hat{r}_1, ..., \hat{r}_T \) that have a convex extension \( K \) satisfying \( C_{-\eta} \subseteq K \subseteq C \). Moreover, the procedure only
probes the approximate oracle at the points $x_1, \ldots, x_T$.

Note that the guarantee $C_{-\eta} \subseteq K \subseteq C$ means that for any point $x_t$ that is $\eta_C$-deep in $C$, i.e., in $C_{-\eta}$, the response $\hat{r}_t$ produced says FEASIBLE, whereas for any $x_t \notin C$ it says INFEASIBLE and gives a hyperplane separating $x_t$ from $K$ (which cannot cut too deep into $C$, i.e., it contains $C_{-\eta}$). At a high-level, such responses allow one to cut off infeasible solutions, but guarantee that there are still ($\eta_C$-deep) solutions with small $f$-value available.

With this additional procedure at hand, we extend Procedure 5 from the main text in the following way to solve constrained optimization: in each step of the procedure, we also send the point $x_t$ queried by the algorithm $\mathcal{A}$ to the $\eta_C$-approximate separation oracle for $C$, receive the response $\hat{r}_t$, pass it through Theorem 74 to obtain the new response $\hat{r}_t$, and send the latter back to $\mathcal{A}$. We call this procedure **Approximate-to-Exact-Constr**, and formally state it as follows:

**Procedure 8. Approximate-to-Exact-Constr($\mathcal{A}, T$)**

For each timestep $t = 1, \ldots, T$:

1. Receive query point $x_t \in B(R)$ from $\mathcal{A}$

2. Send point $x_t$ to the $\eta_f$-approximate first-order oracle and to the $\eta_C$-approximate separation oracle, and receive the approximate first-order information $(\hat{f}_t, \hat{g}_t) \in \mathbb{R} \times \mathbb{R}^d$, and separation response $\hat{r}_t \in \{\text{FEASIBLE}\} \cup \{\text{INFEASIBLE}\} \times \mathbb{R}^d$.

3. Use the online procedures from Theorems 63 and 74 to construct the first-order pair $(\hat{f}_t, \hat{g}_t)$ and separation response $\hat{r}_t$.

4. Send $(\hat{f}_t, \hat{g}_t), \hat{r}_t$ to the algorithm $\mathcal{A}$.

Return the point returned by $\mathcal{A}$.

The proof that this procedure yields Theorem 59 is analogous to the one for the unconstrained case of Theorem 55, so we only sketch it to avoid repetition.
Proof sketch of Theorem 59. Let $F$ and $K$ be the Lipschitz and convex extensions to the answers sent to $A$ that are guaranteed by Theorems 63 and 74, respectively. Approximate-to-Exact-Constr has the same effect as $A$ running on the instance $(F, K)$. One can show that this instance belongs to $I(M + \frac{\eta_f}{2R}, R, \rho - \eta_C)$. Then if $err(\cdot)$ is the error guarantee of $A$ as in the statement of the theorem, this ensures that we return a solution $\bar{x} \in K \cap X$ satisfying

$$F(\bar{x}) \leq \text{OPT}(F, K) + err(T, M + \frac{\eta_f}{2R}, R, \rho - \eta_C),$$

where $\text{OPT}(F, K) := \arg\min\{F(x) : x \in K \cap X\}$. Since $C - \eta_C$ contains a solution $x$ with value $f(x) \leq \text{OPT}(f, C) + \frac{2MR_{\text{lin}}}{\rho}$ (e.g., Lemma 4.7 of [8]), we have $\text{OPT}(f, K) \leq \text{OPT}(f, C) + \frac{2MR_{\text{lin}}}{\rho}$. Finally, using the guarantee $\|F - f\|_{\infty} \leq 2\eta_f T$, we obtain that $f(\bar{x}) \leq \text{OPT}(f, C) + err(T, M + \frac{\eta_f}{2R}, R, \rho - \eta_C) + 4\eta_f T + \frac{2MR_{\text{lin}}}{\rho}$, concluding the proof of the theorem.

The proof of Theorem 60 follows effectively the same reasoning as for Theorem 59; we also sketch it here. The main difference is that one needs to ensure the optimal solution $x^*$ of (3.2) is contained in contained in the auxiliary feasible region $K$ the algorithm $A$ uses; otherwise the additional restrictions imposed by $Z$ may lead to arbitrarily bad solutions, or even $K \cap Z$ being empty (consider for example the case of $Z$ being a singleton on the boundary of $C$ that is then cut off by an approximate separation response). However, since $K$ is guaranteed to contain $C - \eta_C$ and we assume that $\eta_C \leq \rho$, the fact that $x^*$ is $\rho$-deep in $C$ implies that it is also in $K$.

Proof sketch of Theorem 60. Again, let $F$ and $K$ be the Lipschitz and convex extensions as in the previous proof, so that the instance $(F, K)$ belongs to $I(M + \frac{\eta_f}{2R}, R, \rho - \eta_C)$ and $A$ returns a solution $\bar{x} \in K \cap Z$ satisfying $F(\bar{x}) \leq \text{OPT}(F, K) + err(T, M + \frac{\eta_f}{2R}, R, \rho - \eta_C)$, where $\text{OPT}(F, K) := \arg\min\{F(x) : x \in K \cap Z\}$. Recall that $Z$ is assumed to be given and known by the algorithm. Since the optimal solution for $(f, C)$ is assumed to be in $C - \rho$, and $K$ contains $C - \eta_C \supseteq C - \rho$, $K$ contains the optimal solution to to the true instance, $x^* : f(x^*) = \text{OPT}(f, C)$. Finally, using the guarantee $\|F - f\|_{\infty} \leq 2\eta_f T$, we obtain that $f(\bar{x}^*) \leq \text{OPT}(f, C) + err(T, M + \frac{\eta_f}{2R}, R, \rho - \eta_C) + 4\eta_f T$, concluding the proof.
of the theorem.

3.7.1 Computing convex-extensible separation responses

We now prove Theorem 74. The result requires the existence of a convex extension $K$ for the responses $\hat{r}_t$ that we construct, and we need $C_{-\eta} \subseteq K \subseteq C$. We provide a procedure that produces the responses $\hat{r}_1, \ldots, \hat{r}_T$ together with sets $K_1, \ldots, K_T$ so that $K_t \cap C$ is consistent with the responses up to this round, i.e., $\hat{r}_1, \ldots, \hat{r}_t$, and is sandwiched between $C_{-\eta}$ and $C$. The set $K_t$ will consist of all the points that were not excluded by the separating hyperplanes of the responses up to this round. Thus, our main task is to ensure that as $K_t$ evolves, it does not exclude the points $x_t$ that the responses up to now have reported as Feasible (ensuring consistency with previous responses). We also want to ensure that none of the deep points $C_{-\eta}$ is excluded.

Before stating the formal procedure, we give some intuition on how this is accomplished. Suppose one has $K_{t-1}$ satisfying the desired properties. One receives a new point $x_t$ and separation response $\tilde{r}_t$ from the approximate oracle, and we need to construct a response $\hat{r}_t$ and an updated set $K_t$ to maintain the desired properties.

Suppose $\tilde{r}_t$ reports that $x_t$ is Feasible. Our procedure ignores this information, keeps $K_t = K_{t-1}$ and creates a response $\hat{r}_t$ that is Feasible if and only if $x_t \in K_t = K_{t-1}$ (also sending a hyperplane separating $x_t$ from $K_t$ if $x_t \notin K_t$; notice that since this hyperplane does not cut into $K_{t-1}$, we do not need to update this set). Notice that $K_t \cap C$ is indeed consistent with the response $\hat{r}_t$.

The interesting case is when $\tilde{r}_t$ reports that $x_t$ is Infeasible (so $x_t \notin C$) but $x_t \in K_{t-1}$. Thus, $x_t$ cannot belong to $K_t \cap C$ (recall we will construct $K_t \subseteq K_{t-1}$), and so to ensure consistency our response $\hat{r}_t$ needs to report Infeasible and a separating hyperplane that excludes $x_t$. The first idea is to simply use separating hyperplane reported by the approximate oracle. But this can exclude points that were deemed Feasible by our previous responses $\hat{r}_t$ (we call these points $\overline{\text{Feas}}_{t-1}$), which would violate consistency. Thus, we first rotate this hyperplane as little as possible such that it contains all points in $\overline{\text{Feas}}_{t-1}$, and report this rotated hyperplane $H$ (adding it to
$K_{t-1}$ to obtain $K_t$). While this rotation protects the points $\overline{\text{FEAS}}_{t-1}$, we also need to argue that it does not stray too much away from the original approximate separating hyperplane so as to not cut into $C_{-\eta}$.

We now describe the formal procedure in detail. We use $H(g, \bar{x}) := \{x : \langle g, x \rangle \leq \langle g, \bar{x} \rangle \}$ to denote the halfspace with normal $g$ passing through the point $\bar{x}$.

**Procedure 9.**

Initialize $K_0 = \mathbb{R}^d$ and $\overline{\text{FEAS}}_0 = \emptyset$. For each $t = 1, \ldots, T$:

1. Query the $\eta$-approximate feasibility oracle for $C$ at $x_t$, receiving the response $(\text{flag}_t, \tilde{g}_t)$

2. If $\text{flag}_t = \text{Feasible}$ and $x_t \in K_{t-1}$. Define the response $\hat{r}_t = (\text{Feasible}, \star)$, and set $K_t = K_{t-1}$ and $\overline{\text{FEAS}}_t = \overline{\text{FEAS}}_{t-1} \cup x_t$.

3. ElseIf $\text{flag}_t = \text{Feasible}$ but $x_t \notin K_{t-1}$. Set $\hat{g}_t$ be any unit vector such that the halfspace $H(\hat{g}_t, x_t)$ contains $K_{t-1}$. Define the response $\hat{r}_t = (\text{Infeasible}, \hat{g}_t)$. Set $K_t = K_{t-1}$ and $\overline{\text{FEAS}}_t = \overline{\text{FEAS}}_{t-1}$.

4. Else (so $\text{flag}_t = \text{Infeasible}$). Let $\hat{g}_t$ be a unit vector such that the induced halfspace $H(\hat{g}_t, x_t)$ contains $\overline{\text{FEAS}}_{t-1}$ that is the closest to $\tilde{g}_t$ with this property, i.e.

\[
\|\hat{g}_t - \tilde{g}_t\|_2 = \min_{g \in B(1)} \left\{ \|g - \tilde{g}_t\|_2 : H(g, x_t) \supseteq \overline{\text{FEAS}}_{t-1} \right\}.
\]

Define the response $\hat{r}_t = (\text{Infeasible}, \hat{g}_t)$. Set $K_t = K_{t-1} \cap H(\hat{g}_t, x_t)$ and $\overline{\text{FEAS}}_t = \overline{\text{FEAS}}_{t-1}$.

We remark that in Line 4, there indeed exists a halfspace supported at $x_t$ that contains all points in $\overline{\text{FEAS}}_{t-1}$: in this case $x_t \notin C$ (since $\text{flag}_t = \text{Infeasible}$) and by definition $\overline{\text{FEAS}}_{t-1} \subseteq C$, so any halfspace separating $x_t$ from $C$ will do.

**Lemma 75.** For every $t = 1, \ldots, T$, the set $K_t \cap C$, with $K_t$ computed by Procedure 9, is
a convex extension to the responses \( \hat{r}_1, \ldots, \hat{r}_t \). Moreover, all these sets satisfy \( K_t \supseteq C - \eta \).

**Proof.** It suffices to prove this for each iteration, so suppose \( K_{t-1} \) with responses \( \hat{r}_1, \ldots, \hat{r}_{t-1} \) satisfy the lemma. If Lines 2 or 3 of the procedure were executed, it is straightforward to see \( K_t \) satisfies the lemma. If Line 4 executes, \( x_t \notin C \), and so the response \( \hat{r}_t \) is consistent with \( K_t \cap C \) since \( K_t = K_{t-1} \cap H(\hat{g}_t, x_t) \). As \( \text{FAS}_{t-1} \subseteq K_t \) by construction and \( K_t \subseteq K_{t-1} \), \( K_t \cap C \) is consistent with all responses \( \hat{r}_1, \ldots, \hat{r}_t \) made. It remains to show that \( K_t \) contains \( C - \eta \), for which showing \( H(\hat{g}_t, x_t) \) contains it suffices. Notice that since there exists an exact halfspace \( H(g, x_t) \) separating \( x_t \notin C \) that is \( \frac{\eta}{4R} \)-close to \( \tilde{g}_t \) (due to the \( \eta \)-approximate oracle), and \( H(g, x_t) \) contains \( C \) and thus \( \text{FAS}_{t-1} \), we have \( \|\hat{g}_t - \tilde{g}_t\|_2 \leq \frac{\eta}{4R} \). The triangle inequality then reveals \( \|\hat{g}_t - g\|_2 \leq \|\hat{g}_t - \tilde{g}_t\|_2 + \|\tilde{g}_t - g\|_2 \leq \frac{\eta}{2R} \), and then it is easy to see that \( H(\hat{g}_t, x_t) \) contains \( C - \eta \), concluding the proof. \( \square \)
Part II

Quantum Algorithms
Chapter 4

Introduction to Quantum Computing

Quantum computing is a promising emerging technology that aims to harness quantum mechanical effects to solve difficult computational problems more efficiently than classical computers. In light of Part I of this dissertation, in which information complexity in classical settings is studied, it is unclear whether those same lower bounds would transfer to the quantum analogues of the settings studied. Namely, the quantum versions of the oracles discussed in Part I would allow making the same queries but with inputs in superposition, and the oracle would return the superposition of the inputs with their corresponding answers (see (4.9) below for how a function is evaluated in the quantum model). These oracles are at least as powerful as their classical counterparts we have studied here, and so quantum computation may allow for faster algorithms than are possible in a classical framework. In a more general sense, many problems of interest are difficult for classical computers to solve – when classical computation fails us, what can we do?

Quantum computation offers a glimmer of hope for a few of those problems. Consider, for example, the prime factorization problem, in which the product of two prime numbers is given, and one must find two prime numbers whose product is the given number. No classical polynomial-time algorithm exists for this problem, and the
problem is not believed to be in complexity class P. However, a polynomial-time quantum
algorithm for the problem exists due to Peter Shor’s highly influential work from 1995
[99]. Notably, the prime factorization problem seems to have no connection whatsoever
to quantum mechanics – it is perhaps not surprising that one can achieve a speedup with
quantum computing to, for example, simulate the evolution of a quantum mechanical
system, since that problem is inherently a quantum problem. Shor’s algorithm serves
as an example to show that quantum computing has the potential to provide speedups
in mathematical problems that have no clear connection to quantum mechanics at all.
Grover’s search algorithm, first described in [52] fits this theme as well – the algorithm
is often described as finding a marked item in a list of \( N \) unsorted items in \( O(\sqrt{N}) \) time,
which also seems unrelated to quantum mechanics. Yet, quantum computing provides an
advantage over classical computing for this simple unstructured search problem. These
two algorithms use gate-based quantum computing, which is a Turing complete model
of computation in which quantum bits, or qubits, are used and quantum effects like
superposition, entanglement, and quantum measurements can be taken advantage of.

Another exciting, non-Turing-complete computational paradigm in quantum com-
puting is quantum annealing. Seminal work by Farhi et al. [42] on adiabatic quantum
computation spurred interest in what is now implemented as a heuristic in commercially
available devices as quantum annealing, in for example the quantum annealers produced
by D-Wave. These promise to produce good solutions to Ising problems or quadratic un-
constrained binary optimization (QUBO) problems quickly. See [109] for a recent survey
on quantum annealing. Ising or QUBO problems are difficult combinatorial optimization
problems – in fact, they are NP-hard, so they take exponential time for classical
computers to solve, assuming the strong exponential time hypothesis. While quantum
computers, whether gate- or annealing-based, are not expected to be able to solve these
in polynomial-time, quantum annealing stands as a promising heuristic to produce good
solutions for these problems extremely quickly – on D-Wave machines, annealing times
to produce one solution are typically microseconds. The produced solutions are not
perfect and are random, and recent work in [106] has shown that current machines can
be viewed as sampling good solutions to the problem. While current annealers do not
provide practical advantage over classical technologies, on the recent D-Wave Advantage system with 5000 qubits an advantage over classical computation in benchmarks for a particular problem has been shown [103]. Hence, it is not unreasonable to expect that as the available quantum annealing technologies continue to evolve, one may see practical advantage to real problems of interest in the future.

While these technologies are still in their infancy, they provide a possible route, at least in some instances, to the question of what can we do in problems where classical computers fail us? In this spirit, Part II of this dissertation joins the quest of developing and studying quantum computational methods to solve difficult optimization problems.

Chapter 5 presents an image denoising model designed to take advantage of quantum annealing by formulating the denoising problem in QUBO form. The chapter also provides a more technical background overview of quantum annealing.

Chapter 6 works in a gate-based, distributed model of quantum computing. Here, distributed algorithms are presented that are asymptotically faster than their classical analogues. However, practical limitation are also discussed by investigating the exact complexities of these algorithms.

4.1 Background

In this section, we provide some background on two quantum computing paradigms. Subsection 4.1.1 will cover the basics of quantum annealing, while subsection 4.1.2 will introduce gate-based quantum computing. While both of these aim to take advantage of quantum mechanical effects to speed up computations, they are remarkably different from one another. In particular, quantum annealing is not Turing complete (i.e., is not universal in the types of logic and computation it can represent), while gate-based quantum computing is Turing complete (and, in fact is necessarily more powerful than classical computing).
4.1.1 Quantum Annealing

Quantum annealing aims to find the ground state of a given Hamiltonian by essentially physically implementing the Ising spin model evolving over time. A good overview of quantum annealing can be found in [64, 32, 2], among others. We will provide a brief overview here. Consider an Ising model with terms for longitudinal and transverse fields \(^1\) as

\[
\mathcal{H}(t) = -\sum_{ij} J_{ij}\sigma_i\sigma_j - \Gamma(t)\sum_i \sigma_i
\]

\[
=: H_0(\sigma) - \Gamma(t)\sum_i \sigma_i,
\]

where \(H_0\) is the Hamiltonian whose ground state we wish to compute, and \(\Gamma(t)\) is a time dependent amplitude for the transverse field, and the \(\sigma_i \in \{-1, +1\}\) represent the spin values of the particles involved in the model, where we write \(\sigma = (\sigma_1, \ldots, \sigma_n)\) as the entire spin configuration. \(J_{ij}\) represents the *interaction* between particle \(i\) and particle \(j\), and so we shall call \(J\) the *interaction matrix*. This \(J\) determining the interactions between the particles essentially defines the problem at hand. Speaking from an optimization point of view, the goal is find the minimizer of \(H_0(\sigma)\). Noting that (4.1) describes the *energy* of a configuration \(\sigma\), this corresponds to the minimum energy state of (4.1) when the transverse field has amplitude zero. Observe that we have

\[
H_0(\sigma) = -\sigma^T J \sigma,
\]

and so one can already observe that finding the ground state of this Hamiltonian is closely related to quadratic binary optimization, other than that here \(\sigma_i \in \{-1, +1\}\).

If the system (4.3) is evolved over time, writing \(\sigma(t)\) as the configuration at time \(t\), the evolution follows the Schrodinger equation

\[
i\frac{\partial \sigma(t)}{\partial t} = \mathcal{H}(t)\sigma(t),
\]

\(^1\)To the reader unfamiliar with physics but familiar with mathematics, simply think of this as defining a time-dependent objective function, where \(\sigma_i\) are the variables and \(J\) is some square matrix.
noting some abuse of notation using $\sigma(t)$ because this is not in $\{-1, +1\}^n$, but instead in a quantum state space (a superposition of states in $\{-1, +1\}^n$). The Adiabatic Theorem of quantum mechanics states that if $\sigma(0)$ is the ground state of $\mathcal{H}(0)$, and $\mathcal{H}(t)$ evolves sufficiently slowly to $\mathcal{H}(T)$ over time, then $\sigma(t)$ will remain in the ground state of $\mathcal{H}(t)$ for $t \in [0, T]$. Taking this for granted, the core idea inspiring quantum annealing is the following. When $\Gamma(t)$ is extremely large in (4.1), the ground state of $\mathcal{H}(t)$ is known – namely, when $\Gamma(t)$ dominates all the $J_{ij}$, the ground state is obtained by simply setting $\sigma = 0$. The second observation to make is that when $\Gamma(t) = 0$, the ground state of $\mathcal{H}(t)$ is exactly the ground state of $H_0$, which is what we aim to find. Then, according to the Adiabatic Theorem, one can do the following:

1. Set $\sigma = 0$, and set $\Gamma(0)$ to be large enough to dominate all the $J_{ij}$, so that $\sigma$ is the ground state of $\mathcal{H}(0)$.

2. Evolve the system (4.1) sufficiently slowly over time $t \in [0, T]$ letting $\Gamma(t) \to 0$, with $\Gamma(T) = 0$.

3. Measure the configuration $\sigma(T)$ after this evolution process.

If this evolution over time is carried out slowly enough, since $\sigma(0)$ is the ground state of $\mathcal{H}(0)$, the Adiabatic Theorem tells us that $\sigma(T)$ is the ground state of $\mathcal{H}(T)$, which is exactly the ground state of $H_0$ since the system was evolved to have $\Gamma(0) = 0$. Hence, we obtain the minimizer to (4.3) through this process, and it becomes clear that the evolution behaviors of such a system governed by (4.1) may be leveraged for certain optimization problems.

In practice, quantum annealers then aim to physically implement a system governed by the model of (4.1), as a system of coupled interacting qubits. The machines are built to be able to determine the interactions $J_{ij}$ between qubits, and such that both the longitudinal ($H_0$) and transverse ($\Gamma(t)$) fields can be controlled. Typically, instead of decreasing the amplitude of the transverse field slowly enough according to the guarantees of the Adiabatic Theorem, one simply decreases $\Gamma(t)$ much faster. Hence, one may view quantum annealing largely as a heuristic algorithm to find approximate
minimum energy configurations for Ising spin models. In fact, recent research suggests that quantum annealers can practically be viewed as \textit{sampling} low-energy configurations for (4.1). In modern annealing machines, such as those produced by D-Wave, annealing times are typically on the order of microseconds, so that one can quickly sample thousands of potential solutions and then select the best one.

To establish a clear link to optimization, consider a quadratic unconstrained binary optimization (QUBO) problem of the form

$$\min_{x \in \{0,1\}} f_Q = \min_{x \in \{0,1\}} x^T Q x = \min_{x \in \{0,1\}} \sum_{i,j} Q_{ij} x_i x_j,$$

for an $n \times n$ matrix $Q$. We note that this problem and minimizing (4.3) are NP-hard. It is a straightforward exercise to show that this problem can be transformed to the Ising form in (4.1). Hence, given some QUBO instance, one can reformulate it as an Ising problem and use quantum annealing to obtain good solutions to it. From a purely mathematical point of view, one might thus consider quantum annealing as a kind of 'black-box' sampler of good solutions to QUBO or Ising problems. Since currently existing quantum annealers only have a very limited number of qubits available, with limited connectivity (ie. limited interactions between them), it is of interest to find meaningful optimization problems that can be efficiently formulated as QUBO or Ising instances to take advantage of quantum annealers as best as possible. A nice collection of various optimization problems formulated as QUBO instances can be found in [46], demonstrating the variety of problems that quantum annealing has the potential to provide good solutions for.

4.1.2 Gate-Based Quantum Computing

To establish the computational model for gate-based quantum computing, we will first introduce some basic quantum mechanics as it relates to quantum computing. We begin by describing the state space of a single quantum bit, or \textit{qubit}, as

$$|\psi\rangle \in \mathcal{Q}$$

(4.5)
where $\mathcal{Q}$ is a complex Hilbert space with orthonormal basis $|0\rangle, |1\rangle$. Note that as convention on the quantum computing literature, we will use Dirac notation for these quantum states: $|\cdot\rangle$ represents an element of a vector space, while $\langle\cdot|$ represents an element of the dual of that space. Those unfamiliar with the notation may simply think of $|\psi\rangle$ as a vector $\psi \in \mathcal{Q}$, and as $\langle\psi|$ as $\psi^* \in \mathcal{Q}^*$. Notably, the vectors $|0\rangle, |1\rangle$ are vectors, that are simply labeled as 0 and 1. These will represent the quantum bit in question taking on the value 0 or 1, as a classical bit in $\{0,1\}$ would, so defining $\mathcal{Q}$ as being generated by the orthonormal basis $|0\rangle, |1\rangle$ is notationally convenient. A qubit in state $|\psi\rangle$ can be measured with respect to some orthonormal basis $|v_1\rangle, |v_2\rangle$, and the result of that measurement will be $|v_1\rangle$ with probability $\langle\psi, v_1|\psi, v_1\rangle^2$, and $|v_2\rangle$ with probability $\langle\psi, v_2|\psi, v_2\rangle$. Since the probabilities of the measurement outcomes must sum to 1, the state of a qubit must be some unit vector in $\mathcal{Q}$. Hence, any operation on the qubit must also be a unitary operation, that is, one that preserves the norm of $|\psi\rangle$.

For a classical bit $b \in \{0,1\}$, notice that when we have $n$ bits, the state space for a system of these $n$ bits becomes

$$\{0,1\} \times \{0,1\} \times \ldots \times \{0,1\} = \{0,1\}^n,$$

where $\times$ denotes the Cartesian product. For a system of $n$ quantum bits, however, the state space grows with the tensor product instead of the Cartesian product. That is, the state of an $n$-qubit system is described by

$$\mathcal{Q} \otimes \mathcal{Q} \otimes \ldots \mathcal{Q} =: \mathcal{Q}^\otimes_n$$

(4.7)

Recall that the tensor product of two vector spaces $V$ and $W$ with orthonormal bases
\( v_1, \ldots, v_{m_1} \) and \( w_1, \ldots, w_{m_2} \), respectively, is the vector space generated by the orthonormal basis

\[
\begin{align*}
&v_1 \otimes w_1, v_1 \otimes w_2, \ldots, v_1 \otimes w_{m_2}, \\
v_2 \otimes w_1, v_2 \otimes w_2, \ldots, v_2 \otimes w_{m_2}, \\
&\cdots \\
v_{m_1} \otimes w_1, v_{m_1} \otimes w_2, \ldots, v_{m_1} \otimes w_{m_2}.
\end{align*}
\]

Notice that this space has dimension \( m_1 \cdot m_2 \). On the other hand, the Cartesian product \( V \times W \) only has dimension \( m_1 + m_2 \). As such, the dimension of the state space \((4.6)\) of a system of \( n \) classical bits is only \( n \), whereas the dimension of the state space \((4.7)\) of a system of \( n \) quantum bits is \( 2^n \). It becomes apparent that the state space of \( n \) qubits is much richer than the state space of \( n \) classical bits. Now, for a system of \( n \) qubits, the state is some \( |\psi\rangle \in \mathbb{C}^{2^n} \). Again, one can measure the system of qubits with respect to some orthonormal basis for \( \mathbb{C}^{2^n} \), and the probability of observing any outcome \( \nu \) (any of those basis vectors) is given by \( \langle \nu | \psi \rangle^2 \), and thus as in the single qubit case we require \( \psi \) to be a unit vector in \( \mathbb{C}^{2^n} \). For the same reason, any operations on the system of qubits needs to preserve the norm of \( \psi \), and so one models the set of possible quantum operations as unitary matrices acting on \( \mathbb{C} \). We associate the vector \(|0\rangle \times \cdots \times |0\rangle \in \mathbb{C}^{2^n} \) with the (classical) binary string \( 0 \ldots 0 \), the vector \(|0\rangle \times \cdots \times |0\rangle \times |1\rangle \in \mathbb{C}^{2^n} \) with the string \( 0 \ldots 01 \) and so on. Any \( |\psi\rangle \) of the form \(|b_1\rangle \otimes \cdots \otimes |b_n\rangle \) with \( b_1, \ldots, b_n \in \{0,1\} \) is referred to as being in a pure state – note that if one measures such a \( |\psi\rangle \) with respect to an orthonormal basis that \( |b_1\rangle \otimes \cdots \otimes |b_n\rangle \) is a basis vector in, the measurement outcome will be \(|b_1\rangle \otimes \cdots |b_n\rangle \) with probability 1. States that do not take this form, i.e., that are some linear combination of such vectors (noting that the collection of \(|b_1\rangle \otimes \cdots |b_n\rangle, b_1, \ldots, b_n \in \{0,1\} \) indeed form an orthonormal basis for \( \mathbb{C}^{2^n} \)), are referred to as being in superposition. For simplicity, it is convention to write the state \(|b_1\rangle \otimes \cdots |b_n\rangle \) as \(|b_1 b_2 \ldots b_n\rangle \). Then we can write any \( |\psi\rangle \in \mathbb{C}^{2^n} \) as

\[
|\psi\rangle = \sum_{b_1,\ldots,b_n \in \{0,1\}} \alpha_{b_1 b_2 \ldots b_n} |b_1 b_2 \ldots b_n\rangle,
\]

(4.8)
where \( \alpha_{b_1b_2...b_n} \in \mathbb{C} \) is called the amplitude of state \( b_1b_2...b_n \), and since it must be a unit length vector we have
\[
\sum_{b_1,...,b_n \in \{0,1\}} |\alpha_{b_1b_2...b_n}|^2 = 1.
\]

Any operation on the state of this system of \( n \) qubits must preserve this, and so the set of linear unitary operators \( U : \mathbb{Q}^{\otimes n} \rightarrow \mathbb{Q}^{\otimes n} \) are valid operations to be performed on qubits.

**Grover Search** Let us now use this framework to establish a Grover’s Search algorithm [52]. We will establish Grover’s algorithm in a simple case for ease of presentation. Suppose one has some function \( f : X \rightarrow \{0,1\} \) such that there exists exactly one \( x^* \in X \) such that \( f(x) = 1 \); we wish to find that \( x^* \). Assume for simplicity that \( X = \{0,1\}^n \).

To do this classically, it is clear that one needs to evaluate \( f \) up to \( O(|X|) \) times, in the worst case. Even to do this with some constant success probability \( c \) requires \( O(|X|) \) evaluations for the same reason. With no further assumptions on \( f \), one cannot do any better than a simple guess-and-check strategy, so that in the worst case the very last item that one guesses is the one giving \( f(x) = 1 \). Grover search, however, is able to solve this problem in only \( O(\sqrt{|X|}) \) evaluations of \( f \), by taking advantage of quantum effects.

Before describing the algorithm, we need to establish how one evaluates a function \( f : X \rightarrow \{0,1\} \) in the quantum computational model. Since this will be an operation on qubits, we need to be able to represent \( f \) as a unitary linear operator. Recall we are assuming that \( X = \{0,1\}^n \), which can be thought of as labeling each element of the domain of \( f \) with a binary string of length \( n \). In particular, \( n = \lceil \log_2|X| \rceil \) is sufficient for this. Recall from elementary linear algebra that a linear operator is fully defined by what it does to each element of a basis of the domain. For a function \( f \), consider the operator

\[
U_f : \mathbb{Q}^{\otimes n} \times \mathbb{Q} \rightarrow \mathbb{Q}^{\otimes n} \times \mathbb{Q}
\]

\[
(|x\rangle, |y\rangle) \rightarrow (|x\rangle, |f(x) \oplus y\rangle),
\]

(4.9)
for $x = b_1 b_2 \ldots b_n$, with $b_1, \ldots, b_n \in \{0, 1\}$, and $y \in \{0, 1\}$, where $\oplus$ denotes the XOR operation (or equivalently, summation modulo 2). This operator is linear and unitary – it is straightforward to determine that $U$ is linear and that each unit basis vector $(|b_1 b_2 \ldots b_n \rangle, |b_{n+1} \rangle)$, with $b_1 b_2 \ldots b_n, b_{n+1} \in \{0, 1\}$ of the domain is mapped to another unit basis vector, so any such $U$ is indeed a unitary linear operator. Hence, this gives a way of implementing such a function $f$ in the quantum computational model.

We will make use of the following Lemma that provides the two most important operations for Grover search.

**Lemma 76.** Given some $f : X \to \{0, 1\}$, $X = \{0, 1\}^n$, and a system of $n$ qubits in state $|\psi\rangle = \sum_{x \in X} \alpha_x |x\rangle$, the following two operations are unitary and can be implemented with a single evaluation of $U_f$:

\begin{align}
    |\psi\rangle &\to \sum_{x \in X : f(x) = 0} \alpha_x |x\rangle + \sum_{x \in X : f(x) = 1} (-1)\alpha_x |x\rangle \quad \text{(Flip the good amplitudes)} \quad (4.10) \\
    |\psi\rangle &\to \sum_{x \in X} (\mu - \alpha_x) |x\rangle, \quad \mu := \frac{1}{|X|} \sum_{x \in X} \alpha_x \quad \text{(Invert amplitudes about the mean)} \quad (4.11)
\end{align}

Let us call these operations $U_{f,\text{flip}}$ and $U_{f,\text{inmean}}$, respectively. It is an interesting exercise to determine that indeed these are unitary operators and that (4.10) can be implemented with only one evaluation of $U_f$. We refer the interested reader to [93].

Given these two operations, we formally present Grover’s Search algorithm in Algorithm 3.

**Algorithm 3** Grover Search Algorithm

1. Initialize $|\psi\rangle := \frac{1}{\sqrt{|X|}} \sum_{x \in X} |x\rangle$.

2. FOR $t = 1, \ldots, T$ DO
   (a) Set $|\psi\rangle \leftarrow U_{f,\text{flip}} |\psi\rangle$
   (b) Set $|\psi\rangle \leftarrow U_{f,\text{inmean}} |\psi\rangle$

3. Measure $|\psi\rangle_T$ with respect to the standard basis, and report the outcome of the measurement as $x^*$.
   (a) Let $x_t \in P_t$ be the center of gravity of $P_t$ as defined in 8.
   (b) Query the full-information first-order oracle at $x_t$, and store the answers $f(x_t)$ and $\nabla f(x_t)$.
   (c) Set $P_{t+1} := P_t \cap H \subseteq (x_t, \nabla f(x_t))$, and update $P \leftarrow P_{t+1}, t \leftarrow t + 1$.

4. Report the point with the smallest observed function value, $\tilde{x}^* := \arg \min_i f(x_i^*)$.
We forego a formal proof for the guarantees of Grover Search but provide the main ideas here. One initializes $|\psi\rangle$ as the uniform superposition of all possible inputs to $f$, i.e. $|\psi\rangle = \frac{1}{\sqrt{|X|}} \sum_{x \in X} |x\rangle$. Then, one alternates flipping the amplitude of the element $x^*$ with $f(x^*) = 1$, using (4.10), and inverting all the amplitudes about the mean with (4.11). Consider what happens when one does this. Upon initialization, all $\alpha_x$ have value $\frac{1}{\sqrt{|X|}}$. The amplitude for $x^*$ after applying $U_f^{flip}$ to $|\psi\rangle$ becomes $\alpha_{x^*} = -\frac{1}{\sqrt{|X|}}$. Then, assuming $|X| >> 1$, the mean of all the amplitudes is still roughly $\frac{1}{\sqrt{|X|}}$, so that after we apply $U_f^{invmean}$, we have $\alpha_{x^*} = \mu_\alpha - \alpha_{x^*} \approx -2 \frac{1}{\sqrt{|X|}}$. Using the same reasoning, if one applies $U_f^{flip}$ and $U_f^{invmean}$ again, one has $\alpha_{x^*} \approx \frac{3}{\sqrt{|X|}}$. Hence, this increases the amplitude $\alpha_{x^*}$ by roughly $\frac{1}{\sqrt{|X|}}$ each time. Doing this $O(\sqrt{|X|})$ times, one has $\alpha_{x^*} = O(1)$, so that when one measures $|\psi\rangle$ with respect to the standard basis (or any orthonormal basis that includes $|x^*\rangle$), the outcome of that measurement will be $x^*$ with some constant probability. Hence, setting $T = O(\sqrt{|X|})$ allows us to find $x^*$ with some high probability, and in fact Grover search can be derandomized to return $x^*$ with certainty, though we will forego that discussion here (see e.g. [93, Chapter 7] for how this can be done).

Many variations of Grover’s Algorithm exist, such as the just mentioned derandomized version, variants for optimization often referred to as amplitude amplification algorithms, and even distributed versions that will be relevant in Chapter 6. An important note about Grover Search is that the quadratic speedup it provides is for the unstructured search problem – one assumes no knowledge or structure about $X$ or $f$, simply that $X$ is a set of possible inputs for a black-box $f$. This is a severe limitation of the usefulness of the algorithm. In real life, in most cases either the data or the function have some kind of structure. Databases are sorted, objects like graphs may have some kind of structure, or the functions involved may have some kind of known structure like subadditivity or convexity. However, despite this shortcoming, Grover Search serves as a surprising example for what kinds of problems seemingly unrelated to quantum mechanics in any way quantum computation has the potential to provide faster algorithms for.

With this background on quantum computing in mind, in the two forthcoming
chapters we present recent contributions in algorithms using quantum annealing and (distributed) quantum computing.
Chapter 5

Image Denoising via Quantum Annealing

Chapter Abstract

We investigate a framework for binary image denoising via restricted Boltzmann machines (RBMs) that introduces a denoising objective in quadratic unconstrained binary optimization (QUBO) form well-suited for quantum annealing. The denoising objective is attained by balancing the distribution learned by a trained RBM with a penalty term for deviations from the noisy image. We derive the statistically optimal choice of the penalty parameter assuming the target distribution has been well-approximated, and further suggest an empirically supported modification to make the method robust to that idealistic assumption. We also show under additional assumptions that the denoised images attained by our method are, in expectation, strictly closer to the noise-free images than the noisy images are. While we frame the model as an image denoising model, it can be applied to any binary data. As the QUBO formulation is well-suited for implementation on quantum annealers, we test the model on a D-Wave Advantage machine, and also test on data too large for current quantum annealers by approximating QUBO solutions through classical heuristics. The contributions of this chapter are also available in [65].
5.1 Introduction

Quantum annealing (QA) \cite{64, 32, 2} is a promising technology for obtaining good solutions to difficult optimization problems, by making use of quantum interactions to aim to solve Ising or quadratic unconstrained binary optimization (QUBO) instances. Since Ising and QUBO instances are NP-hard, and many other combinatorial optimization problems can be reformulated as Ising or QUBO instances (see e.g. \cite{46}), QA has the potential to become an extremely useful tool for optimization. As the capacities of commercially available quantum annealers continue to improve rapidly, it is of great interest to build models that are well-suited for this emerging technology. Furthermore, QA has promising machine learning applications surrounding Boltzmann Machines (BM), as both QA and BMs are closely connected to the Boltzmann distribution. Boltzmann Machines are a type of generative artificial neural network that aim to learn the distribution of some training data set by fitting a Boltzmann distribution to the data, as described thoroughly in \cite{47, §20}. On the other hand, QA aims to produce approximate minimum energy (maximum likelihood) solutions to a Boltzmann distribution via finding the ground state of the associated Hamiltonian that determines the distribution. Hence, maximum likelihood type problems on BMs are a natural candidate for applying QA in a machine learning framework. We contribute to the goal of furthering useful applications of QA in machine learning in this chapter by building an image denoising model particularly well-suited for implementation via QA.

The task of image denoising is a fundamental problem in image processing and machine learning. In any means of collecting images, there is always a chance of some pixels being afflicted by noise that we wish to remove; see e.g. \cite{18} for a good overview. Accordingly, many classical and data-driven approaches to the image denoising problem have been studied in the literature \cite{20, 102, 48, 94, 26}. This chapter studies a quantum binary image denoising model using Restricted Boltzmann Machines (RBMs henceforth) \cite{47, §20.2} that can take advantage of QA by formulating the denoising problem as a QUBO instance. Specifically, given a trained RBM, we introduce a penalty-based denoising scheme that admits a simple QUBO form, for which we derive the statistically optimal penalty parameter as well as a practically-motivated robustness modification.
The denoising step only needs to solve a QUBO admitting a bipartite graph representation, and so is well-suited for QA. As QA has also shown promise for training BMs [1, 38], our full model lends itself well for denoising images using quantum annealers, and could thus play a role in their future applications since QA can then be leveraged for both the training and denoising steps. The model also shows promise in absence of QA, and our insights presented are not limited to the QA framework, as the QUBO formulation of the denoising problem and its statistical properties we prove may be of independent interest.

The chapter is organized as follows. Section 5.2 gives a summary of background on quantum annealing and Boltzmann Machines. Section 5.3 describes our main contribution of the image denoising model for QAs, and Section 5.4 shows some practical results obtained.

**Remark 77.** We frame our work as a binary image denoising method, although the framework does not depend on the data being images, and can be applied to the denoising of any binary data. This is because the framework does not use any spatial relationships between the pixels, and instead treats the image as a flattened vector whose distribution is to be learned. Hence, the denoising scheme can be applied as-is to any other binary data setting.

### 5.1.1 Contributions and Organization

We provide QUBO-based denoising method for binary images (applicable to general binary data) using restricted Boltzmann machines in Section 5.3. This is done by formulating the denoising objective in equation (5.6) by combining the energy function of the distribution learned by the RBM with a (parameterized) penalty term for deviations from a given noisy image. This objective turns out to have an equivalent QUBO formulation, which is shown in claim 79. In Theorem 82, we derive the optimal choice for the penalty parameter under the assumption that the true images follow the distribution learned by the RBM, which also recovers the maximum a posteriori estimate per Corollary 83, though our model is more flexible, and this flexibility allows for useful practical modifications. Theorem 84 shows that the denoising method yields a result
that is strictly closer (in expectation) to the true image than the noisy image is, under some additional assumptions. Given that these idealistic assumptions won’t be met in reality, we propose a robustness modification in Section 5.3.3 that improves performance empirically. In Section 5.4, as the method lends itself well to quantum annealing, we then implement the method on a D-Wave Advantage 5000-qubit quantum annealer, demonstrating strong empirical performance. Since only small datasets can be tested on the D-Wave machine due to the relatively low number of qubits, we also test the method on a larger dataset, for which we use simulated annealing on a conventional computer in place of quantum annealing to find good solutions the QUBO denoising objective. Though we highlight the method being well-suited for quantum annealers, we emphasize that it may be of independent interest to the machine learning and image processing communities at large.

5.1.2 Related Work

Closely related work of [70] uses a similar model as ours for the image reconstruction task, also solving QUBO formulations via quantum annealing. In the reconstruction task, some subset of pixels is unknown (or obscured or missing), and needs to be restored, whereas our work considers denoising, where which pixels are noise-afflicted is unknown. [48] derives a maximum a posteriori (MAP) estimator for the noise free image as a denoising method in a particular model of binary images that is less general than ours, though we would recover their estimator under a particular choice of our penalty parameter if we were to apply our framework to their model (since we recover MAP in a more general setting). Further, RBMS and quantum annealing have been studied for the classification problem, for instance in [72] and [1]. Other research in the machine learning communities has also studied handling label noise, such as related work in [104], which studies the problem of training models in the presence of noisy labels, whereas our approach is entirely unsupervised (the data need not have any labels to begin with).
5.2 Background

Though we have already discussed the basics of quantum annealing in 4.1.1, we provide here some additional background on QA especially in how it relates to Boltzmann Machines that we make use of in the chapter. Recall that Quantum Annealers make use of quantum interactions with the primary goal of finding the ground state of a Hamiltonian operator by initializing and then evolving a system of coupled qubits over time [63]. In particular, we may view QA as implementing the Ising spin-glass model [87] evolving over time. As the QUBO model is equivalent to the Ising model [46], and QUBO instances can be efficiently transformed to Ising instances, a QA is well suited to provide good solutions to QUBO problems. An optimization problem with QUBO cost function, or energy function, takes the form

\[
\min_{x \in \{0, 1\}^n} f_Q(x) := \min_{x_i \in \{0, 1\}} \sum_{i,j} Q_{ij} x_i x_j
\]

(5.1)

where \( Q \) is a symmetric, \( n \times n \) real-valued matrix. We will occasionally refer to \( Q_{ij} \) as the weight between \( x_i \) and \( x_j \). QUBO is well-known to be NP hard [6], and many combinatorial problems can be reformulated as QUBO instances. See [46, 78] for a thorough presentation of QUBO formulations of various problems. A Boltzmann Distribution using the above QUBO as its energy function takes the form

\[
P_Q^{\text{model}}(x) = \frac{1}{z} \exp(-f(x, Q)),
\]

(5.2)

where \( z \) is a normalizing constant. Note that a parameter called inverse temperature has been fixed to unity and is not explicitly shown in the above expression. In this chapter, we will focus on making use of Boltzmann Machines, a type of generative neural network that fits a Boltzmann Distribution to the training data via making use of latent variables. The goal is to approximate as well as possible the distribution of real data observed. Specifically, we consider Restricted Boltzmann Machines (RBMs), which have seen significant success and frequent use in deep probabilistic models [47]. RBMs consist of an input layer of visible nodes (corresponding to observed / generated data), and a layer of latent, or hidden nodes (serving as a latent representation of the data),
which each have zero intra-group weights. An RBM will learn a distribution across both
the visible and hidden nodes—however, only the visible nodes correspond to data to
be observed or generated, while the hidden nodes remain unobserved (latent). Hence,
the learning problem for the RBM is to learn a distribution such that the marginal
distribution of only the visible nodes approximates the distribution of observed data
as well. The hidden nodes simply serve to give the RBM sufficient expressibility and
structure to be able to achieve this task.

Let \( v \in \{0, 1\}^v \) and \( h \in \{0, 1\}^h \) denote the visible and hidden nodes, respectively.
It will be convenient for us to write \( x = (v, h) \in \{0, 1\}^{v+h} \) as their concatenation. The
probability distribution represented by a RBM is then

\[
P^\text{model}_Q((v, h)) = \frac{1}{Z} \exp(-f((v, h), Q))
\]  

(5.3)

with the restriction that \( Q_{ij} = Q_{ji} = 0 \) if \( i, j \in \{1, \ldots, v\} \) or \( i, j \in \{v + 1, \ldots, v + h\} \).
Hence, we have the simplified energy function

\[
f((v, h), Q) = \sum_{i=1}^{v+h} \sum_{j=1}^{v+h} 2Q_{ij} (v, h)_i (v, h)_j = \sum_{i=1}^{v} \sum_{j=v+1}^{v+h} Q_{ij} v_i h_j + \sum_{i=1}^{v} Q_{ii} v_i^2 + \sum_{i=v+1}^{v+h} Q_{ii} h_i^2
\]

= \( h^T W v + b_v^T v + b_h^T h =: f_{W, b_v, b_h}(v, h) \)  

(5.4)

where \( W \) is the \( v \times h \) matrix consisting of the \( Q_{ij} \) weights between the visible and
hidden nodes, and \( b_v \) and \( b_h \) are vectors of the diagonal entries \( Q_{ii}, i \in \{1, \ldots, v\} \)
corresponding to visible nodes, and \( Q_{ii}, i \in \{n + 1, \ldots, v + h\} \) corresponding to hidden
nodes, respectively. We will write the Boltzmann distribution with this energy function
as \( P^\text{model}_{W, b_v, b_h} \), noting that this is also \( P^\text{model}_Q \) for the appropriate \( Q \).

It is well known that RBMs can universally approximate discrete distributions
[47], making them a powerful model. They are also more easily trained than general
Boltzmann Machines, usually through the contrastive divergence algorithm as described
in [58], or variants thereof.
5.2.1 Training Boltzmann Machines

We first devote some discussion to the training of RBMs. Subsection 5.3.1 then describes how to denoise images via QUBO given a well-trained RBM.

Continuing with the notation as in equation (5.4), the probability distribution represented by a RBM is

\[ P_\theta(v, h) = \frac{1}{z_\theta} \exp(-f_\theta). \]

For simplicity, denote \( \theta = (W, b_v, b_h) \) as the model parameters henceforth. The normalizing constant \( z_\theta \) above is

\[ z_\theta = \sum_{v \in \{0, 1\}^v} \sum_{h \in \{0, 1\}^h} \exp(-f_\theta(v, h)) \]

which becomes intractable quickly even for relatively small values of \( v \) and \( h \). The common training approach aims to maximize the log-likelihood of the data. At a high-level, this will be done by approximating gradients and following a stochastic gradient scheme. However, the RBM work with both the visible and hidden nodes. Recall that only the visible nodes are observed, while the hidden nodes remain latent, and that the goal is to learn a distribution such that the marginal distribution of the visible nodes approximates the distribution of the observed data well. For this reason, since our data consists only of the visible nodes, we need to work with the marginal distribution of the visible nodes. This is given by

\[ P_\theta(v) = \sum_h P_\theta(v, h) = \sum_h \frac{\exp[-f_\theta(v, h)]}{z_\theta} \]

Denote our set training data samples by \( V := \{v^1, ..., v^N\} \). We will use superscripts to indicate training data samples, and reserve subscripts to denote entries of vectors. Then
the log-likelihood is given by

\[
l_\theta(V) = \sum_{k=1}^{N} \log P_\theta(v^k) = \sum_{k=1}^{N} \log \sum_h P_\theta(v^k, h)
\]

\[
= \left( \sum_k \log \sum_h \exp(-f_\theta(v^k, h)) \right) - N \cdot \log z_\theta
\]

\[
= \left( \sum_k \log \sum_h \exp(-f_\theta(v^k, h)) \right) - N \cdot \log \sum_v \sum_h \exp(-f_\theta(v, h)) 
\]  

(5.5)

Now we can calculate the gradient with respect to \( \theta \) as

\[
\nabla l_\theta(V) = \sum_{k=1}^{N} \frac{\sum_h \exp(-f_\theta(v^k, h)) \nabla (-f_\theta(v^k, h))}{\sum_h \exp(-f_\theta(v^k, h))} - N \cdot \frac{\sum_v \sum_h \exp(-f_\theta(v, h)) \nabla (-f_\theta(v, h))}{\sum_v \sum_h \exp(-f_\theta(v, h))}
\]

\[
= \sum_{k=1}^{N} E_{P_\theta(h|v^k)} \left[ -\nabla f_\theta(v^k, h) \right] - N \cdot E_{P_\theta(v, h)} \left[ -\nabla f_\theta(v, h) \right]
\]

\[
= \frac{1}{N} \sum_{k=1}^{N} E_{P_\theta(h|v^k)} \left[ (v^k)^T h + v^k + h \right] - E_{P_\theta(v, h)} \left[ v^T h + v + h \right]
\]

The first term can be computed exactly and efficiently from the data, since the conditional \( P_\theta(h|v) \) admits the simple form \( P(h_j = 1|v) = \text{logistic}(b_h + (v^T W)_j) \); we refer the interested reader to [38] or [47] and will focus on the second term. Due to its intractability to compute (one would have to sum over all possibilities of \( v \) and \( h \)), the most promising approach is to approximate it by sampling from \( P_\theta(v, h) \). Classically, this is done via Gibbs sampling as described in [58]. However, recent research has also investigated using quantum annealers to sample from the relevant Boltzmann distribution, as suggested in [38, 13], which would make QAs useful in the training process since obtaining good Gibbs samples can be expensive. We note that together with our framework, QAs show promise to become useful for both the RBM training and the denoising process in the implementation of our method.
5.3 Image Denoising as Quadratic Unconstrained Binary Optimization

This section is devoted to showing how one can naturally frame the image denoising problem as a QUBO instance over a learned Boltzmann Distribution fit to the data.

5.3.1 Denoising via QUBO

Let us assume we are given a trained Restricted Boltzmann Machine described in Sec. 5.2. The model prescribes to each vector \( x \in \{0, 1\}^v \) the cost \( f_Q(x) \) and corresponding likelihood \( P_{model}^Q(x) \) defined in Eqs. ((5.1)) and ((5.3)), respectively. We will here make the assumption that \( P_{model}^Q \) describes the distribution of our data. Hence, high likelihood vectors in \( P_{model}^Q \) correspond to low cost vectors of \( f_Q \). In particular, note that finding the maximum likelihood argument in ((5.2)) corresponds to finding a solution to the QUBO instance in ((5.1)). However, as mentioned earlier, in the data one observes only the visible nodes, so one only observes vectors \( x \in \{0, 1\}^v \). We denote the marginal distribution of the visible nodes as \( P_{model}^Q|_v \).

Now, supposing this model, our goal is to reconstruct an image that has been affected by noise. The visible portion of our vector will be considered to be a flattened image with \( v \) pixels, black or white corresponding to 0 or 1, respectively, in the binary entries of the vector.

Noise Model

We now describe the noise assumptions we will conduct our analysis under.

Definition 78. For \( x \in \{0, 1\}^v \), we define \( x \) afflicted by salt-and-pepper noise of level \( \sigma \) as the random variable \( \tilde{X}_{x,\sigma} := (x + \varepsilon) \mod 2 \), where \( \varepsilon_i = B_i(p) \sim Bern(\sigma) \), independently.

In other words, a binary image afflicted by salt-and-pepper noise has each pixel independently flipped with probability \( \sigma \). In particular, we are interested in \( \tilde{X}_{X,\sigma} \), where \( X \sim P_{model}^Q|_v \), which is the compound random variable obtained by sampling the visible nodes \( X \) from the learned distribution of the data and then afflicting it with
salt-and-pepper noise. For notational simplicity, will simply write $\tilde{X}$ when the intended subscripts are clear from context.

We remark here that this salt-and-pepper noise model, also sometimes called impulse valued noise, is a natural choice for binary data and can occur in image processing through faulty sensors or pixel elements in cameras; see e.g. [18] for discussion of noise models in digital image processing. Since the pixels (or binary data entries for non-image binary data) only take the values 0 or 1, individual entries can only be corrupted by the value being flipped. Hence, continuous noise models such as Gaussian noise are not appropriate. Further, since the data we can work with on currently available quantum machines are very small, imposing additional structure on the noise does not seem fitting. However, the related problem of image reconstruction, in which some known set of pixels is damaged, is another model appropriate for such data, as studied in [70]. We emphasize that in our noise model, which pixels are affected by noise is random and unknown, leading to the denoising problem.

Suppose we are given a realization $\tilde{x} \in \{0, 1\}^v$ of $\tilde{X}_{X,\sigma}$. The reconstruction process aims to retrieve this original $X$ using $\tilde{x}$ and the trained model through $Q$. The approach we will take begins from the intuition that $X$ is likely to be a high-likelihood image that is close to $\tilde{x}$. To enforce this “closeness” to $\tilde{x}$ while searching for higher likelihood images in our model to remove noise, we add to the cost in ((5.1)) a penalty for deviations from $\tilde{x}$ to formulate the following natural denoising cost function:

$$f_{Q,\tilde{x},\gamma}(x) = f_Q(x) + \gamma \sum_{i=1,\ldots,v} (x_i - \tilde{x}_i)^2$$

for some $\gamma > 0$ that determines the penalty level. Note that this is a function of $x \in \{0, 1\}^{v+h}$, even though $\tilde{x} \in \{0, 1\}$ only consists of visible nodes. The penalty $\gamma$ for deviations is only applied to the visible nodes, while there is no modification to the distribution for the hidden nodes (since they are unobserved). The intuition is that by penalizing deviations from the observed noisy image, the minimizer of this function for a well-chosen $\gamma$ will change a limited number of pixels to find an image that is similar to the noisy image, but has a lower cost, i.e. higher likelihood, under the model, in hopes
of removing the noise.

We show next that this minimizing (5.6) corresponds to solving a QUBO instance.

\textbf{Claim 79.} Defining $\tilde{Q}_\gamma, \tilde{x} \in \mathbb{R}^{(v+h)\times(v+h)}$ by setting $\tilde{Q}_{ij} = Q_{ij}$ if $i \neq j$ and $\tilde{Q}_{ii} = Q_{ii} + \gamma(1 - 2\tilde{x})$ if $i = j$, we have

$$\arg\min_{x} f_{Q, \tilde{x}, \gamma}(x) = \arg\min_{x} f_{\tilde{Q}_\gamma, \tilde{x}}(x).$$

\textbf{Proof.} Observe that

$$f_{Q, \tilde{x}, \gamma}(x) = f_{Q}(x) + \gamma \sum_{i}(x_{i} - \tilde{x}_{i})^{2} = \sum_{i,j} Q_{ij} x_{i} x_{j} + \gamma \sum_{i} x_{i}^{2} - 2x_{i}\tilde{x}_{i} + \tilde{x}_{i}^{2}$$

$$= \sum_{i \neq j} Q_{ij} x_{i} x_{j} + \sum_{i} Q_{ii} x_{i}^{2} + \gamma(x_{i}^{2} - 2x_{i}\tilde{x}_{i} + \tilde{x}_{i}^{2})$$

$$= \sum_{i \neq j} Q_{ij} x_{i} x_{j} + \sum_{i} (Q_{ii} + \gamma(1 - 2\tilde{x}_{i}))x_{i}^{2} + \gamma \tilde{x}_{i}^{2} = f_{\tilde{Q}_\gamma, \tilde{x}}(x) + \sum_{i} \gamma \tilde{x}_{i}$$

Noting that $x_{i} = x_{i}^{2}$ for the above derivation since they are in $\{0, 1\}$ here. Since the $\tilde{x}_{i}$ terms do not depend on $x$, the claim follows. \hfill $\Box$

Hence, solving the QUBO in on the right hand side of equation (5.7) gives us the solution to (5.6). Claim 79 thus tells us that we simply need to modify the diagonal of the original matrix $Q$ of our model by adding $\text{diag}(1 - 2\tilde{x}_{1}, ..., 1 - 2\tilde{x}_{n})$ and then solve the resulting QUBO to get the denoised image. We can then make use of quantum annealing to solve the resulting QUBO of (5.7), or use classical methods and heuristics like simulated annealing instead. We formally spell out the denoising procedure in algorithm \texttt{QUBO\_Denoise}.

<table>
<thead>
<tr>
<th>\texttt{QUBO_Denoise}</th>
</tr>
</thead>
</table>
| **Input:** A matrix $Q$, a noisy image $\tilde{x}$ sampled from the distribution of $\tilde{X}_{X,\sigma}$ with $X \sim P^{\text{model}}_{\gamma}$, and a penalty parameter $\gamma > 0$.  

**Output:** A denoised image $X^{*}_{\gamma, \tilde{x}, Q}$. |

144
1. Set $\tilde{Q}_{ij}^{\gamma,x} = Q_{ij}$ if $i \neq j$ and $\tilde{Q}_{ii}^{\gamma,x} = Q_{ii} + \gamma(1 - 2\tilde{x})$ if $i = j$.

2. Set $X_{\gamma,x,Q}^* := \text{argmin}_x f_{\tilde{Q}_{\gamma,x}}(x)$.

For the remainder of the chapter, $X_{\gamma,x,Q}^*$ will denote the denoised image obtained by applying \texttt{QUBO\_Denoise} with noisy image $\tilde{x}$, penalty parameter $\gamma$, and the distribution-defining matrix $Q$.

\textbf{Remark 80.} Considering the entire process of sampling a noisy image and then denoising it, the measurability of $X_{\gamma,x,Q}^*$ is inherited from the measurability of $\tilde{X}_{X,\sigma}$, which in turn inherits its measurability as compound random variable of the measurable noise and original image $X \sim P_{\text{model}}^Q$.

### 5.3.2 Optimal Choice of penalty parameter $\gamma$

The choice of the parameter $\gamma$ for the proposed image denoising model is clearly crucial to its success, since different choices will result in different solutions. If $\gamma$ is chosen to be too small, there is very little cost to flipping a pixel, and then many pixels may be flipped and the solution may not resemble the noisy image at all anymore. If $\gamma$ is too large, we may be too heavily penalizing flipping pixels, and thus may not be able to get rid of noise effectively. Hence, we now turn towards finding the optimal choice for $\gamma$.

We will evaluate the choice of $\gamma$ via expected overlap:

\textbf{Definition 81.} The expected overlap between two distributions $P$ and a $P'$, is defined by

$$d(P, P') := \mathbb{E}_P \mathbb{E}_{P'} (n - \|X - X'\|_1),$$

where $X \sim P, X' \sim P'$.

We will consider $X \sim P_{\text{model}}^Q|_v$, and $X'$ as $X_{\gamma,x,Q}^*$, the corresponding denoised image, and will also call $d(P, P')$ the expected overlap between $X$ and $X'$. To keep notation simple, for the remainder of this section allow us to write $\tilde{X}$ in place of $\tilde{X}_{X,\sigma}$, with $X$ and $\sigma$ being clear from context.

Our main positive result concerning the choice of $\gamma$ is summarized in the following
Theorem 82. Let \( X \sim P^\text{model}_\theta \) as in (5.2) and \( \tilde{X} \) be the noisy image. Then choosing
\[
\gamma = \log \frac{1 - \sigma}{\sigma}
\]
to obtain \( X^*_\gamma, \tilde{X}, Q \) is optimal with respect to maximizing the expected overlap between \( X \) and \( X^*_\gamma, \tilde{X}, Q \).

Proof. Let \( X \sim P^\text{model}_\theta \) and \( \tilde{X} \) be the noisy image afflicted by salt-and-pepper noise of level \( \sigma \). Then since \( \tilde{X} \) is obtained by flipping pixels with probability \( \sigma \), we have the conditional probability
\[
P_\sigma(\tilde{X} = \tilde{x} | X = x) = \prod_{i=1}^v \left\{ \sigma(\tilde{x}_i - x_i)^2 + (1 - \sigma)[1 - (\tilde{x}_i - x_i)^2] \right\}
\]
\[
= \exp \left[ -\beta_\sigma \sum_{i=1}^v (\tilde{x}_i - x_i)^2 \right] \frac{1}{(1 + e^{-\beta_\sigma})^v},
\]
where \( \beta_\sigma := \log \frac{1 - \sigma}{\sigma} \). In order to infer the original image \( X \) from the noisy one \( \tilde{X} \), we utilize the Bayes formula and calculate the conditional probability \( P^\text{post}_{\beta_\sigma, Q}(X = x | \tilde{X} = \tilde{x}) \).

The overlap of two vectors \( x^* \) and \( x \) is given by
\[
m(x, x^*) := \frac{1}{v + h} \sum_{i=1}^{v+h} (2x_i - 1)(2x^*_i - 1),
\]
the proportion of shared entries. We consider the average (over the noise) of solutions,
\( \bar{X}_{\gamma,\bar{x},Q} \) with
\[
(\bar{X}_{\gamma,\bar{x},Q})_i = \theta \left( \sum_{\{x\}} P_{\text{model}}(x) x_i - \frac{1}{2} \right),
\]  
(5.11)
where \( \theta(x) = 1 \) if \( x > 0 \), otherwise 0, noting that the right hand side represents the inferred pixel value based on the expectation from \( P_{\text{model}}^Q \). We have formally distinguished \( P_{\text{model}}^Q(x) \) from \( P_{\gamma,\bar{Q}}^\text{post}(x|\bar{x}) \), but in fact they are the same. Note that
\[
2(\bar{X}_{\gamma,\bar{x},Q})_i - 1 = \text{sign} \left( \sum_{\{x\}} P_{\text{model}}(x) (2x_i - 1) \right),
\]  
(5.12)
where \( \text{sign}(x) \) is the sign of \( x \). Let \( \alpha_{\sigma,Q} := -\beta_{\sigma} \sum_i (\bar{x}_i - x_i)^2 - \sum_{i,j} Q_{ij} x_i x_j \) for conciseness. In order to evaluate the statistical performance of our method with coefficient \( \gamma \) of penalty term, we calculate the average of overlap as
\[
M_{\beta_{\sigma},Q}^\gamma := \frac{1}{(1 + e^{\beta_{\sigma}})^\nu} \frac{1}{v + h} \sum_i \sum_{\{\bar{x}\},\{x\}} e^{\alpha_{\sigma,Q}} [2(\bar{X}_{\gamma,\bar{x},Q})_i - 1](2x_i - 1). 
\]  
(5.13)
A sum in the right hand side of the above equation holds
\[
\sum_{\{x\}} e^{\alpha_{\sigma,Q}} [2(\mathbb{E}(X_{\gamma,\bar{x},Q}^*)_i - 1)](2x_i - 1) \leq \sum_{\{x\}} e^{\alpha_{\sigma,Q}} [2(\mathbb{E}(X_{\gamma,\bar{x},Q}^*)_i - 1)](2x_i - 1)
\]
\[
\leq \sum_{\{x\}} e^{\alpha_{\sigma,Q}} (2x_i - 1) \left| \frac{\sum_{\{x\}} e^{-\beta_{\sigma} \sum_i (\bar{x}_i - x'_i)^2 - \sum_{i,j} Q_{ij} x'_i x'_j (2x'_i - 1)}}{\sum_{\{x\}} e^{-\beta_{\sigma} \sum_i (\bar{x}_i - x'_i)^2 - \sum_{i,j} Q_{ij} x'_i x'_j (2x'_i - 1)}} \right|
\]
\[
= \sum_{\{x\}} e^{\alpha_{\sigma,Q}} (2x_i - 1) \text{sign} \left( \sum_{\{x\}} P_{\text{model}}^Q(x') (2x'_i - 1) \right)
\]
\[
= \sum_{\{x\}} e^{\alpha_{\sigma,Q}} [2(\bar{X}_{\gamma,\bar{x},Q})_i - 1](2x_i - 1). 
\]  
(5.14)
Hence, the averaged overlap holds

\[
M_{\beta, Q}(\gamma) \leq \frac{1}{(1 + e^{\beta})^v} \frac{1}{\mathbb{E}_1} \frac{1}{v + h} \sum_i \sum_{\{\tilde{x}, x\}} e^{-\beta \sum_i (\tilde{x}_i - x_i)^2 - \sum_{i,j} Q_{ij} x_i x_j} [2(\tilde{X}_{\gamma, \tilde{x}, Q})_i - 1](2x_i - 1) \tag{5.15}
\]

This inequality reveals that the averaged overlap is indeed maximized by setting

\[
\gamma = \beta = \log \frac{1 - \sigma}{\sigma}.
\]

This theorem is based on a known fact in statistical physics of information processing [87] and translates the fact into the setting of our problem. Notably, the optimal choice of \(\gamma\) does not depend on the distribution of the data, but only on the noise level, for which in many real world cases one may have good estimates. The proof of the theorem also reveals the following corollary:

**Corollary 83.** Under the same assumptions of Theorem 82, setting \(\gamma := \log \frac{1 - \sigma}{\sigma}\) makes \(X_{\gamma, \tilde{X}, Q}\) the maximum a posteriori estimator for the original noise-free image \(X\).

The corollary follows from observing that the energy function in the numerator of the posterior distribution ((5.9)) is exactly ((5.6)) with \(\gamma := \frac{1 - \sigma}{\sigma}\), noting that minimizing ((5.6)) is equivalent to maximizing ((5.9)). However, this framework allows for additional flexibility in choosing the \(\gamma\) parameter that is absent in standard MAP estimation. In fact, in sections 5.3.3 and 5.4.1 we go on to demonstrate that in practice, choosing a larger \(\gamma\) may be beneficial for robustness of the method.

Though Theorem 82 derives the optimal choice of \(\gamma\), it does not give any guarantees that the method will yield an improvement in expected overlap, even under its assumptions. Next, we prove a theorem to show that in the case of visible units being independent of one another, our image denoising method produces in expectation *strict* denoising improvements with respect to the expected overlap. For \(c > 0\) and a model
distribution $P_{\text{model}}^Q$ as in (5.2), let $\mathcal{I}_c$ be the set of indices $i$ such that $|Q_{ii}| > c$. These indices correspond to components of $X$ that are either 0 or 1 with probability at least $\frac{1}{1 + e^{-c}}$, depending on whether $Q_{ii}$ is positive or negative, respectively.

**Theorem 84.** Suppose that $Q$ is diagonal, $X \sim P_Q$, and that $\tilde{X}$ is $X$ afflicted by salt-and-pepper noise of level $\sigma$. With $\mathcal{I}_c$ as defined above for $c > 0$, setting $\gamma \geq \log\left(\frac{1-\sigma}{\sigma}\right)$, and assuming that $\mathcal{I}_\gamma \neq \emptyset$, the expected overlap of the denoised image and the true image is strictly larger than the expected overlap of the noisy image and the true image, i.e.

$$E \left[ \sum \mathbb{1}( (X^*_\gamma, \tilde{X}, Q)_i = X_i) \right] > E \left[ \sum \mathbb{1}( \tilde{X}_i = X_i) \right].$$

(5.16)

**Proof.** Let $\mathcal{I}^0 := \{ i \in \mathcal{I}_c : Q_{ii} > 0 \}, \mathcal{I}^1 := \{ i \in \mathcal{I}_c : Q_{ii} < 0 \}$. Intuitively, these are the indices which are likely to be zero or one, respectively. Further, letting $x^i$ denote the vector obtained by flipping entry $i$ of $x$, we have that $|f_Q(x) - f_Q(x^i)| = Q_{ii} > c$ if and only if $i \in \mathcal{I}_c$. Hence, this reveals that $x^*$ solves ((5.6)) by setting $x^*_i = 1 \forall i \in \mathcal{I}^1, x^*_i = 0 \forall i \in \mathcal{I}^0, \text{and } x^*_i = \tilde{x}_i$ otherwise, since the value of $f_Q$ of ((5.1)) is reduced by more than $\gamma$, so that the overall penalized objective ((5.6)) improves despite the $\gamma$ penalty accruing by the pixel flips.

Now, let $X \sim P_{\text{model}}^Q|_v$. Let us compute $P((X^*_\gamma, \tilde{X}, Q)_i = X_i)$. The cases where this happens are: $i \in I^0, X_i = 0$, $i \in I^1, X_i = 1$, or $i \notin I_\gamma$ and pixel $i$ was not flipped by the noise.

We know that if $i \in I^0_\gamma$, $P(X_i = b) \geq \frac{1}{1 + e^{-c}}$, for $b \in \{0, 1\}$, so $P((X^*_\gamma, \tilde{X}, Q)_i = X_i) \geq \frac{1}{1 + e^{-c}}$ for these. For $i \notin I_\gamma$, $P((X^*_\gamma, \tilde{X}, Q)_i = X_i) = 1 - \sigma$, where $\sigma$ is the probability that the pixel was flipped by the noise. On the other hand, $P(\tilde{X}_i = X_i) = 1 - \sigma \forall i$. We characterize

$$E \left[ \sum \mathbb{1}( (X^*_\gamma, \tilde{X}, Q)_i = X_i) \right] > E \left[ \sum \mathbb{1}( \tilde{X}_i = X_i) \right]$$

(5.17)

$$\sum P((X^*_\gamma, \tilde{X}, Q)_i = X_i) > \sum P(\tilde{X}_i = X_i) = n \cdot (1 - \sigma)$$

(5.18)

For the left-hand side, assuming $I_\gamma \neq \emptyset$, we have
\[
\sum P((X^*_{\gamma,\tilde{X},Q})_i = X_i) > \sum_{i \in I_\gamma} \frac{1}{1 + e^{-\gamma}} + \sum_{i \notin I_\gamma} (1 - \sigma) = |I_\gamma| \cdot \frac{1}{1 + e^{-\gamma}} + (n - |I_\gamma|)(1 - \sigma)
\]
so that (5.17) holds when

\[
|I_\gamma| \cdot \frac{1}{1 + e^{-\gamma}} + (n - |I_\gamma|)(1 - \sigma) \geq n(1 - \sigma) \tag{5.19}
\]
\[
\iff |I_\gamma| \neq 0 \quad \text{and} \quad \frac{1}{1 + e^{-\gamma}} \geq 1 - \sigma \iff \gamma \geq \log \left( \frac{1 - \sigma}{\sigma} \right) \quad \text{and} \quad I_\gamma \neq \emptyset, \tag{5.20}
\]
and the theorem is proven.

The assumption that matrix \( Q \) is diagonal is equivalent to the components of \( X \) being independent, which is not realistic with real data. However, since in the RBM model the visible units are independent conditioned on the hidden units, we still consider this independent case to be informative to the denoising method. In fact, if the hidden states were fixed (or known, or recovered correctly), Theorem 84 would apply. We leave it as a tantalizing open question to generalize this result beyond the independent case.

The assumption of nonemptiness of \( I_\gamma \) is a natural one for the denoising task; indeed, when \( I_\gamma \) is empty, no entries of \( Q \) are large in magnitude, which is equivalent to the entries of \( X \) being close to uniformly distributed. In that case, intuitively of course it should not be possible to guarantee that we can denoise an image well if it looks like noise to begin with.

### 5.3.3 Robust Choice of \( \gamma \)

The optimal choice of \( \gamma \) as derived in Theorem 82 relies on the assumption that the observed data comes from the learned distribution, or equivalently that the distribution generating our data has been perfectly learned by the RBM. However, in practice we will always only approximately learn the data distribution. Hence, we do not want to rely too heavily on the exact distribution we have learned when we denoise the images. One may hope to have a more robust method by only changing the value of a pixel
when there is some confidence in the model that the pixel should be flipped. We may thus want to penalize flipping pixels slightly more than we should under the idealistic setting of Theorem 82, which corresponds to choosing a larger $\gamma$ value than $\log \frac{1-\sigma}{\sigma}$, or equivalently using a smaller $\sigma' < \sigma$ value when setting $\gamma := \log \frac{1-\sigma'}{\sigma'}$. We opt for the latter as a means of intentionally biasing $\gamma$ to make the approach more robust for application. Figures 5.2 and 5.3 in Section 5.4 show the effect this proposed robustness modification has, demonstrating indeed that choosing a larger $\gamma$ via intentionally using a smaller $\sigma$ yields positive results. If the true noise level is $\sigma$, our experiments demonstrate that setting to roughly $\gamma := \frac{1-0.75\sigma}{0.75\sigma}$ has a positive effect on performance.

5.4 Empirical Results

This section contains results from implementing the previously described method and comparing it against other denoising approaches. Datasets and code are available on the first author’s GitHub for the purpose of easy reproducibility.

5.4.1 Datasets and Setup

In this subsection, we present empirical results obtained by implementing our model on a quantum annealer, D-Wave’s Advantage_system4.1, which has 5000 qubits and enables embedding of a complete bipartite graph of size $172 \times 172$. Hence, we use $12 \times 12$ pixel images here so that the visible layer is of size 144. We test the method on two different datasets with very differently structured data.

The first dataset is a $12 \times 12$ version of the well-known MNIST dataset [75], created by downsizing the original dataset with nearest-neighbor image downscaling and binarizing pixels. The second dataset we use is a $12 \times 12$ pixel Bars-and-Stripes (BAS) dataset, as has been used in closely related work [70, 38], in which the authors used a smaller $8 \times 8$ version of BAS in order to accommodate a 2000 qubit machine, so we implement a larger $12 \times 12$ version for the 5000 qubit machine we use. Each image consists of binary pixels with either each row or each column sharing the same values, so that each image consists of either “bars” or “stripes”. Some examples of noise-free,
noisy, and denoised images across different noise levels are presented in figure 5.1.

For both datasets we train the RBM by using the classical Contrastive Divergence algorithm first presented in [58], and as described in subsection 5.2.1. The number of hidden units was set to 50 and 64 for BAS and MNIST, respectively. For both datasets, we used learning rate of 0.01, batch size of 50, and 150 epochs as the training hyperparameters. For the BAS data, 4000 images were generated as training data, and 1000 as test data, while for MNIST, we simply used the full MNIST provided training set of 60,000 images and test set of 10,000 images. Noisy images were generated by adding salt-and-pepper noise of level $\sigma$ to images from the test dataset. Given a noisy image, we are then able to embed and solve the resulting denoising QUBO of (5.7) onto a D-Wave
quantum annealer, Advantage_system4.1. A function of D-Wave’s Ocean software, find_embedding, is utilized to find appropriate mappings from variables in a QUBO to physical qubits on D-Wave’s Pegasus graph. A variable in QUBO is often mapped to multiple physical qubits, called chain, that are strongly connected to each other to behave like a single variable. A mapping can be used for every noisy images for each dataset, since their QUBO have the same graph structure. We have prepared in advance 50 sets of the different mappings for each dataset and choose a mapping from the pool at random to embed QUBO of each image. This random selection is done to avoid possible artificial effects on the denoising performance from using only a particular mapping. Parameters for embedding and annealing, i.e., chain_strength and annealing_time, are tuned to maximize the performance. In particular, we set chain_strength as the product of a coefficient $c_0$ and the maximum abstract value among the elements of each QUBO matrix, where we tune $c_0$. The adopted values of the parameters are different between MNIST and BAS but the same values for all the range of $\sigma$. We set $(c_0, \text{annealing\_time}) = (0.6, 50 \mu s), (0.5, 40 \mu s)$ for BAS and MNIST, respectively. The number num_reads of reads of annealing is 100 for each noisy image. We calculate the average of solution of each pixel over the reads to approximate Eq. (5.11) and use it to evaluate the overlap that is proportion of pixels in denoised images that matched the original image. We denoise 200 noisy images for each $\sigma$, which are randomly selected from the pool of test images for each sigma. Note also that for each value of sigma, the different methods compared use the same set of (randomly selected) noisy test images.

5.4.2 Results with Quantum Annealing

Figures 5.2 and 5.3 first investigate the robust choice of $\gamma$ as discussed in Section 5.3.3. This is done by using a biased value of $\sigma$ when setting $\gamma = \log \frac{1-\sigma}{\sigma}$, instead setting $\gamma := \log \frac{1-b\sigma}{b\sigma}$ for some bias factor $b$. The denoising performance for $b \in \{1.25, 1, 0.75, 0.5\}$ are shown, with 95% confidence intervals obtained by bootstrapping. Note that using a bias factor $b = 1$ means using the true value of $\sigma$ for determining $\gamma$.

Based on the empirical performance, using a bias factor of around 0.75 seems to give an improved performance compared to using a bias factor of 1 in both data
Figure 5.2: Proportion of pixels in denoised MNIST images that matched the original image, for different denoising methods with 95% CI error bars.

Figure 5.3: Proportion of pixels in denoised BAS images that matched the original image, for different denoising methods with 95% CI error bars.
sets. A bias factor of 0.5 seems to perform quite well across most noise regimes as well, with largely overlapping confidence regions to the 0.75 parameter setting, though in the low-noise setting for the BAS dataset we observe an adverse effect. The authors thus suggest a setting of 0.75 for the bias factor.

Next, in figures 5.4 and 5.5, we compare our method to popular other denoising methods for binary images on the 12 × 12 MNIST and bars-and-stripes datasets, respectively, across different noise levels. When comparing to other methods, a crucial factor is that we choose \( \gamma \) based off of \( \sigma \), but in practice \( \sigma \) may be unknown. In light of this, we include two versions of our method in these comparisons. First, we use our method with \( \gamma := \log \frac{1-\sigma}{\sigma} \), using the true value of \( \sigma \) without introducing the recommended bias factor. Secondly, we simulate the situation in which the true \( \sigma \) is unknown, and instead we only have a guess for \( \sigma \). To simulate having an approximate guess for \( \sigma \), for each image afflicted by noise of level \( \sigma \), we sample \( \sigma' \) uniformly from an interval of size \( \sigma/2 \) centered at sigma. We then set \( \gamma := \log \frac{1-0.75\sigma'}{0.75\sigma'} \), using a bias factor of 0.75 on with this “guessed” value of \( \sigma \). This is a significantly more realistic way of testing our method, since it gives an idea of how well the method may perform when the true noise level present in the noisy images is unknown and must be guessed. Our implementation here only assumes that the practitioner roughly knows the magnitude of the noise. For example, if the true noise is \( \sigma = 0.2 \), here we sample \( \sigma' \) uniformly from \([0.15, 0.25]\) to simulate the guess.

We compare our method to Gibbs denoising with an RBM [102, section 3.2], median filtering [59], Gaussian filtering [101, chapter 5], and a graph-cut method [48] for denoising. For the Gibbs denoising, we use the same well-trained RBM as for our QUBO-based method, and parameters of the method were carefully tuned for best performance to use 20 Gibbs iterations to then construct the denoised image as the exponentially weighted average of the samples with decay factor 0.8. Notably, as Gibbs-based denoising also requires a well-trained RBM, this method incurs the same computational overhead of training an RBM as our method does. However, it has the disadvantage of requiring
Figure 5.4: Proportion of pixels in denoised MNIST images that matched the original image, for different denoising methods with 95% CI error bars.

Figure 5.5: Proportion of pixels in denoised BAS images that matched the original image, for different denoising methods with 95% CI error bars.
careful tuning of the hyperparameters of the number of Gibbs iterations and decay factor to use, whereas our method of picking $\gamma$ is much more straightforward and shows good results without tuning. For the graph-cut method, the recommended parameter setting in the reference of $\beta = 0.5$ is used. The median filter, Gaussian filter, and Gibbs denoising (excluding the overhead of training the RBM) each have complexity $O(n)$, where $n$ is the number of pixels, whereas the graph-cut method has complexity $O(n^3)$ since a maximum-flow problem is solved on a graph whose nodes are the pixels of the image. Keeping the annealing time and number of reads as constant, the scaling of our method is also $O(n)$. We forego wall-time here, since the software implementations we compare against are specialized for large problems, so comparing walltime for the small problems that can be implemented on current quantum annealers may not be representative. However, we note that for the QUBO denoising as we use up to $50\mu s$ annealing time and 100 reads per image, denoising an image only takes a total of $5ms$ of annealing time in our case.

Results are summarized in figures 5.4 and 5.5. Overall, the QUBO-based method performs quite strongly. Across all noise regimes in the MNIST data, and in most noise regimes in the bars-and-stripes dataset, the method outperforms the others. In particular, for the MNIST data the 95% confidence region for the QUBO method entirely dominates the others. Indeed, we see the good performance that our analysis from Section 5.3 suggests, even when the true $\sigma$ is unknown and instead guessed. Using a guessed $\sigma$ and the robustness modification of Section 5.3.3 makes the method perform as well (if not slightly better) as knowing the true $\sigma$ without the robustness modification. Only in the noise regime of $\sigma \geq 0.2$ in the BAS data does Gibbs denoising outperform our method.

### 5.4.3 Testing on Larger Images

Though we see the the straightforward implementability of our method on quantum annealers as a strong positive, a current drawback on using QAs is the limited data size that can be handled to accomodate their still small qubit capacities. Of course we can still instead test our method on larger datasets by obtaining solutions to the
denoising QUBO (5.6) using other means. In Figure 5.6, we implement our method on a binarized version of the popular MNIST dataset [75] by using simulated annealing [67] to find solutions to (5.6). We particularly choose to test on the full-size MNIST dataset since we could only use a downscaled version on the QA due to size limitations on the input data, so this experiment serves to test our method without this downscaling. All methods are implemented as described in 5.4.1, and again for our method we use a guessed $\sigma$ to simulate the unknown $\sigma$ case and bias the guess for robustness.

5.5 Chapter Conclusions and Future Work

This chapter investigated an image denoising framework via a penalty-based QUBO denoising objective that shows promise both theoretically through its statistical properties and practically through its empirical performance together with the proposed robustness modification. The method is well-suited for implementability on a quantum annealer,
providing an important application of QAs within machine learning through the fundamental image denoising task. Good results are still obtained on larger datasets when the QUBO is only classically approximated by simulated annealing instead, revealing the approach to be promising even in the absence of QAs. As RBMs form a core building block of many deep generative models such as deep Boltzmann machines or deep belief networks [47], a natural next step is to attempt to incorporate this approach into these more complex models, though current hardware limitations on existing quantum annealers are restrictive. Further, since our method takes advantage of QAs for the denoising step, further research into making use of QAs for the training process of RBMs would yield a full image denoising model where both the model training and image denoising make use of QA.

5.5.1 Different Boltzmann Machine Architectures for QA

A very exciting direction to pursue would be to consider different Boltzmann Machine architectures for using the model presented in this chapter. In this work, we used restricted Boltzmann Machines, which have a fully connected bipartite architecture. This is useful in a classical computing framework, because one can sample from this RBM by using Gibbs sampling. However, if one uses QA to sample, as studied e.g. in [100] and more recently in [97], this reason for using the RBM architecture no longer applies. This raises the following question: What architectures are best used for Boltzmann machine implementations on quantum annealers? In fact, with the RBM architecture, on the 5000 qubit machine we used, due to the chip architecture it is only possible to embed an RBM with 174 visible and 174 hidden units, so the RBM architecture is perhaps even impractical. Due to the mismatch between the RBM bipartite architecture and then D-Wave chip architecture, only a small RBM can be embedded into the quantum annealing hardware. The Pegasus topology of the D-Wave Advantage’s chip is shown in figure 5.7.

A desirable architecture would need to satisfy the following:

1. Have sufficient connectivity between visible and hidden nodes, or even visible-to-visible or hidden-to-hidden nodes, to perform well, i.e. to be flexible enough to
learn the distributions of real datasets.

2. Allow for an efficient and scalable embedding into the chip architecture to be able to use Boltzmann machines with a larger amount of visible nodes.

To achieve this, of course the architecture of the chip would need to be carefully taken into account. The prospect of using chip-specific architectures to embed much larger Boltzmann Machines into quantum annealers to process larger datasets is very exciting and should be pursued in the future.

Chapter Acknowledgments

The early stage of the work described in this chapter is based on the work in the g-RIPS Sendai 2021 program. Many thanks Y. Araki, E. Escobar, T. Mihara, V. Q. H. Huynh, H. Kodani, A. T. Lin, M. Shirane, Y. Susa, and H. Suito for collaboration in the program. The authors also acknowledge H. Kobayashi and M. Sato for the use of the computing environment in the program. P.K. thanks Y. Sukurdeep for helpful feedback and discussions.
Chapter 6

Quantum Distributed Algorithms

Chapter Abstract

The CONGEST and CONGEST-CLIQUE models have been carefully studied to represent situations where the communication bandwidth between processors in a network is severely limited. Messages of only $O(\log(n))$ bits of information each may be sent between processors in each round, where $n$ is the number of nodes in the network. The quantum versions of these models allow the processors instead to communicate and compute with quantum bits under the same bandwidth limitations. This leads to the following natural research question: What problems can be solved more efficiently in these quantum models than in the classical ones? Building on existing work, we contribute to this question in two ways. Firstly, we present two algorithms in the Quantum CONGEST-CLIQUE model of distributed computation that succeed with high probability; one for producing an approximately optimal Steiner Tree, and one for producing an exact directed minimum spanning tree, each of which uses $\tilde{O}(n^{1/4})$ rounds of communication and $\tilde{O}(n^{9/4})$ messages, where $n$ is the number of nodes in the network. The algorithms thus achieve a lower asymptotic round and message complexity than any known algorithms in the classical CONGEST-CLIQUE model. At a high level, we achieve these results by combining classical algorithmic frameworks with quantum subroutines. An existing framework for using a distributed version of Grover’s search
algorithm to accelerate triangle finding lies at the core of the asymptotic speedup. Secondly, we carefully characterize the constants and logarithmic factors involved in our algorithms as well as related algorithms, otherwise commonly obscured by $\tilde{O}$ notation. The analysis shows that some improvements are needed to render both our and existing related quantum and classical algorithms practical, as their asymptotic speedups only help for very large values of $n$. The contributions of this chapter are also available in [66].

6.1 Introduction

The classical CONGEST-CLIQUE Model (cCCM henceforth) in distributed computing has been carefully studied as a model central to the field, e.g., [69, 95, 43, 76, 39, 89]. In this model, processors in a network solve a problem whose input is distributed across the nodes under significant communication limitations, described in detail in §6.2. For example, a network of aircraft or spacecraft, satellites, and control stations, all with large distances between them, may have severely limited communication bandwidth to be modeled in such a way. The quantum version of this model, in which quantum bits can be sent between processors, the quantum CONGEST-CLIQUE Model (qCCM), as well as the quantum CONGEST model, have been the subject of recent research [60, 23, 3, 41] in an effort to understand how quantum communication may help in these distributed computing frameworks. For the quantum CONGEST Model, however, [41] showed that many problems cannot be solved more quickly than in the classical model. These include shortest paths, minimum spanning trees, Steiner trees, min-cut, and more; the computational advantages of quantum communication are thus severely limited in the CONGEST setting, though a notable positive result is sub-linear diameter computation in [74]. No comparable negative results exist for the qCCM, and in fact, [60] provides an asymptotic quantum speedup for computing all-pairs shortest path (APSP henceforth) distances. Hence, it is apparent that the negative results of [41] cannot transfer over to the qCCM, so investigating these problems in the qCCM presents an opportunity for contribution to the understanding of how quantum communication may help in these distributed computing frameworks. In this paper, we contribute to this
understanding by formulating algorithms in the qCCM for finding approximately optimal Steiner trees and exact directed minimum spanning trees using $\tilde{O}(n^{1/4})$ rounds – asymptotically fewer rounds than any known classical algorithms. This is done by augmenting the APSP algorithm of [60] with an efficient routing table scheme, which is necessary to make use of the shortest paths information instead of only the APSP distances, and using the resulting subroutine with existing classical algorithmic frameworks. Beyond asymptotics, we also characterize the complexity of our algorithms as well as those of [60, 22, 95, 43] to include the logarithmic and constant factors involved to estimate the scales at which they would be practical, which was not included in the previous work. It should be noted that, like APSP, these problems cannot see quantum speedups in the CONGEST (non-clique) setting as shown in [41]. Our Steiner tree algorithm is approximate and based on a classical polynomial-time centralized algorithm of [71]. Our directed minimum spanning tree problem algorithm follows an approach similar to [43], which effectively has its centralized roots in [77].

### 6.2 Background and Setting

This section provides the necessary background for our algorithms’ settings and the problems they solve.

#### 6.2.1 The CONGEST and CONGEST-CLIQUE Models of Distributed Computing

In the standard CONGEST model, we consider a graph of $n$ processor nodes whose edges represent communication channels. Initially, each node knows only its neighbors in the graph and associated edge weights. In rounds, each processor node executes computation locally and then communicates with its neighbors before executing further local computation. The congestion limitation restricts this communication, with each node able to send only one message of $O(\log(n))$ classical bits in each round to its neighbors, though the messages to each neighbor may differ. Since there are $n$ nodes, assigning them ID labels 1, ..., $n$ means the binary encoding size of a label is $\lceil \log(n) \rceil$ bits – i.e., each message in CONGEST of $O(\log n)$ bits can contain roughly the amount
of information to represent one node ID. In the cCCM, we separate the communication
graph from the problem input graph by allowing all nodes to communicate with each
other, though the same $O(\log(n))$ bits-per-message congestion limitation remains, and
the input graph is still a graph over the $n$ processor nodes of the network. Hence, a
processor node could send $n-1$ different messages to the other $n-1$ nodes in the graph,
with a single node distributing up to $O(n \cdot \log(n))$ bits of information in a single round.
Taking advantage of this way of dispersing information to the network is paramount in
many efficient CONGEST-CLIQUE algorithms. The efficiency of algorithms in these
distributed models is commonly measured in terms of the round complexity, the number
of rounds of communication used in an algorithm to solve the problem in question.
Considering asymptotic behavior of algorithms as $n \to \infty$ then means that both the
input graph and the number of processors is tending to $\infty$. A good overview of these
distributed models can be found in [44].

6.2.2 Quantum Versions of CONGEST and CONGEST-CLIQUE

The quantum models we work in are obtained via the following modification: Instead
of restricting to messages of $O(\log(n))$ classical bits, we allow messages to consist of
$O(\log(n))$ quantum bits, qubits. While a single classical bit can be in the state 0 or
1, and a system of $n$ classical bits can be represented by the state space $\{0, 1\}^n$, recall
from section 4.1.2 that the state of a single quantum bit, or qubit, can be represented by
a unit-length vector in a two-dimensional complex Hilbert space $\mathbb{Q}$, with basis vectors
commonly written as $|0\rangle$ and $|1\rangle$. A qubit in state $|\psi\rangle$ can be measured with respect to
some orthonormal basis $|v_1\rangle, |v_2\rangle$, and the result of that measurement will be $|v_1\rangle$ with
probability $\langle \psi | v_1 \rangle^2$, and $|v_2\rangle$ with probability $\langle \psi | v_2 \rangle^2$. Unit vectors can then represent
the state space of a register of $n$ qubits in the $n$-fold tensor product of the single qubit
state-spaces $\mathbb{Q} \otimes \mathbb{Q} \otimes \cdots \otimes \mathbb{Q}$. Since these must always remain unit vectors (for the
measurement probabilities to sum to 1), operations on qubits are represented by $n \times n$
unitary matrices. Currently, different physical implementations of qubits, such as su-
perconducting qubits, trapped ion qubits, photonic qubits, and more, have been built
and are the subjects of ongoing research. On a high level, the intuitive reason quantum
computers can give advantages in specific problems is that quantum interference effects
can be leveraged to create canceling effects among non-solutions to a problem. The asymptotic quantum speedup for our algorithm stems from taking advantage of a distributed version of Grover’s search algorithm (see Algorithm 3 for the non-distributed version), which can be used to speed up distributed triangle finding in graphs to allow for faster shortest-path computations.

We formally define the qCCM, the setting for our algorithms, as follows:

**Definition 85 (Quantum CONGEST-CLIQUE).** The Quantum CONGEST-CLIQUE Model (qCCM) is a distributed computation model in which an input graph $G = (V, E, W)$ is distributed over a network of $n$ processors, where each processor is represented by a node in $V$. Each node is assigned a unique ID number in $\{1, 2, \ldots, n\}$. Time passes in rounds, each of which consists of the following:

1. Each node may execute unlimited local computation.

2. Each node may send a message consisting of either a register of $O(\log n)$ qubits or a string of $O(\log n)$ classical bits to each other node in the network. Each of those messages may be distinct.

3. Each node receives and saves the messages the other nodes send it.

The input graph $G$ is distributed across the nodes as follows: Each node knows its own ID number, the ID numbers of its neighbors in $G$, the number of nodes $n$ in $G$, and the weights corresponding to the edges it is incident upon. The output solution to a problem must be given by having each node $v \in V$ return the restriction of the global output to $\mathcal{N}_G(u) := \{v : uv \in E\}$, its neighborhood in $G$. No entanglement is shared across nodes initially.

This is an analog of the eCCM, except that quantum bits may be sent in place of classical bits. To clarify the output requirement, in the Steiner tree problem, we require node $u$ to output the edges of the solution tree that are incident upon $u$. Since many messages in our algorithms need not be sent as qubits, we define the qCCM slightly unconventionally, allowing either quantum or classical bits to be sent. We specify those
that may be sent classically. However, even without this modification, the quantum versions of CONGEST and cCCM are at least as powerful as their classical counterparts. This is because any \( n \)-bit classical message can be instead sent as an \( n \)-qubit message of unentangled qubits; for a classical bit reading 0 or 1, we can send a qubit in the state \( |0\rangle \) or \( |1\rangle \) respectively, and then take measurements with respect to the \( \{ |0\rangle , |1\rangle \} \) basis to read the same message the classical bits would have communicated. Hence, one can also freely make use of existing classical algorithms in the qCCM. Further, the assumption that IDs are in \([n]\), with \( n \) known, is not necessary but is convenient; without this assumption, we could have all nodes broadcast their IDs to the entire network and then assign a new label in \([n]\) to each node according to an ordering of the original IDs, resulting in our assumed situation.

**Remark 86.** Definition 85 does not account for how the information needs to be stored. In this paper, it suffices for all information regarding the input graph to be stored classically as long as there is quantum access to that data. We provide some details on this in §6.8.4 of the appendix.

**Remark 87.** No entanglement being shared across nodes initially in definition 85 results in quantum teleportation not being a trivial way to solve problems in the qCCM.

**Example 88.** To provide some intuition on how allowing communication through qubits in this distributed setting can be helpful, we now describe and give an example of distributed Grover search, first described in [74]. The high-level intuition for why quantum computing gives an advantage for search is that quantum operations use quantum interference effects to have canceling effects among non-solutions. Grover search has a generalization called “amplitude amplification” we will use; see [93] for details on these algorithms. Now, for a processor node \( u \) in the network and a Boolean function \( g : X \rightarrow \{ 0, 1 \} \), suppose there exists a classical procedure \( C \) in the cCCM that allows \( u \) to compute \( g(x) \), for any \( x \in X \) in \( r \) rounds. In other words, for some \( x \in X \), within \( r \) rounds of communication with other nodes in the network, the node \( u \) is able to determine \( g(x) \). The quantum speedup will come from computing \( C \) in a quantum superposition, which enables \( g \) to be evaluated with inputs in superposition so that amplitude
amplification can be used for inputs to \( g \). Let \( A^i : \{ x \in X : g(x) = i \} \), for \( i = 0, 1 \), and suppose that \( 0 < |A^i| \leq |X|/2 \). Then classically, node \( u \) can find an \( x \in A^1 \) in \( \Theta(r|X|) \) rounds by checking each element of \( X \). Using the quantum distributed Grover search of [74] enables \( u \) to find such an \( x \) with high probability in only \( \tilde{O}(r\sqrt{|X|}) \) rounds by evaluating the result of computing \( g \) on a superposition of inputs.

We illustrate this procedure in an example case where a node \( u \) wants to inquire whether one of its edges \( uv \) is part of a triangle in \( G \). We first describe a classical procedure for this, followed by the corresponding quantum-distributed search version.

For \( v \in \mathcal{N}_G(u) \), where \( \mathcal{N}_G(u) \) is the set of nodes in \( G \) that share an edge with \( u \), denote by \( \mathcal{I}_v : V \to \{0, 1\} \) the indicator function of \( \mathcal{N}_G(v) \), and by \( g_{uv} : \mathcal{N}_G(u) \to \{0, 1\} \) its restriction to inputs in \( \mathcal{N}_G(u) \). Classically, node \( u \) can evaluate \( g_{uv}(w) \) in two rounds for any \( w \in \mathcal{N}_G(u) \) by sending the ID of \( w \) (of length \( \log n \)) to \( v \), and having \( v \) send back the answer \( \mathcal{I}_v(w) \). Then \( u \) can check \( g_{uv}(w) \) for each \( w \in \mathcal{N}_G(u) \) one at a time to determine whether \( uv \) is part of a triangle in \( G \) or not in \( 2 \cdot |\mathcal{N}_G(u)| \) rounds.

For the distributed quantum implementation, \( u \) can instead initialize a register of \( \log n \) qubits as \( |\psi\rangle_0 := \frac{1}{\sqrt{|\mathcal{N}_G(u)|}} \sum_{x \in \mathcal{N}_G(u)} |x\rangle \), all the inputs for \( g_{uv} \) in equal superposition. To do a Grover search, \( u \) needs to be able to evaluate \( g_{uv} \) with inputs \( |\psi\rangle \) in superposition. For the quantum implementation of \( C \), \( u \) sends a quantum register in state \( |\psi\rangle|0\rangle \) to node \( v \), and has node \( v \) evaluate a quantum implementation of \( \mathcal{I}_v \), which we will consider as a call to an oracle mapping \( |x\rangle|0\rangle \) to \( |x\rangle|\mathcal{I}_v(x)\rangle \) for all \( x \in V \). Node \( v \) sends back the resulting qubit register, and node \( u \) has evaluated \( g_{uv}(|\psi\rangle) \) in 2 rounds. Now, since \( u \) can evaluate \( g_{uv} \) in superposition, node \( u \) may proceed using standard amplitude amplification, using 2 rounds of communication for each evaluation of \( g_{uv} \), so that \( u \) can find an element \( w \in \mathcal{N}_G(u) \) satisfying \( g_{uv}(w) = 1 \) with high probability in \( \tilde{O}(r\sqrt{|\mathcal{N}_G(u)|}) \) rounds if one exists. We note that in this example, \( v \) cannot execute this procedure by itself since it does not know \( \mathcal{N}_G(u) \) (and sending this information to \( v \) would take \( |\mathcal{N}_G(u)| \) rounds), though it is able to evaluate \( \mathcal{I}_v \) in superposition for any \( w \in \mathcal{N}_G(u) \). For any classical procedure \( C \) evaluating a different function from this specific \( g \) (that can be implemented efficiently classically and, therefore, translated to an efficient quantum implementation), the same idea results in the square-root advantage.
to find a desired element such that \( g \) evaluates to 1.

### 6.2.3 Notation and Problem Definitions

For an integer-weighted graph \( G = (V, E, W) \), we will denote \( n := |V|, m := |E|, \) and \( W_e \) the weight of an edge \( e \in E \) throughout the paper. Let \( \delta(v) \subset V \) be the set of edges incident on node \( v \), and \( N_G(u) := \{v : uv \in E\} \) the neighborhood of \( u \in G \). Denote by \( d_G(u, v) \) the shortest-path distance in \( G \) from \( u \) to \( v \). For a graph \( G = (V, E, W) \) two sets of nodes \( U \) and \( U' \), let \( P_G(U, U') := \{uv \in E : u \in U, w \in U'\} \) be the set of edges connecting \( U \) to \( U' \). Let \( P(U) := P(U, U) \) as shorthand. All logarithms will be taken with respect to base 2, unless otherwise stated.

**Definition 89** (Steiner Tree Problem). Given a weighted, undirected graph \( G = (V, E, W) \), and a set of nodes \( Z \subset V \), referred to as Steiner Terminals, output the minimum weight tree in \( G \) that contains \( Z \).

**Definition 90** (Approximate Steiner Tree). For a Steiner Tree Problem with terminals \( Z \) and solution \( S_{OPT} \) with edge set \( E_{S_{OPT}} \), a tree \( T \) in \( G \) containing \( Z \) with edge set \( E_T \) such that

\[
\sum_{uv \in E_T} W_{uv} \leq r \cdot \sum_{uv \in E_{S_{OPT}}} W_{uv}
\]

is called an approximate Steiner Tree with approximation factor \( r \).

**Definition 91** (Directed Minimum Spanning Tree Problem (DMST)). Given a directed, weighted graph \( G = (V, E, W) \) and a root node \( r \in V \), output the minimum weight directed tree \( T^* \) in \( G \) such that there exists a directed path in \( T^* \) from \( r \) to any other node of \( G \). This is also known as the minimum weight arborescence problem.

### 6.3 Contributions

We provide an algorithm for the qCCM that produces an approximate Steiner Tree with high probability (w.h.p.) in \( \tilde{O}(n^{1/4}) \) rounds and an algorithm that produces an exact Directed Minimum Spanning Tree w.h.p. in \( \tilde{O}(n^{1/4}) \) rounds. To do this, we enhance the quantum APSP algorithm of [60] in an efficient way to compute not only
APSP distances but also the corresponding routing tables (described in §6.4) that our algorithms rely on. Further, in addition to these $\tilde{O}$ results, in sections 6.4.7, 6.5.4, and 6.6.3, we characterize the constants and logarithmic factors involved in our algorithms as well as related classical algorithms to contribute to the community’s understanding of their implementability. This reveals that the factors commonly obscured by $\tilde{O}$ notation in related literature, especially the logarithms, have a severe impact on practicality.

We summarize the algorithmic results in the following two theorems:

**Theorem 92.** There exists an algorithm in the Quantum CONGEST-CLIQUE model that, given an integer-weighted input graph $G = (V, E, W)$, outputs a $2(1 - 1/l)$ approximate Steiner Tree with probability of at least $1 - \frac{1}{\text{poly}(n)}$, and uses $\tilde{O}(n^{1/4})$ rounds of computation, where $l$ denotes the number of terminal leaf nodes in the optimal Steiner Tree.

**Theorem 93.** There exists an algorithm in the Quantum CONGEST-CLIQUE model that, given a directed and integer-weighted input graph $G = (V, E, W)$, produces an exact Directed Minimum Spanning Tree with high probability, of at least $1 - \frac{1}{\text{poly}(n)}$, and uses $\tilde{O}(n^{1/4})$ rounds of computation.

### 6.4 APSP and Routing Tables

We first describe an algorithm for the APSP problem with routing tables in the qCCM, for which we combine an algorithm of [60] with a routing table computation from [111]. For this, we reduce APSP with routing tables to triangle finding via distance products as in [22].

#### 6.4.1 Distance Products and Routing Tables

**Definition 94.** A routing table for a node $v$ is a function $R_v : V \rightarrow V$ mapping a vertex $u$ to the first node visited in the shortest path going from $v$ to $u$ other than $v$ itself.

**Definition 95.** The distance product between two $n \times n$ matrices $A$ and $B$ is defined
As the $n \times n$ matrix $A \star B$ with entries:

$$(A \star B)_{ij} = \min_k \{A_{ik} + B_{kj}\}. \quad (6.1)$$

The distance product is also sometimes called the min-plus or tropical product. For shortest paths, we will repeatedly square the graph adjacency matrix with respect to the distance product. For an $n \times n$ matrix $W$ and an integer $k$, let us denote $W^{k,\star} := W \star (W \star (\ldots (W \star W)\ldots ))$ as the $k^{th}$ power of the distance product. For a graph $G = (V, E, W)$ with weighted adjacency matrix $W$ (assigning $W_{uv} = \infty$ if $uv \notin E$), $W^{k,\star}_{uv}$ is the length of the shortest path from $v$ to $u$ in $G$ using at most $k$ hops. Hence, for any $N \geq n$, $W^{N,\star}$ contains all the shortest path distances between nodes in $G$. As these distance products obey standard exponent rules, we may take $N = 2^{\lceil \log n \rceil}$ to recursively compute the APSP distances via taking $\lceil \log n \rceil$ distance product squares:

$$W^{2,\star} = W \star W, \quad W^{4,\star} = (W^{2,\star})^{2,\star}, \ldots, \quad W^{2^{\lceil \log n \rceil} - 1,\star} = (W^{2^{\lceil \log n \rceil} - 1,\star})^{2,\star}. \quad (6.2)$$

This procedure reduces computing APSP distances to computing $\lceil \log n \rceil$ distance products. In the context of the CONGEST-CLIQUE model, each node needs to learn the row of $W^n$ that represents it. As we also require nodes to learn their routing tables, we provide a scheme in §6.4.3 that is well-suited for our setting to extend [60] to also compute routing tables.

### 6.4.2 Distance Products via Triangle Finding

Having established reductions to distance products, we turn to their efficient computation. The main idea is that we can reduce distance products to a binary search in which each step in the search finds negative triangles. This procedure corresponds to [61, Proposition 2], which we describe here, restricting to finding the distance product square needed for Eq. (6.2).

A negative triangle in a weighted graph is a set of edges $\Delta^- = (uv, vw, wu) \subset E^3$ such that $\sum_{e \in \Delta^-} W_e < 0$. Let us denote the set of all negative triangles in a graph $G$ as $\Delta^-_G$. Specifically, we will be interested in each node $v$ being able to output edges
vu ∈ δ(v) such that vu is involved in at least one negative triangle in G. Let us call this problem **FindEdges**, and define it formally as:

**FindEdges**

Input: An integer-weighted (directed or undirected) graph G = (V, E, W) distributed among the nodes, with each node v knowing NG(v), as well as the weights Wvu for each u ∈ NG(v).

Output: For each node v, its output is all the edges vu ∈ E that are involved in at least one negative triangle in G.

**Proposition 96.** If **FindEdges** on an n-node integer-weighted graph G = (V, E, W) can be solved in T(n) rounds, then the distance product A ⋆ B of two n × n matrices A and B with entries in [M] can be computed in T(3n) ⋅ ⌈log₂(2M)⌉ rounds.

**Proof.** Let A and B be arbitrary n × n integer-valued matrices, and D be an n × n matrix initialized to 0. Let each u ∈ V simulate three copies of itself, u1, u2, u3, writing V1, V2, V3 as the sets of copies of nodes in V. Consider the graph G′ = (V1 ∪ V2 ∪ V3, E′, W′), by letting uivj ∈ E′ for ui ∈ Vi, vj ∈ Vj, i ≠ j, taking W′uivj = Auv for u1 ∈ V1, v2 ∈ V2, W′u2v3 = Buv for u2 ∈ V2, v3 ∈ V3, and W′u3v1 = Duv for u3 ∈ V3, v1 ∈ V1. An edge zv is part of a negative triangle in G′ exactly whenever

\[
\min_{u ∈ V} \{A_{vu} + B_{uz}\} < -D_{zv}.
\]

Assuming we can compute **FindEdges** for a k-node graph in T(n) rounds, with a non-positive matrix D = 0 initialized we can apply simultaneous binary searches on Dzv, with values between {−2M, 0}, updating it for each node v after each run of **FindEdges** to find \(\min_{u ∈ V} \{A_{vu} + B_{uz}\}\) for every other node z in T(3n) ⋅ [log(maxv,z∈V {minu∈V {A_{vu} + B_{uz}}})] rounds, since G′ is a tripartite graph with 3n nodes.

**Remark 97.** This procedure can be realized in a single n-node distributed graph by
letting each node represent the three copies of itself since $G'$ is tripartite. The $T(3n)$ stems from each processor node possibly needing to send one message for each node it is simulating in each round of \texttt{FindEdges}. If bandwidth per message is large enough (3 times the bandwidth needed for solving \texttt{FindEdges} in $T(n)$ rounds), then this can be done in $T(n)$ rounds.

So for this binary search, each node $v$ initializes and locally stores $D_{vz} = 0$ for each other $z \in V$, after which we solve \texttt{FindEdges} on $G'$. The node then updates each $D_{vz}$ according to whether or not the edge copies of $vz$ were part of a negative triangle in $G'$, after which \texttt{FindEdges} is computed with the updated values for $D$. This is repeated until all the $\min_{u \in V}\{A_{vu} + B_{uz}\}$ have been determined.

6.4.3 Routing Tables via Efficient Computation of Witness Matrices

For the routing table entries, we also need each node $v$ to know the intermediate node $u$ that is being used to attain $\min_{u \in V}\{W_{vu} + W_{uz}\}$.

\textbf{Definition 98.} For a distance product $A \ast B$ of two $n \times n$ matrices $A, B$, a witness matrix $C$ is an $n \times n$ matrix such that

$$C_{ij} \in \arg\min_{k \in [n]} \{A_{ik} + B_{kj}\}$$

Put simply, a witness matrix contains the intermediate entries used to attain the values in the resulting distance product. We present here a simple way of computing witness matrices along with the distance product by modifying the matrix entries appropriately, first considered by [111]. The approach is well-suited for our algorithm, as we only incur $O(\log n)$ additional calls to \texttt{FindEdges} for a distance product computation with a witness matrix.

For an $n \times n$ integer matrix $W$, obtain matrices $W'$ and $W''$ by taking $W'_{ij} = nW_{ij} + j - 1$ and $W''_{ji} = nW_{ji}$. Set $K = W' \ast W''$.

\textbf{Claim 99.} With $W, W', W''$, and $K$ as defined immediately above,
(i) \[ \left\lfloor \frac{K}{n} \right\rfloor = W^{2,\star} \]

(ii) \((K \mod n) + 1 \) is a witness matrix for \(W^{2,\star}\).

The claim follows from routine calculations of the quantities involved and can be found in the Appendix, §6.8.1.

Hence, we can obtain witness matrices by simply changing the entries of our matrices by no more than a multiplicative factor of \(n\) and an addition of \(n\). Since the complexity of our method depends on the magnitude of the entries of \(W\) logarithmically, we only need logarithmically many more calls to \texttt{FindEdges} to obtain witness matrices along with the distance products, making this simple method well-suited for our approach. More precisely, we can compute \(W^{2,\star}\) with a witness matrix using \(\left\lceil \log \left(2n \cdot \max_{i,j} \{W^{2,\star}_{ij} < \infty\}\right) \right\rceil\) calls to \texttt{FindEdges}. We obtain the following corollary to proposition 96 to characterize the exact number of rounds needed:

**Corollary 100.** If \texttt{FindEdges} on an \(n\)-node integer-weighted graph \(G = (V, E, W)\) can be solved in \(T(n)\) rounds, then the distance product square \(W^{2,\star}\), along with a witness matrix \(H\), can be computed in \(T(3n) \cdot \lceil \log_2(n \cdot \max_{v,z \in V} \{\min_{u \in V} \{W_{vu} + W_{uz}\}\} + n) \rceil\) rounds.

**Proof.** This follows from claim 99 and proposition 96 upon observing that \(\max_{v,z \in V} \{\min_{u \in V} \{W'_{vu} + W''_{uz}\}\} \leq n \cdot \max_{v,z \in V} \{\min_{u \in V} \{W_{vu} + W_{uz}\}\} + n\).}

Once we obtain witness matrices along with the distance product computations, constructing the routing tables for each node along the way of computing APSP is straightforward. In each squaring of \(W\) in Eq. (6.2), each node updates its routing table entries according to the corresponding witness matrix entry observed. It is worth noting that these routing table entries need only be stored and accessed classically so that we avoid using unnecessary quantum data storage.
6.4.4 Triangle Finding

Given the results from sections 6.4.3 and 6.4.2, we have reduced finding both the routing tables and distance product to having each edge learn the edges involved in a negative triangle in the graph. This section will thus describe the procedure to solve the FindEdges subroutine. We state here a central result from [60]:

**Proposition 101.** There exists an algorithm in the quantum CONGEST-CLIQUE model that solves the FindEdges subroutine in $\tilde{O}(n^{1/4})$ rounds.

We will proceed to describe each step of the algorithm to describe the precise round complexity beyond the $\tilde{O}(n^{1/4})$ to characterize the constants involved in the interest of assessing the future implementability of our algorithms.

As a preliminary, we give a message routing lemma of [39] for the congested clique, which will be used repeatedly:

**Lemma 102.** Suppose each node in $G$ is the source and destination for at most $n$ messages of size $O(\log n)$ and that the sources and destinations of each message are known in advance to all nodes. Then all messages can be routed to their destinations in 2 rounds.

We introduce the subproblem FindEdgesWithPromise (FEWP henceforth). Let $\Gamma(u, v)$ denote the number of nodes $w \in V$ such that $(u, v, w)$ forms a negative triangle in $G$.

<table>
<thead>
<tr>
<th><strong>FEWP</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> An integer-weighted graph $G = (V, E, W)$ distributed among the nodes and a set $S \subset \mathcal{P}(V)$, with each node $v$ knowing $\mathcal{N}_G(v)$ and $S$.</td>
</tr>
<tr>
<td><strong>Promise:</strong> For each $uv \in S, \Gamma(u, v) \leq 90 \log n$.</td>
</tr>
<tr>
<td><strong>Output:</strong> For each node $v$, its output is the edges $vu \in S$ that satisfy $\Gamma(u, v) &gt; 0$.</td>
</tr>
</tbody>
</table>
We give here a description of the procedure of [60] to solve **FindEdges** given an algorithm \( A \) to solve **FEWP**. Let \( \varepsilon_A \) be the failure probability of the algorithm \( A \) for an instance of **FEWP**.

### FindEdgesViaFEWP

1. \( S := \mathcal{P}; M := \emptyset; i := 0. \)

2. WHILE \( 60 \cdot 2^i \log n \leq n \):
   
   a). Each node samples each of its edges with probability \( \sqrt{\frac{60 \cdot 2^i \log n}{n}} \), so that we obtain a distributed subgraph \( G' \) of \( G \) consisting of the sampled edges
   
   b). Run \( A \) on \((G', S)\). Denote the output by \( S' \).
   
   c). \( S \leftarrow S \setminus S'; M \leftarrow M \cup S; i \leftarrow i + 1. \)

3. Run \( A \) on \((G, S)\), and call \( S'' \) the output.

4. Output \( M \cup S \).

From step 2 of this above algorithm, it is straightforward to check that this requires a maximum of \( c_n := \lceil \log (\frac{n}{\log \log n}) \rceil + 1 \) calls to the \( A \) subroutine to solve **FEWP**. Further, it succeeds with probability at least \( 1 - c_n/n^3 - c_n/n^38 - (c_n + 1)\varepsilon_A \). We refer the reader to [60, §3] for the proof of correctness. We now turn toward constructing an efficient algorithm for **FEWP**.

To solve this subroutine, we must first introduce an additional labeling scheme over the nodes that will determine how the search for negative triangles will be split up to avoid communication congestion in the network. Assume for simplicity that \( n^{1/4}, \sqrt{n}, n^{3/4} \) are integers. Let \( \mathcal{M} = [n^{1/4}] \times [n^{1/4}] \times [\sqrt{n}] \). Clearly, \( |\mathcal{M}| = n \), and \( \mathcal{M} \) admits a total ordering lexicographically. Since we assume each node \( v_i \in V \) is labeled with unique integer ID \( i \in [n] \), \( v_i \) can select the element in \( \mathcal{M} \) that has place \( i \) in the lexicographic ordering of \( \mathcal{M} \) without communication occurring. Hence, each node \( v \in V \) is associated with a unique triple \((i, j, k) \in \mathcal{M}\). We will refer to the unique node
associated with \((i, j, k) \in M\) as node \(v_{(i,j,k)}\).

The next ingredient is a partitioning scheme of the space of possible triangles. Let \(\mathcal{U}\) be a partition of \(V\) into \(n^{1/4}\) subsets containing \(n^{3/4}\) nodes each, by taking

\[
U_i := \{v_j : j \in \{(i-1) \cdot n^{3/4}, \ldots, i \cdot n^{3/4}\}\}
\]

for \(i = 1, \ldots, n^{1/4}\), and \(\mathcal{U} := \{U_1, \ldots, U_{n^{1/4}}\}\). Apply the same idea to create a partition \(\mathcal{U}'\) of \(\sqrt{n}\) sets of size \(\sqrt{n}\), by taking

\[
U_i' := \{v_j : j \in \{(i-1) \cdot \sqrt{n}, \ldots, i \cdot \sqrt{n}\}\}
\]

for \(i = 1, \ldots, \sqrt{n}\), and \(\mathcal{U} := \{U_1, \ldots, U_{\sqrt{n}}\}\). Let \(\mathcal{V} = \mathcal{U} \times \mathcal{U} \times \mathcal{U}'\). Each node \(v_{(i,j,k)}\) can then locally determine its association with the element \((U_i, U_j, U_k')\) it is assigned to, so this assignment can be done in one round.

We present here the algorithm ComputePairs used to solve the FEWP subroutine.

---

**ComputePairs**

**Input:** An integer-weighted graph \(G = (V, E, W)\) distributed among the nodes, a partition of \(V \times V \times V\) of \((U_i, U_j, U_k')\) associated with each node as above, and a set \(S \subset \mathcal{P}(V)\) such that for \(uv \in S\), \(\Gamma(u, v) \leq 90 \log n\).

**Output:** For each node \(v\), its output is the edges \(vu \in S\) that satisfy \(\Gamma(u, v) > 0\).

1. Every node \(v_{(i,j,k)}\) receives the weights \(W_{uv}, W_{vw}\) for all \(uv \in \mathcal{P}(U_i, U_j)\) and \(vw \in \mathcal{P}(U_j, U_k')\).

2. Every node \(v_{(i,j,k)}\) constructs the set \(\Lambda_k(U_i, U_j) \subset \mathcal{P}(U_i, U_j)\) by selecting every \(uv \in \mathcal{P}(U_i, U_j)\) with probability \(10 \cdot \frac{\log n}{\sqrt{n}}\).

   If \(|\{v \in U_1 : uv \in \Lambda_k(U_i, U_j)\}| > 100n^{1/4}\log n\) for some \(u \in U_j\), abort the
Otherwise, \( v(i,j,k) \) keeps all pairs \( uv \in \Lambda_k(U_i, U_j) \cap S \) and receives the weights \( W_{uv} \) for all of those pairs. Denote those elements of \( \Lambda_k(U_i, U_j) \cap S \) as \( u^k_1 v^k_1, \ldots, u^k_m v^k_m \).

3: Every node \( v(i,j,k) \) checks for each \( l \in [m] \) whether there is some \( U \in U' \) that contains a node \( w \) such that \( (u^k_l, v^k_l, w) \) forms a negative triangle, and outputs all pairs \( u^k_l v^k_l \) for which a negative triangle was found.

With probability at least \( 1 - 2/n \), the algorithm ComputePairs does not terminate at step 2 and every pair \( (u, v) \in S \) appears in at least one \( \Lambda_k(U_i, U_j) \). The details for this result can be found in [60, Lemma 2].

Step 1 requires \( 2n^{1/4}[\log W / \log n] \) rounds and can be implemented fully classically without any qubit communication. Step 2 requires at most \( 200 \log n[\log W / \log n] \) rounds and can also be implemented classically. Step 3 can be implemented in \( \tilde{O}(n^{1/4}) \) rounds quantumly taking advantage of distributed Grover search but would take \( O(\sqrt{n}) \) steps to implement classically. The remainder of this section is devoted to illustrating how this step can be done in \( \tilde{O}(n^{1/4}) \) rounds.

Define the following quantity:

**Definition 103.** For node \( v(i,j,k) \), let

\[
\Delta(i, j, k) := \left\{ (u, v) \in \mathcal{P}(U_i, U_j) \cap S : \exists w \in U'_k \text{ with } (u, v, w) \text{ forming a negative triangle in } G \right\}
\]

For simultaneous quantum searches, we divide the nodes into different classes based on the number of negative triangles they are a part of with the following routine:

**IdentifyClass**

Input: An integer-weighted graph \( G = (V, E, W) \) distributed among the nodes, and
a set $S \subset E$ as in FWP.

Output: For each node $v$, a class $\alpha$ the node belongs to.

1: Every node $u_{(i,j,k)} \in V$ samples each node in $\{v \in V : (u_{(i,j,k)}, v) \in S\}$ with probability $\frac{10 \log n}{n}$, creating a set $\Lambda(u)$ of sampled vertices. If $\max_u |\Lambda(u)| > 20 \log n$, abort the algorithm and report a failure. Otherwise, have each node broadcast $\Lambda(u)$ to all other nodes, and take $R := \cup_{u \in V} \{uv|v \in \Lambda(u)\}$.

2: Each $v_{(i,j,k)} \in V$ computes $d_{i,j,k} := |\{uv \in \mathcal{P}(i,j) \cap R : \exists w \in U'_{k} \text{ such that } \{u,v,w\} \text{ forms a negative triangle in } G\}|$, then determines its class $\alpha$ to be $\min\{c \in \mathbb{N} : d_{i,j,k} < 10 \cdot 2^c \log n\}$.

This uses at most $20 \log n$ rounds (each node sends at most that many IDs to every other node) and can be implemented by having all exchanged messages consist only of classical bits. Using Chernoff’s bound, one can show that the procedure succeeds with probability of at least $1 - 1/n$ as seen in [61, Proposition 5].

Let us make the convenient assumption that $\alpha = 0$ for all $v_{i,j,k}$, which avoids some technicalities around congestion in the forthcoming triangle search. Note that $\alpha \leq \frac{1}{2} \log n$, so we can run successive searches for each $\alpha$ for nodes in with class $\alpha$ in the general case. The general case is discussed in §6.8.2 of the appendix and can also be found in [60], but this case is sufficient to convey the central ideas.

We have all the necessary ingredients to describe the implementation of step 3 of the ComputePairs procedure.

3.1: Each node executes the IdentifyClass procedure.

3.2: For each $\alpha$, for every $l \in [m]$, every node $v_{(i,j,k)}$ in class $\alpha$ executes a quantum search to find whether there is a $U'_{k} \in \mathcal{U}$ with some $w \in U'_{k}$ forming a negative triangle $(u_{l}^{k}, v_{l}^{k}, w)$ in $G$, and then reports all the pairs $u_{l}^{k}v_{l}^{k}$ for which such a $U'_{k}$ was found.

This provides the basis of the triangle-searching strategy. To summarize the
intuition of the asymptotic speedup in this paper: Since the $U'_k$ have size $\sqrt{n}$ (recall that $|U'| = \sqrt{n}$), if each node using a quantum search can search through its assigned $U'_k$ in $\tilde{O}(n^{1/4})$ rounds, simultaneously, we will obtain our desired complexity. We will complete this argument in §6.4.6 and first describe the quantum searches used therein in the following subsection.

6.4.5 Distributed Quantum Searches

With this intuition in mind, we now state two useful theorems of [60] for the distributed quantum searches. Let $X$ denote a finite set throughout this subsection.

**Theorem 104.** Let $g : X \to \{0, 1\}$, if a node $u$ can compute $g(x)$ in $r$ rounds in the CONGEST-CLIQUE model for any $x \in X$, then there exists an algorithm in the Quantum CONGEST-CLIQUE that has $u$ output some $x \in X$ with $g(x) = 1$ with high probability using $\tilde{O}(r\sqrt{|X|})$ rounds.

This basic theorem concerns only single searches, but we need a framework that can perform multiple simultaneous searches. Let $g_1, \ldots, g_m : X \to \{0, 1\}$ and

$$A^0_i := \{x \in X : g_i(x) = 0\}, A^1_i := \{x \in X : g_i(x) = 1\}, \forall i \in [m].$$

Assume there exists an $r$-round classical distributed algorithm $C_m$ that allows a node $u$ upon an input $\chi = (x_1, \ldots, x_m) \in X^m$ to determine and output $(g_1(x_1), \ldots, g_m(x_m))$.

In our use of distributed searches, $X$ will consist of nodes in the network, and searches will need to communicate with those nodes for which the functions $g_i$ are evaluated. To avoid congestion, we will have to consider those $\chi \in X^m$ that have many repeated entries carefully. We introduce some notation for this first. Define the quantity

$$\alpha(\chi) := \max_{I \subseteq [m]} |\{\chi_i = \chi_j : \forall i, j \in I\}|,$$

the maximum number of entries in $\chi$ that are all identical.

Next, given some $\beta \in \mathbb{N}$, assume that in place of $C_m$ we now have a classical algorithm $\tilde{C}_{m, \beta}$ such that upon input $\chi = (x_1, \ldots, x_m) \in X^m$, a node $u$ outputs
$g_1(x_1), \ldots, g_m(x_m)$ if $\alpha(\chi) \leq \beta$ and an arbitrary output otherwise. The following theorem summarizes that such a $\tilde{C}_{m,\beta}$ with sufficiently large $\beta$ is enough to maintain a quantum speedup as seen in the previous theorem:

**Theorem 105.** For a set $X$ with $|X| < m/(36 \log m)$, suppose there exists such an evaluation algorithm $C_{m,\beta}$ for some $\beta > 8m/|X|$ and that $\alpha(\chi) \leq \beta$ for all $\chi \in A_1^1 \times \cdots \times A_m^1$. Then there is a $\tilde{O}(r \sqrt{|X|})$-round quantum algorithm that outputs an element of $A_1^1 \times \cdots \times A_m^1$ with probability at least $1 - 2/m^2$.

The proof can be found in [60, Theorem 3].

### 6.4.6 Final Steps of the Triangle Finding

We continue here to complete the step 3.2 of the **ComputePairs** procedure, armed with Theorem 105. We need simultaneous searches to be executed by each node $v_{(i,j,k)}$ to determine the triangles in $U_i \times U_j \times U_k'$. We provide a short lemma first that ensures the conditions for the quantum searches:

**Lemma 106.** The following statements hold with probability at least $1 - 2/n^2$:

(i): $|\Delta(i, j, k)| \leq 2n$

(ii): $|\Lambda_k(U_i, U_j) \cap \Delta(i, j, k)| \leq 100 \cdot \sqrt{n \log n}$ for $i, j \in [n^{1/2}]$.

The proofs of these statements are technical but straightforward, making use of Chernoff’s bound and union bounds; hence we skip them here. To invoke Theorem 105, we describe a classical procedure first, beginning with an evaluation step, **EvaluationA** implementable in $\tilde{O}(1)$ rounds.

**EvaluationA**

Input: Every node $v_{(i,j,k)}$ receives $m$ elements $(u_{i,j,k}^1, \ldots, u_{i,j,k}^m)$ of $U'$

Promise: For every node $v_{i,j,k}$ and every $w \in U'$, $|L_{i,j,k}^w| \leq 800\sqrt{n \log n}$.

Output: Each node outputs a list of exactly those $u_{i,j,k}^l$ such that there is a negative triangle in $U_i \times U_j \times u_{i,j,k}^l$. 

180
1. Every node \( v(i,j,k) \), for each \( r \in \sqrt{n} \) routes the list \( L^{i,j,t} \) to node \( v(i,j,t) \).

2. Every node \( v(i,j,k) \), for each \( vu \) it received in step 1, sends the truth value of the inequality

\[
\min_{w \in U'_{k}} \{W_{uw} + W_{wv}\} \leq W_{vu} \tag{6.3}
\]

to the node that sent \( vu \).

Each node is the source and destination of up to \( 800n \log n \) messages in step 1, meaning that this step can be implemented in \( 1600 \log n \) rounds. The same goes for step 2, noting that the number of messages is the same, but they need only be single-bit messages (the truth values of the inequalities). Hence, the evaluations for Theorem 105 can be implemented in \( 3200 \log n \) rounds. Now, applying the theorem with \( X = U', \beta = 800\sqrt{n} \log n \), noting that then the assumptions of the theorem hold with probability at least \( 1 - 2/n^2 \) due to Lemma 106, implies that step 3.2 is implementable in \( \tilde{O}(n^{1/4}) \) rounds, with a success probability of at least \( 1 - 2/m^2 \).

For the general case in which we do not assume \( \alpha = 0 \) for all \( i, j, k \) in \text{IdentifyClass}, covered in the appendix, one needs to modify the \text{EvaluationA} procedure in order to implement load balancing and information duplication to avoid congestion in the simultaneous searches. These details can be found in the appendix, where a new labeling scheme and different evaluation procedure \text{EvaluationB}, are described for this, or in [60].

### 6.4.7 Complexity

As noted previously and in [60], this APSP scheme uses \( \tilde{O}(n^{1/4}) \) rounds. Let us characterize the constants and logarithmic factors involved to assess this algorithm’s practical utility. Suppose that in each round, \( 2 \cdot \log n \) qubits can be sent in each message (so that we can send two IDs or one edge with each message), where \( n \) is the number of nodes. For simplicity, let’s assume that for the edge weights \( W \ll n \) and drop \( W \).

1. APSP with routing tables needs \( \log(n) \) distance products with witness matrices.
2. Computing the \(i^{th}\) distance product square for Eq. (6.2) with a witness matrix needs up to \(\log(2^i) = i\) calls to \texttt{FindEdges}, since the entries of the matrix being squared may double each iteration. Then APSP and distance products together make \(\sum_{i=1}^{\lceil \log n \rceil} \frac{[\log(n)](([\log(n)] + 1)}{2} \) calls to \texttt{FindEdges}.

3. Solving \texttt{FindEdges} needs \(\log\left(\frac{n}{60\log n}\right)\) calls to \texttt{FEWP}, using \texttt{FindEdgesViaFEWP}.

4. Step 1 of \texttt{ComputePairs} needs up to \(2 \cdot n^{1/4}\) rounds and step 2 takes up to \(200 \log n\) rounds.

5. Step 1 of \texttt{IdentifyClass} needs up to \(20 \log n\) rounds.

6. In step 2 of \texttt{IdentifyClass}, the \(c_{uw}\) are up to \(\frac{1}{2} \log n\) large, and hence \(\alpha\) may range up to \(\frac{1}{2} \log n\).

7. Step 0 of the \texttt{EvaluationB} procedure needs \(n^{1/4}\) rounds. Steps 1 and 2 of the \texttt{EvaluationB} (or \texttt{EvaluationA}, in the \(\alpha = 0\) case) procedure use a total of \(3200 \log n\) rounds.

8. \texttt{EvaluationB} (or \texttt{EvaluationA}) procedure is called up to \(\log(n)n^{1/4}\) times for each value of \(\alpha\) in step 3.2 of \texttt{ComputePairs}.

Without any improvements, we get the following complexity, using \(3n\) in place of \(n\) for the terms of steps 3-8 due to corollary 100:

\[
\frac{[\log(n)]([\log(n)] + 1)}{2} \log\left(\frac{3n}{60\log 3n}\right) \left(2(3n)^{1/4} + 220 \log 3n + 2(3n)^{1/4} + \frac{1}{2} \log 3n \cdot \log 3n \cdot (3n)^{1/4}3200(\log 3n)\right),
\]

which we will call \(f(n)\), so that \(f(n) = O(n^{1/4} \log^6(n))\), with the largest term being about \(800 \log^6(n)n^{1/4}\), and we have dropped \(W\) to just consider the case \(W \ll n\). We can solve the problem trivially in the (quantum or classical) CONGEST-CLIQUE within \(n \log(W)\) rounds by having each node broadcast its neighbors and the weight on the edge. Let us again drop \(W\) for the case \(W \ll n\) so that in order for the quantum
algorithm to give a real speedup, we will need

\[ f(n) < n, \]

which requires \( n > 10^{18} \) (even with the simpler under-approximation \( 800 \log^8(n)n^{1/4} \) in place of \( f \)). Hence, even with some potential improvements, the algorithm is impractical for a large regime of values of \( n \) even when compared to the trivial CONGEST-CLIQUE \( n \)-round strategy.

For the algorithm of [60] computing only APSP distances, the first term in 6.4 becomes simply \( \lceil \log n \rceil \), so that when computing only APSP distances the advantage over the trivial strategy begins at roughly \( n \approx 10^{16} \).

**Remark 107.** In light of logarithmic factors commonly being obscured by \( \tilde{O} \) notation, we point out that even an improved algorithm needing only \( \log^4(n)n^{1/4} \) would not be practical unless \( n > 10^7 \), for the same reasons. Recall that \( n \) is the number of processors in the distributed network – tens of millions would be needed to make this algorithm worth implementing instead of the trivial strategy. Practitioners should mind the \( \tilde{O} \) if applications are of interest, since even relatively few logarithmic factors can severely limit practicality of algorithms, and researchers should be encouraged to fully write out the exact complexities of their algorithms for the same reason.

**Memory Requirements**

Although in definition 85 we make no assumption on the memory capacities of each node, the trivial \( n \)-round strategy uses at least \( 2 \log(n) |E|^2 \cdot \log(W) \) memory at the leader node that solves the problem. For the APSP problem in question, using the Floyd-Warshall algorithm results in memory requirements of \( 2n^2 \log(n) \cdot \log(nW) \) at the leader node. Hence, we may ask whether the quantum APSP algorithm leads to lower memory requirements. The memory requirement is largely characterized by up to \( 720n^{7/4} \log(n) \log(nW) \) needed in step 0 of the \textbf{EvaluationB} procedure, which can be found in the appendix. This results in a memory advantage for quantum APSP over the trivial strategy beginning in the regime of \( n > 1.6 \cdot 10^{10} \).
Complexity of the Classical Analogue

For completeness, we provide here a characterization of the complexity of a closely related classical algorithm for APSP with routing tables in the CONGEST-CLIQUE as proposed in [22] that has complexity $\tilde{O}(n^{1/3})$. In their framework, the approach to finding witness matrices requires $O(\log^3(n))$ calls to the distance product [22, §3.4], and similarly to our approach $\log(n)$ distance products are required. Their classical algorithm computes distance products in $O(n^{1/3})$ rounds, or under $2\log n$ message bandwidth in up to

$$20n^{1/3}\log(n)^4 =: g(n)$$

rounds, the details of which can be found in the appendix, §6.8.2. Then $g(n) > n$ up until about $n \approx 2.6 \cdot 10^{11}$. As with the quantum APSP, though this algorithm gives the best known asymptotic complexity of $\tilde{O}(n^{1/3})$ in the classical CONGEST-CLIQUE, it also fails to give any real improvement over the trivial strategy across a very large regime of values of $n$. Consequently, algorithms making use of this APSP algorithm, such as [95] or [43], suffer from the same problem of impracticality. However, the algorithm only requires within $4n^{1/3}\log(n)\log(nW) + n\log(n)\log(nW)$ memory per node, which is less than required for the trivial strategy even for $n \geq 4$.

6.5 Algorithm for Approximate Steiner Tree

6.5.1 Algorithm Overview

We present a high-level overview of the proposed algorithm to produce approximately optimal Steiner Trees, divided into four steps.

**Step 1 - APSP and Routing Tables:** Solve the APSP problem as in [60] and add an efficient routing table scheme via triangle finding in $\tilde{O}(n^{1/4})$ rounds, with success probability $(1 - 1/poly(n))$ (this step determines the algorithm’s overall success probability).

**Step 2 - Shortest-path Forest:** Construct a shortest-path forest (SPF), where each
tree consists of exactly one source terminal and the shortest paths to the vertices whose closest terminal is that source terminal. This step can be completed in one round and \( n \) messages, per [95, §3.1]. The messages can be in classical bits.

**Step 3 - Weight Modifications:** Modify the edge weights depending on whether they belong to a tree (set to 0), connect nodes in the same tree (set to \( \infty \)), or connect nodes from different trees (set to the shortest path distance between root terminals of the trees that use the edge). This uses one round and \( n \) messages.

**Step 4 - Minimum Spanning Tree:** Construct a minimum spanning tree (MST) on the modified graph in \( O(1) \) rounds as in [89], and prune leaves of the MST that do not connect terminal nodes since these are not needed for the Steiner Tree.

The correctness of the algorithm follows from the correctness of each step together with the analysis of the classical results of [71], which uses the same algorithmic steps of constructing a shortest path forest and building it into an approximately optimal Steiner Tree.

### 6.5.2 Shortest Path Forest

After the APSP distances and routing tables have been found, we construct a **Shortest Path Forest** (SPF) based on the terminals of the Steiner Tree.

**Definition 108.** *(Shortest Path Forest):* For a weighted, undirected graph \( G = (V,E,W) \) together with a given set of terminal nodes \( Z = \{z_1, \ldots, z_k\} \), a subgraph \( F = (V,E_F,W) \) of \( G \) is called a shortest path forest if it consists of \( |Z| \) disjoint trees \( T_z = (V_z, E_z, W) \) satisfying

i) \( z_i \in T_{z_j} \text{ if and only if } i = j, \text{ for } i, j \in [k] \).

ii) For each \( v \in Z_i, d_G(v, z_i) = \min_{z \in Z} d_G(v, z) \), and a shortest path connecting \( v \) to \( z_i \) in \( G \) is contained in \( T_{z_i} \).

iii) The \( V_{z_i} \) form a partition of \( V \), and \( E_{z_1} \cup E_{z_2} \cdots \cup E_{z_k} = E_F \subset E \)

In other words, an SPF is a forest obtained by gathering, for each node, a shortest
path in $G$ connecting it to the closest Steiner terminal node.

For a node $v$ in a tree, we will let $\text{par}(v)$ denote the parent node of $v$ in that tree, $s(v)$ the Steiner Terminal in the tree that $v$ will be in, and $ID(v) \in [n]$ the ID of node $v \in V$. Let $Q(v) := \{z : d_G(v, z) = \min_{z \in Z} d_G(v, z)\}$ be the set of Steiner Terminals closest to node $v$. We make use of the following procedure for the SPF:

\begin{itemize}
  \item[] \textbf{DistributedSPF}
  \item Input: For each node $v \in G$, APSP distances and the corresponding routing table $R_v$.
  \item Output: An SPF distributed among the nodes.
  \item 1: Each node $v$ sets $s(v) := \arg\min_{z \in Q(v)} ID(z)$ using the APSP information.
  \item 2: Each node $v$ sets $\text{par}(v) := R_v(s(v))$, $R_v$ being the routing table of $v$, and sends a message to $\text{par}(v)$ to indicate this choice. If $v$ receives such a message from another node $u$, it registers $u$ as its child in the SPF.
\end{itemize}

Step 1 in \textbf{DistributedSPF} requires no communication since each node already knows the shortest path distances to all other nodes, including the Steiner Terminals, meaning it can be executed locally. Each node $v$ choosing $\text{par}(v)$ in step 2 can also be done locally using routing table information, and thus step 2 requires 1 round of communication of $n - |Z|$ classical messages, since all non-Steiner nodes send one message.

\textbf{Claim 109.} After executing the \textbf{DistributedSPF} procedure, the trees $T_{z_k} = (V_{z_k}, E_{z_k}, W)$ with $V_{z_k} := \{v \in V : s(v) = z_k\}$ and $E_{z_k} := \{v, \text{par}(v)\} : v \in V_{z_k}$ form an SPF.

\textbf{Proof.} i) holds since each Steiner Terminal is closest to itself. iii) is immediate. To see that ii) holds, note that for $v \in V_{z_k}$, $\text{par}(v) \in V_{z_k}$ and $\{v, \text{par}(v)\} \in E_{z_k}$ as well. Then $\text{par}(\text{par}(\ldots \text{par}(v) \ldots)) = z_k$ and the entire path to $z_k$ lies in $T_{z_k}$. \hfill $\Box$

Hence, after this procedure, we have a distributed SPF across our graph, where
each node knows its label, parent, and children of the tree it is in.

6.5.3 Weight Modified MST and Pruning

Finally, we introduce a modification of the edge weights before constructing an MST on that new graph that will be pruned into an approximate Steiner Tree. These remaining steps stem from a centralized algorithm first proposed by [71] whose steps can be implemented efficiently in the distributed setting, as in [95]. We first modify the edge weights as follows:

Partition the edges $E$ into three sets – tree edges $E_F$ as in 108 that are part of the edge set of the SPF, intra-tree edges $E_{IT}$ that are incident on two nodes in the same tree $T_i$ of the SPF, and inter-tree edges $E_{XT}$ that are incident on two nodes in different trees of the SPF. Having each node know which of these its edges belong to can be done in one round by having each node send its neighbors the ID of the terminal it chose as the root of the tree in the SPF that is a part of. Then the edge weights are modified as follows, denoting the modified weights as $W'$:

(i): For $e = (u, v) \in E_T$, $W'(u, v) := 0$

(ii): For $e = (u, v) \in E_{IT}$, $W'(u, v) := \infty$

(iii): For $e = (u, v) \in E_{XT}$, $W'(u, v) := d_G(u, Z_u) + W(u, v) + d_G(v, Z_v)$,

noting that $d_G(u, s(u))$ is the shortest-path distance in $G$ from $u$ to its closest Steiner Terminal.

Next, we find a minimum spanning tree on the graph $G' = (V, E, W')$, for which we may implement the classical $O(1)$ round algorithm proposed by [89]. On a high level, this constant-round complexity is achieved by sparsification techniques, reducing MST instances to sparse ones, and then solving those efficiently. We skip the details here and refer the interested reader to [89]. After this step, each node knows which of its edges are part of this weight-modified MST, as well as the parent-child relationships in the tree for those edges.
Finally, we prune this MST by removing non-terminal leaf nodes and the corresponding edges. This is done by each node $v$ sending the ID of its parent in the MST to every other node in the graph. As a result, each node can locally compute the entire MST and then decide whether or not it connects two Steiner Terminals. If it does, it decides it is part of the Steiner Tree; otherwise, it broadcasts that it is to be pruned. Each node that has not been pruned then registers the edges connecting it to non-pruned neighbors as part of the Steiner Tree. This pruning step takes 2 rounds and up to $n^2 + n$ classical messages.

### 6.5.4 Overall Complexity and Correctness

In algorithm 6.5.1, after step 1, steps 2, and 3 can each be done within 2 rounds. Walking through [89] reveals that the MST for step 4 can be found in 54 rounds, with an additional 2 rounds sufficing for the pruning. Hence, the overall complexity remains dominated by Eq. (6.4). Hence, the round complexity is $\tilde{O}(n^{1/4})$, which is faster than any known classical CONGEST-CLIQUE algorithm to produce an approximate Steiner tree of the same approximation ratio. However, as a consequence of the full complexity obtained in §6.4.7, the regime of $n$ in which this algorithm beats the trivial strategy of sending all information to a single node is also $n > 10^{18}$. For the same reason, the classical algorithm provided in [95] making use of the APSP subroutine from [22] discussed in §6.4.7 has its complexity mostly characterized by Eq. (6.5), so that the regime in which it provides an advantage over the trivial strategy lies in $n > 10^{11}$. Our algorithm’s correctness follows from the correctness of each step together with the correctness of the algorithm by [71] that implements these steps in a classical, centralized manner.

### 6.6 Directed Minimum Spanning Tree Algorithm

This section will be concerned with establishing Theorem 93 for the Directed Minimum Spanning Tree (DMST) problem, in definition 91. Like [43], we follow the algorithmic ideas first proposed by [77], implementing them in the quantum CONGEST-CLIQUE. Specifically, we will use $\log n$ calls to the APSP and routing tables scheme described
in §6.4, so that in our case, we retrieve complexity $\tilde{O}(n^{1/4})$ and success probability
\[
\left(1 - \frac{1}{\text{poly}(n)}\right)^{\log n} = 1 - \frac{1}{\text{poly}(n)}.
\]

Before describing the algorithm, we need to establish some preliminaries and terminology for the procedures executed during the algorithm, especially the ideas of shrinking vertices into super-vertices and tracking a set $H$ of specific edges as first described in [40]. We use the following language to discuss super-vertices and related objects.

**Definition 110.** A super-vertex set $V^* := \{V^*_1, \ldots, V^*_t\}$ for a graph $G = (V, E, W)$ is a partition of $V$, and each $V^*_i$ is called a super-vertex. We will call a super-vertex simple if $V^*_i$ is a singleton. The corresponding minor $G^* := (V^*, E^*, W^*)$ is the graph obtained by creating edges $(V^*_i, V^*_j)$ with weight $W^*(V^*_i, V^*_j) := \min\{W(v_i, v_j) : v_i \in V^*_i, v_j \in V^*_j\}$.

Notably, we continue to follow the convention of an edge of weight $\infty$ being equivalent to not having an edge. We will refer to creating a super-vertex $V^*$ as contracting the vertices in $V^*$ into a super-vertex.

### 6.6.1 Edmonds’ Centralized DMST Algorithm

We provide a brief overview of the algorithm proposed in [40], which presents the core ideas of the super-vertex-based approach. The following algorithm produces a DMST for $G$:

---

**Edmonds DMST Algorithm**

**Input:** An integer-weighted digraph and a root node $r$.

**Output:** A DMST for $G$ rooted at $r$.

1. Initialize a subgraph $H$ with the same vertex set as $G$ by subtracting for each node the minimum incoming edge weight from all its incoming edges, and selecting exactly one incoming zero-weight edge for each non-root node of $G$. Set $G_0 = G$, $H_0 = H$, $t = 0$.  

---

189
2. WHILE $H_t$ is not a tree:

(a) For each cycle of $H$, contract the nodes on that cycle into a super-vertex. Consider all non-contracted nodes as simple super-vertices, and obtain a new graph $G_{t+1}$ as the resulting minor.

(b) If there is a non-root node of $G_{t+1}$ with no incoming edges, report a failure. Otherwise, obtain a subgraph $H_{t+1}$ by, for each non-root node of $G_{t+1}$, subtracting the minimum incoming edge weight from all its incoming edges, and selecting exactly one incoming zero-weight edge for each non-root, updating $t \leftarrow t + 1$.

3. Let $B_t = H_t$. FOR $k \in (t, t - 1, \ldots, 1)$:

(a) Obtain $B'_{k-1}$ by expanding the non-simple super-vertices of $B_k$ and selecting all but one of the edges for each of the previously contracted cycles of $H_k$ to add to $B_{k-1}$.

4. Return $B_0$.

Note that the edge weight modifications modify the weight of all directed spanning trees equally, so optimality is unaffected. In step 2., if $H_t$ is a tree, it is an optimal DMST for the current graph $G_t$. Otherwise, it contains at least one directed cycle, so that indeed step 2. is valid. Hence, at the beginning of step 3., $B_t$ is a DMST for $G_t$. Then the first iteration produces $B_{t-1}$ a DMST for $G_{t-1}$ since only edges of zero weight were added, and $B_{t-1}$ will have no cycles. The same holds for $B_{t-2}, B_{t-3}, \ldots, B_0$, for which $B_0$ corresponds to the DMST for the original graph $G$. If the algorithm reports a failure at some point, no spanning tree rooted at $r$ exists for the graph, since a failure is reported only when there is an isolated non-root connected component in $G_{t+1}$.

Note that in iteration $t$ of step 2., $H$ has one cycle for each of its connected components that does not contain the root node. Hence, the drawback of this algorithm is that we may apply up to $O(n)$ steps of shrinking cycles. This shortcoming is remedied by a more efficient method of selecting how to shrink nodes into super-vertices in [77], such that only $\log n$ shrinking cycle steps take place.
6.6.2 Lovasz’ Shrinking Iterations

We devote this subsection to discuss the shrinking step of [77] that will be repeated \( \log n \) times in place of step 2. of Edmonds’ algorithm to obtain Lovasz’ DMST algorithm.

Lovasz’ Shrinking Iteration LSI

Input: A directed, weighted graph \( G = (V, E, W) \) and a root node \( r \in V \).

Output: Either a new graph \( G^* \), or a success flag and a DMST \( H \) of \( G \).

1. If there is a non-root node of \( G \) with no incoming edges, report a failure. Otherwise, for each non-root node of \( G \), subtract the minimum incoming edge weight from all its incoming edges. Select exactly one incoming zero-weight edge for each non-root node to create a subgraph \( H \) of \( G \) with those edges.

2. Find all cycles of \( H \), and denote them \( H_1, \ldots, H_C \). If \( H \) has no cycles, abort the iteration and return (SUCCESS, \( H \)). For \( j = 1, \ldots, C \), find the set \( V_j \) of nodes that dipaths in \( H \) from \( H_j \) can reach.

3. Compute the All-Pairs-Shortest-Path distances in \( G \).

4. For each node \( v \in V \), denote \( d_j(v) := \min\{d(v, u) : u \in H_j\} \). For each \( j = 1, \ldots, C \), set \( \beta_j := \min\{d_j(v) : v \in V(G) \setminus V_j\} \) and \( U_j := \{u \in V_j : d_j(u) \leq \beta_j\} \).

5. Create a minor \( G^* \) by contracting each \( U_j \) into a super-vertex \( U_j^* \), considering all other vertices of \( G \) as simple super-vertices \( V_1^*, \ldots, V_k^* \). For each vertex \( N^* \) of \( G^* \), let the edge weights in \( G^* \) be:

\[
W_{N^*U_j^*} = \min\{W_{vu} : v \in N^*, u \in U_j^*\} - \beta_j + \min\{d_j(u) : u \in U_j^*\}
\]

for all \( j = 1, \ldots, C \), and

\[
W_{N^*V^*} = \min\{W_{vV^*} : v \in N^*\}
\]

for all the simple super-vertices \( V^* \) of \( G^* \).
To summarize these iterations: The minimum-weight incoming edge of each node is selected. That weight is subtracted from the weights of every incoming edge to that node, and one of those edges with new weight 0 is selected for each node to create a subgraph $H$. If $H$ is a tree, we are done. Otherwise, we find all cycles of the resulting directed subgraph, then compute APSP and determine the $V_j, U_j,$ and $\beta_j$, which we use to define a new graph with some nodes of the original $G$ contracted into super-vertices.

The main result for the DMST problem in [77] is that replacing (a) and (b) of step 2. in the Edmonds DMST Algorithm, taking the new $H$ obtained at each iteration to be $H_{t+1}$ and the $G^*$ to be $G_{t+1}$, leads to no more than $\lceil \log n \rceil$ such shrinking iterations needed before a success is reported.

**Quantum Distributed Implementation**

Our goal is to implement the Lovasz iterations in the quantum distributed setting in $\tilde{O}(n^{1/4})$ rounds by making use of quantum APSP of §6.4. In the distributed setting, processor nodes cannot directly be shrunk into super-vertices. As in [43], we reconcile this issue by representing the super-vertex contractions within the nodes through soft contractions.

First, note that a convenient way to track what nodes we want to consider merging into a super-vertex is to keep a mapping $sID: V \rightarrow S$, where $S$ is a set of super-vertex IDs, which we can just take to be the IDs of the original nodes. We will refer to a pair of $(G, sID)$ as an annotated graph. An annotated graph naturally corresponds to some minor of $G$, namely, the minor obtained by contracting all vertices sharing a super-vertex ID into a super-vertex.

**Definition 111** (Soft Contractions). For an annotated graph $(G, sID)$, a set of active edges $H$, and active component $H_i$ with corresponding weight modifiers $\beta_i$, and a subset $A \subseteq S$ of super-vertices, the soft contraction of $H_i$ in $G$ is the annotated graph $(G^{H_i}, sID')$ obtained by taking $G^{H_i} = (V, E, W')$ with

- $W'_{uv} = 0$ if $sID(u) = sID(v)$

192
\[ W'_{uv} = W_{uv} + \text{dist}_{G(A)}(v, C(H_i)) - \beta_i \text{ if } u \in V \setminus A \text{ and } v \in A \]
\[ W'_{uv} = W_{uv} \text{ otherwise} \]

and updating the mapping \( sID \) to \( sID' \) defined by \( sID'(v) = sID(v), \forall v \notin A \), \( sID'(v) = \min\{sID(u) : u \in A\} \).

**Quantum Distributed Lovasz’ Iteration**

We provide here a quantum distributed implementation of Lovasz’ iteration that we will form the core of our DMST algorithm.

---

Quantum Distributed Lovasz’ Iteration QDLSI

**Input:** A directed, weighted, graph \( G = (V, E, W) \) with annotations \( sID \) and a subgraph \( H \).

**Output:** A new graph \( G^* \) with annotations \( sID' \), or a success flag and a DMST \( H \) of \( G \).

1: Have all nodes learn all edges of \( H \), as well as the current super-vertices.

2: For each connected component \( H_i \subset H \), denote by \( C(H_i) \) the cycle of \( H_i \). Let \( c(H_i) \) be the node with maximal ID in \( C(H_i) \), which each node can locally compute.

3: Run the quantum algorithm for APSP and routing tables described in §6.4 on this graph, or report a failure if it fails.

4: For each \( i \), determine an edge \( v_i u_i, v_i \notin H_i, u_i \in H_i \) minimizing \( \beta_i := W_{v_iu_i} + d_{G}(u_i, c(H_i)) \), and broadcast both to all nodes in \( H_i \).

5: Each node \( v_i \) in each \( H_i \) applies the following updates **locally**:

   - Soft-contract \( H_i \) at level \( \beta_i \) to soft-contract all super-vertices with distance \( \beta_i \) to \( C(H_i) \) into one super-vertex, with each contracted node updating its super-vertex ID to \( c(H_i) \)
add edge \( v_iu_i \) to \( H \), effectively merging \( H_i \) with another active component of \( H \)

We can follow exactly the steps of Lovasz’s DMST algorithm, distributedly by replacing steps 2-5 of the LSI with this quantum-distributed version. The following ensues:

**Lemma 112.** If none of the APSP and routing table subroutines fail, within \( \lceil \log n \rceil \) iterations of the QDLSI, \( H \) is a single connected component.

**Lemma 113.** With probability \( (1 - \frac{1}{\text{poly}(n)})^{\log n} \), all the APSP and routing table subroutines in step 3 succeed.

Lemmas 112 and 113 then together imply Theorem 93. Within \( \lceil \log n \rceil \) iterations, only one active component remains: the root component. This active component can then be expanded to a full DMST on \( G \) within \( \lceil \log n \rceil \) rounds, as detailed in [43, §7] or the Unpacking procedure in §6.8.3 of the appendix. All messages in the algorithm other than those for computing the APSP in QDLSI may be classical. We provide here the full algorithm for completeness:

**Quantum DMST Algorithm**

Input: An integer-weighted digraph and a root node \( r \).

Output: A DMST for \( G \) rooted at \( r \).

1. Initialize a subgraph \( H \) with the same vertex set as \( G \) by subtracting for each node the minimum incoming edge weight from all its incoming edges, and selecting exactly one incoming zero-weight edge for each non-root node of \( G \). Set \( t = 0, H_0 = H, \) and \( G_0 = G \) with annotations \( sID_0 \) to be the identity mapping.

2. \textbf{WHILE:} \( H_t \) is not a single component

   (a) Run QDLSI with inputs \( H_t, (G_t, sID_t) \) to obtain \( H_{t+1}, (G_{t+1}, sIDt + 1) \)
as outputs. Increment $t \leftarrow t + 1$.

3. Let $T_1 := H_t$. For $k = t, \ldots, 1$: For each super-vertex of the $k^{th}$ iteration of \texttt{QDLSI} applied, simultaneously run the \texttt{Unpacking} procedure with input tree $T_k$ to obtain $T_{k-1}$.

4. Return $T_0$ as the distributed minimum spanning tree.

### 6.6.3 Complexity

In the \texttt{QDLSI}, all steps other than the APSP step 3 of the quantum Lovasz iteration can be implemented within 2 rounds. In particular, to have all nodes know some tree on $G$ for which each node knows its parent, every node can simply broadcast its parent edge and weight. Since this iteration is used up to $\lceil \log(n) \rceil$ times and expanding the DMST at the end of the algorithm also takes logarithmically many rounds, we obtain a complexity dominated by the APSP computation of $\tilde{O}(n^{1/4})$, a better asymptotic rate than any known classical CONGEST-CLIQUE algorithm. However, beyond the $\tilde{O}$, the complexity is largely characterized by $\log(n) \cdot f(n)$, with $f(n)$ as in Eq. (6.4). In order to have $\log(n) f(n) < n$ to improve upon the trivial strategy of having a single node solve the problem, we then need $n > 10^{21}$. Using the classical APSP from [22] in place of the quantum APSP of §6.4 as done in [43] to attain the $\tilde{O}(n^{1/3})$ complexity in the cCCM, one would need $\log(n) \cdot g(n) < n$ to beat the trivial strategy, with $g$ as in Eq. (6.5), or more than $n > 10^{14}$.

### 6.7 Discussion and Future Work

We have provided algorithms in the Quantum CONGEST-CLIQUE model for computing approximately optimal Steiner Trees and exact Directed Minimum Spanning trees that use asymptotically fewer rounds than their classical known counterparts. As Steiner Tree and Minimum Spanning Trees cannot benefit from quantum communication in the CONGEST (non-clique) model, the algorithms reveal how quantum communication can be exploited thanks to the CONGEST-CLIQUE setting. A few open questions remain as well. In particular, there exist many generalizations of the Steiner Tree problem,
so these may be a natural starting point to attempt to generalize the results. A help-
ful overview of Steiner-type problems can be found in [54]. Regarding the DMST, it
may be difficult to generalize a similar approach to closely related problems. Since
the standard MST can be solved in a (relatively small) constant number of rounds in
the classical CONGEST-CLIQUE, no significant quantum speedup is possible. Other
interesting MST-type problems are the bounded-degree and minimum-degree spanning
tree problems. However, even the bounded-degree decision problem on an unweighted
tree, “does $G$ have a spanning tree of degree at most $k$?” is NP-complete, unlike the
DMST, so we suspect that other techniques would need to be employed. [37] provides
a classical distributed approximation algorithm for the problem. Additionally, we have
traced many constants and log factors throughout our description of the above algo-
rithms, which, as shown, would need to be significantly improved for these and related
algorithms to be practical. Hence, a natural avenue for future work is to work towards
such practical improvements. Beyond the scope of the particular algorithms involved,
we hope to help the community recognize the severity with which the practicality of
algorithms is affected by logarithmic factors that may be obscured by $\tilde{O}$ notation, and
thus encourage fellow researchers to present the full complexity of their algorithms be-
yond asymptotics. Particularly in a model like CONGEST-CLIQUE, where problems
can always be solved trivially in $n$ rounds, these logarithmic factors should clearly not
be taken lightly. Further, a question of potential practical interest would be to ask the
following: What algorithms solving the discussed problems are the most efficient with
respect to rounds needed in the CONGEST-CLIQUE in the regimes of $n$ in which the
discussed algorithms are impractical?

**Chapter Acknowledgements**

We are grateful for support from the NASA Ames Research Center, from the NASA
SCaN program, and from DARPA under IAA 8839, Annex 130. PK and DB acknowl-
dge support from the NASA Academic Mission Services (contract NNA16BD14C). The
authors thank Ojas Parekh for helpful input and discussions regarding the arborescence
problem, Shon Grabbe for ongoing useful discussions, and Filip Maciejewski for helpful feedback on the work.
6.8 Chapter Appendix

6.8.1 Proof of claim 99

For an $n \times n$ integer matrix $W$, obtain matrices $W'$ and $W''$ by taking $W'_{ij} = nW_{ij} + j - 1$ and $W''_{ji} = nW_{ji}$. Set $D = W' \ast W''$. We aim to show that $\left\lfloor \frac{D}{n} \right\rfloor = W^{2,*}$ and $(D \mod n) + 1$ is a witness matrix for $W^{2,*}$.

**Proof.**

(i) We have

$$\left\lfloor \frac{D}{n} \right\rfloor_{ij} = \min_{k \in [n]} \{ nW_{ik} + k - 1 + nW_{kj} \} / n = \min_{k \in [n]} \left\{ W_{ik} + W_{kj} + \frac{k - 1}{n} \right\} = W_{ij}^2 + \min_{k \in [n]} \left\{ \frac{k - 1}{n} : W_{ik} + W_{kj} = W_{ij}^2 \right\} = W_{ij}^2.$$

(ii) Next,

$$D_{ij} = nW_{ij}^2 + \min_{k \in [n]} \left\{ k - 1 : W_{ik} + W_{kj} = W_{ij}^2 \right\}$$

gives us

$$(D \mod n) + 1 = \min_{k \in [n]} \left\{ k - 1 : W_{ik} + W_{kj} = W_{ij}^2 \right\} + 1 = \min_{k \in [n]} \left\{ k : W_{ik} + W_{kj} = W_{ij}^2 \right\},$$

which proves the claim.

6.8.2 The $\alpha > 0$ case in IdentifyClass

Part of the strategy of the algorithm in the main body of this chapter was be to assign each $v_{(i,j,k)} \in V$ into classes in accordance with approximately how many negative triangles are in $U_i \times U_j \times U_k'$ before starting the search.

To assign each node to a class, we use the routine IdentifyClass of [60], also described in the main text.
So far, we discussed the special case assuming $\alpha = 0$ in \texttt{IdentifyClass}. Hence we now consider the $\alpha > 0$ case.

For each $\alpha \in \mathbb{N}$, let us denote $c_{i,j,k}$ the smallest nonnegative integer satisfying $d_{i,j,k} < 10 \cdot 2^c \log n$, and

$$V_\alpha := \{v_{(i,j,k)} : c_{i,j,k} = \alpha\} \quad (6.6)$$

$$V_\alpha[i,j] := \{U'_k \in \mathcal{U}' : v_{(i,j,k)} \in V_\alpha\} \quad (6.7)$$

for any $i,j \in [n^{1/4}]$. Notably, $\mathcal{P}(i,j)$ contains at most $\sqrt{n}$ edges, so that $d_{i,j,k} \leq \sqrt{n}$ as well. Hence, $c = \frac{1}{2} \log n$ provides an upper bound for the minimum in step 2. The important immediate consequence is that we only need to consider $V_\alpha$ up to at most $\alpha = \frac{1}{2} \log n$.

**Lemma 114.** The \texttt{IdentifyClass} algorithm and the resulting $V_\alpha$ satisfy the following statements with probability at least $1 - 2/n$:

(i): The algorithm does not abort

(ii): $|\Delta(i,j,k)| \leq 2n$

(iii): For $\alpha > 0$, $v_{(i,j,k)} \in V_\alpha$, we have $2^{\alpha-3}n \leq |\Delta(i,j,k)| \leq 2^{\alpha+1}n$.

(iv): $|\Lambda_x(i,j) \cap \Delta(i,j,k)| \leq 100 \cdot 2^\alpha \sqrt{n} \log n$ for $i,j \in [n^{1/4}]$ and $\alpha \in \mathbb{N}$.

This provides an adapted version of lemma 106 for the $\alpha > 0$ case.

The following lemma provides a tool that will allow for ”duplication” of information to avoid message congestion in the network in the \texttt{EvaluationB} procedure.

**Lemma 115.** For all $\alpha \geq 0$ and $i,j \in [n^{1/4}]$,

$$|V_\alpha[i,j]| \leq \frac{720\sqrt{n} \log n}{2^\alpha} \quad (6.8)$$

**Proof.** The $\alpha = 0$ case is immediate since $|\mathcal{U}'| = \sqrt{n}$, so consider $\alpha \geq 1$. The “promise” in the FEWP subroutine we are in guarantees that for all $(u,v) \in S$, $\Gamma(u,v) \leq 90 \log n$,
so that for any $i, j \in [n^{1/4}]$, each edge in $P(U_i, U_j) \cap S$ has at most $90 \log n$ other nodes forming a negative triangle with it, leading to the inequality

$$\sum_{k: v(i,j,k) \in V_\alpha} |\Delta(i,j,k)| \leq 90n^{3/2}\log n.$$  

Using $|\Delta(i,j,k)| \geq 2^{\alpha - 3}n$ from part (i) of lemma 114, the conclusion follows. \hfill \Box

We now describe the implementation of step 3 of the ComputePairs procedure for the $\alpha > 0$ case.

3.1: Each node executes the IdentifyClass procedure.

3.2: For each $\alpha$:

For every $l \in [m]$, every node $v_{(i,j,k)}$ executes a quantum search to find whether there is a $U'_k \in V_\alpha[U_i, U_j]$ with some $w \in U'_k$ forming a negative triangle $(u^k_l, v^k_l, w)$ in $G$, and then reports all the pairs $u^k_l v^k_l$ for which such a $U'_k$ was found.

The $\alpha = 0$ case was described in the main text. We proceed to describe the classical procedure for invoking theorem 105 to obtain the speedup for the general $\alpha$ case, as in [60, §5.3.2]. Some technical precautions must be taken to avoid congestion of messages between nodes. This crucially relies on information duplication to effectively increase bandwidth between nodes. Lemma 115 provides a strong bound for the size of each $V_\alpha$. For this duplication of the information stored by the relevant nodes, a new labeling scheme is convenient. Suppose for simplicity that $C_\alpha := 2^\alpha/(720 \log n)$ is an integer, and assign each node a label $(u, v, w, y) \in V_\alpha \times [C_\alpha]$, which is possible due to the bound of lemma 115. The following EvaluationB implementable in $O(\log n)$ rounds (using a slightly sharper complexity analysis than [60]) can then be used for invoking theorem 105:

<table>
<thead>
<tr>
<th>EvaluationB</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> A list $(w^k_1, \ldots, w^k_m)$ of elements of $V_\alpha[u, v]$ assigned to each node $k =$</td>
</tr>
</tbody>
</table>
(u, v, x).

Promise: |L^k_w| ≤ 800 · 2^α \sqrt{n} \log n for each node k and all w ∈ V_α[u, v].

Output: Every node k = (u, v, x) outputs for each ℓ ∈ [m] whether some w ∈ w^k_ℓ forms a negative triangle {u^k_ℓ, v^k_ℓ, w}.

0. Every node (u, v, w) ∈ V_α broadcasts the edge information loaded in step 1 of ComputePairs to (u, v, w, y) for each y ∈ [C_α].

1. Every node (u, v, x) splits each L^k_w into smaller sublists L^k_w,1, . . . , L^k_w,C_α for each w, with each sublist containing up to \lceil |L^k_w|/C_α \rceil = \lceil 800 · 720 \sqrt{n} \log^2 n \rceil elements, and sends each L^k_w,y to node (u, v, w, y) along with the relevant edge weights.

2. Every (u, v, w, y) node returns the truth value

\[\min_{w ∈ w} \{W_{uw} + W_{uw}\} \leq W_{uv}\]

to node k for each uw ∈ L^k_w,y received in step 1.

For each value of α, we separately solve step 3.2 of the ComputePairs procedure. Since lemma 115 tells us that there are C_α times more nodes not in V_α than there are in V_α, every node in V_α can use C_α of those nodes not in V_α to relay messages and effectively increase its message bandwidth, which is exactly what EvaluationB takes advantage of. Steps 1 and 2 of the procedure take up to 2 · \lceil |L^k_w|/C_α \rceil ≤ 1600 · \log n rounds, since lists of size \lceil |L^k_w|/C_α \rceil are sent to C_α nodes, and the bound on α gives \lceil |L^k_w| \rceil ≤ 800n \log(n).

**Complexity of the Classical Analogue**

This subsection of the appendix serves to provide some supplemental information to §6.4.7 discussing the complexity of an algorithm for APSP with routing tables in the CONGEST-CLIQUE as proposed in [22] that has complexity \~O(n^{1/3}). Note that [22, corollary 6] applied to APSP distance computations only, whereas the routing table computations are discussed in [22, §3.4]. As shown there, O(\log^3) distance products (without witnesses) are needed to compute one distance product with a witness matrix.
More precisely:

1. Obtaining a witness matrix when witnesses are unique requires $\log(n)$ distance products.

2. The procedure for finding witnesses in the general case calls the procedure to find witnesses in the unique witness case $O(\log^2 n)$ times, or $2 \cdot \log^2 n$ times if $c = 2$ is deemed as sufficient for the success probability.

3. $\log n$ such distance products with witnesses are needed for the APSP algorithm with routing tables.

Then $2 \log^4 n$ distance products are computed in total for one distance product with witnesses. The distance product via the semi-ring matrix multiplication algorithm of [22, §2.1] uses $10n^{1/3}$ rounds ($4n^{1/3}$ for its steps 1 and 2, and $2n^{1/3}$ for step 3) using lemma 102, and hence one obtains the full round complexity of

$$10n^{1/3} \cdot 2 \log(n)^4 = g(n).$$ (6.9)

### 6.8.3 Expanding the DMST in the Distributed Setting

We handle the expansion of the DMST in the same way as in [43, §7], borrowing much of their discussion for our description here. However, as we have computed APSP distances along the way in place of SSSP, ‘unpacking’ the DMST becomes a bit simpler in our case.

Consider a component $H_i$ in one of the iterations of QDLSI, with input graph for the iteration being $G_i$. For each contraction in QDLSI, we determined edges $v_iu_i$, $v_i \notin H_i, u_i \in H_i$ minimizing $\beta_i := W_{v_iu_i} + d_G(u_i, c(H_i))$ to contract nodes. Recall that what happens in the iteration is that the cycle $c(H_i)$ and all nodes that have distance $\beta_i$ to $c(H_i)$ are contracted into one super-vertex. Denote that super-vertex by $V_{H_i,\beta_i}^*$. Let $G_{i+1}$ denote the graph obtained after this contraction. Then our goal, given a DMST $T_{i+1}$ for $G_{i+1}$, is to recover $G_i$ along with a DMST $T_i$ for $G_i$. We make use of the following Unpacking operation of [43, §7]:

202
Unpacking

Input: A digraph $G_{i+1}$ with a DMST $T_{i+1}$ with root $r$, a set of edges $H_i$ as in Quantum DMST Algorithm, a node $V_{H_i, \beta_i}^*$ of $G_{i+1}$ marked as a super-vertex, a set $c(H_i)$ of the nodes contracted into it, and $G_i$ the graph before contracting $c(H_i)$.

Output: A DMST $T_i$ for $G_i$ rooted at $r$.

1: For any $v_1, v_2 \notin V_{H_i, \beta_i}^*$, let edge $v_1v_2 \in T_i$ iff $v_1v_2 \in T_{i+1}$.

2: For $uV_{H_i, \beta_i}^* \in T_{i+1}$, which exists since $T_{i+1}$ is a DMST for $G_{i+1}$, denote the edge $uv^* := \arg\min_{uv: v \in V_{H_i, \beta_i}^*, u: \exists uv \in G_{i+1}} W_{vu} + d_G(u, c(H_i))$. Add $uv^*$ and the shortest path $\zeta$ connecting $v^*$ to $c(H_i)$ to $T_i$.

3: For any edge $V_{H_i, \beta_i}^* u \in T_{i+1}$ outgoing from the contracted super-vertex, add the edge $\arg\min_{vu: v \in V_{H_i, \beta_i}^*, u: \exists vu \in G_i} W_{vu}^{G_i}$ to $T_i$.

4: Add all edges $H_i \setminus \delta^\text{in}(\zeta)$ to $T_i$, where $\delta^\text{in}(\zeta)$ denotes all edges incoming on $\zeta$.

At the end of this procedure, $T_i$ is a DMST for $G_i$ [43, lemma 8]. We now describe how it can be implemented distributedly, needing only classical messages and information. For every contracted super-vertex, the following steps can be implemented at the same time, as will become clear in how the steps are executed for the nodes of each contracted super-vertex. Let us focus on unpacking one super-vertex $V_{H_i, \beta_i}^*$. Each node knows its neighbors in $G_i$, and every node’s super-vertex ID in $G_i$ and $G_{i+1}$, since each node stores this information before the initial contraction to $G_{i+1}$ in QDLSI happens. Hence, step 1 can be done locally at each node without any communication. Step 2 can be done by first having each node $v \in V_{H_i, \beta_i}^*$ send $\beta(u, v)$ to the other nodes in $V_{H_i, \beta_i}^*$, in one round, and then having each node of $V_{H_i, \beta_i}^*$ send to $v^*$ the routing table entry corresponding to its shortest path to $c(H_i)$ in $G_i$, also in one round (the nodes have already computed this information in QDLSI). Then $v^*$ notifies the nodes that are part of $\zeta$, which can then add the appropriate edge to $T_i$, needing yet another round, so that
step 2 can be done in three rounds of classical communication only. Step 3 is handled similarly. For the outgoing edge, each node in $V^*_H,\beta_i$ sends $W^{G_i}_{vu}$ to the other nodes in $V^*_H,\beta_i$, so that the appropriate edge to add to $T_i$ can be determined (in case of a tie, the node with smaller ID can be the one to add the edge), so this can be done in one round.

For step 4, every node in $\zeta$ notifies its neighbors that it is in $\zeta$, after which every node can determine which edges to add to $T_i$. For the unpacking of $V^*_H,\beta_i$, the information and communication for implementing its unpacking is contained in the nodes of $V^*_H,\beta_i$, so we can indeed unpack all vertices synchronously to obtain $G_i$ even when multiple super-vertices were contracted to get $G_{i+1}$. Hence, one layer of unpacking using this procedure can be implemented in 5 rounds (making use of the APSP and routing table information computed earlier before the contractions in QDLSI). Since there are at most $\lceil \log n \rceil$ contraction steps, the unpacking procedure can be implemented in $5 \cdot \lceil \log n \rceil$ rounds.

6.8.4 Information access

In remark 86, we mention that it suffices for all information regarding the input graph to be stored classically, with quantum access to it. Here, we expand on what we mean by that and refer the interested reader to [16] for further details.

While our algorithms use quantum subroutines, the problem instances and their solutions are encoded as classical information. The required quantum access refers to the ability to access the classical data so that computation in superposition of this data is possible. For instance, in the standard (non-distributed) Grover search algorithm, with a problem instance described by a function $g : X \rightarrow 0, 1$, we need the ability to apply the unitary operation $U_w|x\rangle = (-1)^{g(x)}|x\rangle$ to an $N$-qubit superposition state $|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle$, as was also described in Section 4.1.2. If we wish to use the distributed Grover search in example 88, in which the node $u$ leading the search tries to determine whether each edge $uv$ incident on it is part of a triangle in graph $G$, the unitary that node $v$ must be able to evaluate is the indicator function of its neighborhood, and $u$ must be able to apply the Grover diffusion unitary restricted to its neighborhood. Then after initializing the $N$-qubit equal superposition, nodes $u$ and $v$ can send
a register of qubits back and forth between each other, with $v$ evaluating the unitary corresponding to the indicator of its neighborhood and $u$ applying the Grover diffusion operator restricted to its neighborhood. The same ideas transfer over to a distributed quantum implementation of the EvaluationA (or EvaluationB) procedure. There, instead of evaluating unitary operators corresponding to indicators, in step 2, each node $v_{(i,j,k)}$ evaluates the unitary corresponding to the truth values of inequality 6.3 for the evaluation steps. That information is then returned to the node that sent it, which can then apply the appropriate Grover diffusion operator.

In general, quantum random access memory (QRAM) is the data structure that allows queries to the oracle. We can use circuit QRAM in our protocols or could make use of special-purpose hardware QRAM if it were to be realized. This choice does not affect the number of rounds of communication but would affect the efficiency of computation at each node. A main component of the distributed algorithms discussed in this work is quantum query access for each node to its list of edges and their weights in some graph $G$. This information is stored in memory, and the QRAM implementing the query to retrieve it can be called in time $O(\log n)$, resulting in a limited overhead for our algorithms. This retrieval of information takes place locally at each node; hence, this overhead does not add to the round complexity of our algorithms in the CONGEST-CLIQUE setting. We refer to [45] for more details on QRAM.
Conclusion

In this thesis, we have explored various facets of optimization, complexity, and applications in the realm of quantum computing. Through the investigation of information complexity in convex mixed-integer optimization (as discussed in Part I), we have extended our understanding of the inherent difficulty of solving optimization problems in different contexts, especially with access to different types of oracles. By refining the classical notion of first-order information complexity and establishing new lower bounds for those settings, we have shed light on the fundamental challenges associated with solving these problems. Further, the algorithms presented in both Chapters 2 and 3 giving valuable upper bounds on the complexity of these problems also present some novel insights to these problems and how one can use effective strategies under limited or approximate information.

With the lower bounds proven in Part I in mind, a natural question to contemplate is whether emerging quantum computing technologies may offer and help in overcoming these classical computational limitations. The oracles with respect to which we obtain the lower bounds of Part I have more powerful quantum counterparts, which would allow querying the oracle with inputs in superposition and receiving the respective superposition of the oracle’s answers. These types of oracles may be much more powerful. In fact, ongoing research explores exactly this question of quantum query complexity. For example, in [24] the authors formulate a quantum algorithm using fewer queries to a function value oracle (zero’th order oracle) for convex optimization than any known classical algorithm, which needs $\tilde{O}(d)$ queries to solve convex optimization in dimension $d$. If one could use ideas similar to those of Part I of this thesis to prove a superlinear
lower-bound on the number of function evaluations needed for convex optimization, the algorithm of [24] would use provably fewer oracle queries than any algorithm in the classical framework. Moving forward to Part II, we delve into contributions in quantum computing for optimization problems. The formulation of image denoising as a quadratic unconstrained binary problem using Boltzmann Machines and quantum annealing in Chapter 5 demonstrates a practical application of quantum annealing to a fundamental problem of high interest. Although the datasets (images) we can work with on current quantum machines are still very small, the impressive practical performance of the method demonstrates promise for quantum annealing to be a useful technology in the machine learning field.

Furthermore, the introduction of quantum distributed computing models and asymptotically faster algorithms in Chapter 6 provides a step towards surpassing classical computational bounds. While these algorithms may not yet outperform known lower bounds for classical models, they pave the way for future advancements in quantum computing that may eventually achieve such feats. It would be desirable to study classical lower bounds for the Steiner tree and DMST problems discussed in Chapter 6 for this reason, as proving a lower bounds of $\Omega(n^{1/3})$ in the classical model would make the presented algorithms provably faster than any possible classical algorithms. Our exploration into the exact complexity of these algorithms reveals current limitations unless significant improvements are made, emphasizing the need for continued research and development to make such algorithms practical for realistic problem sizes.

Finally, gazing upon all chapters of this thesis and taking a step back to look at the bigger picture, we conclude by acknowledging the impressive power that the intricate and sophisticated mathematical tools used throughout have in solving real problems. To juxtapose that on the one hand problems like image denoising, the Steiner tree problem, or mixed-integer problems like the facility location problem or the traveling salesman problem can be stated and understood in minutes with the fact that on the other hand modern tools from convex analysis, linear algebra, information theory, graph theory, statistics, and even quantum mechanics are involved in crafting solutions to these problems is a wonder in itself.
Bibliography


