STOCHASTIC FRANK-WOLFE METHODS AND HEALTHCARE DIALYSIS SCHEDULING APPLICATIONS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Natthawut Boonsiriphatthanajaroen

August 2026

© 2026 Natthawut Boonsiriphatthanajaroen
ALL RIGHTS RESERVED

STOCHASTIC FRANK-WOLFE METHODS AND HEALTHCARE DIALYSIS SCHEDULING APPLICATIONS

Natthawut Boonsiriphatthanajaroen, Ph.D.

Cornell University 2026

Stochastic constrained optimization is fundamental to decision-making in a wide range of applications, including artificial intelligence, operations research, and healthcare. While projected gradient methods are widely used, they can become computationally expensive due to the projection step, even for linearly constrained problems.

This thesis studies stochastic projection-free optimization through the Frank-Wolfe algorithm and its variants, which are particularly attractive for structured constrained optimization problems. We first establish convergence analyses for the standard and away-step Frank-Wolfe algorithms under appropriate assumptions on the objective function and accurate gradient approximation. We then extend these analyses to biased gradient estimators, the light-tail property of the algorithm's stopping time, and optimization problems with interior optima. To address the unfavorable search directions of the standard and away-step variants, we further analyze the boosted Frank-Wolfe algorithm and demonstrate its performance on unbounded feasible regions in comparison with projected gradient descent and away-step Frank-Wolfe.

Finally, we develop a simulation model of a dialysis clinic and propose technician's policies and patient scheduling methods to improve operational efficiency by reducing patient waiting times and overall completion times. Overall, this the-

sis contributes to both the theory of stochastic constrained optimization and the practice of operations research by advancing stochastic Frank-Wolfe optimization methods and developing simulation-based methodologies for improving dialysis clinic operations.

BIOGRAPHICAL SKETCH

Natthawut Boonsiriphatthanajaroen was born in Bangkok, Thailand. He obtained his Bachelor of Science in Engineering, majoring in Operations Research and Financial Engineering from Princeton University in 2021. He began his Ph.D in the School of Operations Research & Information Engineering at Cornell University, with interests in simulation-optimization algorithms and modeling.

To my family.

ACKNOWLEDGEMENTS

I would first like to express my deepest gratitude to my Ph.D. advisor, Professor Shane Henderson, for his unwavering support, encouragement, and guidance throughout my doctoral journey. His passion for research inspired me to pursue my own curiosity through careful problem solving and exploration. Beyond the countless hours he devoted to discussing ideas and patiently answering even my most elementary questions, he taught me what it means to be a thoughtful researcher—one who thinks critically, asks the right questions, and seeks to understand a problem from multiple perspectives before pursuing new results. I have always enjoyed the process of cultivating curiosity and exploring the many facets of a problem before attempting to solve it, and this mindset is one of the greatest lessons I have learned from him. Beyond research, Shane also provided opportunities that helped me develop valuable skills for my future career, particularly through my experience as a Teaching Assistant, where I strengthened my analytical thinking and communication abilities. These lessons and experiences are invaluable, and I will carry them with me throughout my career.

I would also like to express my sincere gratitude to my committee members, Prof. David Shmoys and Prof. Peter Frazier, for their guidance and inspiration. Taking David's Combinatorial Optimization course was a truly rewarding experience. It deepened my appreciation for algorithmic thinking and demonstrated how elegant algorithms can be applied to solve challenging real-world problems in operations research. I also greatly enjoyed Peter's Mathematics of Deep Learning course. His lectures were exceptionally clear and engaging, and he has a remarkable ability to provide intuitive explanations for complex concepts and recent advances in deep learning. Their teaching has broadened my perspective and enriched my graduate education.

I would like to express my sincere thanks to Prof. Katya Scheinberg. Taking my first optimization course at Cornell, as well as her seminar on derivative-free optimization, inspired me to pursue research in optimization. Her research style, emphasis on rigorous analysis, and elegant techniques have also had a profound influence on my own work in stochastic optimization.

I would like to express my special thanks to Prof. Nikhil Garg and Prof. Andre Lodi for their guidance and support during one of my first research projects in my Ph.D. journey. Working with them provided me with my first experience tackling an applied operations research problem and exposed me to the process of translating practical challenges into mathematical models and solution approaches. This experience was invaluable in shaping my research perspective and equipped me with skills that have prepared me well for future applied research projects.

I would also like to thank everyone in the ORIE community for making my Ph.D. journey both enjoyable and memorable. I am especially grateful to Billy Jin and Liwei Jiang for their generous advice and encouragement throughout graduate school and during the job search process. I am also thankful to Hugo Maniguy, whose friendship and thoughtful conversations about both research and life made my time at Cornell all the more meaningful. Finally, I would like to express my appreciation to my research collaborators—Joe Ye, Rongyi He, and Litong Liu.

I would also like to express my sincere appreciation to the wonderful staff in the ORIE department for their support throughout my Ph.D. I am especially grateful to Henry Lam, Mingming Wang, Odrick Palmer, and Troy Parish for their kindness and assistance. In particular, Henry Lam and Mingming Wang were always incredibly helpful and responsive whenever I had questions or needed guidance. Their support made navigating graduate school much easier, and I truly appreciate all

that they have done.

Lastly, I would like to express my deepest gratitude to my family for their unconditional love and unwavering support throughout my life. No matter the circumstances, they have always stood by my side, believing in me and encouraging me every step of the way. I would not have achieved any of this without their love, sacrifices, and constant presence. This accomplishment is as much theirs as it is mine.

TABLE OF CONTENTS

Biographical Sketch . . . iii
Dedication . . . iv
Acknowledgements . . . v
Table of Contents . . . viii
List of Tables . . . x
List of Figures . . . xi

1 Introduction **1**
1.1 Frank-Wolfe algorithm and variants . . . 4
1.2 Dialysis Clinic . . . 7
1.3 Contribution and Organization . . . 9

2 Standard and Away-step Frank-Wolfe **12**
2.1 Introduction . . . 12
2.2 The Standard and Away-Step Frank-Wolfe Algorithms . . . 20
2.3 Preparatory Results for the Convergence Analysis . . . 24
2.3.1 Results for the Standard Frank-Wolfe Algorithm . . . 25
2.3.2 Results for the Away-step Frank-Wolfe Algorithm . . . 27
2.4 Tail Bounds for Gradient Estimation Error . . . 36
2.4.1 The Bounded Second Moment Case . . . 37
2.4.2 The Sub-Gaussian Case . . . 38
2.5 Complexity with Unbiased Gradient Estimators . . . 39

3 Frank-Wolfe Algorithm Extensions **61**
3.1 Introduction . . . 61
3.2 Complexity with Biased Gradient Estimators . . . 62
3.3 The Time to Reach Near-Optimality Has a Light Tail . . . 82
3.4 Improved Complexity of the standard Frank-Wolfe algorithm with an Interior Optimum . . . 95

4 Boosted Frank-Wolfe **111**
4.1 Introduction . . . 111
4.2 Boosted Frank-Wolfe Algorithm . . . 116
4.3 Preparatory Results for Convergence Analysis . . . 122
4.3.1 Results in the Deterministic Setting . . . 125
4.3.2 Results in Stochastic Setting . . . 129
4.4 Complexity for a Bounded Feasible Region . . . 136
4.5 Extensions of Bounded Boosted Frank-Wolfe . . . 157
4.5.1 Complexity with Biased Gradient Estimators . . . 157
4.5.2 Time to Reach Near-Optimality Has a Light Tail . . . 175
4.5.3 Improved Complexity of the boosted Frank-Wolfe algorithm with an Interior Optimum . . . 185

5 Boosted Frank-Wolfe over an Unbounded Feasible Region **196**
5.1 Introduction . . . 196
5.2 Unbounded Boosted Frank-Wolfe Algorithm . . . 199
5.3 Problems . . . 205
5.4 Initial Experiments . . . 206
5.5 Additional Algorithms . . . 207
5.6 Additional Experiments . . . 209
5.7 Conclusions . . . 212

6 Dialysis Scheduling **215**
6.1 Introduction 215
6.2 Simulation Framework for Dialysis Clinic Operations 223
6.3 Operating Policies 231
6.4 Heuristics for Scheduling 241
6.5 Numerical Experiments 245
6.6 Building the Super Policy via Classification 254
6.7 Conclusions 264

Bibliography **267**

LIST OF TABLES

2.1 Complexity results established in this paper. The column "Sub-Gaussian" indicates whether this is assumed of the gradient estimates or not. If it is not assumed, then we assume bounded variance. 18

LIST OF FIGURES

3.1 An example of zig-zagging behavior of standard Frank-Wolfe. . . . 99

4.1 An example of a standard Frank-Wolfe path (left), and an away-step Frank-Wolfe path (right). . . . 112
4.2 The AFW algorithm mitigates zigzagging behavior in the original Frank-Wolfe algorithm, by moving not just towards vertices, but also away from vertices. The BFW algorithm has even more flexibility in choosing a search direction. . . . 115
4.3 Computing the search direction. In (a), Vertex C solves the linear program, and we project the negative gradient onto u_0, leaving the residual r_1. In (b), Vertex B solves the linear program, and we project the residual, r_1, onto u_1, leaving the residual r_2, which sufficiently improves alignment and thus defines the direction d in (c). In (d), the alignment between the direction d and the negative gradient is highlighted. . . . 120
4.4 The negative gradient $g(x)$ points outward and the search direction is not boosted. . . . 175

5.1 An example of the SAN problem consisting of 9 nodes and 13 edges (left), and an example of the SMFCVX problem with 10 nodes. . . . 205
5.2 An instance of SAN tackled using Algorithm 7 exhibits a plateau in the budget interval $250 - 750$ (left), and an instance of OJN has a plateau in the budget interval $500 - 2000$. . . . 206
5.3 An example of the plateau effect with a 2D problem. The optimal solution x^* lies far away from the vertices below. The search direction will not be boosted and only follows an extreme ray from Linear Programming 207
5.4 A plane (dashed line) perpendicular to the negative gradient is added as a new constraint in an unbounded boosted Frank-Wolfe (left), and away-steps are added to the components of final search direction (right). . . . 208
5.5 An instance of SAN (left), and OJN (right), tackled using variants of Algorithm 4. . . . 209
5.6 Area scatter plot of different variants of BFW solvers (left) and CDF solvability profile at level 0.2 (right) on all test problems. . . . 210
5.7 Area scatter plot on different solvers (left) and CDF solvability profile at level 0.2 (right). . . . 211
5.8 Mean progress curves of SMFCVX-R5 (left) and OPENJACKSON-R6 (right). . . . 212

6.1 Dialysis Treatment Time Distribution 223
6.2 An example of simulation model 229
6.3 An example of simulation results of all patients 230
6.4 Simulation results with SPT (left) and LPT (right) 240
6.5 Simulation results with Quick Rolling Optimization (left) and Greedy Rolling Optimization (right) 241
6.6 An example of push (left) and interlaced (right) schedule 244
6.7 An example of push ascending scheduling (left) and push descending scheduling (right) schedule 245
6.8 Waiting Time under Push scheduling 246
6.9 Waiting Time under Interlaced scheduling 247

6.10 Waiting time on all policies of push asis (left) and interlaced asis (right) 248
6.11 Makespan under Push scheduling 249
6.12 Makespan under Interlaced scheduling 250
6.13 Makespan on all policies of push asis (left) and interlaced asis (right) 251
6.14 Box plots on all policies of push (left) and interlaced (right) scheduling 251
6.15 Combined objective function on all policies of push asis (left) and interlaced asis (right) 253
6.16 Waiting time of Super Policy and other policies 260
6.17 Makespan of Super Policy and other policies 261
6.18 Combined Objective of Super Policy and other policies 262
6.19 Feature importance from the classifier under push asis (left) and push descending (right) 263

CHAPTER 1

INTRODUCTION

Optimization plays a central role in decision-making across a wide range of disciplines, including machine learning, artificial intelligence, operations research, finance, supply chain management, and healthcare. In many of these applications, decision-makers must operate in environments characterized by uncertainty. For example, customer demand in supply chains, service times in healthcare systems, travel times in transportation networks, asset returns in financial markets, and training data in machine learning applications are inherently random and cannot be known exactly in advance. Consequently, optimization models must account for stochasticity in order to produce solutions that perform well under uncertain conditions. Recent advances in artificial intelligence have further increased the importance of large-scale optimization, as modern machine learning models and large language models are often trained using massive datasets that can be viewed as random samples from an underlying data-generating process. As a result, stochastic optimization has emerged as a fundamental framework for decision-making under uncertainty across many scientific and engineering disciplines. In many of these applications, decision variables are subject to operational, physical, or resource limitations, leading naturally to constrained optimization problems of the form

$$\min_{x \in \mathcal{X}} f(x) = \mathbb{E}[F(x, \xi)],$$

where f is an objective function, x denotes the decision variables ξ represents a random vector capturing uncertainty and $\mathcal{X}$ denotes the feasible region. The constraint set $\mathcal{X}$ may represent budget limitations, capacity restrictions, conservation laws, sparsity requirements, risk controls, or other application-specific re-

quirements. As optimization models continue to grow in scale and complexity, the development of efficient algorithms for constrained optimization has become increasingly important.

Constrained optimization arises in numerous modern applications. In operations research, optimization models are widely used for transportation, network flow, scheduling, inventory control, and resource allocation problems [45]. In finance, portfolio optimization problems often incorporate budget and risk constraints, while in machine learning and statistics, constrained optimization formulations appear in data fitting, regression and classification problems [13]. The prevalence of such applications has motivated extensive research into scalable optimization algorithms capable of handling large and structured feasible regions efficiently.

Constraints play a fundamental role in these applications because they capture practical limitations that feasible decisions must satisfy. For example, transportation and network flow problems are subject to flow conservation and capacity constraints, while scheduling problems must satisfy resource availability and precedence constraints. Portfolio optimization models often impose budget, leverage, and risk limits to ensure that investment decisions remain financially viable. In machine learning, sparsity constraints may be used to improve model interpretability, and norm constraints are frequently employed to promote regularization and robustness. Consequently, the feasible region is frequently determined by complex operational, statistical, or physical requirements, and its structure plays a critical role in the design and computational efficiency of optimization algorithms.

$$\arg\min_{x \in \mathcal{X}} \|x - y\|^2$$

denotes the Euclidean projection of y onto the feasible region $\mathcal{X}$. Projection is an attractive mechanism because it guarantees that every iterate remains feasible while preserving many desirable convergence properties of unconstrained gradient methods.

However, the computational cost of projection can vary significantly depending on the geometry of the feasible region. While projections onto simple sets such as boxes or Euclidean balls admit closed-form solutions, projections onto more structured feasible regions may require solving a nontrivial optimization problem at every iteration. For example, projection onto the nuclear norm ball requires a singular value decomposition, which can be computationally expensive for large matrices [14], and projections onto the spectrahedron involve eigenvalue computations [29].

The difficulty of projection is not limited to nonlinear feasible regions. Even for polyhedral feasible sets, projection may be substantially more expensive than linear optimization. For several structured polytopes, Euclidean projection remains computationally demanding. Examples include network flow polytopes, matching polytopes, and matroid polytopes [41], [14]. Consequently, projection-free methods may offer significant computational advantages even when the feasible region is polyhedral.

These computational challenges have motivated the development of projection-free optimization methods that avoid explicit projections while maintaining feasibility. Among such methods, the Frank–Wolfe algorithm and its variants have attracted

significant attention due to their ability to replace projection operations with linear optimization subproblems, which are often substantially easier to solve for many structured feasible regions.

1.1 Frank-Wolfe algorithm and variants

In this thesis, we analyze the convergence behavior of standard Frank-Wolfe and two of its variants for linearly constrained stochastic optimization problems. We investigate these algorithms with a view towards simulation optimization with linear constraints that define a bounded polytope, i.e., we consider the problem

$$\begin{aligned} \min_x & f(x) = \mathbb{E}[F(x,\xi)] \\ \text{s.t. } & Ax \leq b \end{aligned} \tag{1.1}$$

where the set $\mathcal{X} = \{x : Ax \leq b\}$ is a bounded polytope in $\mathbb{R}^d$ with $d < \infty$. The matrix A and vector b are deterministic and known, ξ is some random element, the distribution of which does not depend on x, and the function $F(x,\cdot)$ is real-valued and integrable for all feasible x. $F(x,\xi)$ represents an objective function realization obtained from a single simulation replication at the solution x. We can generate independent realizations of $F(x,\xi)$ and we estimate $f(x)$ using sample averages. This problem is then a *linearly constrained simulation optimization problem* (LCSOP). We assume we have access to gradient estimates. Unbiased gradient estimates can be obtained in many, but not all, simulation applications using infinitesimal perturbation analysis, which can be computed, e.g., through automatic differentiation of simulation replications [28]. Unbiased gradient estimators can also often be obtained using the likelihood ratio method or extensions thereof [33, 68]. We study the convergence behavior of the standard Frank-Wolfe

and its variants under certain function and gradient properties. Our goal is to find a solution x such that $f(x) - f^* \leq \varepsilon$ for some pre-specified $\varepsilon > 0$, where f^* is the optimal objective value, and study the terminal time T_ε when this defined error is achieved.

LCSOPs are important. For example, Cornell University solved a problem during the COVID-19 pandemic that determined surveillance testing capacity for different campus groups [27]. There the goal was to select testing rates for the groups so as to minimize pandemic outcomes like infections over a full semester. A single linear constraint represented the bounded testing capacity. The pandemic outcomes were estimated through a stochastic simulation. This problem can be formulated as a LCSOP. Another classical application is supervised learning, where the objective is to minimize the expected risk subject to a feature-budget constraint [14, 21]. Such constraints arise when only a limited number of features can be selected due to computational, storage, or interpretability considerations. Beyond supervised learning, [21] discusses a variety of additional applications, including traffic assignment problems and collaborative filtering problems, which can also be formulated as constrained optimization problems.

Despite its computational advantages, the standard Frank–Wolfe algorithm can suffer from slow convergence. Since each iteration moves toward a vertex of the feasible region, the iterates may exhibit a zig-zagging behavior, particularly when the optimal solution lies on or near a face of the feasible region. This phenomenon can significantly slow the algorithm's progress. To address this limitation, several works, including [53], [54], and [7], study the away-step Frank–Wolfe algorithm. In addition to moving toward a vertex, away-step Frank–Wolfe allows the iterate to move away from previously selected vertices, thereby reducing the zig-zagging

effect and accelerating convergence.

More recently, [21] proposed the boosted Frank–Wolfe algorithm, which further improves the quality of the search direction. Rather than selecting a single direction toward a vertex, the algorithm combines multiple Frank–Wolfe directions to construct a direction that is better aligned with the negative gradient. As a result, the search direction more closely resembles that of gradient descent while retaining the projection-free nature of Frank–Wolfe methods. These developments demonstrate that substantial improvements in convergence can be achieved by enhancing the search direction without sacrificing the algorithm's computational advantages. With these enhancements provided by the away-step and boosted Frank–Wolfe algorithms, we investigate their convergence behavior for linearly constrained optimization problems.

In addition to the main convergence analyses of the standard, away-step, and boosted Frank–Wolfe algorithms, this dissertation investigates several extensions and properties that provide further insight into their behavior. These include the use of biased gradient estimators, the special case in which the optimal solution lies in the interior of the feasible region, and the light-tail property of the stopping time T_ε. In practice, unbiased gradient estimators are not always available. For example, when infinitesimal perturbation analysis cannot be applied, one often resorts to finite-difference methods, which typically produce biased gradient estimates. We show that the complexity analyses in the unbiased and biased settings are closely related, although the resulting guarantees in the biased case are necessarily weaker. We also show that when the optimal solution lies strictly in the interior of the feasible region, the standard Frank–Wolfe algorithm can achieve significantly faster convergence under appropriate assumptions, attaining rates comparable to those

of away-step Frank–Wolfe. Finally, our analyses reveal a light-tail property of the stopping time T_ε, providing additional probabilistic guarantees on the number of iterations required to reach a desired level of accuracy.

Beyond these theoretical developments, we conduct numerical studies to better understand the practical behavior of the boosted Frank–Wolfe algorithm, particularly in problems with unbounded feasible regions. Returning to the COVID-19 testing example, if the testing-capacity constraint is relaxed and replaced by a cost term that penalizes high testing rates, the resulting problem becomes a linearly constrained stochastic optimization problem with an unbounded feasible region. This formulation more closely reflects real-world decision making, where testing capacity was gradually expanded as the benefits of widespread testing became better understood. Similar unbounded formulations arise in a variety of applications, including machine learning, signal processing, trend filtering, and basis pursuit problems [55, 51, 20]. Through these numerical experiments, we explore the behavior of boosted Frank–Wolfe in settings that are not covered by existing theory and identify several interesting empirical phenomena that motivate future research.

1.2 Dialysis Clinic

In addition to methodological contributions in stochastic optimization, this dissertation explores an application of operations research in healthcare systems through the study of hemodialysis clinic operations. Hemodialysis is a life-sustaining treatment for patients with kidney failure and is typically administered multiple times per week, with each treatment session lasting several hours. As a result, dialysis clinics must coordinate patient appointments, chair utilization, and technician

workloads throughout the day. Inefficient scheduling and operational decisions can lead to long patient waiting times, underutilized resources, and extended clinic completion times. These challenges are further complicated by uncertainty in treatment durations, patient arrivals, and service times, making dialysis clinics a natural application area for simulation and optimization methods.

This dissertation investigates the daily operation of a dialysis clinic from two perspectives: patient scheduling and technician decision-making policies. We develop a discrete-event simulation model that captures patient arrivals, treatment processes, technician activities, and resource utilization within the clinic. Using this simulation framework, we evaluate several scheduling methods and operational policies inspired by both healthcare appointment scheduling and classical job scheduling literature. In particular, we adapt scheduling methods from [78] and technician policies from task-processing heuristics studied in [73, 17, 35, 76]. These methods utilize patient treatment times and arrival information to determine appointment schedules and guide technician decisions throughout the day. The primary objective is to reduce patient waiting times, shorten clinic makespan, and improve overall operational efficiency.

Beyond simulation and optimization, machine learning has emerged as an important tool for improving healthcare appointment scheduling and treatment planning, as demonstrated in [40, 79, 16]. Motivated by these developments, we propose a machine-learning-based "Super Policy" that adapts technician decisions to the characteristics of the patients scheduled on a given day. The policy is trained using simulated data together with patient-specific features, including treatment durations, appointment schedules, and other derived statistics, to identify operational strategies that perform well under different scheduling environments. Through ex-

tensive numerical experiments, we show that the Super Policy consistently achieves performance comparable to or better than existing policies while remaining simple to implement in practice. More broadly, this study demonstrates how simulation, optimization, and machine learning can be integrated to support operational decision making in healthcare systems and improve the efficiency of complex service operations.

1.3 Contribution and Organization

The following works form the foundation of several chapters of this dissertation:

1. *Evaluating Solvers for Linearly Constrained Simulation Optimization* [10]. This work investigates the empirical performance of several algorithms for linearly constrained stochastic optimization problems, including the away-step Frank–Wolfe algorithm. The paper has appeared.
2. *Convergence Analysis of Stochastic Standard and Away-step Frank–Wolfe* [12]. This work develops convergence and complexity analyses for stochastic Frank–Wolfe algorithms under linearly constrained stochastic optimization problems. The results form the theoretical foundation of Chapters 2 and 3, where we establish iteration complexity and sample size complexity. It is currently under submission.
3. *Numerical Results of Stochastic Unbounded Boosted Frank–Wolfe* [11]. This work studies the practical performance of boosted Frank–Wolfe algorithms on stochastic optimization problems with unbounded feasible regions. The has appeared.

Together, these works provide both computational and theoretical perspectives on the development of Frank–Wolfe methods for linearly constrained stochastic optimization.

This dissertation is organized as follows.

1. Chapter 2 studies the convergence behavior of the standard and away-step Frank–Wolfe algorithms under appropriate assumptions on the objective function. Due to the stochastic nature of the problem, we first characterize the accuracy required of gradient estimators and establish conditions under which these accuracy requirements are satisfied. We then derive iteration complexity and sample size complexity results for both algorithms.
2. Chapter 3 extends the analyses of Chapter 2 by introducing additional assumptions on the objective function and problem setting. Under these assumptions, we investigate several extensions, including biased gradient estimators, the case where the optimal solution lies in the interior of the feasible region, and the light-tail property of the stopping time.
3. Chapter 4 analyzes the convergence behavior of the boosted Frank–Wolfe algorithm. Similar to Chapter 2, we establish iteration complexity and sample size complexity results under stochastic gradient estimates and identify conditions under which the algorithm achieves accelerated convergence. We also present several extensions of the boosted Frank–Wolfe analysis that parallel those developed in Chapter 3.
4. Chapter 5 conducts numerical experiments on problems with unbounded feasible regions and compares the performance of the boosted Frank–Wolfe algorithm with projected gradient descent and away-step Frank–Wolfe.

5. Chapter 6 investigates the operation of a dialysis clinic using a discrete-event simulation model that captures patient flow and clinic resources. We evaluate various scheduling methods and technician policies with respect to patient waiting time, clinic makespan, and overall system performance. Building upon these results, we develop a machine-learning-based "Super Policy" that utilizes patient information and historical data to achieve performance comparable to or better than existing policies across a range of scheduling environments.

CHAPTER 2

STANDARD AND AWAY-STEP FRANK-WOLFE

2.1 Introduction

In this chapter, we study convergence behavior of standard and away-step Frank-Wolfe algorithms for the problem

$$\begin{aligned} \min_{x} & f(x) = \mathbb{E}[F(x, \xi)] \\ \text{s.t. } & Ax \leq b \end{aligned} \tag{2.1}$$

The Frank–Wolfe algorithm has been extensively studied for constrained optimization problems due to its ability to avoid costly projection operations. Instead of projecting onto the feasible region, Frank–Wolfe only requires the solution of a linear optimization problem over the feasible set at each iteration. This advantage is particularly important when projections are computationally expensive. For example, projections onto the trace norm ball and spectrahedron can be significantly more costly than solving the corresponding linear optimization problem [5, 29]. Moreover, projection can remain expensive even for certain polytopes, including the Birkhoff polytope, matching polytope, flow polytope, and matroid polytope [41, 14]. As a result, projection-free methods based on linear optimization oracles are often more attractive in these settings.

Despite these advantages, the standard Frank–Wolfe algorithm may converge slowly on polyhedral feasible regions. In particular, when the optimal solution lies on or near a face of the feasible region, the iterates can exhibit a zig-zagging

behavior between vertices, resulting in poor progress toward the optimum. To address this issue, several works have studied the away-step Frank–Wolfe algorithm, which augments the standard Frank–Wolfe direction with away directions that help remove undesirable vertices from the current representation of the iterate [54, 53, 7]. The convergence behavior of away-step Frank–Wolfe depends on geometric properties of the feasible region, such as pyramidal width and related condition measures. Under appropriate assumptions, these works show that away-step Frank–Wolfe achieves significantly faster convergence rates than the standard Frank–Wolfe algorithm in deterministic settings.

Convergence analyses of the Frank-Wolfe algorithm for deterministic objective functions have been extensively studied [7, 53, 54]. The standard Frank-Wolfe algorithm requires $\mathcal{O}(\varepsilon^{-1})$ iterations to achieve ε-optimal solutions, while the away-step Frank-Wolfe algorithm requires $\mathcal{O}(\log(\varepsilon^{-1}))$ iterations. The convergence rate of the away-step variant depends on the geometry of the feasible region, characterized by a property known as the pyramidal width. The pyramidal width does not need to be explicitly computed to be able to implement the algorithms or establish their complexity properties, which is fortunate because it is challenging to compute. Although linear convergence under strong convexity can be achieved by a modified Frank-Wolfe algorithm [31], this improvement comes at the cost of a specialized local search oracle that requires explicit knowledge of all vertices of the polytope, making it impractical for many large-scale problems. Motivated by large-scale data-driven applications, recent work has turned to understanding how these guarantees adapt to stochastic objectives.

The study of the stochastic setting [42, 61, 64], where the objective function has the form (2.1), focuses on finite-sum minimization problems, where the objective

function is the average of n loss terms, $f_1, f_2, \ldots, f_n$, one for each of n data points. This setting exactly matches (2.1) only when the random element ξ has a discrete uniform distribution. This is true for the *sampled* problem in the important special case of sample-average approximation or empirical loss minimization, but not in general. Additionally, [61] and [64] consider objective functions of the form $f_i(x_i^T w)$, where w is the optimization variable and x_i is a given data point. The gradient estimator in the aforementioned studies is composed of n entries, one for each data point. At each iteration, the gradient estimate is updated by randomly selecting a single data point and updating only the corresponding gradient-estimate entry, eliminating the need for separate sample size analysis in gradient estimation. The iteration complexity in this setting is $\mathcal{O}(\varepsilon^{-1})$, with additional dependence on the dataset size n. A related line of work is the stochastic away-step Frank–Wolfe method of [34], which also operates within the finite-sum framework. Their analysis establishes an iteration complexity $\mathcal{O}(\log(\varepsilon^{-1}))$, but this requires additional assumptions on each of the function f_i and comes at the cost of exponentially increasing sample sizes per iteration. These convergence guarantees remain tailored to the finite-sum regime, leaving open the question of how to extend such results to the more general expectation-minimization setting we consider.

The analysis in [90], inspired by the SPIDER gradient-estimation technique of [22] and SARAH gradient-estimation technique of [63], applies to a broad class of objective functions, including finite-sum minimization, expectation minimization, and certain nonconvex problems. The authors establish a total work complexity of $\mathcal{O}(\varepsilon^{-2})$ for the Frank–Wolfe algorithm. Their results rely on additional assumptions, namely a Lipschitz-gradient condition on the component functions f_i in the finite-sum setting and on the stochastic function $F(x, \xi)$ in the expectation-minimization setting. The improved convergence rate is achieved through a

variance-reduction mechanism that uses carefully designed, iteration-dependent step sizes to control the gradient-estimation error. Unfortunately, such step-size schedules are incompatible with the away-step Frank–Wolfe method, where the admissible step size is capped by an iteration-specific upper bound, preventing this variance-reduction strategy from being directly applied. Overall, [90] highlights the critical role of gradient-estimation accuracy and demonstrates how improving gradient quality can significantly enhance both theoretical complexity bounds and algorithmic performance.

The insight that gradient-estimate quality can significantly improve complexity results is echoed in several other works on the convergence of Frank–Wolfe-type methods. Indeed, the quality of gradient approximations has been analyzed without relying on a finite-sum representation of the objective function in [57]. There, it is assumed that the stochastic error in the gradient estimates decreases over iterations, leading to convergence results that guarantee a small error with high probability. However, this study does not examine the required sample size for gradient estimation, which is a central question if Frank-Wolfe algorithms are to be applied in simulation optimization. This motivates the study of adaptive sampling and gradient-estimation strategies that explicitly account for iteration and sample complexity.

Early work on stochastic setting is [41] which acheives iteration complexity of $\mathcal{O}(\varepsilon^{-2})$, however, several works after that have explored methods for improving gradient estimation through variance-reduction techniques and iteration-dependent sample size selection. Results from [42], [58], [69], [62], [75], and [39] establish improved convergence guarantees for the standard Frank–Wolfe algorithm compared with those presented in this paper, often without requiring Lipschitz continuity

of the gradient, unlike the SPIDER-based approach in [90]. Similarly, [80] extends the SCGS framework of [58] by incorporating robust gradient estimators that accommodate sub-Gaussian, weak-exponential, and weak-moment noise distributions. While this approach broadens the range of stochastic settings under which convergence guarantees can be established, it retains the same order of complexity as the original SCGS method. In addition, [6] applies momentum-based variance reduction to improve gradient estimation while requiring only one sample per iteration, although this approach achieves a suboptimal iteration complexity of $\mathcal{O}(\varepsilon^{-3})$. The work in [91] further improves upon this result by requiring only one sample while obtaining a total work complexity of $\mathcal{O}(\varepsilon^{-2})$. However, this method requires Hessian computations and additionally assumes bounded fourth moments of the gradient of the logarithm of the probability distribution.

All of these works rely on deterministic step size policies in which the step sizes diminish over iterations. However, due to the step size rules used in Away-step Frank–Wolfe methods from [7], [54], and [53], the step size must depend explicitly on the estimated gradient, the search direction, and the smoothness constant. Consequently, applying a deterministic diminishing step size policy solely to improve gradient estimation is not suitable for Away-step Frank–Wolfe algorithms, since excessively small step sizes may fail to produce sufficient decrease. While variance reduction may lead to improved performance for the standard Frank–Wolfe algorithm, we instead adopt a unified framework across all algorithms considered in this work, particularly because these variance-reduction approaches are not directly applicable to Away-step Frank–Wolfe methods.

Some other works have explicitly characterized the sample sizes required to achieve a targeted iteration complexity. For instance, [34] establish an iteration complexity

of $\mathcal{O}(\log(\varepsilon^{-1}))$, but this rate is obtained at the cost of exponentially increasing sample sizes across iterations. Similarly, [90] specifies the sample sizes needed for gradient estimation at each iteration, with these requirements depending explicitly on the iteration index.

Sample-complexity results related to ours have been established for other optimization algorithms, and partly inspired the present paper. Works such as [9], [19], and [67] investigate optimization methods that incorporate simulated gradients to achieve an ε-neighborhood of the optimal solution, often using a Lyapunov function framework. In particular, the analysis of the trust-region method in [9] centers on "good events," in which function or gradient estimates are sufficiently accurate. They show that the algorithm can be designed so that these good events occur with high probability, ensuring continued function improvement even in the presence of low-probability "bad events." Moreover, they establish the necessary sample size required to make good events likely, providing important insights into sample complexity in stochastic optimization.

Our primary contribution is to establish new convergence analyses and rate guarantees for the standard Frank-Wolfe and away-step Frank-Wolfe algorithms for the problem (2.1) under convexity and strong convexity assumptions respectively, which extends the formulation studied in [61] and [64]. We also provide convergence analyses under a sub-Gaussian gradient assumption for both the standard and away-step Frank-Wolfe algorithms, thereby extending the results of [80], which consider only the standard Frank-Wolfe method built upon the sliding-gradient framework of [58]. Furthermore, we extend our analysis to accommodate biased gradient estimates, motivated by [3], which studies bias in the context of stochastic gradient descent. Our approach is partly inspired by the Lyapunov function

analyses [9, 67] for trust-region methods. To understand sample size requirements we assume either bounded variance, or a uniform sub-Gaussian property, of the simulated gradients. Compared to [61, 57, 64], and [62], our work examines a more general class of objective functions than those seen in machine learning applications, ensuring relevance in simulation-optimization applications.

Table 2.1 summarizes the complexity results established in this paper when using unbiased gradient estimators, which we view as the primary contribution. The iteration complexity results are established in Theorem 2.5.2. The sample-size complexity results are established in Theorems 2.5.3 and 2.5.4. The total complexity results are the product of the sample size and iteration complexities. When using biased gradient estimators the complexities are the same, though we cannot allow the error, ε to be arbitrarily small, because the bias in the gradient estimators can be disruptive.

Algorithm	**Sub-Gaussian?**	**Sample Size**	**Iteration**	**Total**
Standard	No	$\mathcal{O}\left(\frac{1}{\varepsilon^4}\right)$	$\mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$	$\mathcal{O}\left(\frac{1}{\varepsilon^6}\right)$
Away-Step	No	$\mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$	$\mathcal{O}\left(\frac{1}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{1}{\varepsilon^3}\right)$
Standard	Yes	$\mathcal{O}\left(\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$	$\mathcal{O}\left(\frac{1}{\varepsilon^4}\log\frac{1}{\varepsilon}\right)$
Away-Step	Yes	$\mathcal{O}\left(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{1}{\varepsilon}\right)$	$\mathcal{O}\left(\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon}\right)$

Table 2.1: Complexity results established in this paper. The column "Sub-Gaussian" indicates whether this is assumed of the gradient estimates or not. If it is not assumed, then we assume bounded variance.

A limitation of our approach is that the step sizes we choose depend on the target tolerance ε, which we therefore assume is pre-specified. Thus, if one desires an algorithm that can be stopped "any time," then another approach is needed.

The iteration complexities achieved in Table 2.1 are optimal. However, there are other algorithms that achieve better sample complexity, such as Projected (Stochastic) Gradient Descent (PGD). Compared to PGD, the Frank-Wolfe algorithm has the same iteration complexity but incurs an additional burden in terms of sample size. In [56], which is an extension of [49], it is shown that PGD has iteration complexity $\mathcal{O}(\varepsilon^{-1})$ assuming strong convexity and $\mathcal{O}(\varepsilon^{-2})$ with regular convexity. These complexities are optimal for first-order oracles [1]. In addition, these results are similar to those for Stochastic Gradient Descent (SGD) on unconstrained problems, as discussed in [32], for example, with the analysis for PGD exploiting the fact that the projection step is a contraction. These analyses (of stochastic PGD and SGD) rely on assumptions of unbiased gradients and bounded gradient variance. Under reasonable conditions, these methods do not need to assume sufficiently large sample sizes in each iteration to estimate gradients; a single sample at each iteration is sufficient, although the number of iterations may depend on the sample size that is used. This is a significant advantage relative to our approach, because we need to control the gradient estimation accuracy through sample-size selection. This advantage comes at the cost of projection; if projection is expensive then one might prefer to use Frank-Wolfe algorithms as explored here.

This chapter is organized as follows. Section 2.2 introduces the necessary assumptions and presents the standard and away-step Frank–Wolfe algorithms, together with the intuition behind them. Section 2.3 develops several preparatory results under deterministic dynamics with sufficiently accurate gradient approximations. Section 2.4 establishes key bounds on the sample size required to obtain accurate gradient estimates under two noise assumptions: bounded variance and sub-Gaussian noise. Finally, Section 2.5 introduces the Lyapunov functions for both algorithms and derives their iteration and sample complexity results by building

on the results established in the preceding sections.

2.2 The Standard and Away-Step Frank-Wolfe Algorithms

In this section, we present the main algorithms and provide a high-level overview of the complexity analysis. Throughout, we require Assumption 2.2.1 ($L-$smoothness) and, except when analyzing the case with biased gradient estimators, we require Assumption 2.2.2 (unbiased gradients). We also require Assumption 2.2.3 (convexity) or Assumption 2.2.4 (strong convexity), depending on the context.

Assumption 2.2.1. *The gradient of the objective function f is $L-$Lipschitz continuous, i.e.*

$$||\nabla f(x) - \nabla f(y)|| \leq L||x - y|| \quad \forall x, y \in \mathcal{X}.$$

Assumption 2.2.2. *We have access to an oracle that, at any feasible x, provides independent and identically distributed (i.i.d) vectors $G_1(x), G_2(x), \ldots$ such that $\mathbb{E}[G_1(x)] = \nabla f(x)$.*

Assumption 2.2.3. *The objective function f is convex,*

$$f(y) \geq f(x) + \nabla f(x)^T(y - x) \quad \forall x, y \in \mathcal{X}.$$

Assumption 2.2.4. *The objective function f is $\mu-$strongly convex, i.e., for some $\mu > 0$,*

$$f(y) \geq f(x) + \nabla f(x)^T(y - x) + \frac{\mu}{2}||x - y||^2 \quad \forall x, y \in \mathcal{X}.$$

We explore the standard and away-step Frank-Wolfe algorithms. In iteration k, these algorithms solve a linear program over the feasible region $\mathcal{X}$, where the

objective function coefficients are given by a gradient estimate g_k. This gradient estimate is the average of multiple gradient realizations. In each iteration of the away-step Frank-Wolfe algorithm, we maintain a representation of the current solution as a convex combination of the set $\mathcal{V}$ of vertices of the feasible region, i.e., in iteration k, the solution x_k at that iteration is represented as

$$x_k = \sum_{u \in \mathcal{V}} \alpha_u^{(k)} u,$$

where $\alpha_u^{(k)}$ are non-negative and sum to 1. This representation is not unique, but it plays a central role in the away-step Frank-Wolfe algorithm's definition and analysis. We define the **active vertex set** at iteration k, U_k, to be the set of all vertices with positive weights, i.e. $U_k = \{v \in \mathcal{V} | \alpha_v^{(k)} > 0\}$. Let $N_k = |U_k|$ be the number of active vertices in iteration k.

We start with a feasible solution, and in each iteration, k, we determine a search direction and then take a step of size γ_k in the search direction. The step size γ_k is limited to maintain feasibility. In the standard Frank-Wolfe algorithm, the maximum step size, γ_k^{max}, is 1, representing a step that exactly reaches a vertex of the feasible region. In the away-step Frank-Wolfe algorithm, the maximum step size is determined by the current solution's representation, as discussed below. The step size that is taken, if not constrained by the maximal step size, is given by minimizing an upper bound on the objective function. Indeed, by convexity and $L-$Lipschitz continuity of the gradient, in the search direction d_k,

$$f(x_k + \gamma_k d_k) \leq f(x_k) + \gamma_k \nabla f(x_k)^T d_k + \gamma_k^2 \frac{L}{2} \|d_k\|^2.$$

Minimizing this quadratic in γ_k and accounting for the maximal step size, we get the step size

$$\gamma_k = \min\left\{\gamma_k^{max}, \frac{-g_k^T d_k}{L\|d_k\|^2}\right\}.$$

In the standard Frank-Wolfe algorithm, Algorithm 1, search always proceeds in the direction of a vertex. The algorithm may alternately search in the direction of multiple vertices when approaching an optimal point near a face of the polytope. This results in zig-zagging behavior, which may be more expensive than moving directly towards the face. This observation motivates Algorithm 2, the away-step Frank-Wolfe algorithm, which has the additional ability to move away from a "bad" vertex, i.e., the vertex that produces the worst estimated function improvement.

Algorithm 1 Frank-Wolfe

Initialize a feasible solution $x_0 \in \mathcal{X}$
for $k = 0, 1, 2, ...$ **do**
  Compute an estimate, g_k, of the gradient $\nabla f(x_k)$
  Compute $s_k = \arg\min_{s\in\mathcal{X}} g_k^T s$ by solving a linear program to obtain a vertex solution
  $d_k = s_k - x_k$
  $\gamma_k = \min\{1, \frac{\varepsilon}{2LD^2}\}$
  $x_{k+1} = x_k + \gamma_k d_k$
end for

In the k^{th} iteration of the away-step Frank-Wolfe algorithm with current iterate x_k, we first obtain a gradient estimate, g_k, at x_k and solve the linear program (LP) to get the usual search direction. We then find another direction, the "away direction," through the current iterate, *away* from an active vertex. In doing so, we select the active vertex v_k that maximizes the dot product of $x_k - v_k$ with the negative gradient estimate. Then, we choose the better of the usual search direction and the away direction. To find the away-direction vertex, we iterate over all current active vertices (those in the set U_k). As long as this set is modest in size, this step requires much less computation than solving the LP for the

standard Frank-Wolfe direction. We initialize the active set, U_0, to a singleton set consisting of a vertex solution of the LP $\min_{x\in\mathcal{X}} \mathbb{1}^T x$.

Algorithm 2 Away-Step Frank-Wolfe

Initialize a feasible solution $x_0 \in \mathcal{X}$ and initialize the active set U_0 and coefficients appropriately
for $k = 0, 1, 2, ...$ **do**
  Compute an estimate, g_k, of the gradient $\nabla f(x_k)$
  Compute $s_k = \arg\min_{s\in\mathcal{X}} g_k^T s$ by solving an LP to obtain a vertex solution
  Compute $v_k = \arg\max_{u\in U_k} g_k^T u$
  $d_k^{FW} = s_k - x_k$ (FW direction)
  $d_k^A = x_k - v_k$ (Away direction)
  if $-g_k^T d_k^{FW} \geq -g_k^T d_k^A$ **then**
    $d_k = d_k^{FW}$ and $\gamma_k^{max} = 1$ (normal FW step)
  else
    $d_k = d_k^A$ and set γ_k^{max} as described in the text (away step)
  end if
  $\gamma_k = \min\left\{\gamma_k^{max}, \frac{-g_k^T d_k}{L\|d_k\|^2}\right\}$
  $x_{k+1} = x_k + \gamma_k d_k$
  Update the vertex representation of x_{k+1} and active set as described in the text
end for

The update of the vertex representation and active set depends on whether a normal Frank-Wolfe step or an away step is taken, and also on whether we take the maximal step [54]. For a normal Frank-Wolfe step, the direction vector is $d_k = s_k - x_k$ for some $s_k \in \mathcal{V}$, and thus the maximal step size is $\gamma_k^{max} = 1$. For an away step, the maximum step size is restricted by the representation of the current iterate. An away step gives $x_{k+1} = x_k + \gamma_k(x_k - v_k)$. Thus, for the vertex v_k we have the update $\alpha_{v_k}^{(k+1)} = (1+\gamma_k)\alpha_{v_k}^{(k)} - \gamma_k$. This weight must remain non-negative, so

$$0 \leq \alpha_{v_k}^{(k+1)} = (1 + \gamma_k^{max})\alpha_{v_k}^{(k)} - \gamma_k^{max},$$

implying that

$$\gamma_k^{max} \leq \frac{\alpha_{v_k}^{(k)}}{1 - \alpha_{v_k}^{(k)}}.$$

We also limit the maximum step size of away step to be at most 1, i.e. $\gamma_k^{max} \leq 1$.

Summarizing the above discussion, the updates to the active set and weights are as follows.

1. Normal FW Step:
   - $\gamma_k = \gamma_k^{max} = 1$: Set $U_{k+1} = \{s_k\}$
   - $\gamma_k < \gamma_k^{max} = 1$: Set $U_{k+1} = U_k \cup \{s_k\}$
   - Update the weights $\alpha_{s_k}^{(k+1)} = (1 - \gamma_k)\alpha_{s_k}^{(k)} + \gamma_k$
   - Update the weights $\alpha_v^{(k+1)} = (1 - \gamma_k)\alpha_v^{(k)}$ for all $v \in U_k \setminus \{s_k\}$
2. Away Step:
   - $\gamma_k = \gamma_k^{max}$: Set $U_{k+1} = U_k \setminus \{v_k\}$
   - $\gamma_k < \gamma_k^{max}$: Set $U_{k+1} = U_k$
   - Update the weights $\alpha_{v_k}^{(k+1)} = (1 + \gamma_k)\alpha_{v_k}^{(k)} - \gamma_k$
   - Update the weight $\alpha_v^{(k+1)} = (1 + \gamma_k)\alpha_v^{(k)}$ for all $v \in U_k \setminus \{v_k\}$.

2.3 Preparatory Results for the Convergence Analysis

We assume deterministic dynamics in this section, so we refer to a gradient *approximation* rather than a gradient *estimate*. We suppose, throughout this section, that $k < T_\varepsilon$, so we are yet to reach an ε-optimal point.

The proofs for the standard Frank–Wolfe results in Lemma 2.3.1 follow the classical analysis framework developed in [47] and [14], which are standard references for Frank–Wolfe convergence analysis. In contrast, the proofs for the Away-step Frank–Wolfe results in Lemmas 2.3.2 and 2.3.3 are based on the analytical framework introduced in [7]. Finally, the result in Lemma 2.3.6 combines the novel good

gradient estimate condition introduced in Definition 2.3.2 with the earlier results, particularly Lemmas 2.3.2 and 2.3.3, to establish the main convergence result for the stochastic Away-step Frank–Wolfe algorithm stated in Lemma 2.3.7

2.3.1 Results for the Standard Frank-Wolfe Algorithm

Definition 2.3.1. *Let D be the diameter of the feasible region $\mathcal{X}$. We say that the vector g_k is a* good gradient approximation *at iteration k if*

$$\|g_k - \nabla f(x_k)\| \leq \frac{\varepsilon}{4D}. \tag{2.2}$$

Lemma 2.3.1. *Under Assumption 2.2.3 (convexity), if we have a good gradient approximation at iteration k, then*

$$f(x_{k+1}) - f(x_k) \leq -\beta^{FW}(\varepsilon)\varepsilon,$$

where $\beta^{FW}(\varepsilon) = \min\{\frac{\varepsilon}{8LD^2}, \frac{1}{4}\}$.

Proof. By convexity and Definition 2.3.1,

$$\begin{aligned} f(y) &\geq f(x_k) + \nabla f(x_k)^T(y - x_k) \\ &\geq f(x_k) + g_k^T(y - x_k) - \frac{\varepsilon}{4D}||y - x_k|| \\ &\geq f(x_k) + g_k^T(y - x_k) - \frac{\varepsilon}{4}. \end{aligned}$$

Minimizing both sides, we have

$$f^* \geq f(x_k) + g_k^T(s_k - x_k) - \frac{\varepsilon}{4}.$$

Then, by convexity and the Lipschitz property,

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) + \gamma_k \nabla f(x_k)^T(s_k - x_k) + \frac{L\gamma_k^2}{2}||s_k - x_k||^2 \\ &\leq f(x_k) + \gamma_k g_k^T(s_k - x_k) + \gamma_k \frac{\varepsilon||s_k - x_k||}{4D} + \frac{L\gamma_k^2}{2}||s_k - x_k||^2 \\ &\leq f(x_k) + \gamma_k(f^* - f(x_k) + \frac{\varepsilon}{4}) + \gamma_k \frac{\varepsilon}{4} + \frac{L\gamma_k^2}{2}D^2. \end{aligned}$$

Hence, we have the improvement

$$f(x_{k+1}) - f^* \leq (1 - \gamma_k)(f(x_k) - f^*) + \frac{\gamma_k \varepsilon}{2} + \frac{L\gamma_k^2 D^2}{2}.$$

The step size policy $\gamma_k = \min\{\varepsilon/(2LD^2), 1\}$ corresponds to moving either to a point between the current solution and the vertex s_k or moving all the way to the vertex s_k. *Case I*: Suppose $\gamma_k = \varepsilon/(2LD^2)$. Then, using the fact that the algorithm is not yet terminated, i.e. $f(x_k) - f^* > \varepsilon$,

$$\begin{aligned} f(x_{k+1}) - f^* &\leq (1 - \gamma_k)(f(x_k) - f^*) + \frac{\varepsilon^2}{4LD^2} + \frac{\varepsilon^2}{8LD^2} \\ &= (1 - \gamma_k)(f(x_k) - f^*) + \frac{3\varepsilon^2}{8LD^2} \\ &\leq (1 - \gamma_k)(f(x_k) - f^*) + \frac{3\varepsilon}{8LD^2}(f(x_k) - f^*) \\ &= (1 - \frac{\varepsilon}{2LD^2} + \frac{3\varepsilon}{8LD^2})(f(x_k) - f^*) \\ &= (1 - \frac{\varepsilon}{8LD^2})(f(x_k) - f^*), \end{aligned}$$

or

$$f(x_{k+1}) - f(x_k) \leq -\frac{\varepsilon^2}{8LD^2}.$$

Case II: If $\gamma_k = 1$, then $2LD^2 \leq \varepsilon$. Therefore,

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) + \gamma_k(f^* - f(x_k)) + \frac{\gamma_k \varepsilon}{2} + \frac{L\gamma_k^2 D^2}{2} \\ &\leq f(x_k) - \gamma_k \varepsilon + \frac{\gamma_k \varepsilon}{2} + \frac{L\gamma_k^2 D^2}{2} \\ &\leq f(x_k) - \gamma_k \varepsilon + \frac{\gamma_k \varepsilon}{2} + \frac{\gamma_k^2 \varepsilon}{4} \\ &= f(x_k) - \frac{\varepsilon}{4}. \end{aligned}$$

Combining these two cases gives the result. □

2.3.2 Results for the Away-step Frank-Wolfe Algorithm

Definition 2.3.2. *Fix some $\epsilon_g \in (0, (4D)^{-1})$, where D is the diameter of the feasible region $\mathcal{X}$. We say that the vector g_k is a* good gradient approximation *at iteration k (with error at most ε_g) if*

$$\|g_k - \nabla f(x_k)\| \leq \varepsilon_g(-g_k^T(s_k - v_k)),$$

where s_k and v_k are, respectively, the vertex obtained by solving the LP and the vertex obtained by enumerating over active vertices in Algorithm 2 at iteration k.

From the definition of the direction d_k in Algorithm 2, we have the bound

$$2g_k^T d_k \leq g_k^T(s_k - x_k) + g_k^T(x_k - v_k) = g_k^T(s_k - v_k) \leq 0.$$

A good gradient approximation then satisfies

$$\|g_k - \nabla f(x_k)\| \leq \varepsilon_g(-g_k^T(s_k - v_k)) \leq 2\varepsilon_g(-g_k^T d_k).$$

Under the good gradient approximation assumption that $\varepsilon_g \leq (4D)^{-1}$, and using the fact that $\|d_k\|$ is bounded by the diameter D, the Cauchy-Schwarz inequality gives

$$|(g_k - \nabla f(x_k))^T d_k| \leq 2\varepsilon_g(-g_k^T d_k)D,$$

implying that

$$\nabla f(x_k)^T d_k \leq g_k^T d_k(1 - 2\varepsilon_g D). \tag{2.3}$$

Now, $g_k^T d_k \leq 0$, so (2.3) with $\varepsilon_g \leq (4D)^{-1}$ implies that $\nabla f(x_k)^T d_k \leq 0$. Let x^* denote the optimal solution, with objective function value $f^* = f(x^*)$. Then, by the definition of d_k, the Cauchy-Schwarz Inequality, and convexity of f,

$$\begin{aligned} g_k^T d_k &\leq g_k^T(x^* - x_k) \\ &= \nabla f(x_k)^T(x^* - x_k) + (g_k - \nabla f(x_k))^T(x^* - x_k) \\ &\leq \nabla f(x_k)^T(x^* - x_k) + 2\varepsilon_g(-g_k^T d_k)\|x^* - x_k\| \\ &\leq f^* - f(x_k) + 2\varepsilon_g D(-g_k^T d_k), \end{aligned}$$

which gives

$$g_k^T d_k \leq \frac{f^* - f(x_k)}{1 + 2\varepsilon_g D}. \tag{2.4}$$

From the step size policy,

$$\gamma_k = \min\left\{\gamma_k^{max}, -\frac{g_k^T d_k}{L\|d_k\|^2}\right\},$$

we see that $\gamma_k L\|d_k\|^2 \leq -g_k^T d_k$. We are now ready to state and prove two results on the decrease in function value, corresponding to the two cases in the step size policy.

Lemma 2.3.2. *Suppose Assumption 2.2.3 (convexity), that we have a good gradient approximation at iteration k, and that the step size is at its maximum, i.e. $\gamma_k = \gamma_k^{max}$. Then,*

$$f(x_{k+1}) - f^* \leq \left(1 - \gamma_k \frac{1/2 - 2\varepsilon_g D}{1 + 2\varepsilon_g D}\right)(f(x_k) - f^*).$$

Proof. By $L-$Lipschitz continuity and (2.3),

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) + \gamma_k \nabla f(x_k)^T d_k + \frac{\gamma_k^2}{2} L\|d_k\|^2 \\ &\leq f(x_k) + \gamma_k \nabla f(x_k)^T d_k - \frac{g_k^T d_k}{2}\gamma_k \text{ (by definition of } \gamma_k) \\ &\leq f(x_k) + \gamma_k g_k^T d_k \,(1 - 2\varepsilon_g D - 1/2) \text{ (from (2.3))} \\ &= f(x_k) + \gamma_k g_k^T d_k (1/2 - 2\varepsilon_g D). \end{aligned}$$

Using (2.4), we have that

$$f(x_{k+1}) \leq f(x_k) + \gamma_k g_k^T d_k (1/2 - 2\varepsilon_g D) \leq f(x_k) + \gamma_k (f^* - f(x_k)) \frac{1/2 - 2\varepsilon_g D}{1 + 2\varepsilon_g D}.$$

Subtracting f^* from both sides, we obtain

$$f(x_{k+1}) - f^* \leq \left(1 - \gamma_k \frac{1/2 - 2\varepsilon_g D}{1 + 2\varepsilon_g D}\right)(f(x_k) - f^*).$$

□

Lemma 2.3.3. *Suppose Assumption 2.2.3 (convexity), that we have a good gradient approximation at iteration k, and that the step size is $\gamma_k = -g_k^T d_k/(L\|d_k\|^2)$. Then*

$$f(x_{k+1}) \leq f(x_k) - \frac{(g_k^T(s_k - v_k))^2}{4L\|d_k\|^2}\left(\frac{1}{2} - 2\varepsilon_g D\right). \tag{2.5}$$

Proof. Using (2.3), the expression for γ_k, convexity and the $L-$smoothness property,

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) + \gamma_k \nabla f(x_k)^T d_k + \frac{\gamma_k^2}{2} L\|d_k\|^2 \\ &\leq f(x_k) + \gamma_k g_k^T d_k(1 - 2\varepsilon_g D) + \frac{\gamma_k^2}{2} L\|d_k\|^2 \text{ (from (2.3))} \\ &= f(x_k) - \frac{(g_k^T d_k)^2}{L\|d_k\|^2}(1 - 2\varepsilon_g D) + \frac{(g_k^T d_k)^2}{2L\|d_k\|^2} \text{ (by } \gamma_k \text{ substitution)} \\ &= f(x_k) - \frac{(g_k^T d_k)^2}{L\|d_k\|^2}\left(1 - 2\varepsilon_g D - \frac{1}{2}\right). \end{aligned}$$

Using the fact that $g_k^T d_k \leq \frac{1}{2} g_k^T(s_k - v_k) \leq 0$, we then have

$$f(x_{k+1}) \leq f(x_k) - \frac{(g_k^T(s_k - v_k))^2}{4L\|d_k\|^2}\left(\frac{1}{2} - 2\varepsilon_g D\right).$$

□

Next, we apply some definitions and results for polytopes from [7] to the feasible region $\mathcal{X}$.

Definition 2.3.3. *Let $I(x)$ be the index set of active constraints of a vector x in*

the polytope $\mathcal{X} = \{x : Ax \leq b\}$*, i.e.*

$$I(x) = \{i : A_i x = b_i\},$$

where A_i *is the* i*th row of the matrix* A*.*

Definition 2.3.4. *Similarly, given a set* $U \subseteq \mathcal{V}$ *of vertices of the polytope* $\mathcal{X}$*, the set of active constraints for all points in* U *is*

$$I(U) = \{i : A_i v = b_i \ \forall v \in U\} = \bigcap_{v \in U} I(v).$$

Lemma 2.3.4. *Let* $x \in \mathcal{X}$ *and* $U \subseteq \mathcal{V}$ *satisfy* $x = \sum_{v \in U} \alpha_v v$*, where all vertices in* U *have a positive coefficient* α_v*, and* $\sum_{v \in U} \alpha_v = 1$*. Then* $I(x) = I(U)$*.*

Lemma 2.3.5, below, is from [7]. The result provides a replacement, in our argument, for a polytope's pyramidal width constant, which is unknown and difficult to compute, even in the case of the unit simplex. For any matrix B and any index set I, define B_I to be the submatrix of B consisting of the rows of B whose indices appear in I.

Lemma 2.3.5. *Let* $U \subseteq \mathcal{V}$ *and* $w \in \mathbb{R}^n$ *be given. If there exists* $z \in \mathbb{R}^n$ *such that* $A_{I(U)} z \leq 0$ *and* $w^T z > 0$*, then*

$$\max_{p \in \mathcal{V}, u \in U} w^T (p - u) \geq \frac{\Omega_{\mathcal{X}}}{|U|} \frac{w^T z}{\|z\|},$$

where $\Omega_{\mathcal{X}} = \zeta/\varphi$ *for*

$$\begin{aligned} \zeta &= \min_{v\in\mathcal{V}, i: b_i > A_i v} (b_i - A_i v) \text{ and} \\ \varphi &= \max_{\mathcal{I}\setminus I(\mathcal{V})} \|A_i\|. \end{aligned}$$

Here, $\mathcal{I}$ *is the set of indices of all constraints.*

Lemma 2.3.3 gives a bound on expected function improvement, but the bound relies on the alignment between the estimated gradient and the vertices identified in the away-step method. We are now in a position to give a different bound that is central in deriving a rate of convergence in the next section, by relating the bound of Lemma 2.3.3 to the pyramidal width of the polytope using Lemma 2.3.5. Recall that N is the number of vertices of the polytope $\mathcal{X}$, L is the Lipschitz constant of the gradient and μ is the strong convexity parameter.

Lemma 2.3.6. *Suppose Assumptions 2.2.4 (strong convexity) and 2.2.1 (*$L-$*smoothness), that we have a good gradient approximation at iteration* k*, and that the step size is* $\gamma_k = -g_k^T d_k/(L\|d_k\|^2)$*. Then*

$$f(x_{k+1}) - f^* \leq \left(1 - \left(\frac{\Omega_{\mathcal{X}}}{N}\right)^2 \frac{\mu(1/2 - 2\varepsilon_g D)}{8LD^2(2\varepsilon_g D + 1)^2}\right)(f(x_k) - f^*).$$

Proof. By convexity,

$$f^* \geq f(x_k) + \nabla f(x_k)^T (x^* - x_k).$$

The Cauchy-Schwarz Inequality gives

$$(g_k - \nabla f(x_k))^T(x^* - x_k) \leq \|g_k - \nabla f(x_k)\| \ \|x^* - x_k\| \leq 2\varepsilon_g(-g_k^T d_k)\|x^* - x_k\|.$$

Therefore,

$$f^* \geq f(x_k) + \nabla f(x_k)^T(x^* - x_k) \geq f(x_k) + g_k^T(x^* - x_k) + 2\varepsilon_g(g_k^T d_k)\|x^* - x_k\|.$$

Rearranging gives

$$-g_k^T((x^* - x_k) + 2\varepsilon_g d_k\|x^* - x_k\|) \geq f(x_k) - f^* > 0.$$

Let $z = (x^* - x_k) + 2\varepsilon_g d_k\|x^* - x_k\|$ and $w = -g_k$. We will prove that z, w satisfy the conditions of Lemma 2.3.5. To that end, x_k is the solution at iteration k, and let U be the set of active vertices of x_k, i.e. $U = \{u \in \mathcal{V} : \alpha_u^{(k)} > 0\}$. By the definition of the index set and Lemma 2.3.4, we have that

$$A_{I(U)}x_k = A_{I(x_k)}x_k = b_{I(x_k)} = b_{I(U)}.$$

Moreover, by the definition of the feasible set, $A_{I(U)}x^* \leq b_{I(U)}$ which gives $A_{I(U)}(x^* - x_k) \leq b_{I(U)} - b_{I(U)} = 0$. Next, consider $A_{I(U)}d_k$.

Case I: $d_k = s_k - x_k$. We know that $A_{I(U)}x_k = A_{I(x_k)}x_k = b_{I(x_k)} = b_{I(U)}$ and $A_{I(U)}s_k \leq b_{I(U)}$ by feasibility. Then, $A_{I(U)}(s_k - x_k) \leq 0$.

Case II: $d_k = x_k - v_k$. Because $v_k \in U$, then $A_{I(U)}v_k = b_{I(U)}$, giving $A_{I(U)}(x_k - v_k) = b_{I(U)} - b_{I(U)} = 0$.

In both cases, we have $A_{I(U)}d_k \leq 0$, and so $A_{I(U)}z \leq 0$.

Now, by Lemma 2.3.5 using $w = -g_k$ and $z = 2\varepsilon_g d_k\|x^* - x_k\| + (x^* - x_k)$,

$$\max_{p\in\mathcal{V},u\in U} -g_k^T(p-u) \geq \frac{\Omega_{\mathcal{X}}}{|U|}\frac{-g_k^T z}{\|z\|} \geq \frac{\Omega_{\mathcal{X}}}{|U|}\frac{f(x_k) - f^*}{\|z\|}.$$

By the definitions of v_k and s_k,

$$g_k^T(v_k - s_k) = \max_{p\in\mathcal{V},u\in U} -g_k^T(p-u).$$

Hence, using the fact that $\|u+v\|^2 \leq (\|u\|+\|v\|)^2$ for any vectors u, v, and recalling that N is the number of vertices in $\mathcal{V}$, we have

$$\begin{aligned}
(g_k^T(v_k - s_k))^2 &\geq \left(\frac{\Omega_{\mathcal{X}}}{N}\right)^2 \frac{(f(x_k) - f^*)^2}{\|2\varepsilon_g d_k\|x^* - x_k\| + (x^* - x_k)\|^2} \\
&\geq \left(\frac{\Omega_{\mathcal{X}}}{N}\right)^2 \frac{(f(x_k) - f^*)^2}{(\|2\varepsilon_g d_k\|\,\|x^* - x_k\| + \|x^* - x_k\|)^2} \\
&\geq \left(\frac{\Omega_{\mathcal{X}}}{N}\right)^2 \frac{(f(x_k) - f^*)^2}{(2\varepsilon_g\|d_k\| + 1)^2\|x^* - x_k\|^2} \\
&\geq \left(\frac{\Omega_{\mathcal{X}}}{N}\right)^2 \frac{(f(x_k) - f^*)^2}{(2\varepsilon_g D + 1)^2\|x^* - x_k\|^2}.
\end{aligned}$$

By strong convexity with parameter μ and the fact that x^* is optimal,

$$f(x_k) \geq f^* + \nabla f(x^*)^T(x_k - x^*) + \frac{\mu}{2}\|x^* - x_k\|^2 \geq f^* + \frac{\mu}{2}\|x^* - x_k\|^2,$$

giving

$$\frac{f(x_k) - f^*}{\|x^* - x_k\|^2} \geq \frac{\mu}{2}.$$

Combining this with the earlier inequality we obtain

$$(g_k^T(v_k - s_k))^2 \geq \left(\frac{\Omega_{\mathcal{X}}}{N}\right)^2 \frac{\mu(f(x_k) - f^*)}{2(2\varepsilon_g D + 1)^2}. \tag{2.6}$$

From (2.5), we have

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) - \frac{(g_k^T(s_k - v_k))^2}{4L\|d_k\|^2}\left(\frac{1}{2} - 2\varepsilon_g D\right) \\ &\leq f(x_k) - \left(\frac{\Omega_{\mathcal{X}}}{N}\right)^2 \frac{\mu(f(x_k) - f^*)}{8LD^2}\frac{1/2 - 2\varepsilon_g D}{(2\varepsilon_g D + 1)^2}. \end{aligned}$$

Subtracting f^* from both sides, we get the required result that

$$f(x_{k+1}) - f^* \leq \left(1 - \left(\frac{\Omega_{\mathcal{X}}}{N}\right)^2 \frac{\mu(1/2 - 2\varepsilon_g D)}{8LD^2(2\varepsilon_g D + 1)^2}\right)(f(x_k) - f^*).$$

□

We can summarize the results from this section on the away-step Frank Wolfe algorithm with the following result on the function decrease on any iteration. We say that there is a vertex drop in iteration k if, in iteration k, we take a maximal

away step, so that the vertex v_k is dropped from the active set.

Lemma 2.3.7. *Suppose Assumptions 2.2.4 (strong convexity) and 2.2.1 (L−smoothness), and that we have a good gradient approximation at iteration k. If there is no vertex drop, then*

$$f(x_{k+1}) - f^* \leq (1 - \beta^{aFW})(f(x_k) - f^*),$$

where β^{aFW} does not depend on ε and is given by

$$\beta^{aFW} = \min\left\{\frac{1/2 - 2\varepsilon_g D}{1 + 2\varepsilon_g D}, \left(\frac{\Omega_{\mathcal{X}}}{N}\right)^2 \frac{\mu(1/2 - 2\varepsilon_g D)}{8LD^2(2\varepsilon_g D + 1)^2}\right\}.$$

If there is a vertex drop, then $f(x_{k+1}) - f^* \leq f(x_k) - f^*$.

Proof. When there is no vertex drop then we have either $\gamma_k = 1$ for the case when we move to a vertex, in which case Lemma 2.3.2 gives the result, or $\gamma_k = -g_k^T d_k/(L\|d_k\|^2) < 1$, in which case Lemma 2.3.6 gives the result. If there is a vertex drop (on an away-step), then the step size achieves its maximum, so we apply Lemma 2.3.2, noting that $\gamma_k(1/2 - 2\varepsilon_g D)/(1 + 2\varepsilon_g D) \geq 0$, to conclude that $f(x_{k+1}) - f^* \leq f(x_k) - f^*$. □

2.4 Tail Bounds for Gradient Estimation Error

We establish results on sample-size complexity under two different assumptions, the first of which is more general than the second but leads to a weaker complexity result.

2.4.1 The Bounded Second Moment Case

We now specialize Assumption 2.2.2 (unbiased gradients).

Assumption 2.4.1. *For all feasible x, the i.i.d. random vectors $G_1(x), G_2(x), \ldots$ are unbiased and have bounded (in x) second moment, i.e., $\mathbb{E}G_1(x) = \nabla f(x)$, and there exists a constant $V_g < \infty$ such that, for all $x \in \mathcal{X}$,*

$$E[\|G_1(x) - \nabla f(x)\|^2] \leq V_g.$$

The gradient estimator at any point, x, is simply the sample average, $g(x) = n^{-1} \sum_{i=1}^{n} G_i(x)$. We adopt a constant sample size, n, at any iterate. Let $\|\cdot\|$ denote the 2-norm.

Lemma 2.4.1. *Under Assumption 2.4.1 (bounded variance), for any feasible x, $\mathbb{E}[\|g(x) - \nabla f(x)\|^2] \leq V_g/n$.*

Proof.

$$\begin{aligned}
\mathbb{E}[\|g(x) - \nabla f(x)\|^2] &= \mathbb{E}\left[\left\|\frac{\sum_{i=1}^{n} G_i(x)}{n} - \nabla f(x)\right\|^2\right] \\
&\leq \frac{1}{n^2}\mathbb{E}\left[\sum_{i=1}^{n} \|G_i(x) - \nabla f(x)\|^2\right] \\
&\leq \frac{1}{n^2} n V_g = \frac{V_g}{n}.
\end{aligned}$$

□

Lemma 2.4.2. *Under Assumption 2.4.1 (bounded variance), for any fixed feasible x and any $s > 0$,*

$$\mathbb{P}(\|g(x) - \nabla f(x)\| > s) \leq \frac{V_g}{n s^2}.$$

Proof. Let $\nabla f(x,i), g(x,i)$ and $G(x,i)$ denote the ith components of $\nabla f(x), g(x)$ and $G(x)$ respectively. By Chebyshev's inequality,

$$\begin{aligned}
\Pr(\|g(x) - \nabla f(x)\| > s) &\leq s^{-2}\mathbb{E}[\|g(x) - \nabla f(x)\|^2] \\
&= s^{-2}\sum_{i=1}^{d}\mathbb{E}[(g(x,i) - \nabla f(x,i))^2] \\
&= \frac{\sum_{i=1}^{d}\mathbb{E}[(G(x,i) - \nabla f(x,i))^2]}{ns^2} \\
&= \frac{\mathbb{E}[\|G(x) - \nabla f(x)\|^2]}{ns^2} \\
&\leq \frac{V_g}{ns^2}.
\end{aligned}$$

□

2.4.2 The Sub-Gaussian Case

The convergence rates we can establish under the bounded second moment assumption can be improved if we make the stronger assumption that the vector $G_1(x)$ that represents a single gradient realization at the feasible point x is sub-Gaussian with a parameter that is uniform in x, following [48, 84].

Definition 2.4.1. *A random vector $X \in \mathbb{R}^d$ is sub-Gaussian with parameter σ if*

$$\mathbb{E}[e^{v^T(X-\mathbb{E}[X])}] \leq e^{\frac{\|v\|^2\sigma^2}{2}} \quad \forall v \in \mathbb{R}^d.$$

Assumption 2.4.2. *For all feasible x, the i.i.d. random vectors $G_1(x), G_2(x), \ldots$ representing gradient realizations at the point x are unbiased and sub-Gaussian with parameter $\rho/\sqrt{d}$, where d is the dimension of x.*

Under Assumption 2.4.2 (sub-Gaussian), $G_1(x) - \nabla f(x)$ is *sub-Gaussian* with parameter ρ, uniformly in all feasible points x [48].

Lemma 2.4.3. *Let $g(x) = n^{-1} \sum_{i=1}^{n} G_i(x)$ be a sample average of i.i.d. gradient realizations at x. Under Assumption 2.4.2 (sub-Gaussian), there exists a constant $c > 0$ such that, for all feasible x, $Pr(||g(x) - \nabla f(x)|| \geq s) \leq 2de^{-ncs^2}$, where d is the dimension of x.*

Proof. Fix $s > 0$. Corollary 7 in [48] establishes that, for some positive constant υ,

$$Pr\left(\left\|\sum_{i=1}^{n}(G_i(x) - \nabla f(x))\right\| \leq \upsilon\sqrt{n\rho^2 \log\frac{2d}{\delta}}\right) \geq 1 - \delta$$

for any $\delta > 0$. Set $\delta = 2d\exp(-ns^2\upsilon^{-2}\rho^{-2})$ to obtain the desired result, where $c = \upsilon^{-2}\rho^{-2}$. □

From Lemma 2.4.3, we also have that

$$\mathbb{E}[\|g(x) - \nabla f(x)\|^2] = \int_0^\infty Pr(\|g(x) - \nabla f(x)\|^2 > s)ds \leq \int_0^\infty 2de^{-ncs}ds = \frac{2d}{nc}, \tag{2.7}$$

which exhibits the same dependence on sample size, n^{-1}, as in the bounded variance case.

2.5 Complexity with Unbiased Gradient Estimators

We now develop a complexity analysis through the stochastic process that results from using (random) gradient estimates in the Frank-Wolfe algorithms. When using gradient estimates, $((X_k, G_k, S_k, V_k, \mathcal{U}_k) : k \geq 0)$ is a stochastic process where, on iteration k, X_k is the current solution, G_k is the gradient estimate, S_k is

the vertex obtained by solving the LP, V_k is the candidate "bad" vertex used for away steps, and $\mathcal{U}_k$ is the set of active vertices, all at X_k. We denote realizations of these random quantities by $(x_k, g_k, s_k, v_k, U_k)$.

For $k \geq 0$, let $\mathcal{F}_k$ denote the $\sigma-$algebra generated from $\{(G_i, S_i, V_i, \mathcal{U}_i)\}_{i=0}^{k-1}$ and $\{X_i\}_{i=0}^{k}$. Our algorithms are assumed to run only up to the stopping time $T_\varepsilon = \inf\{k \geq 1 : f(X_k) - f(x^*) \leq \varepsilon\}$; for $k > T_\varepsilon$ we (somewhat arbitrarily) define $X_k = X_{T_\varepsilon}$ and similarly define the other random quantities for $k > T_\varepsilon$.

A good gradient estimate arises when the gradient estimate is sufficiently close to the true gradient, but the definition of "sufficient closeness" must be customized to the problem at hand [9, 67, 19]. In preparation for our definition, let $\varepsilon_g \in (0, (4D)^{-1})$ be a fixed constant, where D is the diameter of the feasible region $\mathcal{X}$.

Definition 2.5.1. *We say, at iteration k, that G_k is a good gradient estimate for the standard Frank-Wolfe algorithm with at least probability p_g if the event*

$$E_k^{(FW)} = \{\|G_k - \nabla f(X_k)\| \leq \varepsilon/4D\}$$

satisfies the condition $\mathbb{P}(E_k^{(FW)}|\mathcal{F}_k) \geq p_g$ on the event $\{k < T_\varepsilon\}$.

Definition 2.5.2. *We say, at iteration k, that G_k is a good gradient estimate for the away-step Frank-Wolfe algorithm with at least probability p_g if the event*

$$E_k^{(aFW)} = \{\|G_k - \nabla f(X_k)\| \leq \varepsilon_g G_k^T(V_k - S_k)\}$$

satisfies the condition $\mathbb{P}(E_k^{(aFW)}|\mathcal{F}_k) \geq p_g$ on the event $\{k < T_\varepsilon\}$.

Lemma 2.5.1. *Under Assumption 2.4.1 (bounded variance), the sample sizes required for each iteration to achieve a good gradient estimate for the standard and*

away-step Frank-Wolfe are $\mathcal{O}(\varepsilon^{-2}(1-p_g)^{-1})$ *and* $\mathcal{O}(\varepsilon^{-1}(1-p_g)^{-1})$ *respectively.*

Proof. For standard Frank-Wolfe, by Lemma 2.4.2 and Definition 2.5.1, take $s = \varepsilon/(4D)$, and then it suffices to choose the sample size n so that

$$n \geq \frac{16V_g D^2}{\varepsilon^2(1-p_g)}. \tag{2.8}$$

Next, consider the away-step Frank-Wolfe algorithm. In view of Definition 2.5.2, we want s in Lemma 2.4.2 to be a lower bound on $|\varepsilon_g G_k^T(V_k - S_k)|$.

We show that if the sample size is chosen such that the estimated gradient $g(x)$ satisfies

$$\|g(x) - \nabla f(x)\| \leq \min\left\{1, \frac{\varepsilon_g \Omega_{\mathcal{X}}}{2\sqrt{2}N}\right\}\sqrt{\mu\varepsilon}, \tag{2.9}$$

then the good gradient approximation condition in Definition 2.3.2 is satisfied. By Cauchy-Schwarz Inequality for any feasible point x,

$$\begin{aligned}\sqrt{\mu\varepsilon}\|x - x^*\| &\geq \|g(x) - \nabla f(x)\| \cdot \|x - x^*\| \\ &\geq (g(x) - \nabla f(x))^T(x^* - x).\end{aligned}$$

Hence, by strong convexity and inequality above,

$$\begin{aligned}
-g(x)^T(x^*-x) &= (\nabla f(x)-g(x))^T(x^*-x)-\nabla f(x)^T(x^*-x)\\
&\geq (\nabla f(x)-g(x))^T(x^*-x)+(f(x)-f^*)+\frac{\mu}{2}\|x^*-x\|^2\\
&\geq -\sqrt{\mu\varepsilon}\|x^*-x\|+(f(x)-f^*)+\frac{\mu}{2}\|x^*-x\|^2\\
&= f(x)-f^*+\frac{\mu}{2}\left(\|x^*-x\|-\sqrt{\frac{\varepsilon}{\mu}}\right)^2-\frac{\varepsilon}{2}\\
&> \frac{f(x)-f^*}{2}+\frac{\mu}{2}\left(\|x^*-x\|-\sqrt{\frac{\varepsilon}{\mu}}\right)^2\\
&\geq \frac{f(x)-f^*}{2}\\
&> 0.
\end{aligned}$$

Therefore $-g(x)^T(x^*-x) > (f(x)-f^*)/2 > 0$, and by definition of feasible set, index set, and Lemma 2.3.4 where U is a set of active vertices of x,

$$A_{I(U)}(x^*-x) = A_{I(U)}x^* - b_{I(U)} \leq b_{I(U)} - b_{I(U)} = 0.$$

By Lemma 2.3.5 we substitute $w=-g(x)$ and $z=x^*-x$. Then, by definition of bad vertex v, and standard Frank-Wolfe vertex s,

$$\begin{aligned}
g(x)^T(v-s) &= \max_{p\in\mathcal{V},u\in\mathcal{U}} -g(x)^T(p-u)\\
&\geq \frac{\Omega_{\mathcal{X}}}{N}\frac{-g(x)^T(x^*-x)}{\|x^*-x\|}\\
&\geq \frac{\Omega_{\mathcal{X}}}{N}\frac{f(x)-f^*}{2\|x^*-x\|}.
\end{aligned}$$

Hence, by strong convexity,

$$\begin{aligned}(g(x)^T(v-s))^2 &\geq \left(\frac{\Omega_{\mathcal{X}}}{N}\right)^2 \frac{(f(x)-f^*)^2}{4\|x^*-x\|^2} \\ &\geq \left(\frac{\Omega_{\mathcal{X}}}{N}\right)^2 \frac{\mu(f(x)-f^*)}{8} \\ &> \left(\frac{\Omega_{\mathcal{X}}}{N}\right)^2 \frac{\mu\varepsilon}{8}.\end{aligned}$$

This implies

$$g(x)^T(v-s) \geq \left(\frac{\Omega_{\mathcal{X}}}{N}\right) \frac{\sqrt{\mu\varepsilon}}{2\sqrt{2}}$$

With the estimated gradient from (2.9),

$$\|g(x)-\nabla f(x)\| \leq \varepsilon_g \left(\frac{\Omega_{\mathcal{X}}}{N}\right) \frac{\sqrt{\mu\varepsilon}}{2\sqrt{2}} \leq \varepsilon_g g(x)^T(v-s).$$

By Lemma 2.4.2, pick s so that

$$s = \min\left\{1, \frac{\varepsilon_g \Omega_{\mathcal{X}}}{2\sqrt{2}N}\right\} \sqrt{\mu\varepsilon}, \tag{2.10}$$

and then it suffices to choose the sample size n so that

$$n \geq \frac{V_g}{\mu\varepsilon \min\{1, \varepsilon_g^2 \Omega_{\mathcal{X}}^2 / 8N^2\}(1-p_g)} \tag{2.11}$$

□

The sample size for the away-step Frank–Wolfe algorithm contains several parameters that can be interpreted and simplified. Some of these parameters encode geometric information about the feasible region, which influences both the required

sample size and the iteration complexity. Fortunately, these geometric quantities admit intuitive interpretations and can be expressed in a simpler form.

From Definition 2.3.2, the parameter ε_g satisfies $\varepsilon_g \in (0, (4D)^{-1})$. For example, it can be chosen as $\varepsilon_g = 1/(8D)$, indicating that $\varepsilon_g = \mathcal{O}(D^{-1})$. Consequently, the quantity $\varepsilon_g \Omega_{\mathcal{X}}$ is of order $\mathcal{O}(\Omega_{\mathcal{X}}/D)$. By the definition of $\Omega_{\mathcal{X}}$ in Lemma 2.3.5, $\Omega_{\mathcal{X}}$ is the minimum distance from a vertex of the feasible region to any constraint that is not incident to that vertex. Since $\Omega_{\mathcal{X}}$ is measured relative to the diameter D, the ratio $\Omega_{\mathcal{X}}/D \in (0, 1]$ characterizes the geometry of $\mathcal{X}$ rather than its size: a small ratio indicates that $\mathcal{X}$ is thin and sharp, whereas a large ratio indicates that it is thick and well-conditioned. This ratio therefore serves as a scale-invariant geometric "condition" number of $\mathcal{X}$. We define

$$\chi = \frac{\Omega_{\mathcal{X}}}{D}. \tag{2.12}$$

For example, consider a square with side length 1. It has $\Omega_{\mathcal{X}} = 1$ and $D = \sqrt{2}$, giving $\chi = 1/\sqrt{2} \approx 0.707$. In contrast, a rectangle with width 1 and length 10 has $\Omega_{\mathcal{X}} = 1$ and $D = \sqrt{101}$, yielding $\chi = 1/\sqrt{101} \approx 0.0995$. Although both shapes have the same value of $\Omega_{\mathcal{X}}$, they have different values of χ. The rectangle has a much smaller value of χ, reflecting its thin and elongated geometry. Furthermore, if the rectangle is uniformly scaled to have width 2 and length 20, then $\chi = \frac{2}{2\sqrt{101}} = \frac{1}{\sqrt{101}} \approx 0.0995$, which is identical to that of the original rectangle. This example illustrates that χ is invariant under scaling and depends only on the geometry of the feasible region, not on its size.

Another parameter appearing in the sample size bound (2.11) is N. It is first introduced in Lemma 2.3.6 as the total number of vertices of $\mathcal{X}$, which serves

as an upper bound on the number of active vertices $|U|$ in Lemma 2.3.5. This quantity can be very large when $\mathcal{X}$ has many vertices, such as when a polytope closely approximates a Euclidean ball. However, by the result in Section 17 of [72], any point in $\mathcal{X}$ can be represented as a convex combination of at most $d+1$ vertices, where d is the dimension of $\mathcal{X}$. Consequently, the active set can always be maintained so that $|U| \leq d+1$. Therefore, the parameter N may also be replaced by the upper bound $d+1$, which depends only on the dimension and is invariant under scaling of the feasible region $\mathcal{X}$.

Consider the stochastic processes $\Phi^{(S)}, \Phi^{(A)}$ obtained from random variables generated from the standard Frank-Wolfe algorithm and the away-step Frank-Wolfe algorithm respectively, where

$$\Phi_k^{(S)} = f(X_k) - f^*, \tag{2.13}$$

and

$$\Phi_k^{(A)} = \nu(f(X_k) - f^*) + (1-\nu)N_k. \tag{2.14}$$

(Recall that N_k is the number of vertices in the active vertex set $\mathcal{U}_k$.) The stochastic processes $\Phi^{(S)}, \Phi^{(A)}$ are bounded because the polytope $\mathcal{X}$ is bounded, implying that f attains both its minimum and maximum on $\mathcal{X}$, and also because $\mathcal{X}$ has a finite number of vertices. Next, we now establish a key result that determines iterate complexity.

Theorem 2.5.1. *There exists $p_g \in (0,1)$, $\nu \in (0,1)$ and $\delta_S, \delta_A \in (0,1)$ (explicit expressions are given in the proof) such that, with the sample size given in* (2.8) *for standard Frank-Wolfe (under Assumptions 2.2.1 (L−smoothness) and 2.2.3 (convexity)), and with the sample size given in* (2.11) *for away-step Frank-Wolfe*

(under Assumptions 2.2.1 (L−smoothness) and 2.2.4 (strong convexity)), on the event $T_\varepsilon > k$,

$$\mathbb{E}\left[\Phi_{k+1}^{(q)} - \Phi_k^{(q)} \middle| \mathcal{F}_k\right] \leq -\delta_q.$$

for $q = S, A$.

Proof. Define Γ to be the maximum possible step size, which is bounded above by 1, as seen in Algorithms 1 and 2. Consider the term $f(x_{k+1}) - f(x_k)$. By smoothness,

$$\begin{aligned} f(x_{k+1}) - f(x_k) &\leq \gamma_k \nabla f(x_k)^T d_k + \gamma_k^2 \frac{L\|d_k\|^2}{2} \\ &= \gamma_k g_k^T d_k + \gamma_k (\nabla f(x_k) - g_k)^T d_k + \gamma_k^2 \frac{L\|d_k\|^2}{2} \\ &\leq \gamma_k g_k^T d_k + \gamma_k D \|\nabla f(x_k) - g_k\| + \gamma_k^2 \frac{LD^2}{2} \\ &\leq \Gamma D \|\nabla f(x_k) - g_k\| + \frac{L\Gamma^2 D^2}{2}. \end{aligned}$$

Let $\mathcal{G}$ be the event of good gradient approximation in iteration k, P_k be its conditional probability given $\mathcal{F}_k$ (which is bounded below by p_g by our sample-size assumption) and $\mathcal{G}'$ be its complement. Let $\mathbb{1}_A$ be the indicator function of the event A. By the Cauchy-Schwarz inequality and Lemma 2.4.1,

$$\begin{aligned} \mathbb{E}[(f(X_{k+1}) - f(X_k))\mathbb{1}_{\mathcal{G}'} | \mathcal{F}_k] &\leq \Gamma D \mathbb{E}[\|G_k - \nabla f(X_k)\| \mathbb{1}_{\mathcal{G}'} | \mathcal{F}_k] + \frac{L\Gamma^2 D^2}{2}(1 - P_k) \\ &\leq \Gamma D \sqrt{\mathbb{E}[\|G_k - \nabla f(X_k)\|^2 | \mathcal{F}_k]} \sqrt{1 - P_k} + \frac{L\Gamma^2 D^2}{2}(1 - P_k) \\ &\leq \Gamma D \sqrt{\frac{V_g}{n_k}} \sqrt{1 - P_k} + \frac{L\Gamma^2 D^2}{2}(1 - P_k), \end{aligned}$$

where n_k is the sample size in iteration k. Then,

$$\mathbb{E}[f(X_{k+1}) - f(X_k)|\mathcal{F}_k] \leq \mathbb{E}[(f(X_{k+1}) - f(X_k))\mathbb{1}_{\mathcal{G}}|\mathcal{F}_k] + \Gamma D\sqrt{\frac{V_g}{n_k}}\sqrt{1-P_k} + \frac{L\Gamma^2 D^2}{2}(1-P_k). \tag{2.15}$$

Part I: The standard Frank-Wolfe case, i.e., $q = S$. Continuing from (2.15), using Lemma 2.3.1 and the sample size given in (2.8) from Lemma 2.5.1, on the event $T_\varepsilon > k$,

$$\begin{aligned}\mathbb{E}[\Phi_{k+1}^{(S)} - \Phi_k^{(S)}|\mathcal{F}_k] &\leq -\beta^{FW}(\varepsilon)\varepsilon p_g + \Gamma D\sqrt{\frac{V_g}{n_k}}\sqrt{1-p_g} + \frac{L\Gamma^2 D^2}{2}(1-p_g) \\ &\leq -\beta^{FW}(\varepsilon)\varepsilon p_g + \frac{\Gamma\varepsilon}{4}(1-p_g) + \frac{L\Gamma^2 D^2}{2}(1-p_g).\end{aligned}$$

Pick $\delta_S = \beta^{FW}(\varepsilon)\varepsilon/2$ and

$$p_g \geq \frac{\Gamma\varepsilon/4 + L\Gamma^2 D^2/2 + \delta_S}{\Gamma\varepsilon/4 + L\Gamma^2 D^2/2 + \beta^{FW}(\varepsilon)\varepsilon}, \tag{2.16}$$

which then yields that, on the event $T_\varepsilon > k$,

$$\mathbb{E}\left[\Phi_{k+1}^{(S)} - \Phi_k^{(S)}\middle|\mathcal{F}_k\right] \leq -\delta_S.$$

Part II: The away-step Frank-Wolfe case, i.e., $q = A$. We condition on $\mathcal{F}_k$, so that $\{G_i, S_i, V_i\}_{i=0}^{k-1}$ and $\{X_i\}_{i=0}^{k}$ are known. Thus, the active set $\mathcal{U}_k$ and the number of active vertices N_k are also known, but G_k, S_k and V_k are not yet available. Suppose we are yet to stop so that $T_\varepsilon > k$ (which is also $\mathcal{F}_k$ measurable).

Consider the number of active vertices N_k. Then

$$N_{k+1} - N_k = \begin{cases} 1 & \text{if no vertex drop and a new active vertex is added,} \\ 1 - N_k & \text{if no vertex drop and next solution is a vertex, and} \\ -1 & \text{if vertex drop.} \end{cases} \tag{2.17}$$

In addition to $\mathcal{G}$ being the event of good gradient approximation at Iteration k, let $\mathscr{V}$ be the event of vertex drop at Iteration k. Then, from Lemma 2.3.7,

$$\begin{aligned} (f(X_{k+1}) - f(X_k))\mathbb{1}_{\mathcal{G}} &= (f(X_{k+1}) - f(X_k)\mathbb{1}_{\mathcal{G}}\mathbb{1}_{\mathscr{V}} + (f(X_{k+1}) - f(X_k))\mathbb{1}_{\mathcal{G}}\mathbb{1}_{\mathscr{V}'} \\ &\leq -\beta^{aFW}\varepsilon\mathbb{1}_{\mathcal{G}}\mathbb{1}_{\mathscr{V}'}. \end{aligned}$$

Thus, $\mathbb{E}[(f(X_{k+1}) - f(X_k))\mathbb{1}_{\mathcal{G}}|\mathcal{F}_k] \leq -\beta^{aFW}\varepsilon Pr(\mathcal{G} \cap \mathscr{V}'|\mathcal{F}_k)$. From (2.17),

$$\begin{aligned} \mathbb{E}[N_{k+1} - N_k|\mathcal{F}_k] &= \mathbb{E}[(N_{k+1} - N_k)\mathbb{1}_{\mathcal{G}'}|\mathcal{F}_k] + \mathbb{E}[(N_{k+1} - N_k)\mathbb{1}_{\mathcal{G}}|\mathcal{F}_k] \\ &\leq \mathbb{E}[\mathbb{1}_{\mathcal{G}'}] + \mathbb{E}[\mathbb{1}_{\mathcal{G}}(\mathbb{1}_{\mathscr{V}'} - \mathbb{1}_{\mathscr{V}})|\mathcal{F}_k] \\ &\leq 1 - p_g + \mathbb{E}[\mathbb{1}_{\mathcal{G}}(\mathbb{1}_{\mathscr{V}'} - \mathbb{1}_{\mathscr{V}})|\mathcal{F}_k]. \end{aligned}$$

For notational convenience, let

$$\Lambda = \Gamma D\sqrt{\frac{V_g}{n_k}}\sqrt{1 - p_g} + \frac{L\Gamma^2 D^2}{2}(1 - p_g).$$

From (2.15), and the definition of the Lyapunov function (2.14), on the event

$T_\varepsilon > k$,

$$
\begin{aligned}
&\mathbb{E}[\Phi_{k+1}^{(A)} - \Phi_k^{(A)} | \mathcal{F}_k] \\
&\quad = \mathbb{E}[(\Phi_{k+1}^{(A)} - \Phi_k^{(A)}) \mathbb{1}_{\mathcal{G}} | \mathcal{F}_k] + \mathbb{E}[(\Phi_{k+1}^{(A)} - \Phi_k^{(A)}) \mathbb{1}_{\mathcal{G}'} | \mathcal{F}_k] \\
&\quad \leq \mathbb{E}[-\nu \beta^{aFW} \varepsilon \mathbb{1}_{\mathcal{G}} \mathbb{1}_{\mathscr{V}'} | \mathcal{F}_k] + \nu\Lambda + (1-\nu)(1-p_g) + \mathbb{E}[(1-\nu)\mathbb{1}_{\mathcal{G}}(\mathbb{1}_{\mathscr{V}'} - \mathbb{1}_{\mathscr{V}}) | \mathcal{F}_k] \\
&\quad = \mathbb{E}[(-\nu\beta^{aFW}\varepsilon + 1 - \nu)\mathbb{1}_{\mathcal{G}} \mathbb{1}_{\mathscr{V}'} | \mathcal{F}_k] - (1-\nu)\mathbb{E}[\mathbb{1}_{\mathcal{G}} \mathbb{1}_{\mathscr{V}} | \mathcal{F}_k] + \nu\Lambda + (1-\nu)(1-p_g) \\
&\quad \leq (-\nu\beta^{aFW}\varepsilon + 1 - \nu) p_g \mathbb{P}(\mathscr{V}' | \mathcal{F}_k, \mathcal{G}) - (1-\nu) p_g \mathbb{P}(\mathscr{V} | \mathcal{F}_k, \mathcal{G}) + \nu\Lambda + (1-\nu)(1-p_g) \\
&\quad \leq \max\{-\nu\beta^{aFW}\varepsilon + 1 - \nu, -(1-\nu)\} p_g + \nu\Lambda + (1-\nu)(1-p_g).
\end{aligned}
$$

Choosing $\nu = 2/(2+\beta^{aFW}\varepsilon)$ equalizes the two terms in the maximum immediately above to $-\nu\beta^{aFW}\varepsilon/2$. Thus, from the sample size choice (2.11) in Lemma 2.5.1,

$$
\begin{aligned}
\mathbb{E}[\Phi_{k+1}^{(A)} - \Phi_k^{(A)} | \mathcal{F}_k] &= -\frac{\nu\beta^{aFW}\varepsilon p_g}{2} + \nu\Lambda + (1-\nu)(1-p_g) \qquad (2.18) \\
&\leq -\frac{\nu\beta^{aFW}\varepsilon p_g}{2} + \left(\nu\Gamma D\sqrt{\mu\varepsilon}\kappa + \nu\frac{L\Gamma^2 D^2}{2} + 1 - \nu\right)(1-p_g),
\end{aligned}
$$

where

$$
\kappa = \min\left\{1, \frac{\varepsilon_g \Omega_{\mathcal{X}}}{2\sqrt{2}N}\right\}. \qquad (2.19)
$$

Choose the probability

$$
p_g \geq \frac{\Gamma D\kappa\sqrt{\mu\varepsilon} + L\Gamma^2 D^2/2 + (3/4)\beta^{aFW}\varepsilon}{\Gamma D\kappa\sqrt{\mu\varepsilon} + L\Gamma^2 D^2/2 + \beta^{aFW}\varepsilon}, \qquad (2.20)
$$

to yield that, on the event $T_\varepsilon > k$,

$$
\mathbb{E}[\Phi_{k+1}^{(A)} - \Phi_k^{(A)} | \mathcal{F}_k] \leq -\frac{\beta^{aFW}\varepsilon/2}{2 + \beta^{aFW}\varepsilon},
$$

which completes the proof with $\delta_A = \frac{1}{2}\beta^{aFW}\varepsilon/(2+\beta^{aFW}\varepsilon)$. □

The deterministic Frank-Wolfe algorithm typically requires $\mathcal{O}(\varepsilon^{-1})$ iterations to reach a point satisfying $f(x) - f^* < \varepsilon$. However, in the presence of gradient approximation errors, the iteration complexities of both the standard and away-step Frank–Wolfe algorithms become slower. Theorem 2.5.2 builds on the results of Theorem 2.5.1, which guarantees an expected decrease in the Lyapunov function, to establish the iteration complexities of both algorithms. The proof is relatively standard, but included to ensure clarity.

Theorem 2.5.2. *Suppose the initial solution $X_0 = x_0$ is deterministic with $f(x_0) - f^* > \varepsilon$. Let $T_\varepsilon = \inf\{k : f(X_k) - f^* \leq \varepsilon\}$.*

1. *Using the standard Frank-Wolfe algorithm under Assumptions 2.2.1 ($L-$smoothness) and 2.2.3 (convexity), along with the sample size given in* (2.8)*, $\mathbb{E}T_\varepsilon = \mathcal{O}(\varepsilon^{-2}(f(x_0) - f^*))$.*
2. *Using the away-step Frank-Wolfe algorithm under Assumptions 2.2.1 ($L-$smoothness) and 2.2.4 (strong convexity), along with the sample size given in* (2.11)*, $\mathbb{E}T_\varepsilon = \mathcal{O}(\varepsilon^{-1}(f(x_0) - f^*))$.*

Proof. Consider a stochastic process $\Phi = (\Phi_k : k \geq 0)$, adapted to $(\mathcal{F}_k : k \geq 0)$ that satisfies the expected ratio decrease as in Theorem 2.5.1 and define a stochastic process $M = (M_k : k \geq 0)$, where $M_k = \delta(k \wedge T_\varepsilon) + \Phi_{k \wedge T_\varepsilon}$ and $a \wedge b = \min\{a, b\}$. Then, using Theorem 2.5.1, on the event $k < T_\varepsilon$, with $\delta = \delta_S$ or δ_A for the standard

or away-step stochastic process respectively,

$$\begin{aligned}\mathbb{E}[M_{k+1}|\mathcal{F}_k] &= \mathbb{E}[\delta((k+1)\wedge T_\varepsilon) + \Phi_{(k+1)\wedge T_\varepsilon}|\mathcal{F}_k] \\ &= \delta(k+1) + \mathbb{E}[\Phi_{k+1}|\mathcal{F}_k] \\ &\leq \delta k + \Phi_k = M_k.\end{aligned}$$

On the event $T_\varepsilon \leq k$, $M_{k+1} = M_k$, so $(M_k : k \geq 0)$ is a non-negative supermartingale. By the optional stopping theorem, and since $X_0 = x_0$ is deterministic with $f(x_0) > f^* + \varepsilon$,

$$\mathbb{E}[M_{T_\varepsilon}] \leq \mathbb{E}[M_0] = \Phi_0.$$

So, $\mathbb{E}[\delta T_\varepsilon + \Phi_{T_\varepsilon}] \leq \Phi_0$, and, since $\Phi_{T_\varepsilon} \geq 0$, $\mathbb{E}[\delta T_\varepsilon] \leq \Phi_0 < \infty$. Therefore, for the standard Frank-Wolfe algorithm, using Lemma 2.3.1,

$$\begin{aligned}\mathbb{E}[T_\varepsilon] &\leq \frac{\Phi_0^{(S)}}{\delta_S} \\ &= \frac{2\Phi_0^{(S)}}{\beta^{FW}(\varepsilon)\varepsilon} \\ &= 2(f(x_0) - f^*)\max\left\{\frac{8LD^2}{\varepsilon^2}, \frac{4}{\varepsilon}\right\} \\ &\sim \mathcal{O}\left(LD^2\frac{f(x_0) - f^*}{\varepsilon^2}\right) \\ &\sim \mathcal{O}\left(\frac{f(x_0) - f^*}{\varepsilon^2}\right)\end{aligned}$$

Similarly, for the away-step Frank-Wolfe algorithm, using Lemma 2.3.7,

$$\begin{aligned}
\mathbb{E}[T_\varepsilon] &\leq \frac{\Phi_0^{(A)}}{\delta_A} \\
&= (\Phi_0^{(A)}) \left(2 + \frac{4}{\beta^{aFW}\varepsilon}\right) \\
&\sim \mathcal{O}\left(\frac{f(x_0) - f^* + |U_0|}{\beta^{aFW}\varepsilon}\right) \\
&\sim \mathcal{O}\left(\frac{LD^2N^2}{\mu\Omega_{\mathcal{X}}^2}\frac{f(x_0) - f^*}{\varepsilon}\right) \\
&\sim \mathcal{O}\left(\frac{f(x_0) - f^*}{\varepsilon}\right)
\end{aligned}$$

□

The proof of Theorem 2.5.2 provides explicit expressions for bounds on $\mathbb{E}T_\varepsilon$, making explicit the dependence on various additional key quantities beyond ε, including the Lipschitz constant L, the diameter of the feasible region D, and the initial optimality gap $f(x_0) - f^*$. The away-step Frank-Wolfe iteration complexity also depends on the strong convexity constant μ, the feasible region's pyramidal width constant $\Omega_{\mathcal{X}}$ and the number of vertices N (or $d+1$ where d is the dimension size).

We are now in a position to provide a result on the required per-iteration sample complexity under the bounded second moment assumption.

Theorem 2.5.3. *The sample sizes in each iteration that are required to achieve a good gradient estimate under the Assumption 2.4.1 (bounded variance) are as follows.*

1. *For the Standard Frank-Wolfe under Assumptions 2.2.1 (L−smoothness) and 2.2.3 (convexity) the needed sample size is $\mathcal{O}(\varepsilon^{-4})$.*
2. *For the away-step Frank-Wolfe under Assumptions 2.2.1 (L−smoothness)*

and 2.2.4 (strong convexity) the needed sample size is $\mathcal{O}(\varepsilon^{-2})$.

Proof. Choose p_g as in (2.16) with $\delta_S = \beta^{FW}(\varepsilon)\varepsilon/2$. Then

$$1 - p_g = \frac{\beta^{FW}(\varepsilon)\varepsilon}{\Gamma\varepsilon/2 + L\Gamma^2 D^2 + 2\beta^{FW}(\varepsilon)\varepsilon}.$$

Hence, the necessary sample size n in (2.8) from Lemma 2.5.1 is (using $\Gamma \leq 1$)

$$\begin{aligned}
\frac{16V_g D^2}{\varepsilon^2(1-p_g)} &= \frac{16V_g D^2}{\varepsilon^3 \beta^{FW}(\epsilon)} \left(\Gamma\varepsilon/2 + L\Gamma^2 D^2 + 2\beta^{FW}(\varepsilon)\varepsilon\right) \\
&\sim \mathcal{O}\left(\frac{16V_g L D^4}{\varepsilon^3 \beta^{FW}(\varepsilon)}\right) \\
&\sim \mathcal{O}(V_g L^2 D^6 \varepsilon^{-4}) \\
&\sim \mathcal{O}(\varepsilon^{-4}).
\end{aligned}$$

This completes the proof for the standard Frank-Wolfe algorithm.

For the away-step Frank-Wolfe algorithm, with the choice of ν, δ_A and p_g from (2.20), we have

$$1 - p_g = \frac{\beta^{aFW}\varepsilon/4}{\Gamma D\kappa\sqrt{\mu\varepsilon} + L\Gamma^2 D^2/2 + \beta^{aFW}\varepsilon}.$$

Then, the necessary sample size in (2.11) from Lemma 2.5.1 is

$$
\begin{aligned}
&\frac{V_g}{\mu\varepsilon \min\{1, \varepsilon_g^2\Omega_{\mathcal{X}}^2/8N^2\}(1-p_g)} \\
&= \frac{V_g}{\mu\varepsilon \min\{1, \varepsilon_g^2\Omega_{\mathcal{X}}^2/8N^2\}} \cdot \frac{\Gamma D\kappa\sqrt{\mu\varepsilon} + L\Gamma^2D^2/2 + \beta_2\varepsilon}{\beta_2\varepsilon/4} \\
&\leq \frac{V_g}{\mu\varepsilon \min\{1, \varepsilon_g^2\Omega_{\mathcal{X}}^2/8N^2\}} \cdot \left(\frac{4D\kappa\sqrt{\mu}}{\beta_2\sqrt{\varepsilon}} + \frac{2LD^2}{\beta_2\varepsilon} + 4\right) \\
&\leq \frac{4V_gD}{\min\{1, \varepsilon_g\Omega_{\mathcal{X}}/2\sqrt{2}N\}\beta_2\varepsilon\sqrt{\mu\varepsilon}} + \frac{4V_gLD^2}{\min\{1, \varepsilon_g^2\Omega_{\mathcal{X}}^2/8N^2\}\mu\beta_2\varepsilon^2} \\
&\sim \mathcal{O}\left(\frac{L^2D^6N^4}{\mu^2\Omega_{\mathcal{X}}^4}\varepsilon^{-2}\right) \\
&\sim \mathcal{O}\left(\varepsilon^{-2}\right).
\end{aligned}
$$

□

We can obtain an improved per-iteration sample complexity if the gradient estimates have lighter tails, as in Theorem 2.5.4 below.

Theorem 2.5.4. *The sample sizes in each iteration that are required to achieve a good gradient estimate under the Assumption 2.4.2 (sub-Gaussian) are as follows.*

1. *For the Standard Frank-Wolfe under Assumptions 2.2.1 (L−smoothness) and 2.2.3 (convexity) the needed sample size is* $\mathcal{O}(\varepsilon^{-2}\log(\varepsilon^{-1}))$.
2. *For the away-step Frank-Wolfe under Assumptions 2.2.1 (L−smoothness) and 2.2.4 (strong convexity) the needed sample size is* $\mathcal{O}(\varepsilon^{-1}\log(\varepsilon^{-1}))$.

Proof. Consider the standard Frank-Wolfe algorithm. From the same Cauchy-

Schwarz argument used in the proof of Theorem 2.5.1 in (2.15) and (2.7), we have

$$\mathbb{E}[\Phi_{k+1}^{(S)} - \Phi_k^{(S)} | \mathcal{F}_k] \leq -\beta^{FW} \varepsilon p_g + \Gamma D \sqrt{\frac{2d}{n_k c}} \sqrt{1 - p_g} + \frac{L\Gamma^2 D^2}{2}(1 - p_g).$$

In particular, the term V_g/n_k is replaced by $2d/n_k c$ from (2.7).

$$\mathbb{E}[\Phi_{k+1}^{(S)} - \Phi_k^{(S)} | \mathcal{F}_k] \leq -\beta^{FW} \varepsilon p_g + \Gamma D \sqrt{\frac{2d}{n_k c}} \sqrt{1 - p_g} + \frac{L\Gamma^2 D^2}{2}(1 - p_g). \quad (2.21)$$

Pick the probability p_g so that

$$1 - p_g \leq \min \left\{ \frac{1}{6}, \frac{\varepsilon^4 c}{4608 d L^2 \Gamma^2 D^6}, \frac{\varepsilon^2}{24 L^2 \Gamma^2 D^4} \right\} \quad (2.22)$$

so that

$$\begin{aligned}
\beta^{FW} \varepsilon (1 - p_g) &\leq \frac{\beta^{FW} \varepsilon}{6}, \\
\Gamma D \sqrt{\frac{2d}{n_k c}} \sqrt{1 - p_g} &\leq \Gamma D \sqrt{\frac{2d}{c}} \sqrt{\frac{c}{2d}} \frac{\varepsilon^2}{48 L \Gamma D^3} = \frac{\varepsilon^2}{48 L D^2} \leq \frac{\beta^{FW} \varepsilon}{6}, \text{ and} \\
\frac{L\Gamma^2 D^2}{2}(1 - p_g) &\leq \frac{\varepsilon^2}{48 L D^2} \leq \frac{\beta^{FW} \varepsilon}{6}.
\end{aligned}$$

Combining these three inequalities,

$$\beta^{FW} \varepsilon (1 - p_g) + \Gamma D \sqrt{\frac{2d}{n_k c}} \sqrt{1 - p_g} + \frac{L\Gamma^2 D^2}{2}(1 - p_g) \leq \frac{\beta^{FW} \varepsilon}{2}.$$

Hence, from (2.21),

$$\mathbb{E}[\Phi_{k+1}^{(S)} - \Phi_k^{(S)} | \mathcal{F}_k] \leq -\beta^{FW} \varepsilon p_g + \Gamma D \sqrt{\frac{2d}{n_k c}} \sqrt{1 - p_g} + \frac{L\Gamma^2 D^2}{2}(1 - p_g) \leq -\frac{\beta^{FW} \varepsilon}{2}.$$

Then, with the chosen constants, the Lyapunov condition as in Theorem 2.5.1 holds for $\delta_S = \beta^{FW}(\varepsilon)\varepsilon/2$. Now take $s = \varepsilon/(4D)$ in Lemma 2.4.3 and select the sample size n so large such that $2de^{-nc\varepsilon^2/(16D^2)}$ is a lower bound of $1 - p_g$. The term involving ε^4 is the determining factor for small ε, which gives

$$n \geq \frac{16D^2}{c\varepsilon^2} \log \frac{9216d^2L^2\Gamma^2D^6}{\varepsilon^4 c} \sim \mathcal{O}\left(\frac{D^2}{\varepsilon^2} \log \frac{LD}{\varepsilon}\right) \sim \mathcal{O}\left(\varepsilon^{-2} \log \varepsilon^{-1}\right),$$

establishing the result for the standard Frank-Wolfe algorithm. For the away-step Frank-Wolfe algorithm, from the proof of Theorem 2.5.1 and (2.7),

$$\mathbb{E}[\Phi_{k+1}^{(A)} - \Phi_k^{(A)} | \mathcal{F}_k] \leq -\frac{\nu\beta^{aFW}\varepsilon p_g}{2} + \nu\left(\Gamma D\sqrt{\frac{2d}{n_k c}}\sqrt{1-p_g} + \frac{L\Gamma^2D^2}{2}(1-p_g)\right) + (1-\nu)(1-p_g),$$

where $\nu = 2/(2 + \beta^{aFW}\varepsilon)$. Pick the probability p_g such that

$$1 - p_g = \frac{(\beta^{aFW}\varepsilon/4)^2}{(\Gamma D\sqrt{2d/c} + L\Gamma^2D^2/2 + \beta^{aFW}\varepsilon)^2}. \tag{2.23}$$

Then,

$$\begin{aligned}
\frac{\nu\beta^{aFW}\varepsilon}{2}(1-p_g) &\leq \frac{\nu\beta^{aFW}\varepsilon}{2}\sqrt{1-p_g} = \frac{\nu\beta^{aFW}\varepsilon}{4} \cdot \frac{\beta^{aFW}\varepsilon/2}{\Gamma D\sqrt{2d/c} + L\Gamma^2D^2/2 + \beta^{aFW}\varepsilon}, \\
\nu\Gamma D\sqrt{\frac{2d}{c}}\sqrt{1-p_g} &\leq \frac{\nu\beta^{aFW}\varepsilon}{4} \cdot \frac{\Gamma D\sqrt{2d/c}}{\Gamma D\sqrt{2d/c} + L\Gamma^2D^2/2 + \beta^{aFW}\varepsilon}, \text{ and} \\
\left(\frac{\nu L\Gamma^2D^2}{2} + 1 - \nu\right)(1-p_g) &\leq \left(\frac{\nu L\Gamma^2D^2}{2} + 1 - \nu\right)\sqrt{1-p_g} \\
&= \left(\frac{\nu L\Gamma^2D^2}{2} + \frac{\nu\beta^{aFW}\varepsilon}{2}\right)\sqrt{1-p_g} \\
&= \frac{\nu\beta^{aFW}\varepsilon}{4} \cdot \frac{L\Gamma^2D^2/2 + \beta^{aFW}\varepsilon/2}{\Gamma D\sqrt{2d/c} + L\Gamma^2D^2/2 + \beta^{aFW}\varepsilon}.
\end{aligned}$$

Combining the three inequalities above, we have

$$\frac{\nu\beta^{aFW}\varepsilon}{2}(1-p_g)+\nu\left(\Gamma D\sqrt{\frac{2d}{n_k c}}\sqrt{1-p_g}+\frac{L\Gamma^2 D^2}{2}(1-p_g)\right)+(1-\nu)(1-p_g)\leq\frac{\nu\beta^{aFW}\varepsilon}{4}.$$

Hence,

$$\begin{aligned}
&\mathbb{E}[\Phi_{k+1}^{(A)}-\Phi_k^{(A)}|\mathcal{F}_k]\\
&\quad\leq\ -\frac{\nu\beta^{aFW}\varepsilon p_g}{2}+\nu\left(\Gamma D\sqrt{\frac{2d}{n_k c}}\sqrt{1-p_g}+\frac{L\Gamma^2 D^2}{2}(1-p_g)\right)+(1-\nu)(1-p_g)\\
&\quad\leq\ -\frac{\nu\beta^{aFW}\varepsilon}{4}.
\end{aligned}$$

Thus we have the desired expected decrease in the Lyapunov function with the same value $\delta_A=\nu\beta^{aFW}\varepsilon/4=\frac{1}{2}\beta^{aFW}\varepsilon/(2+\beta^{aFW}\varepsilon)$ as in Theorem 2.5.1. Next, in view of (2.10), take $s^2=c_1\varepsilon$, where

$$c_1=\min\left\{1,\frac{\varepsilon_g^2\Omega_{\mathcal{X}}^2}{8N^2}\right\}\mu.$$

Then, from Lemma 2.4.3, it suffices to take n so large that $2d\exp(-ncc_1\varepsilon)\leq 1-p_g$, i.e.,

$$\begin{aligned}
n\ &\geq\ \frac{1}{cc_1\varepsilon}\log\frac{8d}{\beta^{aFW}\varepsilon}\left(\Gamma D\sqrt{\frac{2d}{c}}+\frac{\Gamma^2 LD^2}{2}+\beta^{aFW}\varepsilon\right)\\
&\sim\ \mathcal{O}\left(\frac{N^2D^2}{\mu\Omega_{\mathcal{X}}^2}\frac{1}{\varepsilon}\log\left(\frac{NLD}{\mu\Omega_{\mathcal{X}}}\frac{1}{\varepsilon}\right)\right)\\
&\sim\ \mathcal{O}\left(\varepsilon^{-1}\log\varepsilon^{-1}\right)
\end{aligned}$$

□

In summary, from Theorem 2.5.2, Theorem 2.5.3 and Theorem 2.5.4, the total work complexity for standard Frank-Wolfe under Assumption 2.2.1 ($L-$smoothness), Assumption 2.2.3 (convexity) are

1. $\mathcal{O}(\varepsilon^{-6})$ under Assumption 2.4.1 (bounded variance)
2. $\mathcal{O}(\varepsilon^{-4}\log\varepsilon^{-1})$ under Assumption 2.4.2 (sub-Gaussian)

, and total work complexity for away-step Frank-Wolfe under Assumption 2.2.1 ($L-$smoothness), Assumption 2.2.4 (strong convexity) are

1. $\mathcal{O}(\varepsilon^{-3})$ under Assumption 2.4.1 (bounded variance)
2. $\mathcal{O}(\varepsilon^{-2}\log\varepsilon^{-1})$ under Assumption 2.4.2 (sub-Gaussian)

The term $\varepsilon_g\Omega_{\mathcal{X}}$ is of order $\mathcal{O}(\Omega_{\mathcal{X}}/D)$ and is represented by the ratio χ defined in (2.12), which serves as a geometric "condition number"-like measure of the feasible region $\mathcal{X}$. This ratio depends only on the geometry of $\mathcal{X}$ and is invariant under scaling of the feasible region. In addition, by [72], the parameter N can be upper bounded by $d+1$, where d is the dimension of the feasible region. The bound on the expected stopping time obtained in Theorem 2.5.2 for standard and away-step Frank-Wolfe can be expressed more explicitly as

$$\mathcal{O}\left(\frac{LD^2}{\varepsilon^2}\right) \text{ and } \mathcal{O}\left(\frac{LD^2N^2}{\mu\Omega_{\mathcal{X}}^2\varepsilon}\right)$$

respectively. Under Assumption 2.4.1 (bounded variance), Theorem 2.5.3 yields

the standard and away-step Frank-Wolfe's sample size complexity of

$$\mathcal{O}\left(V_g L^2 D^6 \varepsilon^{-4}\right) \text{ and } \mathcal{O}\left(\frac{V_g L^2 D^6 N^4}{\mu^2 \Omega_{\mathcal{X}}^4} \varepsilon^{-2}\right)$$

respectively. Under Assumption 2.4.2 (sub-Gaussian), by Theorem 2.5.4 the sample size complexity becomes

$$\mathcal{O}\left(\frac{D^2}{\varepsilon^2} \log \frac{dLD}{\varepsilon}\right) \text{ and } \mathcal{O}\left(\frac{N^2 D^2}{\mu \Omega_{\mathcal{X}}^2} \frac{1}{\varepsilon} \log\left(\frac{dNLD}{\mu \Omega_{\mathcal{X}}} \frac{1}{\varepsilon}\right)\right)$$

respectively. Combining the iteration and sample size complexities gives the corresponding total work complexities. Under the bounded-variance assumption, the total work complexity for standard and away-step Frank-Wolfe are

$$\mathcal{O}\left(V_g L^3 D^8 \varepsilon^{-6}\right) \text{ and } \mathcal{O}\left(\frac{V_g L^3 D^8 N^6}{\mu^3 \Omega_{\mathcal{X}}^6} \varepsilon^{-3}\right)$$

respectively whereas under the sub-Gaussian assumption, they are

$$\mathcal{O}\left(\frac{LD^4}{\varepsilon^4} \log \frac{dLD}{\varepsilon}\right) \text{ and } \mathcal{O}\left(\frac{LD^4 N^4}{\mu^2 \Omega_{\mathcal{X}}^4} \frac{1}{\varepsilon^2} \log\left(\frac{dNLD}{\mu \Omega_{\mathcal{X}}} \frac{1}{\varepsilon}\right)\right).$$

By the definition of χ in (2.12), the complexity bounds for the away-step Frank–Wolfe algorithm can be expressed in terms of this geometric ratio. The iteration complexity can be written as

$$\mathcal{O}\left(\frac{LN^2}{\mu \chi^2 \varepsilon}\right).$$

Similarly, under the bounded variance and sub-Gaussian assumptions, the sample

size complexities can be written as

$$\mathcal{O}\left(\frac{V_g L^2 D^2 N^4}{\mu^2 \chi^4}\varepsilon^{-2}\right) \text{ and } \mathcal{O}\left(\frac{N^2}{\mu\chi^2}\frac{1}{\varepsilon}\log\left(\frac{dNL}{\mu\chi}\frac{1}{\varepsilon}\right)\right).$$

respectively. Therefore, the total complexities of the away-step Frank–Wolfe algorithm under the bounded variance and sub-Gaussian assumptions can be written as

$$\mathcal{O}\left(\frac{V_g L^3 D^2 N^6}{\mu^3 \chi^6}\varepsilon^{-3}\right) \text{ and } \mathcal{O}\left(\frac{LN^4}{\mu^2\chi^4}\frac{1}{\varepsilon^2}\log\left(\frac{dNL}{\mu\chi}\frac{1}{\varepsilon}\right)\right),$$

respectively, where N can be further upper bounded by $d+1$, with d denoting the dimension of the affine hull of the feasible region.

CHAPTER 3

FRANK-WOLFE ALGORITHM EXTENSIONS

3.1 Introduction

In the previous chapter, we established convergence guarantees for the standard and away-step Frank–Wolfe algorithms in the stochastic setting, where the objective takes the form

$$\min_{x \in \mathcal{X}} f(x) = \mathbb{E}[F(x, \xi)].$$

In particular, the analysis focused on the case of unbiased gradient estimators and characterized the corresponding iteration complexities for both variants of the algorithm.

This chapter extends the analysis along several important directions that arise naturally in stochastic optimization problems. First, we investigate the behavior of Frank–Wolfe methods under biased gradient estimators. While unbiasedness is a standard assumption in theoretical analyses, it is often violated in practice for some simulation-optimization problems where, for example, gradients cannot be obtained using Infinitesimal Perturbation Analysis (IPA), but with Finite Differences. Understanding this regime is therefore essential for bridging the gap between theory and practical implementations. We show that meaningful convergence guarantees can still be established in the presence of bias, and we characterize how the magnitude and structure of the bias affect the overall complexity. At the same time, biased gradient estimates introduce inherent limitations: unlike the unbiased case, persistent bias may prevent exact convergence to the true optimum and instead lead to convergence to a neighborhood whose size depends on the bias

level.

Second, we study the distributional properties of the time required to reach an ε-optimal solution. Beyond expectation-based guarantees, we establish that the corresponding hitting time exhibits light-tailed behavior by showing that it admits a finite moment generating function in a neighborhood of the origin. This provides a stronger form of probabilistic control on the algorithm's convergence.

Finally, we consider the regime in which the optimal solution lies in the interior of the feasible region $\mathcal{X}$. In this setting, the geometry of the problem allows for improved complexity guarantees. We show that standard Frank–Wolfe algorithm benefits from this additional structure under strong convexity assumption, leading to faster convergence rate in the same order as away-step Frank-Wolfe's.

3.2 Complexity with Biased Gradient Estimators

We have now established the desired complexity bounds when unbiased gradient estimators are available. Sometimes, bias in gradient estimators is unavoidable. For example, in simulation this happens when one cannot use so-called *direct* gradient estimation methods [28]. In that setting, there is a limit to the attainable accuracy, though restricted convergence results are still possible, e.g., see [3] on gradient descent with biased gradient estimates. Here we obtain related results for the standard and away-step Frank-Wolfe methods.

Earlier, Algorithm 1 and Lemma 2.3.1 were stated in terms of the target error ε. In particular, Algorithm 1 specifies a step size rule that depends explicitly on this quantity. In the following analysis, however, the target error ε will be replaced by

a new neighborhood error ε', which depends on additional parameters associated with the feasible region $\mathcal{X}$. Consequently, Algorithm 1 applies the step size rule

$$\gamma_k = \min\left\{1, \frac{\varepsilon'}{2LD^2}\right\}$$

For the remainder of this section, all proofs for both the standard and Away-step Frank-Wolfe algorithms are carried out using the modified target error ε', which will be defined in the statement of the corresponding corollary.

Perhaps surprisingly, with biased gradients the attainable convergence neighborhood in the standard Frank–Wolfe analysis depends explicitly on the diameter D of the feasible region $\mathcal{X}$. We prove such a result below and then show through examples that this dependence on D is necessary. In contrast, for the away-step Frank–Wolfe method, the neighborhood does not depend on D; instead, it is governed by the problem's geometric and functional parameters, namely the strong convexity constant μ and the geometry of the feasible region characterized by N. We will discuss an intuition for these observations after giving the results and examples.

Our first result assumes biased gradient estimates with bounded second moments.

Assumption 3.2.1. *For all feasible x, the i.i.d. random vectors $G_1(x), G_2(x), \ldots$ can be written as $G(x,\xi) = \nabla f(x) + b(x) + \eta(x,\xi)$, where $b(x)$ is a bias term with $\|b(x)\| \leq B$ for some constant $B > 0$ and all x, and $\eta(x,\xi)$ is mean 0 stochastic error with bounded second moment, i.e., $\mathbb{E}[\eta(x,\xi)] = 0$ and $\mathbb{E}[\|\eta(x,\xi)\|^2] \leq V_g$ for all x.*

Corollary 3.2.1. *Suppose that Assumption 3.2.1 (bounded variance) holds.*

1. *Under Assumptions 2.2.1 (L−smoothness) and 2.2.3 (convexity) the standard Frank-Wolfe algorithm has the same iteration and sample-size complexities as given in Theorems 2.5.2 and 2.5.3, but we redefine the terminal time to be* $T_\varepsilon = \min\{k : f(x_k) - f^* \leq \varepsilon + 4BD\}$.

2. *Under Assumptions 2.2.1 (L−smoothness) and 2.2.4 (strong convexity) the away-step Frank-Wolfe algorithm has the same iteration and sample-size complexities as given in Theorems 2.5.2 and 2.5.3, but we redefine the terminal time to be* $T_\varepsilon = \min\{k : f(x_k) - f^* \leq \varepsilon + 2B\sqrt{\varepsilon}/(\kappa\sqrt{\mu}) + B^2/(\kappa^2\mu)\}$ *where the constant* κ *was defined in* (2.19).

Proof. Consider Lemmas 2.4.1 and 2.4.2. We now have

$$\begin{aligned}\mathbb{E}[\|g(x) - \nabla f(x)\|^2] &= \mathbb{E}[\|g(x) - \mathbb{E}[g(x)]\|^2] + \mathbb{E}[\|\nabla f(x) - \mathbb{E}[g(x)]\|^2] \\ &= \mathbb{E}[\|g(x) - \mathbb{E}[g(x)]\|^2] + \mathbb{E}[\|\nabla f(x) - \mathbb{E}[G(x,\xi)]\|^2] \\ &\leq \frac{V_g}{n} + B^2, \end{aligned} \tag{3.1}$$

and, by the triangle inequality,

$$\mathbb{P}(\|g(x) - \nabla f(x)\| > s + B) \leq \mathbb{P}(\|g(x) - \mathbb{E}[g(x)]\| > s) \leq \frac{V_g}{ns^2}. \tag{3.2}$$

Part I: Standard Frank-Wolfe. Define a new neighborhood radius as $\varepsilon' = \varepsilon + 4BD$. By Definition 2.3.1, and Lemma 2.3.1, we have the same result, but where ε' replaces ε and $\beta^{FW}(\varepsilon')$ replaces $\beta^{FW}(\varepsilon)$. In particular, we approximate the

gradient so that $\|g(x) - \nabla f(x)\| \leq \varepsilon'/(4D)$. To that end, from (3.2),

$$\mathbb{P}\left(\|g(x) - \nabla f(x)\| > \frac{\varepsilon'}{4D}\right) = \mathbb{P}\left(\|g(x) - \nabla f(x)\| > \frac{\varepsilon}{4D} + B\right) \leq \frac{16V_g D^2}{n\varepsilon^2}.$$

Thus, similar to Lemma 2.5.1, we require the same sample size

$$n \geq \frac{16V_g D^2}{\varepsilon^2(1-p_g)}. \tag{3.3}$$

Then, using the Lyapunov function as defined in (2.13), the analogous result to (2.15) in the proof of Theorem 2.5.1 is

$$\mathbb{E}[\Phi_{k+1}^{(S)} - \Phi_k^{(S)} | \mathcal{F}_k] \leq -\beta^{FW}(\varepsilon')\varepsilon' p_g + \gamma_k D\sqrt{\frac{V_g}{n_k} + B^2}\sqrt{1-p_g} + \frac{L\Gamma^2 D^2}{2}(1-p_g),$$

where we do not yet replace the stepsize γ_k by an upper bound. We want to upper bound the right hand side away from 0. To that end, choose the probability p_g so that

$$p_g \geq \max\left\{\frac{2}{1+\sqrt{1+L^2D^2/4B^2}}, \frac{2}{1+\sqrt{2}}, \frac{\Gamma\varepsilon/4 + L\Gamma^2D^2/2 + \beta^{FW}(\varepsilon')\varepsilon'/4}{\Gamma\varepsilon/4 + L\Gamma^2D^2/2 + \beta^{FW}(\varepsilon')\varepsilon'/2}\right\}.$$

The definition of $\beta_1(\epsilon)$ gives

$$\beta^{FW\,2}(\varepsilon') = \min\left\{\left(\frac{\varepsilon'}{8LD^2}\right)^2, \frac{1}{16}\right\} = \min\left\{\left(\frac{\varepsilon + 4BD}{8LD^2}\right)^2, \frac{1}{16}\right\} \geq \min\left\{\frac{B^2}{4L^2D^2}, \frac{1}{16}\right\},$$

which then implies that

$$\max\left\{\frac{2}{1+\sqrt{1+L^2D^2/4B^2}}, \frac{2}{1+\sqrt{2}}\right\} \geq \frac{2}{1+\sqrt{1+4\beta^{FW^2}(\varepsilon')L^2D^2/B^2}}.$$

Therefore,

$$p_g \geq \frac{2}{1+\sqrt{1+4\beta^{FW^2}(\varepsilon')L^2D^2/B^2}},$$

which then implies that

$$\begin{aligned}\frac{p_g}{\sqrt{1-p_g}} &\geq \frac{2}{1+\sqrt{1+4\beta^{FW^2}(\varepsilon')L^2D^2/B^2}} \cdot \frac{\sqrt{\sqrt{1+4\beta^{FW^2}(\varepsilon')L^2D^2/B^2}+1}}{\sqrt{\sqrt{1+4\beta^{FW^2}(\varepsilon')L^2D^2/B^2}-1}} \\ &= \frac{2}{\sqrt{4\beta^{FW^2}(\varepsilon')L^2D^2/B^2}} = \frac{B}{\beta^{FW}(\varepsilon')LD}.\end{aligned}$$

Recall that the stepsize, γ_k, is bounded by $\varepsilon'/(2LD^2)$ in the standard Frank-Wolfe method, so, from the inequality we just established,

$$\gamma_k DB\sqrt{1-p_g} \leq \frac{\varepsilon'}{2LD^2}DB\sqrt{1-p_g} \leq \frac{\beta^{FW}(\varepsilon')\varepsilon' p_g}{2}.$$

Therefore, from (3.3) and our choice of p_g,

$$
\begin{aligned}
\mathbb{E}[\Phi_{k+1}^{(S)} - \Phi_k^{(S)} | \mathcal{F}_k] &\leq -\beta^{FW}(\varepsilon')\varepsilon' p_g + \gamma_k D \sqrt{\frac{V_g}{n_k} + B^2} \sqrt{1-p_g} + \frac{L\Gamma^2 D^2}{2}(1-p_g) \\
&\leq -\beta^{FW}(\varepsilon')\varepsilon' p_g + \gamma_k D \sqrt{\frac{V_g}{n_k}} \sqrt{1-p_g} + \gamma_k D B \sqrt{1-p_g} + \frac{L\Gamma^2 D^2}{2}(1-p_g) \\
&\leq -\beta^{FW}(\varepsilon')\varepsilon' p_g + \frac{\Gamma\varepsilon}{4}(1-p_g) + \frac{\beta^{FW}(\varepsilon')\varepsilon' p_g}{2} + \frac{L\Gamma^2 D^2}{2}(1-p_g) \\
&= \frac{-\beta^{FW}(\varepsilon')\varepsilon' p_g}{2} + \frac{\Gamma\varepsilon}{4}(1-p_g) + \frac{L\Gamma^2 D^2}{2}(1-p_g) \\
&= (1-p_g)\left(\frac{\Gamma\varepsilon}{4} + \frac{L\Gamma^2 D^2}{2} + \frac{\beta^{FW}(\varepsilon')\varepsilon'}{2}\right) - \frac{\beta^{FW}(\varepsilon')\varepsilon'}{2} \\
&\leq -\frac{\beta^{FW}(\varepsilon')\varepsilon'}{4},
\end{aligned}
$$

establishing the desired negative drift that is the conclusion of Theorem 2.5.1 for $\delta_S = \beta^{FW}(\varepsilon')\varepsilon'/4$. As in Theorem 2.5.2, we then have iteration complexity of the order $\mathcal{O}\left(LD^2(f(x_0) - f^*)(\varepsilon')^{-2}\right)$, which is bounded above by $\mathcal{O}\left(LD^2(f(x_0) - f^*)\varepsilon^{-2}\right)$. Moreover, due to (3.2) and the argument of Lemma 2.5.1, we still have the same sample size complexity as in Theorem 2.5.3.

Part II: Away-step Frank-Wolfe. Definition 2.3.2 of a good gradient remains the same, which ensures that Lemma 2.3.5 still applies. Under good gradient approximate, by (2.6) from Lemma 2.3.6, we now have

$$
(g_k^T(v_k - s_k))^2 \geq \left(\frac{\Omega_{\mathcal{X}}}{N}\right)^2 \frac{\mu\varepsilon'}{2(2\varepsilon_g D + 1)^2},
$$

where $\varepsilon' = \varepsilon + 2B\sqrt{\varepsilon}/(\kappa\sqrt{\mu}) + B^2/(\kappa^2\mu) = (\sqrt{\varepsilon} + B/(\kappa\sqrt{\mu}))^2$, which replaces ε, and

$$
\kappa = \min\left\{1, \frac{\varepsilon_g \Omega_{\mathcal{X}}}{2\sqrt{2}N}\right\}.
$$

We now approximate the gradient so that

$$\begin{aligned} \|g(x) - \nabla f(x)\| &\leq \kappa\sqrt{\mu\varepsilon} + B \\ &= \kappa\sqrt{\mu\varepsilon'} \\ &\leq \frac{\varepsilon_g \Omega_{\mathcal{X}}}{2\sqrt{2}N}\sqrt{\mu\varepsilon'}. \end{aligned}$$

By Lemma 2.5.1, the analogous result holds with ε' instead of ε where

$$\|g(x) - \nabla f(x)\| \leq \frac{\varepsilon_g \Omega_{\mathcal{X}}}{2\sqrt{2}N}\sqrt{\mu\varepsilon'} \leq \varepsilon_g g_k^T(v_k - s_k).$$

From (3.2) substitute $s = \kappa\sqrt{\mu\varepsilon}$, giving

$$\mathbb{P}(\|g(x) - \nabla f(x)\| > \kappa\sqrt{\mu\varepsilon} + B) \leq \frac{V_g}{n\kappa^2\mu\varepsilon}.$$

Thus, similar to Lemma 2.5.1, we require the sample size

$$n \geq \frac{V_g}{\kappa^2\mu\varepsilon(1-p_g)} = \frac{V_g}{\mu\varepsilon \min\{1, \varepsilon_g^2\Omega_{\mathcal{X}}^2/8N^2\}(1-p_g)}.$$

Therefore,

$$\Gamma D\sqrt{\frac{V_g}{n_k}}\sqrt{1-p_g} \leq \Gamma D\kappa\sqrt{\mu\varepsilon}(1-p_g).$$

Now, choose the probability p_g such that

$$p_g \geq \max\left\{\frac{2}{1+\sqrt{1+\frac{\beta^{aFW\,2}B^2}{4\Gamma^2D^2\kappa^4\mu^2}}},\ \frac{\Gamma D\kappa\sqrt{\mu\varepsilon} + L\Gamma^2D^2/2 + (5/8)\beta^{aFW}\varepsilon'}{\Gamma D\kappa\sqrt{\mu\varepsilon} + L\Gamma^2D^2/2 + (3/4)\beta^{aFW}\varepsilon'}\right\}.$$

Because $\varepsilon' = (\sqrt{\varepsilon} + B/(\kappa\sqrt{\mu}))^2 \geq B^2/(\kappa^2\mu)$ it follows that

$$p_g \geq \frac{2}{1+\sqrt{1+\frac{\beta^{aFW\,2}B^2}{4\Gamma^2D^2\kappa^4\mu^2}}} \geq \frac{2}{1+\sqrt{1+\frac{\beta^{aFW\,2}\varepsilon'}{4\Gamma^2D^2\kappa^2\mu}}}.$$

This implies that

$$\frac{p_g}{\sqrt{1-p_g}} \geq \frac{2}{1+\sqrt{1+\frac{\beta^{aFW\,2}\varepsilon'}{4\Gamma^2D^2\kappa^2\mu}}} \cdot \frac{\sqrt{\sqrt{1+\frac{\beta^{aFW\,2}\varepsilon'}{4\Gamma^2D^2\kappa^2\mu}}+1}}{\sqrt{\sqrt{1+\frac{\beta^{aFW\,2}\varepsilon'}{4\Gamma^2D^2\kappa^2\mu}}-1}} = \frac{4\Gamma D\kappa\sqrt{\mu}}{\beta^{aFW}\sqrt{\varepsilon'}}.$$

From this inequality and the definition of ε',

$$\Gamma DB\sqrt{1-p_g} = \Gamma D(\sqrt{\varepsilon'}-\sqrt{\varepsilon})\kappa\sqrt{\mu}\sqrt{1-p_g} \leq \Gamma D\sqrt{\varepsilon'}\kappa\sqrt{\mu}\sqrt{1-p_g} \leq \frac{\beta^{aFW}\varepsilon' p_g}{4}.$$

Define the Lyapunov function $\Phi^{(A)}$ as in (2.14) where we analogously take $\nu = 2/(2+\beta^{aFW}\varepsilon')$ as in the analysis of unbiased gradient estimators. Then, by the

choice of p_g, above, and the inequality (2.18) in the proof of Theorem 2.5.1,

$$
\begin{aligned}
&\mathbb{E}[\Phi_{k+1}^{(A)} - \Phi_k^{(A)} | \mathcal{F}_k] \\
&\leq -\frac{\nu\beta^{aFW}\varepsilon' p_g}{2} + \nu\Gamma D\sqrt{\frac{V_g}{n_k} + B^2}\sqrt{1-p_g} + \frac{\nu L\Gamma^2 D^2}{2}(1-p_g) + (1-\nu)(1-p_g) \\
&\leq -\frac{\nu\beta^{aFW}\varepsilon' p_g}{2} + \nu\Gamma D\sqrt{\frac{V_g}{n_k}}\sqrt{1-p_g} + \nu\Gamma DB\sqrt{1-p_g} + \left(\frac{\nu L\Gamma^2 D^2}{2} + 1 - \nu\right)(1-p_g) \\
&\leq -\frac{\nu\beta^{aFW}\varepsilon' p_g}{2} + \nu\Gamma D\kappa\sqrt{\mu\varepsilon}(1-p_g) + \frac{\nu\beta^{aFW}\varepsilon' p_g}{4} + \left(\frac{\nu L\Gamma^2 D^2}{2} + 1 - \nu\right)(1-p_g) \\
&= (1-p_g)\left(\frac{\nu\beta^{aFW}\varepsilon'}{4} + \nu\Gamma D\kappa\sqrt{\mu\varepsilon} + \frac{\nu L\Gamma^2 D^2}{2} + 1 - \nu\right) - \frac{\nu\beta^{aFW}\varepsilon'}{4} \\
&= (1-p_g)\left(\frac{\nu\beta^{aFW}\varepsilon'}{4} + \nu\Gamma D\kappa\sqrt{\mu\varepsilon} + \frac{\nu L\Gamma^2 D^2}{2} + \frac{\nu\beta^{aFW}\varepsilon'}{2}\right) - \frac{\nu\beta^{aFW}\varepsilon'}{4} \\
&= (1-p_g)\left(\nu\Gamma D\kappa\sqrt{\mu\varepsilon} + \frac{\nu L\Gamma^2 D^2}{2} + \frac{3\nu\beta^{aFW}\varepsilon'}{4}\right) - \frac{\nu\beta^{aFW}\varepsilon'}{4} \\
&\leq -\frac{\nu\beta^{aFW}\varepsilon'}{8} = -\frac{\beta^{aFW}\varepsilon'/4}{(2+\beta^{aFW}\varepsilon')}.
\end{aligned}
$$

We have established the desired negative drift that is the conclusion of Theorem 2.5.1 for $\delta_A = \frac{1}{4}\beta^{aFW}\varepsilon'/(2+\beta^{aFW}\varepsilon')$ and $\nu = 2/(2+\beta^{aFW}\varepsilon')$. Hence, as in the proof of Theorem 2.5.2, we obtain iteration complexity of the order

$$\mathcal{O}\left(\frac{LD^2N^2}{\mu\Omega_{\mathcal{X}}^2}\frac{f(x_0)-f^*}{\varepsilon'}\right)$$

which is, at worst,

$$\mathcal{O}\left(\frac{LD^2N^2}{\mu\Omega_{\mathcal{X}}^2}\frac{f(x_0)-f^*}{\varepsilon}\right).$$

From (3.2) and the choice of p_g, we still have the same sample-size complexity as in Theorem 2.5.3. □

We can also derive an analogous result when the gradient estimators have bounded

bias and sub-Gaussian noise.

Assumption 3.2.2. *For all feasible x, the i.i.d. random vectors $G_1(x), G_2(x), \ldots$ can be written as $G(x,\xi) = \nabla f(x) + b(x) + \eta(x,\xi)$, where $b(x)$ is a bias term with $\|b(x)\| \leq B$ for some constant $B > 0$ and all x, and for all x, $\eta(x,\xi)$ is sub-Gaussian with parameter $\rho/\sqrt{d}$, where d is the dimension of x.*

Corollary 3.2.2. *Suppose that Assumption 3.2.2 (sub-Gaussian) holds.*

1. *Under Assumptions 2.2.1 ($L-$smoothness) and 2.2.3 (convexity) the standard Frank-Wolfe algorithm has the same iteration and sample-size complexities as given in Theorems 2.5.2 and 2.5.4, but we redefine the terminal time to be $T_\varepsilon = \min\{k : f(x_k) - f^* \leq \varepsilon + 4BD\}$.*
2. *Under Assumptions 2.2.1 ($L-$smoothness) and 2.2.4 (strong convexity), the away-step Frank-Wolfe algorithm has the same iteration and sample-size complexities as given in Theorems 2.5.2 and 2.5.4, but we redefine the terminal time to be $T_\varepsilon = \min\{k : f(x_k) - f^* \leq \varepsilon + 2B\sqrt{\varepsilon}/(\kappa\sqrt{\mu}) + B^2/(\kappa^2\mu)\}$ where κ was defined in* (2.19).

Proof.

By Lemma 2.4.3 and (2.7),

$$\mathbb{E}[\|g(x) - \nabla f(x)\|^2] \leq \frac{2d}{nc} + B^2 \tag{3.4}$$

and

$$\mathbb{P}(\|g(x) - \nabla f(x)\| > s + B) \leq \mathbb{P}(\|g(x) - \mathbb{E}[g(x)]\| > s) \leq 2de^{-ncs^2}. \tag{3.5}$$

Part I: Standard Frank-Wolfe. Define $\varepsilon' = \varepsilon + 4BD$. As in Corollary 3.2.1, to ensure a good gradient estimate we define $s = \varepsilon/(4D)$ and apply (3.5) to get

$$\mathbb{P}(\|g(x) - \nabla f(x)\| > s + B) \leq 2de^{-nc\varepsilon^2/16D^2}.$$

Setting this expression equal to $1 - p_g$ yields the necessary sample size

$$n \geq \frac{16D^2}{c\varepsilon^2} \log\left(\frac{2d}{1-p_g}\right).$$

With the Lyapunov function defined in (2.13), analogously to (2.15) in the proof of Theorem 2.5.1,

$$\mathbb{E}[\Phi_{k+1}^{(S)} - \Phi_k^{(S)} | \mathcal{F}_k] \leq -\beta^{FW}(\varepsilon')\varepsilon' p_g + \Gamma D\sqrt{\frac{2d}{n_k c} + B^2}\sqrt{1-p_g} + \frac{L\Gamma^2 D^2}{2}(1-p_g).$$

Choose the probability p_g so that

$$p_g \geq \max\left\{ \frac{2}{1+\sqrt{1+\frac{B^2}{\Gamma^2(2d/c+B^2)}}}, \frac{2}{1+\sqrt{1+\frac{4B^4}{L^2D^2\Gamma^2(2d/c+B^2)}}}, \frac{2L\Gamma^2D^2 + \beta^{FW}\varepsilon'}{2L\Gamma^2D^2 + 2\beta^{FW}\varepsilon'} \right\}.$$

The definition of $\beta^{FW}(\epsilon')$ implies that

$$\beta^{FW\,2}(\varepsilon') = \min\left\{ \left(\frac{\varepsilon'}{8LD^2}\right)^2, \frac{1}{16} \right\} = \min\left\{ \left(\frac{\varepsilon + 4BD}{8LD^2}\right)^2, \frac{1}{16} \right\} \geq \min\left\{ \frac{B^2}{4L^2D^2}, \frac{1}{16} \right\},$$

so that

$$\beta^{FW\,2}(\varepsilon')\varepsilon'^2 \geq \min\left\{ \frac{B^2}{4L^2D^2}, \frac{1}{16} \right\} \cdot (4BD)^2 \geq \min\left\{ \frac{4B^4}{L^2}, B^2D^2 \right\}.$$

Thus,

$$p_g \geq \max\left\{ \frac{2}{1+\sqrt{1+\frac{B^2}{\Gamma^2(2d/c+B^2)}}},\ \frac{2}{1+\sqrt{1+\frac{4B^4}{L^2D^2\Gamma^2(2d/c+B^2)}}} \right\} \geq \frac{2}{1+\sqrt{1+\frac{\beta^{FW\,2}(\varepsilon')\varepsilon'^2}{\Gamma^2D^2(2d/c+B^2)}}}.$$

Then,

$$\begin{aligned}
\frac{p_g}{\sqrt{1-p_g}} &\geq \frac{2}{1+\sqrt{1+\frac{\beta^{FW\,2}(\varepsilon')\varepsilon'^2}{\Gamma^2D^2(2d/c+B^2)}}} \cdot \frac{\sqrt{\sqrt{1+\frac{\beta^{FW\,2}(\varepsilon')\varepsilon'^2}{\Gamma^2D^2(2d/c+B^2)}}+1}}{\sqrt{\sqrt{1+\frac{\beta^{FW\,2}(\varepsilon')\varepsilon'^2}{\Gamma^2D^2(2d/c+B^2)}}-1}} \\
&= \frac{2}{\sqrt{\frac{\beta^{FW\,2}(\varepsilon')\varepsilon'^2}{\Gamma^2D^2(2d/c+B^2)}}} \\
&= \frac{2\Gamma D\sqrt{2d/c+B^2}}{\beta^{FW}(\varepsilon')\varepsilon'}.
\end{aligned}$$

Therefore,

$$\Gamma D\sqrt{\frac{2d}{c}+B^2}\sqrt{1-p_g} \leq \frac{\beta^{FW}(\varepsilon')\varepsilon' p_g}{2}.$$

By the choice of the probability p_g, above,

$$\begin{aligned}
\mathbb{E}[\Phi_{k+1}^{(S)} - \Phi_k^{(S)}|\mathcal{F}_k] &\leq -\beta^{FW}(\varepsilon')\varepsilon' p_g + \Gamma D\sqrt{\frac{2d}{n_k c}+B^2}\sqrt{1-p_g} + \frac{L\Gamma^2D^2}{2}(1-p_g) \\
&\leq -\beta^{FW}(\varepsilon')\varepsilon' p_g + \frac{\beta^{FW}(\varepsilon')\varepsilon' p_g}{2} + \frac{L\Gamma^2D^2}{2}(1-p_g) \\
&= (1-p_g)\left(\frac{L\Gamma^2D^2}{2} + \frac{\beta^{FW}(\varepsilon')\varepsilon'}{2}\right) - \frac{\beta^{FW}(\varepsilon')\varepsilon'}{2} \\
&\leq -\frac{\beta^{FW}(\varepsilon')\varepsilon'}{4}.
\end{aligned}$$

We have therefore established the negative drift that is the conclusion of Theorem 2.5.1 for $\delta_S = \beta^{FW}(\varepsilon')\varepsilon'/4$. As in Theorem 2.5.2, we then have iteration complexity of the order $\mathcal{O}\left(LD^2(f(x_0)-f^*)(\varepsilon')^{-2}\right)$, which is bounded above by

$\mathcal{O}\left(LD^2(f(x_0) - f^*)\varepsilon^{-2}\right)$. Moreover, due to (3.5), the argument of Lemma 2.5.1, and the choice of p_g, we still have the same sample size complexity as in Theorem 2.5.4.

Part II: Away-step Frank-Wolfe. Define $\varepsilon' = (\sqrt{\varepsilon} + B/(\kappa\sqrt{\mu}))^2$ as in Part II of the proof of Corollary 3.2.1. The definition of a good gradient approximation, Definition 2.3.2, stays the same. Analogously to Corollary 3.2.1, we approximate the gradient so that

$$\|g(x) - \nabla f(x)\| \leq \kappa\sqrt{\mu\varepsilon} + B \leq \varepsilon_g g_k^T(v_k - s_k).$$

From (3.5) substitute $s = \kappa\sqrt{\mu\varepsilon}$, we have

$$\mathbb{P}(\|g(x) - \nabla f(x)\| > \kappa\sqrt{\mu\varepsilon} + B) \leq 2de^{-nc\kappa^2\mu\varepsilon}.$$

Thus, we require the sample size

$$n \geq \frac{1}{c\kappa^2\mu\varepsilon}\log\left(\frac{2d}{1-p_g}\right).$$

Choose the probability p_g such that

$$p_g \geq \max\left\{\frac{2}{1+\sqrt{1+\frac{{\beta^{aFW}}^2B^4}{\Gamma^2D^2\kappa^4\mu^2(2d/c+B^2)}}},\ \frac{4L\Gamma^2D^2+5\beta^{aFW}\varepsilon'}{4L\Gamma^2D^2+6\beta^{aFW}\varepsilon'}\right\}.$$

Because $\varepsilon' = (\sqrt{\varepsilon} + B/(\kappa\sqrt{\mu}))^2 \geq B^2/(\kappa^2\mu)$,

$$p_g \geq \frac{2}{1+\sqrt{1+\frac{\beta^{aFW\,2}B^4}{4\Gamma^2D^2\kappa^4\mu^2(2d/c+B^2)}}} \geq \frac{2}{1+\sqrt{1+\frac{\beta^{aFW\,2}\varepsilon'^2}{4\Gamma^2D^2(2d/c+B^2)}}}.$$

This implies that

$$\begin{aligned}
\frac{p_g}{\sqrt{1-p_g}} &\geq \frac{2}{1+\sqrt{1+\frac{\beta^{aFW\,2}\varepsilon'^2}{4\Gamma^2D^2(2d/c+B^2)}}} \cdot \frac{\sqrt{\sqrt{1+\frac{\beta^{aFW\,2}\varepsilon'^2}{4\Gamma^2D^2(2d/c+B^2)}}+1}}{\sqrt{\sqrt{1+\frac{\beta^{aFW\,2}\varepsilon'^2}{4\Gamma^2D^2(2d/c+B^2)}}-1}} \\
&= \frac{2}{\sqrt{\frac{\beta^{aFW\,2}\varepsilon'^2}{4\Gamma^2D^2(2d/c+B^2)}}} \\
&= \frac{4\Gamma D}{\beta^{aFW}\varepsilon'}\sqrt{\frac{2d}{c}+B^2}.
\end{aligned}$$

Therefore,

$$\Gamma D\sqrt{\frac{2d}{n_k c}+B^2}\sqrt{1-p_g} \leq \Gamma D\sqrt{\frac{2d}{c}+B^2}\sqrt{1-p_g} \leq \frac{\beta^{aFW}\varepsilon' p_g}{4}.$$

Define the Lyapunov function $\Phi^{(A)}$ as in (2.14) where we analogously take $\nu = 2/(2+\beta^{aFW}\varepsilon')$ as in the analysis of unbiased gradient estimators. Then, by the

choice of p_g, above, and the inequality (2.18) in the proof of Theorem 2.5.1,

$$
\begin{aligned}
&\mathbb{E}[\Phi_{k+1}^{(A)} - \Phi_k^{(A)} | \mathcal{F}_k] \\
&\leq -\frac{\nu\beta^{aFW}\varepsilon' p_g}{2} + \nu\Gamma D\sqrt{\frac{2d}{n_k c} + B^2}\sqrt{1-p_g} + \frac{\nu L\Gamma^2 D^2}{2}(1-p_g) + (1-\nu)(1-p_g) \\
&\leq -\frac{\nu\beta^{aFW}\varepsilon' p_g}{4} + \frac{\nu L\Gamma^2 D^2}{2}(1-p_g) + (1-\nu)(1-p_g) \\
&= (1-p_g)\left(\frac{\nu\beta^{aFW}\varepsilon'}{4} + \frac{\nu L\Gamma^2 D^2}{2} + 1 - \nu\right) - \frac{\nu\beta^{aFW}\varepsilon'}{4} \\
&= (1-p_g)\left(\frac{\nu L\Gamma^2 D^2}{2} + \frac{3\nu\beta^{aFW}\varepsilon'}{4}\right) - \frac{\nu\beta^{aFW}\varepsilon'}{4} \\
&\leq -\frac{\nu\beta^{aFW}\varepsilon'}{8} = -\frac{\beta^{aFW}\varepsilon'/4}{2+\beta^{aFW}\varepsilon'}.
\end{aligned}
$$

Thus we have established the desired negative drift that is the conclusion of Theorem 2.5.1, where $\delta_A = \frac{1}{4}\beta^{aFW}\varepsilon'/(2+\beta^{aFW}\varepsilon')$ and $\nu = 2/(2+\beta^{aFW}\varepsilon')$. Hence, we have the same iteration complexity as in Theorem 2.5.2. In addition, with the choice of p_g above, by (3.2), we also have the same sample size complexity as in Theorem 2.5.4. □

In the standard Frank–Wolfe algorithm with biased gradient estimates, the achievable error in the objective function value relative to the optimal solution, $\varepsilon + 4BD$, depends on the diameter D of the feasible region, as shown in Corollaries 3.2.1 and 3.2.2. In particular, a larger feasible region leads to a larger potential error. This dependence may, at first, appear counterintuitive: why should the achievable accuracy depend on the size of the feasible region? This dependence arises because, at each iteration, the standard Frank-Wolfe method solves a linear program and generates the search direction $d_k = s_k - x_k$, where s_k is a vertex of the feasible set. Consequently, the direction $s_k - x_k$ can have a magnitude that is close to the diameter D. Under Assumptions 2.2.1 ($L-$smoothness) and 2.2.3 (convexity), and

accounting for the gradient error term $\|g(x) - \nabla f(x)\|$, we obtain

$$f(x_{k+1}) \leq f(x_k) + \gamma_k(f^* - f(x_k)) + \gamma_k \|g_k - \nabla f(x_k)\| \cdot \|s_k - x_k\| + \frac{\gamma_k^2 L D^2}{2}.$$

Here, the error term $\|g_k - \nabla f(x_k)\|$ can be on the order of the bias constant B no matter how much sampling is performed, while the search direction satisfies $\|s_k - x_k\| \leq D$. Their product therefore introduces a positive term of order BD. On the other hand, the decrease in the objective comes from the term $f^* - f(x_k) < 0$. When $f(x_k)$ is already close to f^*, this quantity may be small relative to the BD term, depending on the geometry and curvature of the function f. As a result, the bias–diameter interaction can dominate the descent, explaining why the convergence neighborhood scales with both B and D.

The following examples provide further evidence that the achievable neighborhood size can depend on the diameter D for the standard Frank-Wolfe method. Assume that the feasible region is a thin and tall rectangle with coordinates $(0, 0), (0.01, 0), (0, R)$ and $(0.01, R)$ where $R > 0$ is assumed large and therefore close to the diameter D.

Example 3.2.1. Let the true function f and the bias $b(x)$ in the gradient estimators be

$$f(x_1, x_2) = 100x_1 + \sqrt{x_2^2 + 1} \text{ and } b(x) = \begin{pmatrix} 1 \\ -2 \end{pmatrix}.$$

(We assume no stochastic component to the gradient estimates.) The function f is convex. The bias is constant and we take $B = \sqrt{5}$. Moreover, $\|\nabla f(x)\| > B$ for

all feasible x. The gradient estimator is (deterministically)

$$g(x) = \nabla f(x) + b(x) = \begin{bmatrix} 101 \\ \frac{x_2}{\sqrt{x_2^2+1}} - 2 \end{bmatrix}.$$

At any feasible x, the target vertex obtained by solving the linear program in the standard Frank-Wolfe method will be $(0, R)$, because of the signs of the components of the vector g, which are positive and negative respectively. Thus, if we start at the vertex $(0.01, R)$ then the second component of the solutions will always be R, and we fail to make any progress towards the optimal solution at $(0, 0)$. Thus, the error $f(x_1, x_2) - f(0, 0)$ can be written as

$$f(x_1, x_2) - f(0, 0) = 100x_1 + \sqrt{x_2^2 + 1} - 1,$$

and starting from the vertex $(0.01, R)$ this error is bounded below by R, which is of the order of the diameter D.

Example 3.2.1 may suggest that, in the presence of bias, one cannot guarantee any progress at all, but in other examples, *some* progress is possible, though the error is still of the order BD, where B is the bias constant and D is the diameter of the feasible region.

Example 3.2.2. Let the true function f and the bias $b(x)$ in the gradient estimators be

$$f(x_1, x_2) = \frac{1}{2}(x_1 + 30)^2 + \frac{1}{100}(x_2 + 30)^2 \text{ and } b(x) = \begin{pmatrix} 1 \\ -20 \end{pmatrix}.$$

(We assume no stochastic component to the gradient estimates.) The function f is quadratic and hence strongly convex. The bias is constant and we can take $B = \|b(x)\| = \sqrt{401}$. Moreover $\|\nabla f(x)\| > B$ for all feasible x. The gradient estimator is (deterministically)

$$g(x) = \nabla f(x) + b(x) = \begin{bmatrix} x_1 + 31 \\ \frac{x_2+30}{50} - 20 \end{bmatrix}.$$

If the algorithm is initiated at the top-right corner $(0.01, R)$ and $R < 970$ then the target vertex obtained by solving the linear program in the standard Frank-Wolfe algorithm on *all* iterations gives the vertex $s = (0, R)$, for the same reason as in Example 3.2.1. However, the optimal solution is $(0, 0)$, so the error

$$\begin{aligned} f(x_1, x_2) - f(0, 0) &= \frac{x_1^2}{2} + 30x_1 + \frac{x_2^2}{100} + 0.6x_2 \\ &\geq \frac{x_2^2}{100} + 0.6x_2 \\ &= \frac{x_2}{2}\left(\frac{x_2}{50} + 1.2\right) \\ &= \frac{R}{2}\left(\frac{R}{50} + 1.2\right), \end{aligned}$$

which is of the same order as DB. On the other hand, if $R > 970$, then the sign of the second component of the gradient flips when $x_2 > 970$ and progress towards $(0, 0)$ is now possible when $x_2 > 970$. Thus, the standard Frank-Wolfe algorithm can make progress in the presence of bias in the gradient estimates, even though a function error of order smaller than D cannot be guaranteed.

For the away-step Frank–Wolfe method, the only feasible away direction is away from the top-right corner $(0.01, R)$, which coincides with the direction toward the

vertex $s = (0, R)$. Consequently, in this example, the away-step Frank–Wolfe method behaves similarly to the standard Frank–Wolfe method and therefore exhibits a comparable level of error. However, this error does not actually depend on the diameter D. From Corollaries 3.2.1 and 3.2.2, the dominant additional error term is $B^2/(\kappa^2\mu)$, where

$$\kappa = \min\left\{1, \frac{\varepsilon_g \Omega_{\mathcal{X}}}{2\sqrt{2}N}\right\},$$

and $\varepsilon_g \in (0, (4D)^{-1})$. The term $\varepsilon_g \Omega_{\mathcal{X}}$ is of order $\mathcal{O}(\Omega_{\mathcal{X}}/D)$ and is represented by the ratio χ defined in (2.12), which serves as a geometric "condition number"-like measure of the feasible region $\mathcal{X}$. This ratio depends only on the geometry of $\mathcal{X}$ and is invariant under scaling of the feasible region. In addition, by [72], the parameter N can be upper bounded by $d + 1$, where d is the dimension of the feasible region. Therefore, the term $B^2/(\kappa^2\mu)$ is independent of D and depends only on N, μ, and the bias level B^2. In the example above, $\mu = 1/50$, and because $R/50 + 1.2$ is in the order of B, then the error

$$\frac{R}{2}\left(\frac{R}{50} + 1.2\right) = \frac{1}{2} \cdot 50\frac{R}{50}\left(\frac{R}{50} + 1.2\right) \sim \mathcal{O}(B^2/\mu).$$

In particular, consider the more general objective function

$$f(x_1, x_2) = A_1(x_1 + c_1)^2 + A_2(x_2 + c_2)^2,$$

where $(A_1, A_2, c_1, c_2 > 0)$, together with the same feasible region as in the example above. The strong convexity constant is $\mu = 2\min\{A_1, A_2\}$. Suppose the bias is

given by

$$b(x) = \begin{bmatrix} b_1 \\ -b_2 \end{bmatrix},$$

where $|b_2|$ is significantly larger than $|b_1|$ (for example, in the previous example $(b_1 = 1)$ and $(b_2 = 20)$), and assume that

$$b_2 > 2A_2(x_2 + c_2), \qquad 0 \le x_2 \le R.$$

so that the estimated gradient's direction is similar to the example above. If the algorithm is initialized at the top-right corner $(0.01, R)$, then the iterates follow the same path as in the previous example and eventually approach the top-left corner $(0, R)$. The resulting optimality gap at $(0, R)$ is

$$\begin{aligned} f(0,R) - f(0,0) &= A_2\big((R + c_2)^2 - c_2^2\big) \\ &= A_2 R(R + 2c_2) \\ &= \frac{1}{2A_2} \cdot (A_2 R) \cdot \big(A_2(R + 2c_2)\big). \end{aligned}$$

Since $b_2 > 2A_2(x_2 + c_2)$ for all $0 \le x_2 \le R$, in particular, $b_2 > 2A_2(R + c_2)$, which implies

$$A_2 R \le b_2, \qquad A_2(R + 2c_2) \le b_2.$$

Therefore,

$$f(0,R) - f(0,0) \le \frac{b_2^2}{2A_2} \le \frac{b_2^2}{\mu} \le \frac{B^2}{\mu}.$$

Thus, the error remains of order $\mathcal{O}(B^2/\mu)$.

The presence of the diameter D in the attainable function accuracy in the standard

Frank–Wolfe method can be attributed to the fact that the search direction $s_k - x_k$ can have magnitude comparable to the diameter D, and this corrupts the estimated solutions substantially. By contrast, the away-step Frank–Wolfe method does not have this dependence on D. It selects directions based on active vertices, which reflect a local geometry of the current iterate. Moreover, its search direction is related to the pairwise direction $s_k - v_k$, the properties of which are closely related to the ratio between $\Omega_{\mathcal{X}}$ and D (χ), and N, as stated in Lemma 2.3.5. Finally, the size of the convergence neighborhood also depends on the strong convexity constant μ. A similar dependence appears in the work of [3] on gradient descent with biased gradients, where μ arises under the Polyak–Łojasiewicz (PL) condition; note that strong convexity implies the PL condition.

3.3 The Time to Reach Near-Optimality Has a Light Tail

This section establishes additional convergence properties of the standard and away-step Frank–Wolfe algorithms. In Section 2.5, we showed that the iteration complexities of the standard and away-step Frank–Wolfe algorithms are $\mathcal{O}(\varepsilon^{-2})$ and $\mathcal{O}(\varepsilon^{-1})$, respectively. Under Assumption 2.4.2 (sub-Gaussian), however, it can be further shown that the terminal time T_ε admits a finite moment generating function in a neighborhood of 0. All order of complexities in this section are similar to those in Section 2.5. To establish this result, we first prove a useful lemma that is used to bound the error in the case of a poor gradient approximation.

Lemma 3.3.1. *Let X be a non-negative random variable with the property that*

$$\mathbb{P}(X \geq s) \leq \kappa e^{-cs^2}$$

for some $c > 0$. Then for any positive t,

$$\mathbb{E}[e^{tX}] \leq Ce^{t^2/2c}$$

where $C = 1 + \kappa\sqrt{2\pi/e}$

Proof. By the definition of expectation, we have

$$\begin{aligned}
\mathbb{E}[e^{tX}] &= \int_0^\infty \mathbb{P}(e^{tX} \geq s)ds \\
&= \int_0^1 \mathbb{P}(e^{tX} \geq s)ds + \int_1^\infty \mathbb{P}(e^{tX} \geq s)ds \\
&= \int_0^1 1ds + \int_1^\infty \mathbb{P}(e^{tX} \geq s)ds \\
&= 1 + \int_1^\infty \mathbb{P}(e^{tX} \geq s)ds.
\end{aligned}$$

Substitute s with e^{ts},

$$\begin{aligned}
E[e^{tX}] &= 1 + \int_0^\infty te^{ts}\mathbb{P}(X \geq s)ds \\
&\leq 1 + \int_0^\infty t\kappa e^{ts-cs^2}ds \\
&= 1 + \int_0^\infty t\kappa e^{\frac{t^2}{4c}-c(s-\frac{t}{2c})^2}ds \\
&\leq 1 + te^{t^2/4c}\kappa\int_{-\infty}^\infty e^{-c(s-\frac{t}{2c})^2}ds \\
&\leq 1 + \kappa te^{t^2/4c}\sqrt{\frac{\pi}{c}}.
\end{aligned}$$

Consider the function g such that

$$g(t) = te^{-t^2/4c}, t \geq 0.$$

This function g is maximized at $t = \sqrt{2c}$, and then

$$g(t) \leq g(\sqrt{2c}) = \sqrt{2c/e} \text{ for all } t \geq 0.$$

Therefore,

$$\begin{aligned} E[e^{tX}] &\leq 1 + \kappa t e^{t^2/4c} \sqrt{\frac{\pi}{c}} \\ &= 1 + \kappa e^{t^2/2c} (t e^{-t^2/4c}) \sqrt{\frac{\pi}{c}} \\ &\leq 1 + \kappa e^{t^2/2c} \sqrt{\frac{2c}{e}} \sqrt{\frac{\pi}{c}} \\ &= 1 + \kappa \sqrt{\frac{2\pi}{e}} e^{t^2/2c} \\ &\leq \left(1 + \kappa \sqrt{\frac{2\pi}{e}}\right) e^{t^2/2c}. \end{aligned}$$

□

Consider the stochastic processes $\Phi^{(FW)}, \Phi^{(aFW)}$ obtained from random variables generated from the standard Frank-Wolfe algorithm and the away-step Frank-Wolfe algorithm respectively, where

$$\Phi_k^{(FW)} = \exp(f(X_k) - f^*), \tag{3.6}$$

and

$$\Phi_k^{(aFW)} = \exp(\nu(f(X_k) - f^*) + (1 - \nu)N_k). \tag{3.7}$$

Lemma 3.3.2. *There exists $p_g \in (0, 1)$, and $\nu \in (0, 1)$ such that, for standard Frank-Wolfe (under Assumptions 2.2.1 (Lipschitz gradient) and 2.2.3 (convexity)),*

and for away-step Frank-Wolfe (under Assumptions 2.2.1 (Lipschitz gradient) and 2.2.4 (strong convexity)), on the event $T_\varepsilon > k$,

$$\mathbb{E}\left[\frac{\Phi_{k+1}^{(FW)}}{\Phi_k^{(FW)}}\middle|\mathcal{F}_k\right] \leq e^{-\beta^{FW}(\varepsilon)\varepsilon/2},$$

and,

$$\mathbb{E}\left[\frac{\Phi_{k+1}^{(aFW)}}{\Phi_k^{(aFW)}}\middle|\mathcal{F}_k\right] \leq e^{-\frac{\beta^{aFW}{}_\varepsilon}{2(2+\beta^{aFW}{}_\varepsilon)}}.$$

Proof. The proof is similar to the proof from Theorem 2.5.1, but we consider the ratio Φ_{k+1}/Φ_k. Define Γ to be the maximum possible step size, which is bounded above by 1. By smoothness,

$$\begin{aligned} f(x_{k+1}) - f(x_k) &\leq \gamma_k \nabla f(x_k)^T d_k + \gamma_k^2 \frac{L\|d_k\|^2}{2} \\ &= \gamma_k g_k^T d_k + \gamma_k (\nabla f(x_k) - g_k)^T d_k + \gamma_k^2 \frac{L\|d_k\|^2}{2} \\ &\leq \gamma_k g_k^T d_k + \gamma_k D\|\nabla f(x_k) - g_k\| + \gamma_k^2 \frac{LD^2}{2} \\ &\leq \Gamma D\|\nabla f(x_k) - g_k\| + \frac{L\Gamma^2 D^2}{2}. \end{aligned}$$

By Lemma 2.4.3, there exists a positive constant c such that

$$\mathbb{P}(\|g(x) - \nabla f(x)\|) \leq 2de^{-ncs^2}$$

where n is the sample size of the gradient estimates $g(x)$. By Lemma 3.3.1, where

X is substituted with $\|\nabla f(x) - g(x)\|$, κ with $2d$, t with $2\Gamma D$, and c with $n_k c$,

$$\mathbb{E}[\exp(2\Gamma D\|\nabla f(x_k) - g(x_k)\|)] \leq Ce^{\frac{(2\Gamma D)^2}{2n_k c}} = Ce^{\frac{2\Gamma^2 D^2}{n_k c}}.$$

where $C = 1 + 2d\sqrt{2\pi/e}$. Let $\mathcal{G}$ be the event of good gradient approximation in iteration k, P_k be its conditional probability given $\mathcal{F}_k$ (which is bounded below by p_g by our sample-size assumption) and $\mathcal{G}'$ be its complement, like Theorem 2.5.1. By Cauchy-Schwarz Inequality, and the inequality above,

$$\begin{aligned}
&\mathbb{E}[\exp(f(X_{k+1}) - f(X_k))\mathbb{1}_{\mathcal{G}'}|\mathcal{F}_k] \\
&\leq \mathbb{E}\left[\exp\left(\Gamma D\|\nabla f(x_k) - g_k\| + \frac{L\Gamma^2 D^2}{2}\right)\mathbb{1}_{\mathcal{G}'}|\mathcal{F}_k\right] \\
&\leq \exp\left(\frac{L\Gamma^2 D^2}{2}\right)\mathbb{E}\left[\exp\left(\Gamma D\|\nabla f(x_k) - g_k\|\right)\mathbb{1}_{\mathcal{G}'}|\mathcal{F}_k\right] \\
&\leq \exp\left(\frac{L\Gamma^2 D^2}{2}\right)\sqrt{\mathbb{E}\left[\exp\left(2\Gamma D\|\nabla f(x_k) - g_k\|\right)|\mathcal{F}_k\right]}\sqrt{1 - P_k} \\
&\leq C\exp\left(\frac{L\Gamma^2 D^2}{2}\right)\exp\left(\frac{2\Gamma^2 D^2}{n_k c}\right)\sqrt{1 - P_k}
\end{aligned}$$

where n_k is the sample size in iteration k and $C = 1 + 2d\sqrt{\frac{\pi}{c}}$. Then,

$$\begin{aligned}
\mathbb{E}[\exp(f(X_{k+1}) - f(X_k))|\mathcal{F}_k] \leq \mathbb{E}[\exp(f(X_{k+1}) - f(X_k))\mathbb{1}_{\mathcal{G}}|\mathcal{F}_k] + \\
C\exp\left(\frac{L\Gamma^2 D^2}{2}\right)\exp\left(\frac{2\Gamma^2 D^2}{n_k c}\right)\sqrt{1 - P_k}. \quad (3.8)
\end{aligned}$$

Part I: Standard Frank-Wolfe From Lemma 2.3.1, for $k < T_\varepsilon$,

$$\frac{\Phi_{k+1}^{(FW)}}{\Phi_k^{(FW)}} \leq \begin{cases} \exp(-\beta^{FW}\varepsilon) & \text{if we have a good gradient estimate, and} \\ \exp(f(X_{k+1}) - f(X_k)) & \text{if we have a bad gradient estimate.} \end{cases}$$

Taking conditional expectations, on the event $T_\varepsilon > k$, using (3.8),

$$\begin{aligned}\mathbb{E}\left[\frac{\Phi_{k+1}^{(FW)}}{\Phi_k^{(FW)}}\middle|\mathcal{F}_k\right] &\leq \exp(-\beta^{FW}(\varepsilon)\varepsilon)p_g + C\exp\left(\tfrac{L\Gamma^2D^2}{2} + \tfrac{2\Gamma^2D^2}{n_k c}\right)\sqrt{1-p_g} \\ &\leq \exp(-\beta^{FW}(\varepsilon)\varepsilon)p_g + C\exp\left(\tfrac{L\Gamma^2D^2}{2} + \tfrac{2\Gamma^2D^2}{c}\right)\sqrt{1-p_g},\end{aligned}$$

where p_g is the lower bound of the probability of a good gradient approximation. Pick the probability p_g such that

$$1 - p_g = \frac{e^{-\beta^{FW}(\varepsilon)\varepsilon}(1 - e^{-\beta^{FW}(\varepsilon)\varepsilon/2})^2}{C^2 e^{L\Gamma^2D^2 + \frac{4\Gamma^2D^2}{c}}} \tag{3.9}$$

so that

$$\begin{aligned}\mathbb{E}\left[\frac{\Phi_{k+1}^{(FW)}}{\Phi_k^{(FW)}}\middle|\mathcal{F}_k\right] &\leq \exp(-\beta^{FW}(\varepsilon)\varepsilon)p_g + C\exp\left(\frac{L\Gamma^2D^2}{2} + \frac{2\Gamma^2D^2}{c}\right)\sqrt{1-p_g} \\ &\leq \exp(-\beta^{FW}(\varepsilon)\varepsilon)p_g + \exp\left(\frac{-\beta^{FW}(\varepsilon)\varepsilon}{2}\right)\cdot\left(1 - \exp\left(\frac{-\beta^{FW}(\varepsilon)\varepsilon}{2}\right)\right) \\ &= e^{-\beta^{FW}(\varepsilon)\varepsilon/2} + (p_g - 1)e^{-\beta^{FW}(\varepsilon)\varepsilon} \\ &\leq e^{-\beta^{FW}(\varepsilon)\varepsilon/2}.\end{aligned}$$

Thus on the event $T_\varepsilon > k$,

$$\mathbb{E}\left[\frac{\Phi_{k+1}^{(FW)}}{\Phi_k^{(FW)}}\middle|\mathcal{F}_k\right] \leq e^{-\beta^{FW}(\varepsilon)\varepsilon/2}.$$

Part II: Away-step Frank-Wolfe By Algorithm 2, the number of active vertices

N_k evolves according to one of the scenarios below.

$$N_{k+1} - N_k = \begin{cases} 1 & \text{if no vertex drop and a new active vertex added,} \\ 1 - N_k & \text{if no vertex drop and move to a vertex, and} \\ -1 & \text{if vertex drop.} \end{cases}$$

From Lemma 2.3.7, we have for $k < T_\varepsilon$,

$$\frac{\Phi_{k+1}^{(aFW)}}{\Phi_k^{(aFW)}} \leq \begin{cases} \exp(\nu(-\beta^{aFW}\varepsilon) + (1-\nu)) & \text{if good gradient estimate} \\ & \text{and no vertex drop,} \\ \exp(-(1-\nu)) & \text{if good gradient estimate} \\ & \text{and vertex drop, and} \\ \exp(\nu(f(X_{k+1}) - f(X_k)) + (1-\nu)) & \text{if bad gradient estimate.} \end{cases}$$

As in Theorem 2.5.1, let $\mathscr{V}$ be the event of vertex drop at Iteration k. Then,

$$\begin{aligned} & \mathbb{E}\left[\frac{\Phi_{k+1}^{(aFW)}}{\Phi_k^{(aFW)}} \mathbb{1}_{\mathcal{G}} \middle| \mathcal{F}_k\right] \\ &= \mathbb{E}\left[\exp(\nu(f(X_{k+1}) - f(X_k)) + (1-\nu)(N_{k+1} - N_k))\mathbb{1}_{\mathcal{G}} \middle| \mathcal{F}_k\right] \\ &= \mathbb{E}\left[\exp(\nu(f(X_{k+1}) - f(X_k)) + (1-\nu)(N_{k+1} - N_k))\mathbb{1}_{\mathcal{G}}(\mathbb{1}_{\mathscr{V}} + \mathbb{1}_{\mathscr{V}'}) \middle| \mathcal{F}_k\right] \\ &\leq \mathbb{E}\left[\exp(-(1-\nu))\mathbb{1}_{\mathcal{G}}\mathbb{1}_{\mathscr{V}} + \exp(-\nu\beta^{aFW}\varepsilon + 1 - \nu)\mathbb{1}_{\mathcal{G}}\mathbb{1}_{\mathscr{V}'} \middle| \mathcal{F}_k\right] \\ &\leq \exp(-(1-\nu))p_g\mathbb{P}(\mathscr{V}|\mathcal{F}_k, \mathcal{G}) + \exp(-\nu\beta^{aFW}\varepsilon + 1 - \nu)p_g\mathbb{P}(\mathscr{V}'|\mathcal{F}_k, \mathcal{G}). \end{aligned}$$

Also, using (3.8),

$$\begin{aligned}
&\mathbb{E}\left[\frac{\Phi_{k+1}^{(aFW)}}{\Phi_k^{(aFW)}}\mathbb{1}_{\mathcal{G}'}\middle|\mathcal{F}_k\right] \\
&= \mathbb{E}\left[\exp(\nu(f(X_{k+1})-f(X_k))+(1-\nu)(N_{k+1}-N_k))\mathbb{1}_{\mathcal{G}'}|\mathcal{F}_k\right] \\
&\leq \mathbb{E}\left[\exp(\nu(f(X_{k+1})-f(X_k)))\exp(1-\nu)\mathbb{1}_{\mathcal{G}'}|\mathcal{F}_k\right] \\
&= \exp(1-\nu)\mathbb{E}\left[\exp(\nu(f(X_{k+1})-f(X_k)))\mathbb{1}_{\mathcal{G}'}|\mathcal{F}_k\right] \\
&\leq C\exp(1-\nu)\exp\left(\frac{\nu L\Gamma^2D^2}{2}+\frac{2\nu^2\Gamma^2D^2}{n_kc}\right)\sqrt{1-P_k}.
\end{aligned}$$

Then for $k < T_\varepsilon$,

$$\begin{aligned}
\mathbb{E}\left[\frac{\Phi_{k+1}^{(aFW)}}{\Phi_k^{(aFW)}}\middle|\mathcal{F}_k\right] &\leq \max\{\exp(-(1-\nu)),\exp(-\nu\beta^{aFW}\varepsilon+1-\nu)\}p_g \\
&\quad + C\exp(1-\nu)\exp\left(\frac{\nu L\Gamma^2D^2}{2}+\frac{2\nu^2\Gamma^2D^2}{n_kc}\right)\sqrt{1-p_g}.
\end{aligned}$$

Pick $\nu = \frac{2}{2+\beta^{aFW}\varepsilon} < 1$ so that

$$-(1-\nu) = -\nu\beta^{aFW}\varepsilon+1-\nu = -\frac{\beta^{aFW}\varepsilon}{2+\beta^{aFW}\varepsilon}.$$

Thus,

$$\begin{aligned}
&\mathbb{E}\left[\frac{\Phi_{k+1}^{(aFW)}}{\Phi_k^{(aFW)}}\middle|\mathcal{F}_k\right] \\
&\leq \exp(-(1-\nu))p_g + C\exp(1-\nu)\exp\left(\frac{\nu L\Gamma^2D^2}{2}+\frac{2\nu^2\Gamma^2D^2}{n_kc}\right)\sqrt{1-p_g} \\
&\leq \exp(-(1-\nu))p_g + C\exp(1-\nu)\exp\left(\frac{L\Gamma^2D^2}{2}+\frac{2\Gamma^2D^2}{c}\right)\sqrt{1-p_g} \\
&= \exp\left(-\frac{\beta^{aFW}\varepsilon}{2+\beta^{aFW}\varepsilon}\right)p_g + C\exp\left(\frac{\beta^{aFW}\varepsilon}{2+\beta^{aFW}\varepsilon}+\frac{L\Gamma^2D^2}{2}+\frac{2\Gamma^2D^2}{c}\right)\sqrt{1-p_g}.
\end{aligned}$$

Pick the probability p_g such that

$$1 - p_g = \frac{e^{3(\nu-1)}(1 - e^{(\nu-1)/2})^2}{C^2 e^{L\Gamma^2 D^2 + 4\Gamma^2 D^2/c}} \tag{3.10}$$

so that

$$\begin{aligned}
&\mathbb{E}\left[\frac{\Phi_{k+1}^{(aFW)}}{\Phi_k^{(aFW)}}\middle|\mathcal{F}_k\right] \\
&\leq \exp(-(1-\nu))p_g + \exp(1-\nu)\exp\left(\frac{3}{2}(\nu-1)\right)\left(1 - \exp\left(-\frac{1-\nu}{2}\right)\right) \\
&= \exp(-(1-\nu))p_g + \exp\left(-\frac{1-\nu}{2}\right)\left(1 - \exp\left(-\frac{1-\nu}{2}\right)\right) \\
&= \exp\left(-\frac{1-\nu}{2}\right) + \exp(-(1-\nu))(p_g - 1) \\
&\leq \exp\left(-\frac{1-\nu}{2}\right) = \exp\left(-\frac{\beta^{aFW}\varepsilon}{2(2+\beta^{aFW}\varepsilon)}\right).
\end{aligned}$$

□

Theorem 3.3.1. *Suppose the initial solution $X_0 = x_0$ is deterministic with $f(x_0) - f^* > \varepsilon$. Let $T_\varepsilon = \inf\{k : f(X_k) - f^* \leq \varepsilon\}$, and the gradient estimation satisfies Assumption 2.4.2 (sub-Gaussian). Then the standard and away-step Frank-Wolfe algorithms both have the same iteration complexities as Theorem 2.5.2. In addition, for both algorithms the stopping time T_ε has a finite moment generating function in a neighborhood of 0.*

Proof. Consider the stochastic process $\Phi = (\Phi_k : k \geq 0)$, adapted to $(\mathcal{F}_k : k \geq 0)$ that satisfies the expected ratio decrease as in Lemma 3.3.2 and define a stochastic process $(M_k : k \geq 0)$ where $M_k = e^{(k \wedge T_\varepsilon)\delta}\Phi_{k \wedge T_\varepsilon}$. Then, on the event $k < T_\varepsilon$, for $\delta = \beta^{FW}(\varepsilon)\varepsilon/2$ and $\beta^{aFW}\varepsilon/(2(2+\beta^{aFW}))$ for the standard or away-step stochastic

process respectively,

$$\begin{aligned}\mathbb{E}[M_{k+1}|\mathcal{F}_k] &= e^{(k+1)\delta}\mathbb{E}[\Phi_{k+1}|\mathcal{F}_k] \\ &\leq e^{(k+1)\delta}e^{-\delta}\Phi_k \\ &= e^{\delta k}\Phi_k = M_k.\end{aligned}$$

On the event $T_\varepsilon \leq k$, $M_{k+1} = M_k$. So $(M_k : k \geq 0)$ is a non-negative supermartingale. By the optional stopping theorem, and since $X_0 = x_0$ is deterministic with $f(x_0) > f^* + \varepsilon$,

$$\mathbb{E}[M_{T_\varepsilon}] \leq \mathbb{E}[M_0] = \Phi_0.$$

So, $\mathbb{E}[e^{\delta T_\varepsilon}\Phi_{T_\varepsilon}] \leq \Phi_0$, and, since $\Phi_k \geq 1$ for all non-negative integers k, $\mathbb{E}[e^{\delta T_\varepsilon}] \leq \Phi_0 < \infty$, establishing the finiteness of the moment generating function in a neighborhood of 0. By concavity of the logarithm function,

$$\mathbb{E}[\delta T_\varepsilon] \leq \log \mathbb{E}[e^{\delta T_\varepsilon}] \leq \mathbb{E}[e^{\delta T_\varepsilon}\Phi_{T_\varepsilon}] = \log \mathbb{E}[M_{T_\varepsilon}] \leq \log \Phi_0.$$

Therefore, for the standard Frank-Wolfe algorithm, using Lemma 2.3.1,

$$\mathbb{E}[T_\varepsilon] \leq \frac{\log \Phi_0^{(FW)}}{\beta^{FW}(\varepsilon)\varepsilon/2} = \frac{2\log \Phi_0^{(FW)}}{\beta^{FW}(\varepsilon)\varepsilon} = 2(f(x_0)-f^*)\max\left\{\frac{8LD^2}{\varepsilon^2}, \frac{4}{\varepsilon}\right\} \sim \mathcal{O}\left(\frac{f(x_0)-f^*}{\varepsilon^2}\right).$$

Similarly, for the away-step Frank-Wolfe algorithm, $\log \Phi_0^{(aFW)} = \nu(f(x_0) - f(x^*)) + (1-\nu)N$ where N is the initial number of active vertices. For our choice of ν,

$$1 - \nu = 1 - \frac{2}{2 + \beta^{aFW}\varepsilon} = \frac{\beta^{aFW}\varepsilon}{2 + \beta^{aFW}\varepsilon} \sim \mathcal{O}(\varepsilon),$$

so that

$$\begin{aligned}\mathbb{E}[T_\varepsilon] &\leq \frac{\log \Phi_0^{(aFW)}}{\beta^{aFW}\varepsilon/(2(2+\beta^{aFW}\varepsilon))} \\ &= (\log \Phi_0^{(aFW)}) \left(2 + \frac{4}{\beta^{aFW}\varepsilon}\right) \\ &\sim \mathcal{O}\left(\frac{f(x_0) - f^* + N}{\varepsilon}\right) \\ &= \mathcal{O}\left(\frac{f(x_0) - f^*}{\varepsilon}\right).\end{aligned}$$

Here, β^{aFW} as stated in Lemma 2.3.7 does not depend on ε since

$$\beta^{aFW} = \min\left\{\frac{1/2 - 2\varepsilon_g D}{1 + 2\varepsilon_g D}, \left(\frac{\Omega_{\mathcal{X}}}{N}\right)^2 \frac{\mu(1/2 - 2\varepsilon_g D)}{8LD^2(2\varepsilon_g D + 1)^2}\right\}$$

□

Theorem 3.3.2. *The sample sizes in each iteration that are required to achieve a good gradient estimate under the Assumption 2.4.2 (sub-Gaussian) using Lyapunov functions defined in* (3.6) *and* (3.7) *for the standard and away-step Frank-Wolfe methods respectively are as follows.*

1. *For the Standard Frank-Wolfe method under Assumptions 2.2.1 (Lipschitz gradient) and 2.2.3 (Convexity) the needed sample size is* $\mathcal{O}(\varepsilon^{-2}\log(\varepsilon^{-1}))$.
2. *For the away-step Frank-Wolfe method under Assumptions 2.2.1 (Lipschitz gradient) and 2.2.4 (strong convexity) the needed sample size is* $\mathcal{O}(\varepsilon^{-1}\log(\varepsilon^{-1}))$.

Proof. Take $s = \varepsilon/(4D)$ in Lemma 2.4.3 and select the sample size n so large such

that $2de^{-nc\varepsilon^2/(16D^2)}$ is a lower bound of $1-p_g$. Thus,

$$n \geq \frac{16D^2}{c\varepsilon^2}\log\left(\frac{2d}{1-p_g}\right).$$

Using probability p_g from (3.9), and the fact that for positive x, $1-x \leq e^{-x}$,

$$\begin{aligned}
n &\geq \frac{16D^2}{c\varepsilon^2}\left(\log(2d)+\log\frac{C^2e^{L\Gamma^2D^2+4\Gamma^2D^2/c}e^{\beta^{FW}(\varepsilon)\varepsilon}}{(1-e^{-\beta^{FW}(\varepsilon)\varepsilon/2})^2}\right)\\
&\geq \frac{16D^2}{c\varepsilon^2}\left(\log(2d)+\log\frac{C^2e^{L\Gamma^2D^2+4\Gamma^2D^2/c}e^{\beta^{FW}(\varepsilon)\varepsilon}}{(\beta^{FW}(\varepsilon)\varepsilon/2)^2}\right)\\
&= \frac{16D^2}{c\varepsilon^2}\left(\log(2d)+L\Gamma^2D^2+\frac{4\Gamma^2D^2}{c}+2\log C+\beta^{FW}(\varepsilon)\varepsilon+2\log\frac{2}{\beta^{FW}(\varepsilon)\varepsilon}\right)\\
&\sim \mathcal{O}\left(\frac{D^2}{\varepsilon^2}\log\frac{2}{\min\{\varepsilon/4,\varepsilon^2/8LD^2\}}\right)\\
&\sim \mathcal{O}\left(\frac{D^2}{\varepsilon^2}\log\frac{LD}{\varepsilon}\right).
\end{aligned}$$

which is of order $\mathcal{O}(\varepsilon^{-2}\log\varepsilon^{-1})$. For the away-step Frank-Wolfe method, in view of (2.10), take $s^2 = c_1\varepsilon$, where

$$c_1 = \min\left\{1, \frac{\varepsilon_g^2\Omega_{\mathcal{X}}^2}{8N^2}\right\}\mu.$$

Then, from Lemma 2.4.3, it suffices to take n so large that $2d\exp(-ncc_1\varepsilon) \leq 1-p_g$. Then,

$$n \geq \frac{1}{cc_1\varepsilon}\log\left(\frac{2d}{1-p_g}\right).$$

Using the probability p_g from (3.10) and the fact that for any positive x, $1-x \leq$

e^{-x}, we require a sample size

$$
\begin{aligned}
n &\geq \frac{1}{cc_1\varepsilon}\log\left(\frac{2d}{1-p_g}\right) \\
&\geq \frac{1}{cc_1\varepsilon}\left(\log(2d)+\log\frac{C^2e^{L\Gamma^2D^2+4\Gamma^2D^2/c}e^{3(1-\nu)}}{(1-e^{(\nu-1)/2})^2}\right) \\
&\geq \frac{1}{cc_1\varepsilon}\left(\log(2d)+\log\frac{4C^2e^{L\Gamma^2D^2+4\Gamma^2D^2/c+3(1-\nu)}}{(1-\nu)^2}\right) \\
&\geq \frac{1}{cc_1\varepsilon}\left(\log(2d)+2\log 4C+L\Gamma^2D^2+\frac{4\Gamma^2D^2}{c}+3(1-\nu)+2\log\frac{1}{1-\nu}\right) \\
&= \frac{1}{cc_1\varepsilon}\left(\log(2d)+2\log 4C+L\Gamma^2D^2+\frac{4\Gamma^2D^2}{c}+\frac{3\beta^{aFW}\varepsilon}{2+\beta^{aFW}\varepsilon}+2\log\frac{2+\beta^{aFW}\varepsilon}{\beta^{aFW}\varepsilon}\right) \\
&\sim \mathcal{O}\left(\frac{1}{\varepsilon}\log\frac{1}{\beta^{aFW}\varepsilon}\right) \\
&\sim \mathcal{O}\left(\frac{1}{\varepsilon}\log\frac{NLD}{\mu\Omega_{\mathcal{X}}\varepsilon}\right) \\
&\sim \mathcal{O}\left(\varepsilon^{-1}\log\varepsilon^{-1}\right)
\end{aligned}
$$

which is of order $\mathcal{O}(\varepsilon^{-1}\log\varepsilon^{-1})$ □

Using the Lyapunov functions defined in (3.6) and (3.7), we thus obtain the same iteration and sample complexity results as in Chapter 2. In addition, under Assumption 2.4.2 (sub-Gaussian) on sub-Gaussian gradient estimates, we establish that the stopping time T_ε admits a finite moment generating function. The convergence analysis in Lemma 3.3.2 relies on Lemma 3.3.1, which crucially depends on the sub-Gaussian assumption. In contrast, the bounded variance assumption in 2.4.1 (bounded variance) is insufficient for this result, as it only controls the first and second moments of $\|\nabla f(x)-g(x)\|$, whereas Assumption 2.4.2 (sub-Gaussian) provides control over all moments.

If we instead assume only 2.4.1 (bounded variance), then in the presence of a poor gradient estimate, the difference $f(X_{k+1}) - f(X_k)$ can be as large as $2\max_{x\in\mathcal{X}} |f(x)|$. This corresponds to a worst-case scenario in which the iterate moves to a point with nearly maximal function value, potentially exceeding the initial value $f(x_0)$. In this sense, a bad gradient estimate at iteration k effectively causes a "restart" of the iterate x_k. To still guarantee convergence with a finite moment generating function of the terminal time T_ε, as in Theorem 3.3.1, one would need either a sample size that scales exponentially with $\max_{x\in\mathcal{X}} |f(x)|$, or a sufficiently long sequence of consecutive "good" gradient estimates.

Under Assumption 2.4.2 (sub-Gaussian), however, we can directly bound $\mathbb{E}[\exp(f(X_{k+1}) - f(X_k))]$, which controls the increase of the Lyapunov function via Lemma 3.3.1. This enables us to show that the Lyapunov functions for both the standard and away-step Frank–Wolfe algorithms decrease in expectation. In contrast, under only a bounded variance assumption, achieving such control would again require either consecutive good iterations or prohibitively large sample sizes scaling exponentially with $\max_{x\in\mathcal{X}} |f(x)|$.

3.4 Improved Complexity of the standard Frank-Wolfe algorithm with an Interior Optimum

This section presents the convergence analysis of the stochastic standard Frank–Wolfe Algorithm 1 under the additional assumption that the global optimal solution lies strictly in the interior of the bounded feasible region $\mathcal{X}$. In Section 2.3, several preparatory results for the stochastic standard Frank–Wolfe algorithm were established under the good gradient approximation condition introduced in Defi-

nition 2.3.1. In particular, Lemma 2.3.1 shows the improvement in the objective function value under convexity and L-smoothness assumptions. However, unlike projection-based methods, strong convexity alone is generally insufficient to improve the convergence rate of the standard Frank–Wolfe algorithm due to the possible poor alignment between the Frank–Wolfe search direction and the negative gradient.

This section demonstrates that the additional assumption of an interior optimal solution, combined with strong convexity, leads to significantly improved convergence guarantees. In particular, the iteration complexity and sample complexity can be improved to the same order as those of the Away-step Frank–Wolfe algorithm. The analysis highlights the important role of the interior optimal solution in strengthening the alignment between the search direction and the negative gradient. Moreover, it illustrates how the interaction between the step size rule and the good gradient approximation condition influences the resulting objective function improvement rate.

Assumption 3.4.1. *The global optimum x^* lies in the strict interior of the feasible region $\mathcal{X}$.*

For the standard Frank–Wolfe algorithm, the *curvature constant* of a convex function f is defined as

$$C_f = \sup_{\substack{x,s\in\mathcal{X} \\ \gamma\in[0,1] \\ y=x+\gamma(s-x)}} \frac{2}{\gamma^2}(f(y) - f(x) - \nabla f(x)^T(y-x))$$

According to [47], this quantity is closely related to the diameter D of the feasible region and the Lipschitz constant L of the gradient, satisfying the bound $C_f \leq$

LD^2. Lemma 7 from [47] shows the proof of this inequality using the smoothness property and the definition of C_f. By $L-$smoothness and convexity,

$$f(y) - f(x) - \nabla f(x)^T(y-x) \leq \frac{L}{2}\|y-x\|^2,$$

for any x, y. By definition of C_f,

$$C_f \leq \sup_{\substack{x,s\in\mathcal{X}\\ \gamma\in[0,1]\\ y=x+\gamma(s-x)}} \frac{2}{\gamma^2}\cdot\frac{L}{2}\|y-x\|^2 = \sup_{\substack{x,s\in\mathcal{X}\\ \gamma\in[0,1]}} \frac{L}{\gamma^2}\cdot\gamma^2\|s-x\|^2 \leq LD^2.$$

In the deterministic convergence analysis of the standard Frank–Wolfe algorithm presented in [53], the curvature constant C_f naturally appears in place of the term LD^2. When f is linear, we have $C_f = 0$. Furthermore, under the additional assumptions that the objective function is strongly convex and that the global optimal solution lies in the interior of the feasible region, [53] introduces a refined curvature-related constant, analogous to C_f, defined as follows.

Definition 3.4.1. *The objective function f is strongly convex, and its global optimum lies strictly in the interior of the feasible region $\mathcal{X}$. Define the strong convexity constant μ_f^{FW} by*

$$\mu_f^{FW} = \inf_{\substack{x\in\mathcal{X}\setminus\{x^*\}\\ \bar{s}=\bar{s}(x,x^*,\mathcal{X}),\\ \gamma\in(0,1],\\ y=x+\gamma(\bar{s}-x)}} \frac{2}{\gamma^2}(f(y)-f(x)-\nabla f(x)^T(y-x))$$

where $\bar{s}(x,x^*,\mathcal{X})$ denotes the intersection point between the boundary of $\mathcal{X}$ and the ray originating from x and passing through x^*.

Because the optimal solution x^* lies strictly in the interior of $\mathcal{X}$, it follows that

$\mu_f^{FW} > 0$. In contrast, when x^* lies on the boundary of the feasible region, the quantity μ_f^{FW} can become zero. Indeed, consider the case where $\gamma = 1$ and $\bar{s} = x^*$, so that $y = x^*$. By the strong convexity of f,

$$f(x^*) \geq f(x) + \nabla f(x)^T(x^* - x) + \frac{\mu}{2}\|x - x^*\|^2.$$

As the iterate x approaches x^*, the term $\|x - x^*\|^2$ can become arbitrarily small. Consequently, the corresponding infimum in the definition of μ_f^{FW} may approach zero, implying that $\mu_f^{FW} = 0$ when the optimal solution lies on the boundary of the feasible region. According to [53], the constant μ_f^{FW} is also related to several geometric and smoothness quantities, including the curvature constant C_f, the diameter D of the feasible region, and the interior distance $\delta_{x^*,\mathcal{X}} = \inf_{s \in \partial\mathcal{X}} \|s - x^*\| > 0$ where $\partial\mathcal{X}$ denotes the boundary of the feasible region $\mathcal{X}$. In particular, these quantities satisfy the relationship

$$\mu_f^{FW} \leq C_f \quad \text{and} \quad \mu_f^{FW} \geq \mu\delta_{x^*,\mathcal{X}}^2.$$

For the standard Frank-Wolfe algorithm, even under the assumption of strong convexity, one cannot in general improve either the sample size required to satisfy the gradient approximation condition in Definition 2.3.1 or the improvement rate established in Lemma 2.3.1. This limitation arises because the alignment between the negative gradient and the Frank–Wolfe search direction $s - x$ can become arbitrarily poor and, in extreme cases, nearly orthogonal. As a result, strong convexity alone is insufficient to translate curvature information into a substantially improved descent direction or faster convergence rate. Figure 3.1 illustrates the characteristic zig-zagging behavior that occurs when the optimal solution x^* lies

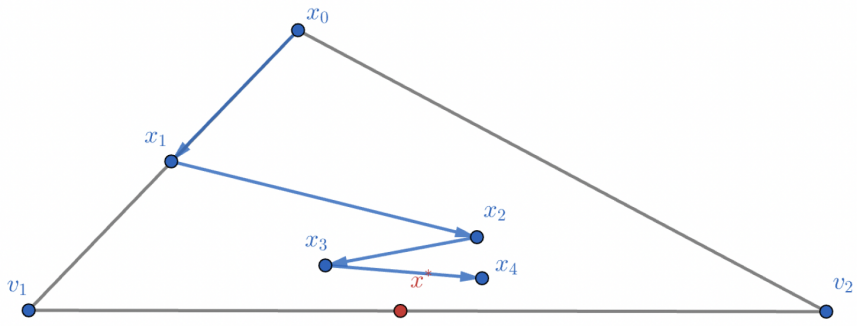


Figure 3.1: An example of zig-zagging behavior of standard Frank-Wolfe.

on the boundary of the feasible region. As the iterates approach x^*, the Frank-Wolfe search direction alternates between extreme points (for example, v_1 and v_2), causing the alignment between the negative gradient and the search direction $s - x$ to deteriorate. In particular, the quantity $-\nabla f(x)^T(s - x)$ may become arbitrarily small, and thus the angle between $-\nabla f(x)$ and $s - x$ may approach orthogonality. However, this term is the main contributor to the objective decrease in Lemma 2.3.1. Consequently, even under strong convexity, one can only guarantee modest improvement in the objective value. To see this more clearly, consider the deterministic version of the standard Frank–Wolfe Algorithm 1 with search direction $d_k = s_k - x_k$. By strong convexity,

$$\begin{aligned} -\nabla f(x_k)^T d_k &= -\nabla f(x_k)^T(s_k - x_k) \\ &\geq -\nabla f(x_k)^T(x^* - x_k) \\ &\geq f(x_k) - f(x^*) + \frac{\mu}{2}\|x^* - x\|^2, \end{aligned}$$

When the algorithm has not yet terminated, the optimality gap satisfies $f(x_k) - f(x^*) > \varepsilon$. However, as x_k approaches x^*, the term $\|x^* - x_k\|^2$ may become arbitrarily small, implying that the lower bound above can still be as small as

$\mathcal{O}(\varepsilon)$. Thus, strong convexity alone does not prevent the alignment term from becoming weak near the solution. In contrast, when the optimal solution x^* lies strictly in the interior of $\mathcal{X}$, a fundamentally different behavior emerges. In the deterministic setting, the definition of the Frank-Wolfe direction s, together with the positive constant μ_f^{FW}, ensures a significantly stronger alignment between the negative gradient and the search direction. By the proof of Theorem 3 in [53],

$$-\nabla f(x_k)^T d_k = -\nabla f(x_k)^T (s_k - x) \geq \sqrt{2\mu_f^{FW}(f(x_k) - f(x^*))}.$$

Under the interior optimality assumption together with strong convexity, the alignment term $-\nabla f(x)^T(s - x)$ is therefore bounded below by a substantially larger positive quantity as the iterates approach optimality. In particular, near termination, the lower bound in the standard strongly convex case remains of order $\mathcal{O}(\varepsilon)$, whereas in the interior optimum case the lower bound becomes $\mathcal{O}(\sqrt{\varepsilon})$, which is significantly larger for small ε. Moreover, this is of the same order as the lower bound obtained for away-step Frank-Wolfe. By Theorem 3.1 in [7], if d_k denotes the away-step Frank-Wolfe search direction, then

$$-\nabla f(x_k)^T d_k \geq \sqrt{\frac{\Omega_{\mathcal{X}}}{\kappa N}\varepsilon},$$

for a positive constant κ. Since the smoothness property gives

$$f(x_{k+1}) \leq f(x_k) + \nabla f(x_k)^T d_k + \frac{L\gamma_k^2 \|d_k\|^2}{2},$$

the term $\nabla f(x_k)^T d_k$ provides the dominant contribution to the objective decrease. Consequently, both away-step Frank-Wolfe and standard Frank–Wolfe with inte-

rior optimality and strong convexity achieve substantially larger decreases in the objective function compared with the standard boundary-optimum case. The remainder of this section develops the convergence analysis under Assumption 3.4.1 (interior optimum), leveraging the constant μ_f^{FW} to quantify this improved geometric alignment. In particular, we show that both the iteration complexity and the sample size complexity achieve the same order as those of the away-step Frank-Wolfe algorithm. In this section, under Assumption 3.4.1 (interior optimum), the constant μ_f^{FW} is strictly positive, which guarantees a positive lower bound on the alignment term $-\nabla f(x)^T(s-x)$ in the deterministic setting. This improved geometric property leads to a stronger decrease in the objective function compared to the general boundary-optimal case. To establish the resulting convergence guarantees, we apply Algorithm 1 with the modified step size policy described below.

Algorithm 3 Interior Standard Frank-Wolfe

Initialize a feasible solution $x_0 \in \mathcal{X}$
for $k = 0, 1, 2, ...$ **do**
  Compute an estimate, g_k, of the gradient $\nabla f(x_k)$
  Compute $s_k = \arg\min_{s \in \mathcal{X}} g_k^T s$ by solving a linear program to obtain a vertex solution
  $d_k = s_k - x_k$
  $\gamma_k = \min\left\{1, \sqrt{2\mu_f^{FW}\varepsilon}/(2LD^2)\right\}$
  $x_{k+1} = x_k + \gamma_k d_k$
end for

Next, we present the convergence analysis of the standard Frank–Wolfe algorithm under Assumption 3.4.1 (interior optimum). The analysis relies on the constant μ_f^{FW} introduced in Definition 3.4.1, which captures the improved geometric alignment induced by the interior optimum condition. We begin by introducing a good gradient approximation condition for this setting, analogous to the definition given in Definition 2.3.1.

Definition 3.4.2. *We say that the vector g_k is a* good gradient approximation *at*

iteration k if

$$\|g_k - \nabla f(x_k)\| \leq \frac{\sqrt{2\mu_f^{FW}\varepsilon}}{4D}.$$

Lemma 3.4.1. *Under Assumption 2.2.4 (strong convexity) and 3.4.1 (interior optimum), if we have a good gradient approximation at iteration k, then*

$$f(x_{k+1}) - f(x_k) \leq -\beta^{iFW}\varepsilon,$$

where $\beta^{iFW} = \min\left\{\frac{\mu_f^{FW}}{4LD^2}, \frac{\sqrt{2\mu_f^{FW}}}{4\sqrt{\varepsilon}}\right\}$.

Proof. In the context of Definition 3.4.1 and the definition of s_k in Algorithm 3, consider the current iterate x_k, γ where we substitute $y = x_k + \gamma(\overline{s} - x_k) = x^*$. Then,

$$\begin{aligned}
\frac{\gamma^2}{2}\mu_f^{FW} &\leq f^* - f(x_k) - \gamma\nabla f(x_k)^T(\overline{s} - x_k) \\
&= f^* - f(x_k) - \gamma g_k^T(\overline{s} - x_k) + \gamma(g_k - \nabla f(x_k))^T(\overline{s} - x_k) \\
&\leq f^* - f(x_k) - \gamma g_k^T(s_k - x_k) + \gamma\|g_k - \nabla f(x_k)\|D.
\end{aligned}$$

Rearranging the inequality gives

$$\begin{aligned}
&f(x_k) - f^* \\
&\leq -\frac{\gamma^2}{2}\mu_f^{FW} - \gamma g_k^T(s_k - x_k) + \gamma\|g_k - \nabla f(x_k)\|D \\
&= -\frac{\mu_f^{FW}}{2}\left(\gamma + \frac{g_k^T(s_k - x_k) - \|g_k - \nabla f(x_k)\|D}{\mu_f^{FW}}\right)^2 + \frac{(g_k^T(s_k - x_k) - \|g_k - \nabla f(x_k)\|D)^2}{2\mu_f^{FW}} \\
&\leq \frac{(g_k^T(s_k - x_k) - \|g_k - \nabla f(x_k)\|D)^2}{2\mu_f^{FW}}.
\end{aligned}$$

By the definition of s_k, we have $g_k^T(s_k - x_k) \leq 0$, and while the algorithm is not terminated, $f(x_k) - f^* > \varepsilon$, so that

$$\varepsilon < \frac{(g_k^T(s_k - x_k) - \|g_k - \nabla f(x_k)\|D)^2}{2\mu_f^{FW}},$$

which implies that

$$g_k^T(x_k - s_k) + \|g_k - \nabla f(x_k)\|D \geq \sqrt{2\mu_f^{FW}\varepsilon}. \tag{3.11}$$

By convexity, the $L-$smoothness property, and (3.11),

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) + \gamma_k \nabla f(x_k)^T(s_k - x_k) + \frac{L\gamma_k^2}{2}\|s_k - x_k\|^2 \\ &\leq f(x_k) + \gamma_k g_k^T(s_k - x_k) + \gamma_k\|g_k - \nabla f(x_k)\|D + \frac{L\gamma_k^2 D^2}{2} \\ &\leq f(x_k) - \gamma_k\sqrt{2\mu_f^{FW}\varepsilon} + 2\gamma_k\|g_k - \nabla f(x_k)\|D + \frac{L\gamma_k^2 D^2}{2}. \end{aligned}$$

By the definition of good gradient approximation in Definition 3.4.2,

$$f(x_{k+1}) \leq f(x_k) - \frac{\gamma_k}{2}\sqrt{2\mu_f^{FW}\varepsilon} + \frac{L\gamma_k^2 D^2}{2}. \tag{3.12}$$

The step size policy from Algorithm 3, $\gamma_k = \min\left\{\sqrt{2\mu_f^{FW}\varepsilon}/(2LD^2), 1\right\}$ corresponds to moving either to a point between the current solution and the vertex s_k or moving all the way to the vertex s_k.

Case I: Suppose $\gamma_k = \sqrt{2\mu_f^{FW}\varepsilon}/(2LD^2)$. Then, using the fact that the algorithm

is not yet terminated, i.e. $f(x_k) - f^* > \varepsilon$ and (3.12),

$$f(x_{k+1}) - f(x_k) \leq -\frac{\mu_f^{FW}\varepsilon}{4LD^2}.$$

Case II: If $\gamma_k = 1$, then $2LD^2 \leq \sqrt{2\mu_f^{FW}\varepsilon}$. Therefore,

$$f(x_{k+1}) - f(x_k) \leq -\frac{\sqrt{2\mu_f^{FW}\varepsilon}}{4}.$$

Combining these two cases gives the result. □

We next establish similar results as in Chapter 2 Section 2.5, using the good gradient approximation definition below

Definition 3.4.3. *We say, at iteration k, that G_k is a good gradient estimate for the standard Frank-Wolfe algorithm under interior optimum assumption with at least probability p_g if the event*

$$E_k^{(iFW)} = \left\{ \|G_k - \nabla f(X_k)\| \leq \frac{\sqrt{2\mu_f^{FW}\varepsilon}}{4D} \right\}$$

satisfies the condition $\mathbb{P}(E_k^{(iFW)}|\mathcal{F}_k) \geq p_g$ on the event $\{k < T_\varepsilon\}$.

Lemma 3.4.2. *Under Assumption 2.4.1 (bounded variance), Assumption 2.2.4 (strong convexity), and Assumption 3.4.1 (interior optimum) the sample size required for each iteration to achieve a good gradient estimate for the interior standard Frank-Wolfe is $\mathcal{O}(\varepsilon^{-1}(1-p_g)^{-1})$.*

Proof. Take $s = \sqrt{2\mu_f^{FW}\varepsilon}/4D$ in Lemma 2.4.2 and then, by Definition 3.4.3, it

suffices to choose the sample size n so that

$$n \geq \frac{8V_g D^2}{\varepsilon \mu_f^{FW}(1-p_g)}. \tag{3.13}$$

□

Consider the stochastic processes $\Phi^{(iFW)}$ similar to the standard Frank-Wolfe's Lyapunov function in (2.13),

$$\Phi_k^{(iFW)} = f(X_k) - f^*. \tag{3.14}$$

The stochastic processes $\Phi^{(iFW)}$ are bounded because the polytope $\mathcal{X}$ is bounded. We now establish a key result that determines iteration and sample size complexity.

Theorem 3.4.1. *Under Assumptions 3.2.1 (bounded variance), 2.2.1 (L−smoothness) , 2.2.4 (strong convexity), and 3.4.1 (interior optimum), there exists $p_g \in (0,1)$ such that, with the sample size given in* (3.13) *for interior standard Frank-Wolfe, on the event $T_\varepsilon > k$,*

$$\mathbb{E}\left[\Phi_{k+1}^{(iFW)} - \Phi_k^{(iFW)} \middle| \mathcal{F}_k\right] \leq -\frac{\beta^{iFW}}{2}.$$

Proof. Similar to Theorem 2.5.1, we have the result from (2.15) that

$$\mathbb{E}[f(X_{k+1}) - f(X_k)|\mathcal{F}_k] \leq \mathbb{E}[(f(X_{k+1}) - f(X_k))\mathbb{1}_{\mathcal{G}}|\mathcal{F}_k] + \Gamma D\sqrt{\frac{V_g}{n_k}}\sqrt{1-P_k} + \frac{L\Gamma^2 D^2}{2}(1-P_k).$$

Using Lemma 3.4.1 and the sample size given in (3.13), on the event $T_\varepsilon > k$,

$$\begin{aligned}\mathbb{E}[\Phi_{k+1}^{(iFW)} - \Phi_k^{(iFW)}|\mathcal{F}_k] &\leq -\beta^{iFW}\varepsilon p_g + \Gamma D\sqrt{\frac{V_g}{n_k}}\sqrt{1-p_g} + \frac{L\Gamma^2 D^2}{2}(1-p_g) \\ &\leq -\beta^{iFW}\varepsilon p_g + \Gamma\sqrt{\frac{\varepsilon\mu_f^{FW}}{8}}(1-p_g) + \frac{L\Gamma^2 D^2}{2}(1-p_g).\end{aligned}$$

Pick probability p_g such that

$$p_g \geq \frac{\Gamma\sqrt{\varepsilon\mu_f^{FW}/8} + L\Gamma^2 D^2/2 + \beta^{iFW}\varepsilon/2}{\Gamma\sqrt{\varepsilon\mu_f^{FW}/8} + L\Gamma^2 D^2/2 + \beta^{iFW}\varepsilon}, \tag{3.15}$$

which then yields that, on the event $T_\varepsilon > k$,

$$\mathbb{E}\left[\Phi_{k+1}^{(iFW)} - \Phi_k^{(iFW)}\middle|\mathcal{F}_k\right] \leq -\frac{\beta^{iFW}}{2}.$$

□

Theorem 3.4.2. *Suppose the initial solution $X_0 = x_0$ is deterministic with $f(x_0) - f^* > \varepsilon$. Let $T_\varepsilon = \inf\{k : f(X_k) - f^* \leq \varepsilon\}$. Under Assumptions 3.2.1 (bounded variance), 2.2.1 ($L-$smoothness) and 2.2.4 (strong convexity), and 3.4.1 (interior optimum), with the sample given in* (3.13) *for interior standard Frank-Wolfe algorithm, $\mathbb{E}T_\varepsilon = \mathcal{O}(\varepsilon^{-1}(f(x_0) - f^*))$.*

Proof. Define the stochastic process M_k as in Theorem 2.5.2 with $\delta = \beta^{iFW}/2$. Because

$$\beta^{iFW} = \min\left\{\frac{\mu_f^{FW}}{4LD^2}, \frac{\sqrt{2\mu^{FW_f}}}{4\sqrt{\varepsilon}}\right\},$$

then the analogous result as in Theorem 2.5.2 holds where

$$\mathbb{E}[T_\varepsilon] \leq \frac{\Phi_0^{(iFW)}}{\delta} = \frac{2\Phi_0^{(iFW)}}{\beta^{iFW}\varepsilon} \sim \mathcal{O}\left(LD^2\frac{f(x_0)-f^*}{\varepsilon}\right) \sim \mathcal{O}\left(\varepsilon^{-1}(f(x_0)-f^*)\right).$$

□

Theorem 3.4.3. *The sample sizes in each iteration that are required to achieve a good gradient estimate under the Assumption 2.4.1 (bounded variance), Assumptions 2.2.1 (L−smoothness), 2.2.4 (strong convexity), and 3.4.1 (interior optimum) for interior standard Frank-Wolfe are $\mathcal{O}(\varepsilon^{-2})$.*

Proof. For the interior standard Frank-Wolfe algorithm, with the choice p_g from (3.15),

$$1-p_g = \frac{\beta^{iFW}\varepsilon/2}{\Gamma\sqrt{\varepsilon\mu_f^{FW}/8} + L\Gamma^2D^2/2 + \beta^{iFW}\varepsilon},$$

The necessary sample size in (3.13) is

$$\begin{aligned}\frac{8V_gD^2}{\varepsilon\mu_f^{FW}(1-p_g)} &= \frac{16V_gD^2}{\beta^{iFW}\mu_f^{FW}\varepsilon^2}\left(\Gamma\sqrt{\varepsilon\mu_f^{FW}/8} + L\Gamma^2D^2/2 + \beta^{iFW}\varepsilon\right)\\ &= 16V_gD^2\left(\frac{1}{2\sqrt{2}\beta^{iFW}\sqrt{\mu_f^{FW}}\varepsilon\sqrt{\varepsilon}} + \frac{LD^2}{2\beta^{iFW}\mu_f^{FW}\varepsilon^2} + \frac{1}{\mu_f^{FW}\varepsilon}\right),\end{aligned}$$

which is dominated by

$$\frac{16V_gLD^4}{2\beta^{iFW}\mu_f^{FW}\varepsilon^2} = \frac{8V_gLD^4}{\mu_f^{FW}\varepsilon^2}\max\left\{\frac{LD^2}{\mu_f^{FW}}, \sqrt{\frac{\varepsilon}{\mu_f^{FW}}}\right\} \sim \mathcal{O}\left(\frac{L^2D^6}{{\mu_f^{FW}}^2\varepsilon^2}\right) \sim \mathcal{O}(\varepsilon^{-2}).$$

□

Theorem 3.4.4. *The sample size in each iteration that are required to achieve*

a good gradient estimate for the interior standard Frank-Wolfe algorithm under the Assumptions 2.4.2 (sub-Gaussian), 2.2.1 (L−smoothness) and 2.2.4 (strong convexity) is $\mathcal{O}(\varepsilon^{-1}\log(\varepsilon^{-1}))$.

Proof. Similar to the analysis in Theorem 3.4.4, the inequality in (2.21) implies that

$$\mathbb{E}[\Phi_{k+1}^{(iFW)} - \Phi_k^{(iFW)}|\mathcal{F}_k] \leq -\beta^{iFW}\varepsilon p_g + \Gamma D\sqrt{\frac{2d}{n_k c}}\sqrt{1-p_g} + \frac{L\Gamma^2 D^2}{2}(1-p_g).$$

Pick the probability p_g so that

$$1 - p_g \leq \min\left\{\frac{1}{6}, \frac{\varepsilon^4 c}{4608 d L^2\Gamma^2 D^6}, \frac{\varepsilon^2}{24 L^2\Gamma^2 D^4}\right\}$$

Then,

$$\beta^{iFW}\varepsilon(1-p_g) + \Gamma D\sqrt{\frac{2d}{n_k c}}\sqrt{1-p_g} + \frac{L\Gamma^2 D^2}{2}(1-p_g) \leq \frac{\beta^{iFW}\varepsilon}{2}.$$

Therefore,

$$\begin{aligned}\mathbb{E}[\Phi_{k+1}^{(iFW)} - \Phi_k^{(iFW)}|\mathcal{F}_k] &\leq -\beta^{iFW}\varepsilon p_g + \Gamma D\sqrt{\frac{2d}{n_k c}}\sqrt{1-p_g} + \frac{L\Gamma^2 D^2}{2}(1-p_g)\\ &\leq -\frac{\beta^{iFW}\varepsilon}{2}.\end{aligned}$$

The analogous results hold as in Theorem 3.4.1 and Theorem 3.4.2, and we have the same iteration complexity as in Theorem 3.4.2. Now take $s = \sqrt{2\mu_f^{FW}\varepsilon}/(4D)$ in Lemma 2.4.3 and select the sample size n so large such that $2de^{-nc\mu_f^{FW}\varepsilon/(8D^2)}$ is a lower bound of $1-p_g$ where c is a constant depending on the parameter ρ

from Assumption 2.4.2 and shown in Lemma 2.4.3. The term involving ε^4 is the determining factor for small ε, which gives

$$\begin{aligned} n &\geq \frac{8D^2}{c\mu_f^{FW}\varepsilon}\log\frac{9216d^2L^2\Gamma^2D^6}{\varepsilon^4 c} \\ &= \frac{8D^2}{c\mu_f^{FW}\varepsilon}\left(\log\frac{9216}{c}+2\log\frac{dL\Gamma D^3}{\varepsilon^2}\right) \\ &\leq \frac{8D^2}{c\mu_f^{FW}\varepsilon}\left(\log\frac{9216}{c}+2\log\frac{dLD^3}{\varepsilon^2}\right), \end{aligned}$$

which is dominated by

$$\frac{16D^2}{c\mu_f^{FW}\varepsilon}\log\frac{dL\Gamma D^3}{\varepsilon^2} \sim \mathcal{O}\left(\frac{D^2}{\mu_f^{FW}\varepsilon}\log\frac{LD}{\varepsilon}\right) \sim \mathcal{O}(\varepsilon^{-1}\log\varepsilon^{-1}).$$

□

In summary, from Theorem 3.4.2 and Theorem 3.4.3, the total work complexity for standard Frank-Wolfe algorithm under Assumptions 2.2.1 ($L-$smoothness), 2.2.4 (strong convexity), and 3.4.1 (interior optimum) are

1. $\mathcal{O}(\varepsilon^{-3})$ under Assumption 2.4.1 (bounded variance), and
2. $\mathcal{O}(\varepsilon^{-2}\log\varepsilon^{-1})$ under Assumption 2.4.2 (sub-Gaussian)

The bound on the expected stopping time obtained in Theorem 3.4.2 can be expressed more explicitly as

$$\mathbb{E}[T_\varepsilon] \leq \frac{2(f(x_0)-f^*)}{\beta^{iFW}\varepsilon} \sim \mathcal{O}\left(\frac{LD^2}{\mu_f^{FW}\varepsilon}\right).$$

Under Assumption 2.4.1 (bounded variance), Theorem 3.4.2 yields the sample size

complexity

$$\mathcal{O}\left(\frac{V_g L^2 D^6}{{\mu_f^{FW}}^2 \varepsilon^2}\right),$$

while under Assumption 2.4.2 (sub-Gaussian), the sample size complexity becomes

$$\mathcal{O}\left(\frac{D^2}{\mu_f^{FW}\varepsilon}\log\frac{dDL}{\varepsilon}\right).$$

Combining the iteration and sample size complexities gives the corresponding total work complexities. Under the bounded-variance assumption, the total work complexity is

$$\mathcal{O}\left(\frac{V_g L^3 D^8}{{\mu_f^{FW}}^3 \varepsilon^3}\right),$$

whereas under the sub-Gaussian assumption, it is

$$\mathcal{O}\left(\frac{LD^4}{{\mu_f^{FW}}^2 \varepsilon^2}\log\frac{dDL}{\varepsilon}\right).$$

These expressions highlight the dependence of the overall complexity on the smoothness constant L, the variance parameter V_g, strong convexity interior parameter μ_f^{FW}, the feasible-region diameter D, and the dimension d.

CHAPTER 4

BOOSTED FRANK-WOLFE

4.1 Introduction

Previously, we analyzed the convergence behavior of the standard and Away-step Frank-Wolfe algorithms for the problem

$$\min_x f(x) = \mathbb{E}[F(x, \xi)]$$
$$\text{s.t. } Ax \leq b.$$

For the standard Frank-Wolfe algorithm, as the iterates approach the optimal solution, the method often exhibits a zig-zagging behavior because the iterate remains constrained by the weights assigned to vertices selected in earlier iterations. In Figure 4.1, the standard Frank–Wolfe algorithm is held back by the vertex x_0, which serves as the initial point, since its associated weight does not decrease to zero fast enough. As the iterates move closer to the optimal solution x^*, the search direction $v - x$ tends to become increasingly orthogonal to the direction $x - x^*$. Consequently, the alignment between the search direction and the direction toward optimality deteriorates, leading to slower progress and the characteristic zig-zagging behavior of the standard Frank-Wolfe algorithm.

An important extension of the deterministic standard Frank–Wolfe algorithm was analyzed in [53], where the authors established an improved iteration complexity of $\mathcal{O}(\log \varepsilon^{-1})$. However, this improvement requires more than strong convexity alone. In addition to strong convexity, it assumes that the global optimum x^* lies strictly in the interior of the feasible region $\mathcal{X}$. Under this interior-optimality assumption,

the alignment between the Frank–Wolfe search direction and the negative gradient can be significantly improved, thereby avoiding the severe deterioration responsible for the zig-zagging behavior described above. Nevertheless, the interior-optimality assumption may not hold in many practical applications, where optimal solutions often lie on or near the boundary of the feasible region. Consequently, it remains important to develop algorithmic modifications that improve the convergence behavior of Frank–Wolfe methods without relying on this additional geometric assumption. This motivation led to the development of variants such as Away-step Frank–Wolfe, which achieves linear convergence under strong convexity even when the optimal solution lies on the boundary of a polytope.

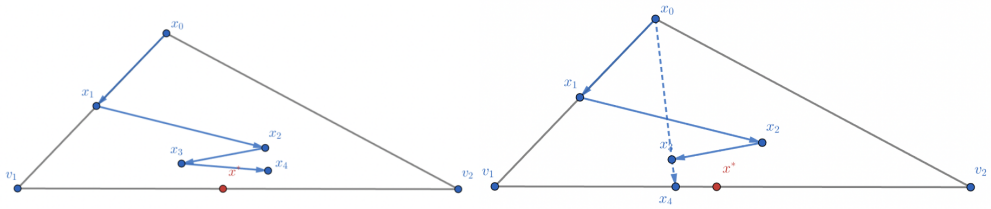


Figure 4.1: An example of a standard Frank-Wolfe path (left), and an away-step Frank-Wolfe path (right).

Away-step Frank–Wolfe was proposed in [89] to mitigate the zig-zagging behavior by allowing the algorithm to reduce and eventually eliminate the weight associated with previously selected vertices, such as the vertex x_0. Figure 4.1 illustrates that away-step Frank–Wolfe can converge significantly faster than standard Frank–Wolfe by moving away from unfavorable vertices, such as v_0. Let v^g and v^b denote the good and bad vertices, respectively, where the algorithm intends to move toward v^g and away from v^b, as determined in Algorithm 2. The search direction is therefore either $v^g - x$ or $x - v^b$. Consequently, all possible search directions are of the form $\pm(v - x)$, where v is a vertex of the feasible region $\mathcal{X}$. Compared with standard Frank–Wolfe, the algorithm has access to a richer set of search directions, which generally improves the alignment between the negative gradient and

the chosen direction. This stronger alignment leads to larger objective decreases and faster convergence.

However, this modification is still restricted to directions toward or away from vertices and therefore may not align as closely with the negative gradient direction as gradient descent methods. In fact, the convergence analyses in [54] and [7] show that the convergence rate of Away-step Frank–Wolfe depends on the Pyramidal Width ($\Omega_{\mathcal{X}}$), a geometric constant of the feasible region that is closely related to its dimensionality. This dependence reflects the fact that the available search directions remain constrained by the geometry of the vertex set. Furthermore, away-step Frank–Wolfe requires additional bookkeeping to maintain the current iterate as a convex combination of vertices, since this representation is needed to determine which vertex should be selected for an away step. Several attempts have been made to further improve Frank–Wolfe variants; however, these approaches typically rely on stronger assumptions regarding either the objective function or the feasible region. For example, [59, 30] assume that the feasible region is strongly convex, an assumption that does not apply to our linearly constrained problem setting.

Boosted Frank–Wolfe, proposed in [21], addresses the limitations of standard and Away-step Frank–Wolfe by constructing search directions that more closely approximate the negative gradient using vertices of the feasible region $\mathcal{X}$. The central idea is to refine a search direction d so that it aligns as closely as possible with the negative gradient while preserving the projection-free nature of Frank–Wolfe methods through a conical decomposition. In particular, if the negative gradient

satisfies

$$-g \in \sum_j \lambda_j (v_j - x),$$

for the current iterate x, vertices v_j, and nonnegative coefficients λ_j, then a feasible search direction can be constructed as

$$d = \frac{\sum_j \lambda_j (v_j - x)}{\Lambda},$$

where

$$\Lambda = \sum_j \lambda_j.$$

By construction, this normalization guarantees that $x + d \in \mathcal{X}$. From a geometric perspective, standard Frank–Wolfe is restricted to directions of the form $v - x$, while Away-step Frank–Wolfe expands this set to $\pm(v - x)$, where v is a vertex of $\mathcal{X}$. Boosted Frank–Wolfe further enlarges the set of admissible directions by allowing conic combinations of the vectors $v - x$ for all vertices $v \in \mathcal{V}$. This richer collection of search directions enables substantially better alignment with the negative gradient. Consequently, boosted Frank–Wolfe combines the projection-free advantage of Frank–Wolfe methods with search directions that more closely resemble those of gradient descent. By iteratively improving this alignment, the algorithm is able to mimic gradient-descent behavior without requiring explicit projections onto the feasible region. In [21], numerical experiments are conducted on several benchmark problems, including sparse signal recovery [81], constrained logistic regression [38], and traffic assignment [8]. The results demonstrate that the boosted Frank–Wolfe algorithm consistently outperforms the away-step Frank–Wolfe algorithm across these problem classes. In the unbounded setting, the same framework naturally extends to include extreme rays, further expanding the set

of feasible search directions. An example of different solution paths from different algorithms are illustrated in Figure 4.2. This chapter extends the boosted Frank-Wolfe algorithm from [21], which is proposed for deterministic problems, to stochastic settings, and develops convergence analyses that are parallel to analyses in Chapter 2 and 3.

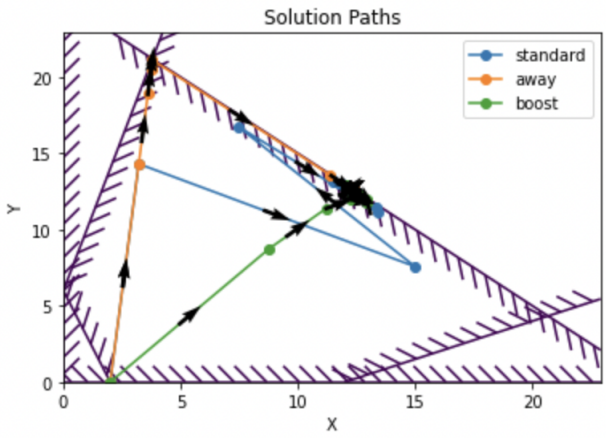


Figure 4.2: The AFW algorithm mitigates zigzagging behavior in the original Frank-Wolfe algorithm, by moving not just towards vertices, but also away from vertices. The BFW algorithm has even more flexibility in choosing a search direction.

This chapter is organized as follows. Section 4.2 introduces the stochastic boosted Frank–Wolfe algorithm, adapted from the deterministic algorithm of [21], and provides the intuition behind its search direction computation procedure. Section 4.3 develops the preparatory results required for the convergence analysis, including conditions for good gradient approximation and the corresponding sample size requirements. Section 4.4 presents the iteration and sample size complexity results for the stochastic boosted Frank–Wolfe algorithm. Finally, Section 4.5 extends the analysis to several additional settings, yielding complexity results whose structure parallels those developed in Chapter 3.

4.2 Boosted Frank-Wolfe Algorithm

The away-step Frank-Wolfe algorithm 2 was introduced to mitigate the zig-zagging behavior of the standard Frank-Wolfe algorithm. However, its search direction can still be poorer than that of gradient descent. Algorithm 4, the boosted Frank-Wolfe algorithm, improves alignment between negative gradient and search direction. Within each iteration, k, it attempts to represent the negative gradient as a conic combination of the vertices, $\mathcal{V}$. This approach implicitly addresses the optimization subproblem

$$\min_{d \in cone(\mathcal{V}-x_k)} \frac{1}{2}\| - g_k - d\|^2, \tag{4.1}$$

where g_k is the estimated gradient of the solution x_k at iteration k, d is the search direction, and *cone* refers to the conic combination. The domain over which we explore consists of the conic combinations of $\mathcal{V} - x_k$, which is standard. Within each iteration, the algorithm successively refines the direction d by attempting to improve its alignment with the negative estimated gradient, $-g_k$, by projecting onto a one-dimensional line using the Non-Negative Matching Pursuit Algorithm [60]. Alignment is measured by the function $align$, defined as

$$align(u, v) = \begin{cases} \frac{u^T v}{||u|| \cdot ||v||} & \text{if } u, v \neq 0, \\ -1 & \text{if } u = 0 \text{ or } v = 0, \end{cases} \tag{4.2}$$

which is, essentially, the cosine of the angle between the vectors u and v. The algorithm successively improves the alignment, in each iteration solving a linear program that identifies the vertex that best can be used to reduce the difference between the negative gradient and the current search direction, stopping only when it fails to improve the alignment by at least some minimal threshold, δ. In Algorithm 4, T denotes the maximum number of these alignment-improvement

steps permitted in the Boosted Direction Oracle. When $T = 1$, BFW reduces to the standard Frank-Wolfe method, and we define the search direction d_k to be "non-boosted", otherwise, it is boosted. The maximum step size γ_k^{max} is set to

Algorithm 4 Boosted Frank-Wolfe (BFW).

Initialize x_0 to be a vertex v, maximum number of oracle steps T, an improvement tolerance δ.
for $k = 0, 1, 2, ...$ **do**
  Estimate the gradient $\nabla f(x_k)$ by g_k
  $d_k, \Lambda_k \leftarrow$ Boosted Direction Oracle(T, δ, g_k)
  $d_k \leftarrow d_k / \Lambda_k$
  if d_k not boosted **then**
    $\gamma_k = \min\{1, \frac{\varepsilon}{2LD^2}\}$ (standard Frank-Wolfe step)
  else
    $\gamma_k = \min\{1, -\frac{g_k^T d_k}{L\|d_k\|^2}\}$
  end if
  $x_{k+1} \leftarrow x_k + \gamma_k d_k$
end for

1 since in the case of boosted d_k, the search direction is normalized where in the non-boosted case, the maximum step size is the same as the one in the standard Frank-Wolfe algorithm 1. In each boosted iteration, the search direction d_k can be represented as

$$d_k = \sum_j \lambda_j (v_j - x_k)$$

where $\sum_j \lambda_j = 1$ and the v_js are vertices of $\mathcal{X}$. Then, for any step size $\gamma_k \leq 1$,

$$x_k + \gamma_k d_k = x_k + \gamma_k \sum_j \lambda_j (v_j - x_k) = \gamma_k \sum_j \lambda_j v_j + (1 - \gamma_k) x_k,$$

which is a linear combination between two feasible points. Then, $x_k + \gamma_k d_k$ is also feasible for all $\gamma_k \in [0, 1]$. Since the Boosted Direction Oracle within the loop in Algorithm 4 also contains a loop, we define another term below for convenience.

Definition 4.2.1. *In the boosted Frank–Wolfe algorithm 4, each main iteration calls the Boosted Direction Oracle, given in Algorithm 5. To distinguish the it-*

erations of the oracle from the main iterations of the algorithm, we refer to each iteration of Algorithm 5 as a ***sub-iteration****.*

In the search direction computation of Algorithm 5, the number of sub-iterations (as defined in Definition 4.2.1) may be either one or greater than one. If the number of sub-iterations is one, then the alignment between the negative gradient and the search direction cannot be further improved. In this case, Algorithm 5 solves the linear program only once and returns the resulting search direction. Since this direction is obtained directly from the linear minimization oracle without any refinement, it coincides with the standard Frank–Wolfe direction and no boosting occurs. We refer to such iterations as *non-boosted* iterations. On the other hand, if Algorithm 5 performs more than one sub-iteration, then the search direction is iteratively refined to improve its alignment with the negative gradient. In this case, we say that the search direction is *boosted*. We formalize these iteration types below:

Definition 4.2.2. *Consider the boosted Frank–Wolfe algorithm in Algorithm 4. If the search direction computation in Algorithm 5 requires exactly one sub-iteration, then the resulting search direction is called* ***non-boosted*** *(search direction not boosted). Otherwise, if the search direction computation requires more than one sub-iteration, then the resulting search direction is called* ***boosted****. We also refer to the iteration associated with these types of search direction as non-boosted and boosted iterations respectively.*

The Boosted Direction Oracle in Algorithm 5 was proposed in [21] and is adapted from the Non-Negative Matching Pursuit algorithm of [60]. At each sub-iteration, the oracle updates the search direction d_t to improve its alignment, defined in (4.2), with the negative gradient. Equivalently, the algorithm seeks to increase the

Algorithm 5 Boosted direction oracle.

Accept as input x, the current solution, T, the maximum number of iterations, $\delta \in (0,1)$, a parameter to indicate sufficient alignment, and g, the estimated gradient
Initialize d_0 to be a zero vector for the direction, $\Lambda = 0$ to be a normalizing term.
for $t = 0, 1, 2, ..., T-1$ **do**
  $r_t \leftarrow -g - d_t$; this is the running residual
  $v_t \leftarrow \arg\max_{u \in \mathcal{X}} r_t^T u$
  $w \leftarrow v_t - x$
  $u_t \leftarrow \arg\max_{u \in \{w, -d_t/\|d_t\|\}} r_t^T u$;
  $\lambda_t \leftarrow r_t^T u_t / \|u_t\|^2$
  $d' \leftarrow d_t + \lambda_t u_t$
  if $align(-g, d') - align(-g, d_t) \geq \delta$ **then**
    $d_{t+1} \leftarrow d'$
    $\Lambda = \begin{cases} \Lambda + \lambda_t & \text{if } u_t = w \\ \Lambda(1 - \lambda_t/\|d_t\|) & \text{if } u_t = -d_t/\|d_t\| \end{cases}$
  else
    break
  end if
end for
return d_t, Λ

cosine of the angle between the search direction and the negative gradient.

The comparison between $v_t - x$ and $-d_t/\|d_t\|$ in Algorithm 5, introduced in [21], is also adapted from [60], where it was originally developed for solving the direction subproblem (4.1). This step enables the Boosted Direction Oracle to reduce the weights of previously selected components $v - x_k$, effectively allowing the algorithm to partially "move back" from earlier choices while preserving the conic-combination structure of the search direction. An example of boosted direction oracle is illustrated in Figure 4.3.

Let $h(d)$ denote the objective function in (4.1) for the deterministic setting at iteration k, where g_k is replaced by $\nabla f(x_k)$, namely

$$h(d) = \frac{1}{2}\|\nabla f(x_k) + d\|^2.$$

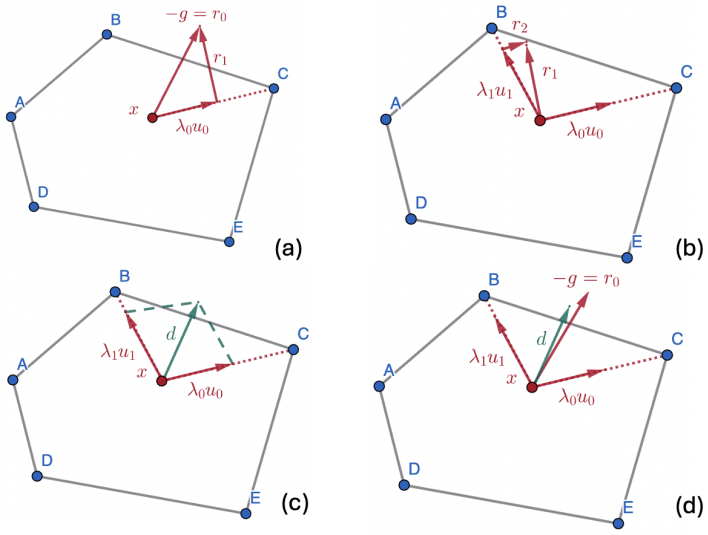


Figure 4.3: Computing the search direction. In (a), Vertex C solves the linear program, and we project the negative gradient onto u_0, leaving the residual r_1. In (b), Vertex B solves the linear program, and we project the residual, r_1, onto u_1, leaving the residual r_2, which sufficiently improves alignment and thus defines the direction d in (c). In (d), the alignment between the direction d and the negative gradient is highlighted.

According to [60], if $\nabla h(d_t)^T z \geq 0$ for all feasible z, and d_t is not a minimizer of h, then

$$\nabla h(d_t)^T \left(-\frac{d_t}{\|d_t\|} \right) < 0.$$

To prove this result, suppose that d_t is not a minimizer of h. Then there exists a point $d' \neq d_t$, where $d' = \sum_j \lambda'_j (v_j - x)$ such that $h(d') < h(d_t)$. Assume, for the sake of contradiction, that

$$\nabla h(d_t)^T \left(-\frac{d_t}{\|d_t\|} \right) \geq 0,$$

We will show that this assumption contradicts the existence of d'. Since

$$\nabla h(d) = \nabla f(x_k) + d,$$

the assumption implies

$$(\nabla f(x_k) + d_t)^T \left(-\frac{d_t}{\|d_t\|} \right) \geq 0,$$

or equivalently,

$$\nabla f(x_k)^T d_t + \|d_t\|^2 \leq 0.$$

Because $h(d') < h(d_t)$, then

$$\begin{aligned} h(d') &< h(d_t) \\ &= \frac{\|\nabla f(x_k)\|^2}{2} + \nabla f(x_k)^T d_t + \frac{\|d_t\|^2}{2} \\ &\leq \frac{\|\nabla f(x_k)\|^2}{2} - \frac{\|d_t\|^2}{2}, \end{aligned}$$

which can be simplified as

$$\nabla f(x_k)^T d' + \frac{\|d'\|^2}{2} < -\frac{\|d_t\|^2}{2}.$$

Moreover, because $\nabla h(d_t)^T z \geq 0$ for any feasible z, then $\nabla h(d_t)^T (v - x_k) \geq 0$ for any vertex v. Therefore,

$$\begin{aligned} -\frac{\|d_t\|^2}{2} &> \nabla f(x_k)^T d' + \frac{\|d'\|^2}{2} \\ &= \nabla f(x_k)^T d' + d_t^T d' + \frac{\|d'\|^2}{2} - d_t^T d' \\ &= \nabla h(d_t)^T d' + \frac{\|d'\|^2}{2} - d_t^T d' \\ &= \sum_j \lambda'_j \nabla h(d_t)^T (v_j - x_k) + \frac{\|d'\|^2}{2} - d_t^T d' \\ &\geq \frac{\|d'\|^2}{2} - d_t^T d', \end{aligned}$$

which can be simplified as

$$0 > \frac{1}{2}\|d_t - d'\|^2.$$

This is a contradiction. Then,

$$\nabla h(d_t)^T \left(-\frac{d_t}{\|d_t\|}\right) < 0.$$

Hence, the Non-Negative Matching Pursuit algorithm of [60] includes the direction $-d_t/\|d_t\|$ among the candidate directions to ensure that the algorithm does not terminate at a suboptimal point. By allowing movement in this direction, the Boosted Direction Oracle can continue improving the objective value even when no feasible atom $v_t - x$ provides a descent direction, thereby avoiding premature termination and enabling further refinement of the search direction.

Next, we will be analyzing this Boosted Frank Wolfe algorithm's properties, along with its convergence. The goal is to carry out the algorithm until the condition $f_k - f^* \leq \varepsilon$ is satisfied, i.e. we define the terminal time T_ε of our algorithm to be $T_\varepsilon = \inf\{k \geq 1 : f_k - f^* \leq \varepsilon\}$.

4.3 Preparatory Results for Convergence Analysis

The convergence analysis of the boosted Frank–Wolfe algorithm relies on Assumption 2.2.4 (strong convexity). A fundamental consequence of strong convexity is the so-called μ-*gradient dominated* property, also known as the *Polyak–Łojasiewicz (PL) inequality*. This inequality establishes a relationship between the objective gap and the gradient norm, and it plays a key role in the convergence results developed throughout this section.

Lemma 4.3.1. *Under Assumption 2.2.4 (strong convexity),*

$$f(x) - f^* \leq \frac{||\nabla f(x)||^2}{2\mu}$$

where f^* is the minimum of f. This property is referred as $\mu-$gradient dominated or Polyak–Łojasiewicz (PL) inequality.

Proof. By the strong convexity of f, we have, for all x, y,

$$f(y) \geq f(x) + \nabla f(x)^T(y - x) + \frac{\mu}{2}||y - x||^2.$$

Taking the minimum with respect to y on both sides, the left-hand side becomes f^*. On the right-hand side, the expression is a strongly convex quadratic function of y, which attains its minimum at $y = x - \frac{\nabla f(x)}{\mu}$. Substituting this minimizer into the right-hand side yields

$$f^* \geq f(x) - \frac{||\nabla f(x)||^2}{\mu} + \frac{||\nabla f(x)||^2}{2\mu} = f(x) - \frac{||\nabla f(x)||^2}{2\mu}$$

where f^* is the optimal (minimum) value. Therefore,

$$f(x) - f^* \leq \frac{||\nabla f(x)||^2}{2\mu}$$

□

We define the iteration types in Definition 4.2.2. Each type has a different effect on the alignment of the search direction. For boosted iterations, we can guarantee

a minimum level of alignment between search direction and the negative gradient. In contrast, for non-boosted iterations, the search direction coincides with that of the standard Frank-Wolfe algorithm, for which the alignment may be poor. Since the alignment quantity defined in (4.2) is the cosine of the angle between the search direction and the negative gradient, a larger alignment value (closer to 1) corresponds to a smaller angle between the two vectors. Therefore, higher alignment indicates that the search direction more closely approximates the negative gradient direction.

The following lemma shows that, whenever an iteration is boosted, the alignment between the search direction and the negative gradient admits a positive lower bound. Consequently, boosted iterations guarantee a minimum level of alignment, ensuring that the search direction remains sufficiently close to the negative gradient.

Lemma 4.3.2. *Let η_k be the alignment between search direction d_k and negative gradient $-g_k$, i.e.*

$$\eta_k = \frac{-g_k^T d_k}{\|g_k\|\|d_k\|} = align(-g_k, d_k).$$

If we have at least two sub-iterations as defined in Definition 4.2.1 from the boosted oracle in Algorithm 5 at step k, i.e., the iteration k is boosted, then $\eta_k \geq \delta$.

Proof. Let t denote the number of sub-iterations performed by the Boosted Direction Oracle at iteration k, where $t \geq 2$. Let d_{k_t} denote the search direction at iteration k after sub-iteration t. At initialization, the oracle sets $d_{k_0} = 0$.

By Algorithm 5, the first sub-iteration updates the search direction by solving a linear program, which is equivalent to a standard Frank–Wolfe step. Therefore,

d_{k_1} is parallel to $s - x_k$, where s is the solution of $\min_{z \in \mathcal{X}} g_k^T z$. By the optimality of s, $g_k^T(s - x_k) \geq 0$. Since d_{k_1} is parallel to $s - x_k$, it follows that

$$\eta_k = align(-g_k, d_{k_t}) \geq align(-g_k, d_{k_1}) + \delta \geq \delta.$$

□

We now present several preliminary results that will be used throughout the convergence analysis. The next subsection reviews the deterministic convergence analysis of boosted Frank–Wolfe from [21], which serves as the foundation for establishing the iteration and sample size complexities in the stochastic setting considered later in this chapter. In addition, [21] introduces an assumption on the frequency of boosted iterations, which leads to significantly stronger convergence guarantees. This assumption is particularly important because boosted iterations produce search directions that are more closely aligned with the negative gradient than those obtained during non-boosted iterations. As a result, boosted iterations contribute more substantially to the decrease of the objective function and play a key role in the overall convergence behavior of the algorithm.

4.3.1 Results in the Deterministic Setting

In the deterministic setting, the algorithm is identical to Algorithm 4 and Algorithm 5, except that the estimated gradient g_k is replaced by the true gradient $\nabla f(x_k)$. The analysis in [21] establishes iteration complexity results for the boosted Frank–Wolfe algorithm in both the general setting and the setting where a sufficiently large proportion of iterations are boosted. The latter case yields substantially stronger convergence guarantees due to the improved alignment between the search direction and the negative gradient during boosted iterations.

Lemma 4.3.3. *[21] Under Assumptions 2.2.1 (L−smoothness) and 2.2.4 (strong*

convexity), in the deterministic boosted Frank-Wolfe algorithm, we have an improvement

$$f(x_{k+1})-f^* = \begin{cases} \left(1-\beta^{dbFW}\right)\left(f(x_k)-f^*\right) \text{ if iteration } k \text{ is boosted or is a full-length FW step} \\ \left(1-\frac{f(x_k)-f^*}{2LD^2}\right)\left(f(x_k)-f^*\right) \text{ otherwise,} \end{cases}$$

where $\beta^{dbFW} = \min\{\frac{1}{2}, \frac{\delta^2\mu}{L}\}$.

Proof. By the $L-$smoothness assumption in 2.2.1, we have

$$f(x_{k+1}) \leq f(x_k) + \gamma_k \nabla f(x_k)^T d_k + \frac{L}{2}\gamma_k^2 \|d_k\|^2.$$

If we use the step size $\gamma_k = \frac{-\nabla f(x_k)^T d_k}{L\|d_k\|^2}$, then by Lemma 4.3.2 and Lemma 4.3.1,

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) + \gamma_k \nabla f(x_k)^T d_k + \frac{L}{2}\gamma_k^2 \|d_k\|^2. \\ &= f(x_k) - \frac{(\nabla f(x_k)^T d_k)^2}{2L\|d_k\|^2} \\ &\leq f(x_k) - \frac{\delta^2 \|\nabla f(x_k)\|^2}{2L} \\ &\leq f(x_k) - \frac{\delta^2\mu}{L}(f(x_k)-f^*). \end{aligned}$$

Hence,

$$f(x_{k+1}) - f^* \leq \left(1-\frac{\delta^2\mu}{L}\right)(f(x_k)-f^*).$$

However, if the search direction did not get boosted, we proceed to apply standard Frank-Wolfe direction with the step size policy $\gamma_k = \min\{\frac{-\nabla f(x_k)^T d_k}{L\|d_k\|^2}, 1\}$. If the step

size is 1, then $L\|d_k\|^2 \leq -\nabla f(x_k)^T d_k$. We have,

$$\begin{aligned} f(x_{k+1}) - f^* &\leq f(x_k) - f^* + \nabla f(x_k)^T d_k - \frac{\nabla f(x_k)^T d_k}{2} \\ &\leq f(x_k) - f^* - \frac{f(x_k) - f^*}{2} \\ &= \frac{1}{2}(f(x_k) - f^*). \end{aligned}$$

If the step size is $\frac{-\nabla f(x_k)^T d_k}{L\|d_k\|^2}$, then

$$\begin{aligned} f(x_{k+1}) - f^* &\leq f(x_k) - f^* - \frac{(\nabla f(x_k)^T d_k)^2}{L\|d_k\|^2} \\ &\leq \left(1 - \frac{f(x_k) - f^*}{2LD^2}\right)(f(x_k) - f^*) \\ &\leq \left(1 - \frac{\varepsilon}{2LD^2}\right)(f(x_k) - f^*). \end{aligned}$$

□

It is well known that the standard Frank–Wolfe algorithm achieves an iteration complexity of $\mathcal{O}(\varepsilon^{-1})$. Consequently, the deterministic boosted Frank–Wolfe algorithm also has a worst-case iteration complexity of $\mathcal{O}(\varepsilon^{-1})$. However, the actual convergence rate depends on how frequently boosted iterations occur. If boosted iterations dominate the overall process, the algorithm can achieve an improved complexity of $\mathcal{O}(\log \varepsilon^{-1})$. In [21], the following assumption is introduced to guarantee a sufficient number of boosted iterations where it assumes that at any iteration, there is always a sufficient number of boosted iterations.

Assumption 4.3.1. *After k iterations, the number of boosted iterations is at least ωk for all $k \geq 1$, for some $0 < \omega \leq 1$. We refer to this assumption as the* ***sufficiently boosted*** *assumption.*

This assumption is adapted from [21], where it is assumed that after k iterations, the number of boosted iterations is at least ωk^p for some $p \in (0, 1]$. The authors further note that the convergence behavior observed in practice is consistent with the case $p = 1$. Motivated by this observation, we restrict our attention to the stronger condition corresponding to $p = 1$ and refer to Assumption 4.3.1 as the *sufficiently boosted* assumption.

The above assumption ensures that the number of boosted iterations is always sufficiently large, growing linearly with the number of past iterations.

Lemma 4.3.4. *In boosted deterministic Frank-Wolfe algorithm with assumptions 2.2.1 ($L-$smoothness), 2.2.4 (strong convexity) and 4.3.1 (sufficiently boosted), the iteration complexity is $\mathcal{O}(\log \varepsilon^{-1})$.*

Proof. By Lemma 4.3.3, at iteration k,

$$f(x_k)-f^* \leq \left(1-\beta^{dbFW}\right)^{\omega k}\left(1-\frac{\varepsilon}{2LD^2}\right)^{k-\omega k}(f(x_0)-f^*) \leq \left(1-\beta^{dbFW}\right)^{\omega k}(f(x_0)-f^*).$$

In order to achieve the ε-neighborhood where $f(x_k) - f^* \leq \varepsilon$, we require the number of iteration at least

$$k \geq \left(\frac{1}{\omega \log(1/(1-\beta^{dbFW}))} \cdot \log \frac{f(x_0)-f^*}{\varepsilon}\right) \sim \mathcal{O}(\log \varepsilon^{-1})$$

where $\beta^{dbFW} = \min\left\{\frac{1}{2}, \frac{\delta^2\mu}{L}\right\}$ □

Assumption 4.3.1 (sufficiently boosted) was introduced in [21] based on empirical observations of the algorithm's behavior in practice. Under this assumption,

the deterministic boosted Frank–Wolfe algorithm achieves an iteration complexity of $\mathcal{O}(\log \varepsilon^{-1})$, which matches the convergence rate of the deterministic away-step Frank–Wolfe algorithm. Motivated by these deterministic results, we now turn to the stochastic setting. In contrast to the deterministic case, the convergence analysis of stochastic boosted Frank–Wolfe must account for errors arising from gradient estimation. Consequently, we first establish conditions under which the gradient estimate is sufficiently accurate to guarantee the objective-function improvements required for convergence.

4.3.2 Results in Stochastic Setting

In Chapter 2, we define the good gradient estimate conditions for the standard Frank–Wolfe and Away-step Frank–Wolfe algorithms using the target accuracy ε together with the feasible-region-dependent constant known as the Pyramidal Width ($\Omega_{\mathcal{X}}$). However, the boosted Frank–Wolfe framework differs fundamentally because its search direction is constructed iteratively to improve the alignment with the negative gradient, as described in Algorithm 5. Consequently, determining whether a stochastic gradient estimate is sufficiently accurate must account for its alignment with the resulting search direction.

In the boosted direction oracle of Algorithm 5, the parameter δ controls the quality of the alignment between the negative estimated gradient and the constructed search direction. Motivated by this geometric property, we define the good gradient estimate condition in terms of the alignment value η_k at iteration k, which measures the alignment between the negative estimated gradient and the search direction d_k.

Definition 4.3.1. Good gradient approximation for boosted Frank-Wolfe in

boosted iterations. *Fix some positive number* $\varepsilon_g \in (0, 1/2)$. *The vector* g_k *is a good gradient approximation at iteration* k *if*

$$\|g_k - \nabla f(x_k)\| \leq \varepsilon_g \|g_k\| \eta_k,$$

when η_k *is the alignment between* $-g_k$ *and the search direction* d_k*, i.e.*

$$\eta_k = -\frac{g_k^T d_k}{\|g_k\| \|d_k\|}.$$

Lemma 4.3.5. *Suppose Assumptions 2.2.1 (L−smoothness) and 2.2.4 (strong convexity), and suppose that at step* k*, the number of sub-iterations of the boosted oracle is at least two (search direction is boosted) and the step size is* $\gamma_k = \frac{-g_k^T d_k}{L\|d_k\|^2} = \frac{\eta_k \|g_k\|}{L\|d_k\|}$*. Then, under the good gradient Definition 4.3.1 (good gradient approximation for boosted Frank-Wolfe in boosted iterations),*

$$f(x_{k+1}) - f^* \leq \left(1 - \frac{2\mu}{L}\left(\frac{1}{2} - \varepsilon_g\right)\left(\frac{\delta}{\varepsilon_g \delta + 1}\right)^2\right)(f(x_k) - f^*)$$

Proof. Because η_k is the cosine measure between $-g_k$ and d_k, then $-1 \leq \eta_k \leq 1$. By the Triangle Inequality,

$$\varepsilon_g \eta_k \|g_k\| \geq \|g_k - \nabla f(x_k)\| \geq \|\nabla f(x_k)\| - \|g_k\|,$$

or,

$$\|g_k\| \geq \frac{\|\nabla f(x_k)\|}{\varepsilon_g \eta_k + 1}. \tag{4.3}$$

Using the property of strong convexity and Lemma 4.3.1 (PL inequality),

$$\begin{aligned}\|g_k\|^2 &\geq \frac{\|\nabla f(x_k)\|^2}{(\varepsilon_g\eta_k+1)^2} \\ &\geq \frac{2\mu(f(x_k)-f^*)}{(\varepsilon_g\eta_k+1)^2}.\end{aligned}$$

Using the property of $L-$Lipschitz and convexity,

$$f(x_{k+1}) \leq f(x_k) + \gamma_k \nabla f(x_k)^T d_k + \frac{L}{2}\gamma_k^2 \|d_k\|^2.$$

From the good gradient estimate assumption and the Cauchy Schwartz Inequality,

$$(\nabla f(x_k) - g_k)^T d_k \leq \|\nabla f(x_k) - g_k\| \, \|d_k\| \leq \varepsilon_g \eta_k \|g_k\| \, \|d_k\|,$$

or,

$$\nabla f(x_k)^T d_k \leq g_k^T d_k + \varepsilon_g \eta_k \|g_k\| \, \|d_k\|.$$

Therefore, with the definition of γ_k and η_k,

$$\begin{aligned}f(x_{k+1}) &\leq f(x_k) + \gamma_k \nabla f(x_k)^T d_k + \frac{L}{2}\gamma_k^2 \|d_k\|^2 \\ &\leq f(x_k) + \gamma_k \varepsilon_g \eta_k \|g_k\| \, \|d_k\| + \gamma_k g_k^T d_k + \frac{L}{2}\gamma_k^2 \|d_k\|^2 \\ &= f(x_k) + \gamma_k(\varepsilon_g \eta_k \|g_k\| \, \|d_k\| + g_k^T d_k + \frac{L}{2}\gamma_k \|d_k\|^2) \\ &= f(x_k) + \gamma_k(\varepsilon_g \eta_k \|g_k\| \, \|d_k\| - \eta_k \|g_k\| \, \|d_k\| + \frac{L}{2}\frac{\eta_k\|g_k\|}{L\|d_k\|}\|d_k\|^2) \\ &= f(x_k) + \gamma_k(\varepsilon_g \eta_k \|g_k\| \, \|d_k\| - \eta_k \|g_k\| \, \|d_k\| + \frac{\eta_k\|g_k\| \, \|d_k\|}{2}) \\ &= f(x_k) + \gamma_k \eta_k \|g_k\| \, \|d_k\| (\varepsilon_g - 1/2) \\ &= f(x_k) - \frac{(\eta_k\|g_k\|)^2}{L}(1/2 - \varepsilon_g).\end{aligned}$$

Because

$$\|g_k\|^2 \geq \frac{2\mu(f(x_k) - f^*)}{(\varepsilon_g \eta_k + 1)^2},$$

then

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) - \frac{(\eta_k \|g_k\|)^2}{L}(1/2 - \varepsilon_g) \\ &\leq f(x_k) - \frac{2\mu}{L}\left(\frac{\eta_k}{\varepsilon_g \eta_k + 1}\right)^2 (1/2 - \varepsilon_g)(f(x_k) - f^*). \end{aligned}$$

Hence,

$$f(x_{k+1}) - f^* \leq \left(1 - \frac{2\mu}{L}\left(\frac{1}{2} - \varepsilon_g\right)\left(\frac{\eta_k}{\varepsilon_g \eta_k + 1}\right)^2\right)(f(x_k) - f^*).$$

By lemma 4.3.2,

$$f(x_{k+1}) - f^* \leq \left(1 - \frac{2\mu}{L}\left(\frac{1}{2} - \varepsilon_g\right)\left(\frac{\delta}{\varepsilon_g \delta + 1}\right)^2\right)(f(x_k) - f^*).$$

□

When the search direction is boosted, we can choose a step size that yields a multiplicative decrease in the optimality gap. This improvement is significantly stronger than the decrease obtained by the standard Frank–Wolfe algorithm or by non-boosted iterations, as characterized in Lemma 2.3.1. In this regard, boosted iterations provide a rate of improvement analogous to the good-step improvement of the away-step Frank–Wolfe algorithm described in Lemma 2.3.6. The precise improvement guarantee for boosted iterations is established in Lemma 4.3.6. For non-boosted iterations, the boosted Frank–Wolfe algorithm behaves identically to

the standard Frank–Wolfe algorithm. Consequently, the improvement obtained in a non-boosted iteration follows directly from Lemma 2.3.1 and relies on the good-gradient condition given in Definition 2.3.1. Throughout this chapter, Definition 2.3.1 may also be referred to as the *good gradient approximation for boosted Frank–Wolfe in non-boosted iterations.*

Lemma 4.3.6. *Suppose Assumptions 2.2.1 (L−smoothness) and 2.2.4 (strong convexity), and that we have a good gradient approximation Definition 2.3.1 (good gradient approximation for boosted Frank-Wolfe in non-boosted iterations) and 4.3.1 (good gradient approximation for boosted Frank-Wolfe in boosted iterations) at iteration k, then*

$$f(x_{k+1}) - f(x_k) = \begin{cases} -\beta^{FW}(\varepsilon)\varepsilon \ \textit{for non-boosted iterations} \\ -\beta^{bFW}\varepsilon \ \textit{for boosted iterations} \end{cases}$$

where β^{bFW} *is given by*

$$\beta^{bFW} = \frac{2\mu(1/2 - \varepsilon_g)\delta^2}{L(\varepsilon_g\delta + 1)^2}.$$

Proof. For non-boosted iterations, the result is the same as the proof of Lemma 2.3.1 with the decrease $\beta^{FW}(\varepsilon)\varepsilon$, while in the boosted iterations, we have the result from Lemma 4.3.5 with the decrease $\beta^{bFW}\varepsilon$. □

We now introduce an alternative good gradient approximation condition for the setting in which Assumption 4.3.1 (sufficiently boosted) holds. Under this assumption, sufficiently boosted iterations dominate the overall convergence behavior because they contribute substantially more to the decrease of the objective function.

As a result, unlike the standard Frank–Wolfe setting considered in Lemma 2.3.1, it is not necessary to use large sample sizes to guarantee sufficient objective decrease at every iteration, particularly during non-boosted iterations.

Instead, for non-boosted iterations, it is sufficient to select smaller sample sizes that merely ensure any increase in the objective value remains controlled. Since the overall convergence behavior is primarily driven by the boosted iterations, occasional modest increases in the objective function during non-boosted iterations can be tolerated without affecting the overall convergence guarantees.

Definition 4.3.2. Good gradient approximation for boosted Frank-Wolfe with sufficiently boosted assumption in non-boosted iterations. *Let D be the diameter of the feasible region $\mathcal{X}$. We say that the vector g_k is a good gradient approximation at iteration k if*

$$\|g_k - \nabla f(x_k)\| \leq \frac{\sqrt{\varepsilon}}{4D}. \tag{4.4}$$

Lemma 4.3.7. *Under Assumptions 2.2.1 ($L-$smoothness) and 2.2.4 (strong convexity), if we have a good gradient approximation at iteration k as stated in Definition 4.3.2 (good gradient approximation for boosted Frank-Wolfe with sufficiently boosted assumption in non-boosted iterations), then*

$$f(x_{k+1}) - f(x_k) \leq \frac{\omega \beta^{bFW} \varepsilon}{8}.$$

Proof. By convexity and Definition 4.3.2 (good gradient approximation for boosted

Frank-Wolfe with sufficiently boosted assumption in non-boosted iterations),

$$\begin{aligned} f(y) &\geq f(x_k) + \nabla f(x_k)^T (y - x_k) \\ &\geq f(x_k) + g_k^T (y - x_k) - \frac{\sqrt{\varepsilon}}{4D} ||y - x_k|| \\ &\geq f(x_k) + g_k^T (y - x_k) - \frac{\sqrt{\varepsilon}}{4}. \end{aligned}$$

Minimizing both sides, we have

$$f^* \geq f(x_k) + g_k^T (s_k - x_k) - \frac{\sqrt{\varepsilon}}{4}.$$

Then, by convexity and the Lipschitz property,

$$\begin{aligned} f(x_{k+1}) &\leq f(x_k) + \gamma_k \nabla f(x_k)^T (s_k - x_k) + \frac{L\gamma_k^2}{2} ||s_k - x_k||^2 \\ &\leq f(x_k) + \gamma_k g_k^T (s_k - x_k) + \gamma_k \frac{\sqrt{\varepsilon} ||s_k - x_k||}{4D} + \frac{L\gamma_k^2}{2} ||s_k - x_k||^2 \\ &\leq f(x_k) + \gamma_k (f^* - f(x_k) + \frac{\sqrt{\varepsilon}}{4}) + \gamma_k \frac{\sqrt{\varepsilon}}{4} + \frac{L\gamma_k^2}{2} D^2. \end{aligned}$$

Hence, we have the improvement

$$f(x_{k+1}) - f(x_k) \leq -\gamma_k (f(x_k) - f^*) + \frac{\gamma_k \sqrt{\varepsilon}}{2} + \frac{LD^2 \gamma_k^2}{2}.$$

By the adjusted step size policy

$$\gamma_k = \min \left\{ 1, \frac{1}{\sqrt{LD^2}} \right\} \frac{\omega \beta^{bFW} \sqrt{\varepsilon}}{8},$$

$$\frac{\gamma_k \sqrt{\varepsilon}}{2} \leq \frac{\sqrt{\varepsilon}}{2} \cdot \frac{\omega \beta^{bFW} \sqrt{\varepsilon}}{8} = \frac{\omega \beta^{bFW} \varepsilon}{16},$$

and,

$$\frac{LD^2\gamma_k^2}{2} \leq \frac{LD^2}{2} \cdot \frac{1}{LD^2} \left(\frac{\omega\beta^{bFW}}{8} \right)^2 \varepsilon \leq \frac{\omega\beta^{bFW}\varepsilon}{16}.$$

Then,

$$f(x_{k+1}) - f(x_k) \leq \frac{\omega\beta^{bFW}\varepsilon}{8}.$$

□

4.4 Complexity for a Bounded Feasible Region

We first establish the convergence analysis of the boosted Frank–Wolfe algorithm under Assumptions 2.2.1 (L-smoothness), 2.2.4 (strong convexity), and 4.3.1 (sufficiently boosted iterations). To obtain sufficiently accurate stochastic gradient estimates, the sample sizes must be chosen so that the estimated gradients satisfy the good gradient approximation conditions introduced in Definition 4.3.2 (good gradient approximation for boosted Frank-Wolfe with sufficiently boosted assumption in non-boosted iterations) and Definition 4.3.1 (good gradient approximation for boosted Frank-Wolfe in boosted iterations). Using these two conditions, we now introduce a unified definition involving the corresponding stochastic quantities below.

Definition 4.4.1. Good gradient approximation for boosted Frank-Wolfe with sufficiently boosted assumption. *We say, at iteration k, that G_k is a good gradient estimate for the standard Frank-Wolfe algorithm with at least probability p_g if the event*

$$E_k^{(bFW)} = \left\{ \frac{\|G_k - \nabla f(X_k)\|}{\|G_k\|} \leq \varepsilon_g\delta \text{ and } \|G_k - \nabla f(x_k)\| \leq \sqrt{\varepsilon}/4D \right\}$$

satisfies the condition $\mathbb{P}(E_k^{(bFW)}|\mathcal{F}_k) \geq p_g$ on the event $\{k < T_\varepsilon\}$.

Lemma 4.4.1. *Under Assumption 2.4.1 (bounded variance), and 4.3.1 (sufficiently boosted) the sample size required for each iteration to achieve a good gradient estimate as in Definition 4.4.1 (good gradient approximation for boosted Frank-Wolfe with sufficiently boosted assumption) for boosted Frank-Wolfe is* $\mathcal{O}(\varepsilon^{-1}(1-p_g)^{-1})$.

Proof. We first show that if $\|\nabla f(x) - g(x)\| \leq \varepsilon_g \delta \sqrt{\mu\varepsilon/2}$, then we have $\|g(x) - \nabla f(x)\|/\|g(x)\| \leq \varepsilon_g \delta$. Because $\varepsilon_g, \delta < 1$, then if $\|\nabla f(x) - g(x)\| \leq \varepsilon_g \delta \sqrt{\mu\varepsilon/2}$ we have

$$\sqrt{\frac{\mu\varepsilon}{2}} \geq \varepsilon_g \delta \sqrt{\frac{\mu\varepsilon}{2}} \geq \|\nabla f(x) - g(x)\| \geq \|\nabla f(x)\| - \|g(x)\|.$$

By the strong convexity assumption in 2.2.4, and Lemma 4.3.1, while Algorithm 4 is not terminated,

$$\|\nabla f(x)\|^2 \geq 2\mu(f(x_k) - f^*) > 2\mu\varepsilon.$$

Therefore,

$$\|g(x)\| \geq \|\nabla f(x)\| - \sqrt{\frac{\mu\varepsilon}{2}} \geq \sqrt{2\mu\varepsilon} - \sqrt{\frac{\mu\varepsilon}{2}} = \sqrt{\frac{\mu\varepsilon}{2}}.$$

Hence,

$$\frac{\|\nabla f(x) - g(x)\|}{\|g(x)\|} \leq \frac{\varepsilon_g \delta \sqrt{\frac{\mu\varepsilon}{2}}}{\sqrt{\frac{\mu\varepsilon}{2}}} = \varepsilon_g \delta.$$

Then, by Lemma 2.4.2, it is sufficient to select a sample size such that $\|\nabla f(x) - g(x)\| \leq \varepsilon_g \delta \sqrt{\mu\varepsilon/2}$, which requires sample size

$$n \geq \frac{2V_g}{\varepsilon_g^2 \delta^2 \mu\varepsilon(1-p_g)}, \tag{4.5}$$

and then $\|g(x) - \nabla f(x)\|/\|g(x)\| \leq \varepsilon_g \delta$. In order to satisfy the condition $\|g(x) -$

$\nabla f(x)\| \leq \sqrt{\varepsilon}/4D$, by Lemma 2.4.2 , we require the sample size

$$n \geq \frac{16V_g D^2}{\varepsilon(1-p_g)}.$$

Therefore selecting the sample size

$$n \geq \max\left\{\frac{16V_g D^2}{\varepsilon(1-p_g)},\ \frac{2V_g}{\varepsilon_g^2 \delta^2 \mu \varepsilon(1-p_g)}\right\} \sim \mathcal{O}(\varepsilon^{-1}(1-p_g)^{-1}) \tag{4.6}$$

is sufficient to satisfy good gradient estimation. □

We now establish the main complexity results. Theorem 4.4.1 provides an upper bound on the expected stopping time $\mathbb{E}[T_\varepsilon]$ under appropriate assumptions and sample size requirements. The key assumption is the sufficiently boosted assumption (Assumption 4.3.1), which requires that a sufficiently large proportion of the iterations are boosted, as defined in Definition 4.2.2. Since boosted iterations yield substantially larger improvements in the objective value than non-boosted iterations, their cumulative effect dominates the slower progress made during non-boosted iterations. As a result, the overall convergence rate matches the same order as that of the away-step Frank–Wolfe algorithm analyzed in Chapter 2.

Unlike the analysis of away-step Frank–Wolfe, which establishes improvement at every iteration, our analysis distinguishes between boosted and non-boosted iterations. We show that boosted iterations provide significant decreases in the objective value, while any increase incurred during non-boosted iterations is uniformly bounded. The combined effect of these two types of iterations yields a net decrease in the objective value and ultimately leads to fast convergence.

Theorem 4.4.1. *Suppose the initial solution $X_0 = x_0$ is deterministic with $f(x_0) - f^* > \varepsilon$. Let $T_\varepsilon = \inf\{k : f(X_k) - f^* \leq \varepsilon\}$. Using the boosted Frank-Wolfe algorithm under Assumptions 2.2.1 (smoothness), 2.2.4 (strong convexity) and 4.3.1 (sufficiently boosted), along with the sample size given in* (4.6) *from Lemma 4.4.1, $\mathbb{E}T_\varepsilon = \mathcal{O}(\varepsilon^{-1}(f(x_0) - f^*))$.*

Proof. Define Γ to be the maximum possible step size, which is bounded above by 1. Similar to Theorem 2.5.1, by (2.15), on the event $\{k < T_\varepsilon\}$,

$$\mathbb{E}[f(X_{k+1}) - f(X_k)|\mathcal{F}_k] \leq \mathbb{E}[(f(X_{k+1}) - f(X_k))\mathbb{1}_{\mathcal{G}}|\mathcal{F}_k] + \Gamma D\sqrt{\frac{V_g}{n_k}}\sqrt{1 - P_k} + \frac{L\Gamma^2 D^2}{2}(1 - P_k).$$

By choosing sample size n_k from Lemma 4.4.1 and (4.6) such that

$$n_k \geq \max\left\{\frac{16V_g D^2}{\varepsilon(1 - p_g)}, \frac{2V_g}{\varepsilon_g^2 \delta^2 \mu\varepsilon(1 - p_g)}\right\},$$

we have

$$\mathbb{E}[(f(X_{k+1}) - f(X_k))|\mathcal{F}_k] \leq \mathbb{E}[(f(X_{k+1}) - f(X_k))\mathbb{1}_{\mathcal{G}}|\mathcal{F}_k] + \Gamma D\kappa(1 - p_g) + \frac{L\Gamma^2 D^2}{2}(1 - p_g),$$

where

$$\kappa = \min\left\{\frac{\sqrt{\varepsilon}}{4D}, \varepsilon_g \delta\sqrt{\frac{\mu\varepsilon}{2}}\right\}.$$

Define a random variable B_k,

$$B_k = \begin{cases} 1 & \text{if } k^{th} \text{ iteration is boosted} \\ 0 & \text{otherwise.} \end{cases}$$

By Assumption 4.3.1 (sufficiently boosted), $\sum_{i=0}^{k-1} B_i \geq \omega k$. Define the stochastic process, similar to the one used with the standard Frank-Wolfe algorithm from (2.13), $\Phi^{(bFW)} = (\Phi_k^{(bFW)} : k \geq 0)$ adapted to the filtration $(\mathcal{F}_k : k \geq 0)$, where

$$\Phi_k^{(bFW)} = f(X_k) - f^*. \tag{4.7}$$

Define another stochastic process $M = (M_k : k \geq 0)$ by

$$M_k = \Phi_{k \wedge T_\varepsilon}^{(bFW)} + \frac{\beta^{bFW}\varepsilon}{2} \sum_{i=0}^{(k \wedge T_\varepsilon)-1} B_i - \frac{\beta^{bFW}\omega\varepsilon}{4} \sum_{i=0}^{(k \wedge T_\varepsilon)-1} (1 - B_i), \tag{4.8}$$

and $M_0 = \Phi_0^{(bFW)}$. By this definition, on the event $\{k < T_\varepsilon\}$,

$$\begin{aligned}
M_k &= \Phi_k^{(bFW)} + \frac{\beta^{bFW}\varepsilon}{2} \sum_{i=0}^{k-1} B_i - \frac{\beta^{bFW}\omega\varepsilon}{4} \sum_{i=0}^{k-1} (1 - B_i) \\
&\geq \frac{\beta^{bFW}\varepsilon}{2} \sum_{i=0}^{k-1} B_i - \frac{\beta^{bFW}\omega\varepsilon}{4} \sum_{i=0}^{k-1} (1 - B_i) \\
&= \frac{\beta^{bFW}\varepsilon}{2} \sum_{i=0}^{k-1} B_i + \frac{\beta^{bFW}\omega\varepsilon}{4} \sum_{i=0}^{k-1} (B_i - 1) \\
&\geq \frac{\beta^{bFW}\varepsilon}{2} \omega k + \frac{\beta^{bFW}\omega\varepsilon}{4} (\omega k - k) \\
&= \frac{\beta^{bFW}\varepsilon\omega k}{2} \left(1 - \frac{1-\omega}{2}\right) \\
&= k\omega(1+\omega)\frac{\beta^{bFW}\varepsilon}{4} \\
&\geq 0.
\end{aligned}$$

For the case $k \geq T_\varepsilon$, the analogous result holds where k is replaced by T_ε. Under the good gradient estimate $\mathcal{G}$ by Lemma 4.3.5, Lemma 4.3.7, and definition of

boosted and non-boosted from Definiton 4.2.2,

$$f(x_{k+1}) - f(x_k) = \begin{cases} \beta^{bFW}\omega\varepsilon/8 & \text{if iteration } k \text{ is non-boosted,} \\ -\beta^{bFW}\varepsilon & \text{if iteration } k \text{ is boosted.} \end{cases}$$

By the definition of M_k, on the event $\{k < T_\varepsilon\}$,

$$M_{k+1} - M_k = (f(X_{k+1}) - f(X_k)) + \frac{\beta^{bFW}\varepsilon}{2}B_k - \frac{\beta^{bFW}\omega\varepsilon}{4}(1 - B_k).$$

Therefore, under the good gradient estimate $\mathcal{G}$ and on the event $\{k < T_\varepsilon\}$,

$$M_{k+1} - M_k \leq \begin{cases} -\beta^{bFW}\varepsilon\omega/8 & \text{if } B_k = 0, \\ -\beta^{bFW}\varepsilon/2 & \text{if } B_k = 1. \end{cases}$$

Then, under the good gradient estimate $\mathcal{G}$ and on the event $\{k < T_\varepsilon\}$,

$$M_{k+1} - M_k \leq \max\left\{-\frac{\beta^{bFW}\varepsilon\omega}{8}, -\frac{\beta^{bFW}\varepsilon}{2}\right\} = -\frac{\beta^{bFW}\varepsilon\omega}{8}.$$

However, under the bad gradient estimate $\mathcal{G}'$ and on the event $\{k < T_\varepsilon\}$,

$$\begin{aligned} M_{k+1} - M_k &= (f(X_{k+1}) - f(X_k)) + \frac{\beta^{bFW}\varepsilon}{2}B_k - \frac{\beta^{bFW}\omega\varepsilon}{4}(1 - B_k) \\ &\leq (f(X_{k+1}) - f(X_k)) + \frac{\beta^{bFW}\varepsilon}{2} \\ &\leq (f(X_{k+1}) - f(X_k)) + \frac{\varepsilon}{2}. \end{aligned}$$

Taking the expectation on $M_{k+1} - M_k$ on the event $\{k < T_\varepsilon\}$, we have

$$\begin{aligned}
&\mathbb{E}[M_{k+1} - M_k | \mathcal{F}_k] \\
&= \mathbb{E}\left[(f(X_{k+1}) - f(X_k)) + \frac{\beta^{bFW}\varepsilon}{2} B_k - \frac{\beta^{bFW}\omega\varepsilon}{4}(1 - B_k)\middle| \mathcal{F}_k\right] \\
&\leq -\frac{\beta^{bFW}\varepsilon\omega}{8} p_g + \left(\Gamma D\kappa + \frac{L\Gamma^2 D^2}{2} + \frac{\varepsilon}{2}\right)(1 - p_g).
\end{aligned}$$

Pick the probability p_g such that

$$p_g \geq \frac{\Gamma D\kappa + L\Gamma^2 D^2/2 + \varepsilon/2}{\Gamma D\kappa + L\Gamma^2 D^2/2 + \varepsilon/2 + \beta^{bFW}\varepsilon\omega/8}. \tag{4.9}$$

Then, on the event $\{k < T_\varepsilon\}$, $\mathbb{E}[M_{k+1} - M_k | \mathcal{F}_k] \leq 0$. On the event $\{k \geq T_\varepsilon\}$, we have $M_{k+1} = M_k$. So, $(M_k : k \geq 0)$ is a non-negative super-martingale. By the Optional Stopping Theorem, and since $X_0 = x_0$ is deterministic with $f(x_0) > f^* + \varepsilon$,

$$\mathbb{E}[M_{T_\varepsilon}] \leq M_0.$$

Because $\Phi_k^{(bFW)} \geq 0$ for any $k \geq 0$, then

$$\begin{aligned}
\Phi_0^{(bFW)} &\geq \mathbb{E}\left[\Phi_k^{(bFW)} + \frac{\beta^{bFW}\varepsilon}{2}\sum_{i=0}^{T_\varepsilon - 1} B_i - \frac{\beta^{bFW}\omega\varepsilon}{4}\sum_{i=0}^{T_\varepsilon - 1}(1 - B_i)\right] \\
&= \frac{\beta^{bFW}\varepsilon}{2}\mathbb{E}\left[\sum_{i=0}^{T_\varepsilon - 1} B_i\right] - \frac{\beta^{bFW}\omega\varepsilon}{4}\mathbb{E}\left[\sum_{i=0}^{T_\varepsilon - 1}(1 - B_i)\right] \\
&= \frac{\beta^{bFW}\varepsilon}{2}\mathbb{E}\left[\sum_{i=0}^{T_\varepsilon - 1} B_i\right] + \frac{\beta^{bFW}\omega\varepsilon}{4}\mathbb{E}\left[\sum_{i=0}^{T_\varepsilon - 1}(B_i - 1)\right] \\
&\geq \frac{\beta^{bFW}\varepsilon}{2}\omega\mathbb{E}[T_\varepsilon] + \frac{\beta^{bFW}\omega\varepsilon}{4}(\omega - 1)\mathbb{E}[T_\varepsilon] \\
&= \frac{\beta^{bFW}\omega(1 + \omega)\varepsilon}{4}\mathbb{E}[T_\varepsilon].
\end{aligned}$$

Thus,

$$\mathbb{E}[T_\varepsilon] \leq \frac{4\Phi_0^{(bFW)}}{\beta^{bFW}\omega(1+\omega)\varepsilon} \sim \mathcal{O}(\varepsilon^{-1}(f(x_0) - f^*)).$$

□

Theorem 4.4.2. *The sample sizes in each iteration that are required to achieve a good gradient estimate for boosted Frank-Wolfe under Assumption 2.2.1 (L−smooth), Assumption 2.2.4 (strong convexity), Assumption 4.3.1 (sufficiently boosted) are*

1. $\mathcal{O}(\varepsilon^{-2})$ *under Assumption 2.4.1 (bounded variance).*
2. $\mathcal{O}(\varepsilon^{-1}\log\varepsilon^{-1})$ *under Assumption 2.4.2 (sub-Gaussian).*

Proof. By the sample size from (4.6) in Lemma 4.4.1, the required sample size is

$$n \geq \max\left\{\frac{16V_g D^2}{\varepsilon(1-p_g)}, \frac{2V_g}{\varepsilon_g^2\delta^2\mu\varepsilon(1-p_g)}\right\},$$

where the probability p_g is taken from Theorem 4.4.1 in (4.9) , where

$$p_g \geq \frac{\Gamma D\kappa + L\Gamma^2 D^2/2 + \varepsilon/2}{\Gamma D\kappa + L\Gamma^2 D^2/2 + \varepsilon/2 + \beta^{bFW}\varepsilon\omega/8}.$$

Under the bounded variance assumption from 2.4.1, we have a similar result for sample size complexity as in Theorem 2.5.3, which is $\mathcal{O}(L^2D^2\mu^{-1}\omega^{-1}\max\{D^2,\mu^{-1}\}\varepsilon^{-2}) \sim \mathcal{O}(\varepsilon^{-2})$.

With the sub-Gaussian assumption 2.4.2, we have a similar result as in Theorem 2.5.4. Using the same stochastic process M_k as defined in (4.8), but under

Assumption 2.4.2 (sub-Gaussian),

$$\mathbb{E}[M_{k+1}-M_k|\mathcal{F}_k] = -\frac{\beta^{bFW}\varepsilon\omega}{8}p_g+\left(\Gamma D\sqrt{\frac{2d}{n_k c}}\sqrt{1-p_g}+\frac{L\Gamma^2D^2}{2}(1-p_g)+\frac{\varepsilon}{2}(1-p_g)\right).$$

Pick the probability p_g so that

$$1-p_g \leq \min\left\{\frac{1}{2}, \frac{{\beta^{bFW}}^2\varepsilon^2 c}{2048d\Gamma^2D^2}, \frac{\beta^{bFW}\varepsilon}{16L\Gamma^2D^2+16\varepsilon}\right\} \tag{4.10}$$

and then

$$\begin{aligned}
-\frac{\beta^{bFW}\varepsilon}{8}p_g &\leq -\frac{\beta^{bFW}\varepsilon}{16},\\
\Gamma D\sqrt{\frac{2d}{n_k c}}\sqrt{1-p_g} &\leq \Gamma D\sqrt{\frac{2d}{c}}\sqrt{\frac{c}{2d}}\frac{\beta^{bFW}\varepsilon}{32\Gamma D} = \frac{\beta^{bFW}\varepsilon}{32}, \text{ and}\\
\left(\frac{L\Gamma^2D^2}{2}+\frac{\varepsilon}{2}\right)(1-p_g) &\leq \left(\frac{L\Gamma^2D^2}{2}+\frac{\varepsilon}{2}\right)\frac{\beta^{bFW}\varepsilon}{(16L\Gamma^2D^2+16\varepsilon)} \leq \frac{\beta^{bFW}\varepsilon}{32}.
\end{aligned}$$

Combining these three inequalities,

$$\mathbb{E}[M_{k+1}-M_k|\mathcal{F}_k] \leq -\frac{\beta^{bFW}\varepsilon}{16}+\frac{\beta^{bFW}\varepsilon}{32}+\frac{\beta^{bFW}\varepsilon}{32} = 0.$$

Then, we have the same iteration complexity as stated in Theorem 4.4.1 under the sub-Gaussian assumption. In addition, by Lemma 2.4.3 take $s = \min\{\varepsilon_g\delta\sqrt{\mu\varepsilon/2}, \sqrt{\varepsilon}/(4D)\}$ and select the sample size n so large such that $2de^{-nc\min\{\varepsilon_g^2\delta^2\mu\varepsilon/2, \varepsilon/(16D^2)\}}$ is an upper bound of $1-p_g$. Then,

$$n \geq \max\left\{\frac{2}{c\mu\varepsilon\varepsilon_g^2\delta^2}, \frac{16D^2}{c\varepsilon}\right\}\log\frac{2048d\Gamma^2D^2}{{\beta^{bFW}}^2\varepsilon^2 c} \sim \mathcal{O}\left(\frac{D^2}{\varepsilon}\log\frac{LD}{\varepsilon}\right) \sim \mathcal{O}(\varepsilon^{-1}\log\varepsilon^{-1}).$$

□

In summary, from Theorem 4.4.1 and Theorem 4.4.2, the total work complexity for boosted Frank-Wolfe algorithm under Assumptions 2.2.1 ($L-$smoothness), 2.2.4 (strong convexity), and 4.3.1 (sufficiently boosted) are

1. $\mathcal{O}(\varepsilon^{-3})$ under Assumption 2.4.1 (bounded variance), and
2. $\mathcal{O}(\varepsilon^{-2}\log\varepsilon^{-1})$ under Assumption 2.4.2 (sub-Gaussian)

The bound on the expected stopping time obtained in Theorem 4.4.1 can be expressed more explicitly as

$$\mathbb{E}[T_\varepsilon] \leq \frac{4(f(x_0)-f^*)}{\beta^{bFW}\omega(1+\omega)\varepsilon} \leq \frac{4(f(x_0)-f^*)}{\beta^{bFW}\omega\varepsilon} \sim \mathcal{O}\left(\frac{L}{\mu\omega\varepsilon}\right).$$

Thus, the iteration complexity improves as the proportion of boosted iterations, represented by ω, increases. Similarly, under Assumption 2.4.1 (bounded variance), Theorem 4.4.2 yields the sample size complexity

$$\mathcal{O}\left(\frac{V_g L^2 D^2}{\mu\omega}\max\left\{D^2, \frac{1}{\mu}\right\}\frac{1}{\varepsilon^2}\right),$$

while under Assumption 2.4.2 (sub-Gaussian), the sample size complexity becomes

$$\mathcal{O}\left(\frac{1}{\varepsilon}\max\left\{D^2, \frac{1}{\mu}\right\}\log\frac{dDL}{\mu\varepsilon}\right).$$

Combining the iteration and sample size complexities gives the corresponding total work complexities. Under the bounded-variance assumption, the total work complexity is

$$\mathcal{O}\left(\frac{V_g L^3 D^2}{\mu^2\omega^2}\max\left\{D^2, \frac{1}{\mu}\right\}\frac{1}{\varepsilon^3}\right),$$

whereas under the sub-Gaussian assumption, it is

$$\mathcal{O}\left(\frac{L}{\mu\omega\varepsilon^2}\max\left\{D^2,\frac{1}{\mu}\right\}\log\frac{dDL}{\mu\varepsilon}\right).$$

These expressions highlight the dependence of the overall complexity on the smoothness constant L, the variance parameter V_g the strong convexity parameter μ, the feasible-region diameter D, dimension d, and the proportion of boosted iterations ω.

In [21], the deterministic boosted Frank–Wolfe algorithm achieves an iteration complexity of $\mathcal{O}(\log\varepsilon^{-1})$ under the sufficiently boosted assumption, since each iteration yields a multiplicative decrease in the objective value, similar to the away-step Frank–Wolfe algorithm analyzed in [53] and [54]. In contrast, in the stochastic setting, the need to estimate gradients introduces additional uncertainty and slows the convergence rate. As a result, we obtain an iteration complexity of $\mathcal{O}(\varepsilon^{-1})$. Furthermore, to maintain sufficiently accurate gradient approximations and guarantee the required decrease conditions, the sample size complexity is $\mathcal{O}(\varepsilon^{-2})$ under a bounded-variance assumption and $\mathcal{O}(\varepsilon^{-1}\log\varepsilon^{-1})$ under a sub-Gaussian assumption, as established in Theorem 4.4.1 and Theorem 4.4.2.

Next, we establish convergence results for the boosted Frank–Wolfe algorithm without Assumption 4.3.1 (sufficiently boosted iterations). When this assumption does not hold, non-boosted iterations may become dominant, and the algorithm effectively behaves similarly to the standard Frank–Wolfe method, since non-boosted steps closely resemble those of Algorithm 1. In this regime, the convergence behavior is therefore comparable to that of the standard Frank–Wolfe algorithm, and we recover results analogous to those presented in Theorem 2.5.2 and The-

orem 2.5.4. We define a good gradient approximation event without relying on Assumption 4.3.1 (sufficiently boosted iterations), in order to ensure sufficient decrease in both boosted and non-boosted iterations. As a consequence, this setting requires larger sample sizes compared to the case where the sufficiently boosted assumption holds.

Definition 4.4.2 (good gradient approximation for boosted Frank-Wolfe without sufficiently boosted assumption) below combines the notions of a good gradient estimate for boosted iterations from Definition 4.3.1 (good gradient approximation for boosted Frank-Wolfe in boosted iterations) and for standard Frank–Wolfe iterations from Definition 2.3.1 (or good gradient approximation for boosted Frank-Wolfe in non-boosted iterations). This is because, during non-boosted iterations, the boosted Frank–Wolfe algorithm reduces to the standard Frank–Wolfe algorithm and therefore requires the corresponding gradient accuracy condition.

Definition 4.4.2. Good gradient approximation for boosted Frank-Wolfe without sufficiently boosted assumption. *We say, at iteration k, that G_k is a good gradient estimate for the boosted Frank-Wolfe algorithm with at least probability p_g if the event*

$$E_k^{(bFW)} = \left\{ \frac{\|G_k - \nabla f(X_k)\|}{\|G_k\|} \leq \varepsilon_g \delta \quad \textit{and} \quad \|G_k - \nabla f(x_k)\| \leq \varepsilon/4D \right\}$$

satisfies the condition $\mathbb{P}(E_k^{(bFW)} | \mathcal{F}_k) \geq p_g$ on the event $\{k < T_\varepsilon\}$.

Lemma 4.4.2. *Under Assumption 2.4.1 (bounded variance), the sample size required for each iteration to achieve a good gradient estimate for boosted Frank-Wolfe is $\mathcal{O}(\varepsilon^{-2}(1-p_g)^{-1})$.*

Proof. We first show that if $\|\nabla f(x) - g(x)\| \leq \varepsilon_g \delta \sqrt{\mu\varepsilon/2}$, then we have $\|g(x) -$

$\nabla f(x)\|/\|g(x)\| \leq \varepsilon_g\delta$. Because $\varepsilon_g, \delta < 1$, then if $\|\nabla f(x) - g(x)\| \leq \varepsilon_g\delta\sqrt{\mu\varepsilon/2}$ we have

$$\sqrt{\frac{\mu\varepsilon}{2}} \geq \varepsilon_g\delta\sqrt{\frac{\mu\varepsilon}{2}} \geq \|\nabla f(x) - g(x)\| \geq \|\nabla f(x)\| - \|g(x)\|.$$

By strong convexity assumption in 2.2.4, and Lemma 4.3.1, while the Algorithm 4 is not terminated,

$$\|\nabla f(x)\|^2 \geq 2\mu(f(x_k) - f^*) > 2\mu\varepsilon.$$

Therefore,

$$\|g(x)\| \geq \|\nabla f(x)\| - \sqrt{\frac{\mu\varepsilon}{2}} \geq \sqrt{2\mu\varepsilon} - \sqrt{\frac{\mu\varepsilon}{2}} = \sqrt{\frac{\mu\varepsilon}{2}}.$$

Hence,

$$\frac{\|\nabla f(x) - g(x)\|}{\|g(x)\|} \leq \frac{\varepsilon_g\delta\sqrt{\frac{\mu\varepsilon}{2}}}{\sqrt{\frac{\mu\varepsilon}{2}}} = \varepsilon_g\delta.$$

Then, by Lemma 2.4.2, it is sufficient to select sample size such that $\|\nabla f(x) - g(x)\| \leq \varepsilon_g\delta\sqrt{\mu\varepsilon/2}$, which requires sample size

$$n \geq \frac{2V_g}{\varepsilon_g^2\delta^2\mu\varepsilon(1-p_g)} \tag{4.11}$$

so that $\|g(x) - \nabla f(x)\|/\|g(x)\| \leq \varepsilon_g\delta$ holds. By the sample size of the standard Frank-Wolfe algorithm from (2.8), selecting the sample size such that

$$n \geq \frac{16V_gD^2}{\varepsilon^2(1-p_g)},$$

we have $\|g(x) - f(x)\| \leq \varepsilon/4D$. Hence, selecting the maximum of the two sample sizes, i.e. selecting n such that

$$n \geq \max\left\{\frac{16V_gD^2}{\varepsilon^2(1-p_g)}, \frac{2V_g}{\varepsilon_g^2\delta^2\mu\varepsilon(1-p_g)}\right\} \tag{4.12}$$

which is in the order of $\mathcal{O}(\varepsilon^{-2}(1-p_g)^{-1})$ is sufficient to satisfy the good gradient approximation of Definition 4.4.2 (good gradient approximation for boosted Frank-Wolfe without boosted assumption). □

Theorem 4.4.3. *Suppose the initial solution $X_0 = x_0$ is deterministic with $f(x_0) - f^* > \varepsilon$. Let $T_\varepsilon = \inf\{k : f(X_k) - f^* \leq \varepsilon\}$. Using the boosted Frank-Wolfe algorithm under Assumptions 2.2.1 (L−smooth) and 2.2.4 (strong convexity), along with the sample size from* (4.12)*, $\mathbb{E}T_\varepsilon = \mathcal{O}(\varepsilon^{-2}(f(x_0) - f^*))$.*

Proof. Define Γ to be the maximum possible step size, which is bounded above by 1. Similar to Theorem 2.5.1, by (2.15),

$$\mathbb{E}[f(X_{k+1}) - f(X_k)|\mathcal{F}_k] \leq \mathbb{E}[(f(X_{k+1}) - f(X_k))\mathbb{1}_{\mathcal{G}}|\mathcal{F}_k] + \Gamma D\sqrt{\frac{V_g}{n_k}}\sqrt{1-P_k} + \frac{L\Gamma^2 D^2}{2}(1-P_k).$$

where $\mathcal{G}$ is the event of good gradient approximation for boosted Frank-Wolfe without sufficiently boosted assumption as defined in 4.4.2. Let $\mathcal{B}$ be the event of boosted search direction, then by Lemma 4.3.6,

$$\begin{aligned}
\mathbb{E}[(f(X_{k+1}) - f(X_k))\mathbb{1}_{\mathcal{G}}|\mathcal{F}_k] &= \mathbb{E}[(f(X_{k+1}) - f(X_k))\mathbb{1}_{\mathcal{G}}(\mathbb{1}_{\mathcal{B}} + \mathbb{1}_{\mathcal{B}'})|\mathcal{F}_k] \\
&\leq -\beta^{bFW}\varepsilon\mathbb{E}[\mathbb{1}_{\mathcal{G}}\mathbb{1}_{\mathcal{B}}|\mathcal{F}_k] - \beta^{FW}(\varepsilon)\varepsilon\mathbb{E}[\mathbb{1}_{\mathcal{G}}\mathbb{1}_{\mathcal{B}'}|\mathcal{F}_k] \\
&= -\beta^{bFW}\varepsilon p_g\mathbb{P}(\mathcal{B}|\mathcal{F}_k, \mathcal{G}) - \beta^{FW}(\varepsilon)\varepsilon p_g\mathbb{P}(\mathcal{B}'|\mathcal{F}_k, \mathcal{G})
\end{aligned}$$

By choosing sample size n_k from Lemma 4.4.2 and (4.12) such that

$$n_k \geq \max\left\{\frac{16V_g D^2}{\varepsilon^2(1-p_g)}, \frac{2V_g}{\varepsilon_g^2\delta^2\mu\varepsilon(1-p_g)}\right\},$$

then

$$\mathbb{E}[(f(X_{k+1}) - f(X_k))|\mathcal{F}_k] \leq -\beta^{bFW}\varepsilon p_g \mathbb{P}(\mathcal{B}|\mathcal{F}_k, \mathcal{G}) - \beta^{FW}(\varepsilon)\varepsilon p_g \mathbb{P}(\mathcal{B}'|\mathcal{F}_k, \mathcal{G}) + \Gamma D\kappa(1 - p_g) + \frac{L\Gamma^2 D^2}{2}(1 - p_g), \quad (4.13)$$

where

$$\kappa = \min\left\{\frac{\varepsilon}{4D}, \varepsilon_g \delta\sqrt{\frac{\mu\varepsilon}{2}}\right\}.$$

Recall that $B_k = 1$ if the k^{th} iteration's search direction is boosted and $B_k = 0$ otherwise. Consider the stochastic process

$$\frac{\beta^{bFW}\varepsilon}{2}B_{k\wedge T_\varepsilon} + \frac{\beta^{FW}(\varepsilon)\varepsilon}{2}(1 - B_{k\wedge T_\varepsilon}).$$

Taking the expectation under the event that $k < T_\varepsilon$,

$$\begin{aligned}
&\mathbb{E}\left[\frac{\beta^{bFW}\varepsilon}{2}B_{k\wedge T_\varepsilon} + \frac{\beta^{FW}(\varepsilon)\varepsilon}{2}(1 - B_{k\wedge T_\varepsilon})\middle|\mathcal{F}_k\right] \\
&= \mathbb{E}\left[\left(\frac{\beta^{bFW}\varepsilon}{2}B_{k\wedge T_\varepsilon} + \frac{\beta^{FW}(\varepsilon)\varepsilon}{2}(1 - B_{k\wedge T_\varepsilon})\right)(\mathbb{1}_{\mathcal{G}} + \mathbb{1}_{\mathcal{G}'})\middle|\mathcal{F}_k\right]
\end{aligned}$$

Because $\beta^{bFW}, \beta^{FW}(\varepsilon)\varepsilon < 1$, then

$$\left(\frac{\beta^{bFW}\varepsilon}{2}B_{k\wedge T_\varepsilon} + \frac{\beta^{FW}(\varepsilon)\varepsilon}{2}(1 - B_{k\wedge T_\varepsilon})\right)\mathbb{1}_{\mathcal{G}'} \leq \max\left\{\frac{\beta^{bFW}\varepsilon}{2}, \frac{\beta^{FW}(\varepsilon)\varepsilon}{2}\right\}\mathbb{1}_{\mathcal{G}'} \leq \frac{\varepsilon}{2}\mathbb{1}_{\mathcal{G}'}.$$

Therefore,

$$\begin{aligned}
&\mathbb{E}\left[\frac{\beta^{bFW}\varepsilon}{2}B_{k\wedge T_\varepsilon}+\frac{\beta^{FW}(\varepsilon)\varepsilon}{2}(1-B_{k\wedge T_\varepsilon})\middle|\mathcal{F}_k\right]\\
&\leq\ \mathbb{E}\left[\left(\frac{\beta^{bFW}\varepsilon}{2}B_{k\wedge T_\varepsilon}+\frac{\beta^{FW}(\varepsilon)\varepsilon}{2}(1-B_{k\wedge T_\varepsilon})\right)\mathbb{1}_{\mathcal{G}}(\mathbb{1}_{\mathcal{B}}+\mathbb{1}_{\mathcal{B}'})\middle|\mathcal{F}_k\right]+\frac{\varepsilon}{2}\mathbb{E}[\mathbb{1}_{\mathcal{G}'}|\mathcal{F}_k]\\
&\leq\ \mathbb{E}\left[\left(\frac{\beta^{bFW}\varepsilon}{2}B_{k\wedge T_\varepsilon}+\frac{\beta^{FW}(\varepsilon)\varepsilon}{2}(1-B_{k\wedge T_\varepsilon})\right)\mathbb{1}_{\mathcal{G}}(\mathbb{1}_{\mathcal{B}}+\mathbb{1}_{\mathcal{B}'})\middle|\mathcal{F}_k\right]+\frac{\varepsilon}{2}(1-p_g)\\
&\leq\ \frac{\beta^{bFW}\varepsilon}{2}p_g\mathbb{P}(\mathcal{B}|\mathcal{F}_k,\mathcal{G})+\frac{\beta^{FW}(\varepsilon)\varepsilon}{2}\mathbb{P}(\mathcal{B}'|\mathcal{F}_k,\mathcal{G})+\frac{\varepsilon}{2}(1-p_g).
\end{aligned}$$

Hence,

$$\begin{aligned}
\mathbb{E}\left[\frac{\beta^{bFW}\varepsilon}{2}B_{k\wedge T_\varepsilon}+\frac{\beta^{FW}(\varepsilon)\varepsilon}{2}(1-B_{k\wedge T_\varepsilon})\middle|\mathcal{F}_k\right]&\leq\frac{\beta^{bFW}\varepsilon}{2}p_g\mathbb{P}(\mathcal{B}|\mathcal{F}_k,\mathcal{G})\\
&\quad+\frac{\beta^{FW}(\varepsilon)\varepsilon}{2}\mathbb{P}(\mathcal{B}'|\mathcal{F}_k,\mathcal{G})+\frac{\varepsilon}{2}(1-p_g).\quad(4.14)
\end{aligned}$$

Consider a stochastic process $\Phi^{(bFW)}=(\Phi_k^{(bFW)}:k\geq 0)$, adapted to $(\mathcal{F}_k:k\geq 0)$ where $\Phi_k^{(bFW)}=f(X_k)-f^*$, and define a stochastic process $M=(M_k:k\geq 0)$, where

$$M_k=\Phi_{k\wedge T_\varepsilon}^{(bFW)}+\left(\frac{\beta^{FW}(\varepsilon)\varepsilon}{2}\sum_{i=0}^{(k\wedge T_\varepsilon)-1}(1-B_i)+\frac{\beta^{bFW}\varepsilon}{2}\sum_{i=0}^{(k\wedge T_\varepsilon)-1}B_i\right)\qquad(4.15)$$

and $M_0 = \Phi_0^{(bFW)}$. Then, by (4.13) and (4.14), on the event $k < T_\varepsilon$,

$$
\begin{aligned}
&\mathbb{E}[M_{k+1} - M_k | \mathcal{F}_k] \\
&= \mathbb{E}[\Phi^{(bFW)}_{(k+1)\wedge T_\varepsilon} - \Phi^{(bFW)}_k | \mathcal{F}_k] + \mathbb{E}\left[\frac{\beta^{FW}(\varepsilon)\varepsilon}{2}(1 - B_{k\wedge T_\varepsilon}) + \frac{\beta^{bFW}\varepsilon}{2} B_{k\wedge T_\varepsilon} \middle| \mathcal{F}_k\right] \\
&= \mathbb{E}[f(X_{k+1}) - f(X_k) | \mathcal{F}_k] + \mathbb{E}\left[\frac{\beta^{FW}(\varepsilon)\varepsilon}{2}(1 - B_k) + \frac{\beta^{bFW}\varepsilon}{2} B_k \middle| \mathcal{F}_k\right] \\
&\leq -\frac{\beta^{bFW}\varepsilon}{2} p_g \mathbb{P}(\mathcal{B} | \mathcal{F}_k, \mathcal{G}) - \frac{\beta^{FW}(\varepsilon)\varepsilon}{2} p_g \mathbb{P}(\mathcal{B}' | \mathcal{F}_k, \mathcal{G}) + \left(\Gamma D\kappa + \frac{L\Gamma^2 D^2}{2} + \frac{\varepsilon}{2}\right)(1 - p_g).
\end{aligned}
$$

Because the probability of the event $\mathcal{B}$ and $\mathcal{B}'$ given $\mathcal{F}_k, \mathcal{G}$ sum up to 1, then

$$
\mathbb{E}[M_{k+1} - M_k | \mathcal{F}_k] \leq -\min\left\{\frac{\beta^{bFW}\varepsilon}{2}, \frac{\beta^{FW}(\varepsilon)\varepsilon}{2}\right\} p_g + \left(\Gamma D\kappa + \frac{L\Gamma^2 D^2}{2} + \frac{\varepsilon}{2}\right)(1 - p_g). \tag{4.16}
$$

By selecting probability p_g such that

$$
p_g \geq \frac{\Gamma D\kappa + \frac{L\Gamma^2 D^2}{2} + \frac{\varepsilon}{2}}{\Gamma D\kappa + \frac{L\Gamma^2 D^2}{2} + \frac{\varepsilon}{2} + \min\left\{\frac{\beta^{bFW}\varepsilon}{2}, \frac{\beta^{FW}(\varepsilon)\varepsilon}{2}\right\}} \tag{4.17}
$$

then

$$
\mathbb{E}[M_{k+1} - M_k | \mathcal{F}_k] \leq 0.
$$

On the event $T_\varepsilon \leq k$, $M_{k+1} = M_k$, so $(M_k : k \geq 0)$ is a non-negative super-martingale. By the optional stopping theorem, and since $X_0 = x_0$ is deterministic with $f(x_0) > f^* + \varepsilon$,

$$
\mathbb{E}[M_{T_\varepsilon}] \leq \mathbb{E}[M_0] = \Phi_0^{(bFW)}.
$$

So, $\mathbb{E}[\Phi^{(bFW)}_{T_\varepsilon} + (\beta^{FW}(\varepsilon)\varepsilon/2)(T_\varepsilon - S^B_{T_\varepsilon}) + (\beta^{bFW}\varepsilon/2) S^B_{T_\varepsilon}] \leq \Phi_0^{(bFW)}$ where $S^B_k = \sum_{i=0}^{k-1} B_k$ is the total number of boosted iterations until iteration k. Since $\Phi^{(bFW)}_{T_\varepsilon} \geq$

0, $\Phi_0^{(bFW)} < \infty$. Therefore, for the boosted Frank-Wolfe algorithm, we have

$$\begin{aligned}\Phi_0^{(bFW)} &\geq \mathbb{E}[\Phi_{T_\varepsilon}^{(bFW)} + (\beta^{FW}(\varepsilon)\varepsilon/2)(T_\varepsilon - S_{T_\varepsilon}^B) + (\beta^{bFW}\varepsilon/2)S_{T_\varepsilon}^B] \\ &\geq \mathbb{E}[\min\{\beta^{FW}(\varepsilon)\varepsilon/2, \beta^{bFW}\varepsilon/2\}T_\varepsilon].\end{aligned}$$

Thus,

$$\mathbb{E}[T_\varepsilon] \leq \frac{\Phi_0^{(bFW)}}{\min\{\beta^{FW}(\varepsilon)\varepsilon/2, \beta^{bFW}\varepsilon/2\}} \sim \mathcal{O}\left(LD^2\frac{f(x_0) - f^*}{\varepsilon^2}\right) \sim \mathcal{O}\left(\frac{f(x_0) - f^*}{\varepsilon^2}\right).$$

In the case that the Assumption 4.3.1 (sufficiently boosted) holds, $S_{T_\varepsilon}^B \geq \omega T_\varepsilon$, then

$$\mathbb{E}[(\beta^{bFW}/2)\omega T_\varepsilon] \leq \Phi_0^{(bFW)}.$$

Thus,

$$\mathbb{E}[T_\varepsilon] \leq \left(\frac{2\Phi_0^{(bFW)}}{\omega\beta^{bFW}\varepsilon}\right) \sim \mathcal{O}\left(\frac{f(x_0) - f^*}{\varepsilon}\right),$$

which is the same order obtained in Theorem 4.4.1. □

The sample size complexity of the boosted Frank–Wolfe algorithm (Algorithm 4) is the same as that of the standard Frank–Wolfe algorithm, since it requires at least the same sample sizes to obtain sufficiently accurate gradient approximations during non-boosted iterations.

Theorem 4.4.4. *The sample sizes in each iteration that are required to achieve a good gradient estimate for boosted Frank-Wolfe under Assumption 2.2.1 (L−smooth), Assumption 2.2.4 (strong convexity) without assuming sufficient boosting are*

1. $\mathcal{O}(\varepsilon^{-4})$ *under Assumption 2.4.1 (bounded variance)*

2. $\mathcal{O}(\varepsilon^{-2}\log\varepsilon^{-1})$ *under Assumption 2.4.2 (sub-Gaussian)*

Proof. From (4.12), the required sample size is

$$n \geq \max\left\{\frac{16V_gD^2}{\varepsilon^2(1-p_g)}, \frac{2V_g}{\varepsilon_g^2\delta^2\mu\varepsilon(1-p_g)}\right\},$$

where the probability p_g is taken from (4.17) in Theorem 4.4.3,

$$p_g \geq \frac{\Gamma D\kappa + \frac{L\Gamma^2D^2}{2} + \frac{\varepsilon}{2}}{\Gamma D\kappa + \frac{L\Gamma^2D^2}{2} + \frac{\varepsilon}{2} + \min\left\{\frac{\beta^{bFW}\varepsilon}{2}, \frac{\beta^{FW}(\varepsilon)\varepsilon}{2}\right\}}.$$

Under the bounded variance assumption from 2.4.1, we have a similar result for sample size complexity as in Theorem 2.5.3, which is $\mathcal{O}(V_gL^2D^6\varepsilon^{-4}) \sim \mathcal{O}(\varepsilon^{-4})$. With the sub-Gaussian assumption 3.2.2, we have a similar result as (4.16) in Theorem 4.4.3. Using the same stochastic process M_k as defined in (4.15), similar to (4.16) but under Assumption 3.2.2 (sub-Gaussian) we have ,

$$\mathbb{E}[M_{k+1}-M_k|\mathcal{F}_k] \leq -\min\left\{\frac{\beta^{bFW}\varepsilon}{2}, \frac{\beta^{FW}(\varepsilon)\varepsilon}{2}\right\}p_g + \Gamma D\sqrt{\frac{2d}{n_kc}}\sqrt{1-p_g} + \frac{L\Gamma^2D^2}{2}(1-p_g).$$

Similar to the proof of Theorem 2.5.4, pick the probability p_g so that

$$1-p_g \leq \min\left\{\frac{1}{3}, \frac{m_\varepsilon^2c}{18d\Gamma^2D^2}, \frac{2m_\varepsilon}{3L\Gamma^2D^2}\right\} \tag{4.18}$$

where

$$m_\varepsilon = \min\left\{\frac{\beta^{bFW}\varepsilon}{2}, \frac{\beta^{FW}(\varepsilon)\varepsilon}{2}\right\},$$

so that

$$\begin{aligned} m_\varepsilon(1-p_g) &\le \frac{m_\varepsilon}{3}, \\ \Gamma D\sqrt{\frac{2d}{n_k c}}\sqrt{1-p_g} \le \Gamma D\sqrt{\frac{2d}{c}}\sqrt{\frac{c}{2d}}\frac{m_\varepsilon}{3\Gamma D} &= \frac{m_\varepsilon}{3}, \text{ and} \\ \frac{L\Gamma^2 D^2}{2}(1-p_g) &\le \frac{m_\varepsilon}{3}. \end{aligned}$$

Combining these three inequalities,

$$\mathbb{E}[M_{k+1} - M_k|\mathcal{F}_k] \le -\frac{2m_\varepsilon}{3} + \frac{m_\varepsilon}{3} + \frac{m_\varepsilon}{3} = 0.$$

Then, similar to Theorem 4.4.3, we have the same iteration complexity under sub-Gaussian assumption. In addition, by Lemma 2.4.3 take $s = \min\{\varepsilon_g\delta\sqrt{\mu\varepsilon/2}, \varepsilon/(4D)\}$ and select the sample size n so large that $2de^{-nc\min\{\varepsilon_g^2\delta^2\mu\varepsilon/2, \varepsilon^2/(16D^2)\}}$ is a lower bound of $1-p_g$. The term involving m_ε^2 or $\min\{\beta^{bFW}\varepsilon/2, \beta^{FW}(\varepsilon)\varepsilon/2\}^2$ is the determining factor for small ε, which gives

$$n \ge \max\left\{\frac{2}{c\mu\varepsilon\varepsilon_g^2\delta^2}, \frac{16D^2}{c\varepsilon^2}\right\}\log\frac{36d^2\Gamma^2D^2}{m_\varepsilon^2 c} \sim \mathcal{O}\left(\frac{D^2}{\varepsilon^2}\log\frac{LD}{\varepsilon}\right) \sim \mathcal{O}\left(\varepsilon^{-2}\log\varepsilon^{-1}\right).$$

□

In summary, from Theorem 4.4.3 and Theorem 4.4.4, the total work complexity for boosted Frank-Wolfe algorithm under Assumptions 2.2.1 ($L-$smoothness) and 2.2.4 (strong convexity) are

1. $\mathcal{O}(\varepsilon^{-6})$ under Assumption 2.4.1 (bounded variance)
2. $\mathcal{O}(\varepsilon^{-4}\log\varepsilon^{-1})$ under Assumption 2.4.2 (sub-Gaussian)

The bound on the expected stopping time obtained in Theorem 4.4.3 can be expressed more explicitly as

$$\mathbb{E}[T_\varepsilon] \leq \frac{2(f(x_0) - f^*)}{\min\{\beta^{FW}(\varepsilon)\varepsilon, \beta^{bFW}\varepsilon\}} \sim \mathcal{O}\left(\max\left\{\frac{LD^2}{\varepsilon^2}, \frac{L}{\mu\varepsilon}\right\}\right).$$

Similarly, under Assumption 2.4.1 (bounded variance), Theorem 4.4.4 yields the sample size complexity

$$\mathcal{O}\left(V_g L^2 D^6 \varepsilon^{-4}\right),$$

while under Assumption 2.4.2 (sub-Gaussian), the sample size complexity becomes

$$\mathcal{O}\left(\max\left\{\frac{D^2}{\varepsilon^2}, \frac{1}{\mu\varepsilon}\right\} \log dD \max\left\{\frac{L^2}{\mu^2\varepsilon^2}, \frac{L^2D^4}{\varepsilon^4}\right\}\right).$$

Combining the iteration and sample size complexities gives the corresponding total work complexities. Under the bounded-variance assumption, the total work complexity is

$$\mathcal{O}\left(V_g L^3 D^8 \varepsilon^{-6}\right),$$

whereas under the sub-Gaussian assumption, it is

$$\mathcal{O}\left(\frac{LD^4}{\varepsilon^4} \log \frac{dDL}{\varepsilon}\right).$$

These expressions highlight the dependence of the overall complexity on the smoothness constant L, the variance parameter V_g the strong convexity parameter μ, the feasible-region diameter D and the dimension d.

4.5 Extensions of Bounded Boosted Frank-Wolfe

This section presents an extended convergence analysis of the boosted Frank–Wolfe algorithm. The analysis parallels that of Chapter 3, including biased gradient estimators, light-tail properties of the stopping time, and the interior optimum setting. Under Assumption 4.3.1 (sufficiently boosted iterations), the results are similar to those obtained for the Away-step Frank–Wolfe algorithm discussed in Chapter 3. In contrast, when this assumption does not hold, we recover convergence results that are analogous to those of the standard Frank–Wolfe algorithm.

4.5.1 Complexity with Biased Gradient Estimators

In the presence of biased gradient estimates, the same complexity results as in Corollary 3.2.1 and Corollary 3.2.2 for the standard Frank–Wolfe algorithm are expected to hold. This is because non-boosted iterations may dominate the overall iteration count, effectively reducing the method to standard Frank–Wolfe behavior. First, the target accuracy ε must be replaced by a modified tolerance ε', defined as the maximum of the error levels associated with boosted and non-boosted iterations, allowing the algorithm to proceed consistently under this unified criterion. Second, selecting the sample size for boosted iterations according to (4.11) alone is insufficient, as it does not guarantee accurate gradient estimates during non-boosted iterations. Instead, choosing the sample size according to (4.12) ensures uniformly good gradient estimates across all iterations.

Lemma 4.5.1. *Suppose that Assumption 3.2.1 (bounded variance) holds, and also suppose that in a non-boosted iteration of boosted Frank-Wolfe algorithm we have*

$$\|\nabla f(x_k) - g(x_k)\| \leq \varepsilon_g \delta \sqrt{\frac{\mu\varepsilon}{2}} + B,$$

then, we have the upper bound of the objective function's difference

$$f(x_{k+1}) - f(x_k) \leq \frac{\beta^{bFW}\varepsilon'\omega}{8}$$

for a new error ε' *such that*

$$\varepsilon' = \frac{1}{2\mu}\left(\sqrt{2\mu\varepsilon} + B\left(\frac{1+\varepsilon_g\delta}{\varepsilon_g\delta}\right)\right)^2$$

Proof. By Lemma 2.3.1, we have an analogous result where

$$f^* \geq f(x_k) + g_k^T(s_k - x_k) - \|f(x_k) - g_k\|D,$$

and,

$$f(x_{k+1}) - f(x_k) \leq \gamma_k(f^* - f(x_k)) + 2\gamma_k D\|f(x_k) - g_k\| + \frac{LD^2\gamma_k^2}{2}.$$

By imposing the gradient estimation error where

$$\|\nabla f(x) - g(x)\| \leq \varepsilon_g\delta\sqrt{\frac{\mu\varepsilon}{2}} + B,$$

we have,

$$\begin{aligned}
\|\nabla f(x) - g(x)\| &\leq \varepsilon_g \delta \sqrt{\frac{\mu\varepsilon}{2}} + B \\
&= \varepsilon_g \delta \sqrt{\frac{\mu}{2}} \left(\sqrt{\varepsilon} + B\sqrt{\frac{2}{\mu}} \cdot \frac{1}{\varepsilon_g \delta} \right) \\
&\leq \varepsilon_g \delta \sqrt{\frac{\mu}{2}} \left(2\sqrt{\varepsilon} + B\sqrt{\frac{2}{\mu}} \cdot \left(\frac{1 + \varepsilon_g \delta}{\varepsilon_g \delta} \right) \right) \\
&= \varepsilon_g \delta \sqrt{2\mu\varepsilon'}
\end{aligned}$$

Hence,

$$f(x_{k+1}) - f(x_k) \leq 2\gamma_k D \varepsilon_g \delta \sqrt{2\mu\varepsilon'} + \frac{LD^2\gamma_k^2}{2}.$$

By selecting the step size γ_k in Algorithm 1 such that

$$\gamma_k = \min \left\{ \frac{\beta^{bFW}\omega\sqrt{\varepsilon'}}{32D\varepsilon_g\sqrt{2\mu}}, \frac{\sqrt{\beta^{bFW}\omega\varepsilon'}}{4\sqrt{LD^2}} \right\},$$

we have

$$\begin{aligned}
f(x_{k+1}) - f(x_k) &\leq 2\gamma_k D \varepsilon_g \delta \sqrt{2\mu\varepsilon'} + \frac{LD^2\gamma_k^2}{2} \\
&= \frac{\beta^{bFW}\varepsilon'\omega}{16} + \frac{\beta^{bFW}\varepsilon'\omega}{16} \\
&= \frac{\beta^{bFW}\varepsilon'\omega}{8}.
\end{aligned}$$

□

Lemma 4.5.2. *Suppose that Assumption 3.2.1 (bounded variance) holds, and also suppose that in a boosted iteration of the boosted Frank-Wolfe algorithm we have*

$$\|\nabla f(x) - g(x)\| \leq \varepsilon_g \delta \sqrt{\frac{\mu\varepsilon}{2}} + B,$$

where

$$\varepsilon' = \frac{1}{2\mu}\left(\sqrt{2\mu\varepsilon} + B\left(\frac{1+\varepsilon_g\delta}{\varepsilon_g\delta}\right)\right)^2.$$

Then, we can satisfy the good gradient estimate inequality below in this iteration where

$$\frac{\|\nabla f(x) - g(x)\|}{\|g(x)\|} \leq \varepsilon_g\delta,$$

This is similar to the Definition 4.3.1 (good gradient approximation for boosted Frank-Wolfe in boosted iterations).

Proof. Suppose that

$$\|\nabla f(x) - g(x)\| \leq \varepsilon_g\delta\sqrt{\frac{\mu\varepsilon}{2}} + B.$$

Because $\varepsilon_g, \delta < 1$, then,

$$\sqrt{\frac{\mu\varepsilon}{2}} + B \geq \varepsilon_g\delta\sqrt{\frac{\mu\varepsilon}{2}} + B \geq \|\nabla f(x) - g(x)\| \geq \|\nabla f(x)\| - \|g(x)\|.$$

By strong convexity and the definition of ε', we have

$$\|\nabla f(x)\|^2 \geq 2\mu(f(x) - f^*) > 2\mu\varepsilon' = \left(\sqrt{2\mu\varepsilon} + B\left(\frac{1+\varepsilon_g\delta}{\varepsilon_g\delta}\right)\right)^2.$$

Therefore,

$$\|g(x)\| \geq \|\nabla f(x)\| - \sqrt{\frac{\mu\varepsilon}{2}} - B \geq \sqrt{2\mu\varepsilon'} - \sqrt{\frac{\mu\varepsilon}{2}} - B.$$

Hence,

$$\frac{\|\nabla f(x) - g(x)\|}{\|g(x)\|} \leq \frac{\varepsilon_g \delta \sqrt{\frac{\mu\varepsilon}{2}} + B}{\sqrt{2\mu\varepsilon'} - \sqrt{\frac{\mu\varepsilon}{2}} - B} = \frac{\varepsilon_g \delta \sqrt{\frac{\mu\varepsilon}{2}} + B}{\sqrt{2\mu\varepsilon} + B\left(\frac{1+\varepsilon_g\delta}{\varepsilon_g\delta}\right) - \sqrt{\frac{\mu\varepsilon}{2}} - B} = \varepsilon_g \delta.$$

□

Corollary 4.5.1. *Suppose that Assumptions 2.2.1 (L−smoothness), 2.2.4 (strong convexity), and 4.3.1 (sufficiently boosted) hold.*

1. *Under Assumption 3.2.1 (bounded variance), boosted Frank-Wolfe algorithm has the same iteration and sample-size complexities as in Theorem 4.4.1 and Theorem 4.4.2 with terminal time* $T_{\varepsilon'} = \min\{k : f(x_k) - f^* \leq \varepsilon'\}$.
2. *Assumption 3.2.2 (sub-Gaussian), boosted Frank-Wolfe algorithm has the same iteration and sample-size complexities as away-step Frank-Wolfe's in Theorem 4.4.1 and Theorem 4.4.2 with terminal time* $T_{\varepsilon'} = \min\{k : f(x_k) - f^* \leq \varepsilon'\}$.

where

$$\varepsilon' = \frac{1}{2\mu}\left(\sqrt{2\mu\varepsilon} + B\left(\frac{1+\varepsilon_g\delta}{\varepsilon_g\delta}\right)\right)^2.$$

Proof. Part I: Bounded Variance Biased Gradients By Lemmas 2.4.1 and 2.4.2,

$$\mathbb{E}[\|g(x) - \nabla f(x)\|^2] \leq \frac{V_g}{n} + B^2 \tag{4.19}$$

and,

$$\mathbb{P}(\|g(x) - \nabla f(x)\| > s + B) \leq \frac{V_g}{ns^2}. \tag{4.20}$$

By Lemma 4.5.2, pick s such that

$$s = \varepsilon_g \delta \sqrt{\frac{\mu\varepsilon}{2}}$$

then,

$$\mathbb{P}(\|g(x) - \nabla f(x)\| > s + B) \leq \frac{V_g}{ns^2} = \frac{V_g}{n} \cdot \frac{2\varepsilon_g^2\delta^2}{\mu\varepsilon} = \frac{2V_g\varepsilon_g^2\delta^2}{n\mu\varepsilon}.$$

By Lemma 2.4.2, pick the sample size such that

$$n \geq \frac{2V_g}{\varepsilon_g^2\delta^2\mu\varepsilon(1-p_g)} \tag{4.21}$$

will satisfy the condition that the probability

$$\mathbb{P}\left(\|g(x) - \nabla f(x)\| \leq \varepsilon_g \delta \sqrt{\frac{\mu\varepsilon}{2}} + B\right)$$

is at least p_g. Define the new error

$$\varepsilon' = \frac{1}{2\mu}\left(\sqrt{2\mu\varepsilon} + B\left(\frac{1+\varepsilon_g\delta}{\varepsilon_g\delta}\right)\right)^2.$$

Because under the good gradient estimate setting, we approximate the gradient so that

$$\|g(x) - \nabla f(x)\| \leq \varepsilon_d \delta \sqrt{\frac{\mu\varepsilon}{2}} + B,$$

by Lemma 4.5.1, for non-boosted iterations,

$$f(x_{k+1}) - f(x_k) \leq \beta^{bFW}\varepsilon'\omega/8.$$

By Lemma 4.5.2, we can also ensure that the event

$$\frac{\|g(x) - \nabla f(x)\|}{\|g(x)\|} \leq \varepsilon_g\delta$$

holds with probability at least p_g. Then, with a good gradient approximation above we can ensure $\|g(x) - \nabla f(x)\|/\|g(x)\| \leq \varepsilon_g\delta$ with the new error ε'. By Lemma 4.3.5 with the good gradient estimate in boosted iterations,

$$f(x_{k+1}) - f(x_k) \leq -\beta^{bFW}\varepsilon'.$$

Then with a good gradient estimate,

$$f(x_{k+1}) - f(x_k) \leq \begin{cases} \beta^{bFW}\varepsilon'\omega/8 \text{ for non-boosted iterations} \\ -\beta^{bFW}\varepsilon' \text{ for boosted iterations.} \end{cases}$$

Here, ε' is the new error of $f(x_k) - f^*$ which replaces ε. Define a stochastic process $(M_k : k \geq 0)$ similar to the process as in (4.8) from Theorem 4.4.1 where ε is replaced with ε'. Then, using the Lyapunov function as defined in (4.7), the

analogous result to (2.15) in the proof of Theorem 2.5.1 is

$$\begin{aligned}
&\mathbb{E}[M_{k+1} - M_k | \mathcal{F}_k] \\
&\leq \; -\frac{\beta^{bFW}\varepsilon'\omega}{8} p_g + \gamma_k D \sqrt{\frac{V_g}{n_k} + B^2} \sqrt{1 - p_g} + \left(\frac{L\Gamma^2 D^2}{2} + \frac{\varepsilon}{2} \right) (1 - p_g) \\
&\leq \; -\frac{\beta^{bFW}\varepsilon'\omega}{8} p_g + \gamma_k DB \sqrt{1 - p_g} + \left(\Gamma D \varepsilon_g \delta \sqrt{\frac{\mu\varepsilon}{2}} + \frac{L\Gamma^2 D^2}{2} + \frac{\varepsilon}{2} \right) (1 - p_g).
\end{aligned}$$

Select probability p_g such that

$$p_g \geq \max \left\{ \frac{2}{1 + \sqrt{1 + \frac{\beta^{bFW^2} B^2}{256\Gamma^2 D^2 \mu^2}}}, \frac{\Gamma D \varepsilon_g \delta \sqrt{\mu\varepsilon/2} + L\Gamma^2 D^2/2 + \varepsilon/2}{\Gamma D \varepsilon_g \delta \sqrt{\mu\varepsilon/2} + L\Gamma^2 D^2/2 + \varepsilon/2 + \beta^{bFW}\varepsilon'\omega/16} \right\} \tag{4.22}$$

Because

$$\varepsilon' = \left(\sqrt{\varepsilon} + \frac{B}{\sqrt{2\mu}} \cdot \left(\frac{1 + \varepsilon_g \delta}{\varepsilon_g \delta} \right) \right)^2 \geq \frac{B^2}{2\mu},$$

it follows that

$$p_g \geq \frac{2}{1 + \sqrt{1 + \frac{\beta^{bFW^2} \omega^2 B^2}{256\Gamma^2 D^2 \mu^2}}} \geq \frac{2}{1 + \sqrt{1 + \frac{\beta^{bFW^2} \omega^2 \varepsilon'}{128\Gamma^2 D^2 \mu}}}.$$

This implies that

$$\frac{p_g}{\sqrt{1 - p_g}} \geq \frac{2}{1 + \sqrt{1 + \frac{\beta^{bFW^2} \omega^2 \varepsilon'}{128\Gamma^2 D^2 \mu}}} \cdot \frac{\sqrt{\sqrt{1 + \frac{\beta^{bFW^2} \omega^2 \varepsilon'}{128\Gamma^2 D^2 \mu}} + 1}}{\sqrt{\sqrt{1 + \frac{\beta^{bFW^2} \omega^2 \varepsilon'}{128\Gamma^2 D^2 \mu}} - 1}} = \frac{16\Gamma D \sqrt{2\mu}}{\beta^{bFW} \omega \sqrt{\varepsilon'}}.$$

From this inequality and the definition of ε',

$$\Gamma DB \sqrt{1 - p_g} \leq \Gamma D \sqrt{2\mu\varepsilon'} \sqrt{1 - p_g} \leq \frac{\beta^{bFW}\varepsilon'\omega p_g}{16}.$$

Then,

$$\begin{aligned}
&\mathbb{E}[M_{k+1} - M_k | \mathcal{F}_k] \\
&\leq -\frac{\beta^{bFW}\varepsilon'\omega}{8} p_g + \gamma_k DB\sqrt{1-p_g} + \left(\Gamma D\varepsilon_g \delta \sqrt{\frac{\mu\varepsilon}{2}} + \frac{L\Gamma^2 D^2}{2} + \frac{\varepsilon}{2}\right)(1-p_g) \\
&\leq -\frac{\beta^{bFW}\varepsilon'\omega}{8} p_g + \frac{\beta^{bFW}\varepsilon'\omega}{16} p_g + \left(\Gamma D\varepsilon_g \delta \sqrt{\frac{\mu\varepsilon}{2}} + \frac{L\Gamma^2 D^2}{2} + \frac{\varepsilon}{2}\right)(1-p_g) \\
&= -\frac{\beta^{bFW}\varepsilon'\omega}{16} p_g + \left(\Gamma D\varepsilon_g \delta \sqrt{\frac{\mu\varepsilon}{2}} + \frac{L\Gamma^2 D^2}{2} + \frac{\varepsilon}{2}\right)(1-p_g) \\
&\leq 0.
\end{aligned}$$

Hence, as in the proof of Theorem 4.4.1 , we obtain iteration complexity of the order

$$\mathbb{E}[T_{\varepsilon'}] \leq \frac{4\Phi_0^{(bFW)}}{\beta^{bFW}\omega(1+\omega)\varepsilon'} \sim \mathcal{O}(\varepsilon'^{-1}(f(x_0) - f^*))$$

which is also upper bounded by $\mathcal{O}(\varepsilon^{-1}(f(x_0) - f^*))$. For sample size complexity, we select the sample size as in (4.21), and probability p_g from (4.22) then

$$\begin{aligned}
n &\geq \frac{2V_g}{\varepsilon_g^2 \delta^2 \mu\varepsilon(1-p_g)} \\
&\geq \frac{2V_g}{\varepsilon_g^2 \delta^2 \mu\varepsilon}\left(1 + \frac{16\Gamma D\varepsilon_g\delta\sqrt{\mu\varepsilon/2} + 8L\Gamma^2D^2 + 8\varepsilon}{\beta^{bFW}\varepsilon'\omega}\right) \\
&\sim \mathcal{O}\left(\frac{1}{\varepsilon\varepsilon'}\right) \\
&\sim \mathcal{O}(\varepsilon^{-2}).
\end{aligned}$$

Part II: The case of sub-Gaussian Biased Gradients

By Lemma 2.4.3 and (2.7),

$$\mathbb{E}[\|g(x) - \nabla f(x)\|^2] \leq \frac{2d}{nc} + B^2,$$

and

$$\mathbb{P}(\|g(x) - \nabla f(x)\| > s + B) \leq \mathbb{P}(\|g(x) - \mathbb{E}[g(x)]\| > s) \leq 2de^{-ncs^2}.$$

Pick the value s such that

$$s = \varepsilon_g \delta \sqrt{\frac{\mu\varepsilon}{2}}.$$

This yields the necessary sample size

$$n \geq \frac{2}{c\varepsilon_g^2 \delta^2 \mu\varepsilon} \log\left(\frac{2d}{1-p_g}\right). \tag{4.23}$$

Define the stochastic process M_k in the same way as (4.8) in Theorem 4.4.1 Then,

$$\mathbb{E}[M_{k+1} - M_k | \mathcal{F}_k] \leq -\frac{\beta^{bFW}\varepsilon'\omega}{8} p_g + \gamma_k D \sqrt{\frac{2d}{n_k c} + B^2}\sqrt{1-p_g} + \left(\frac{L\Gamma^2 D^2}{2} + \frac{\varepsilon}{2}\right)(1-p_g) \tag{4.24}$$

Choose the probability p_g such that

$$p_g \geq \max\left\{\frac{2}{1 + \sqrt{1 + \frac{\beta^{bFW\,2} B^4 \omega^2}{1024\mu^2\Gamma^2 D^2 (2d/c + B^2)}}},\ \frac{L\Gamma^2 D^2/2 + \varepsilon/2}{L\Gamma^2 D^2/2 + \varepsilon/2 + \beta^{bFW}\varepsilon'/16}\right\}. \tag{4.25}$$

Because

$$\varepsilon' = \left(\sqrt{\varepsilon} + \frac{B}{\sqrt{2\mu}} \cdot \left(\frac{1 + \varepsilon_g\delta}{\varepsilon_g\delta}\right)\right)^2 \geq \frac{B^2}{2\mu},$$

then,

$$p_g \geq \frac{2}{1+\sqrt{1+\frac{\beta^{bFW^2}B^4\omega^2}{1024\mu^2\Gamma^2D^2(2d/c+B^2)}}} \geq \frac{2}{1+\sqrt{1+\frac{\beta^{bFW^2}\varepsilon'^2\omega^2}{256\Gamma^2D^2(2d/c+B^2)}}}.$$

This implies that

$$\begin{aligned}
\frac{p_g}{\sqrt{1-p_g}} &\geq \frac{2}{1+\sqrt{1+\frac{\beta^{bFW^2}\varepsilon'^2\omega^2}{256\Gamma^2D^2(2d/c+B^2)}}} \cdot \frac{\sqrt{\sqrt{1+\frac{\beta^{bFW^2}\varepsilon'^2\omega^2}{256\Gamma^2D^2(2d/c+B^2)}}+1}}{\sqrt{\sqrt{1+\frac{\beta^{bFW^2}\varepsilon'^2\omega^2}{256\Gamma^2D^2(2d/c+B^2)}}-1}} \\
&= \frac{2}{\sqrt{\frac{\beta^{bFW^2}\varepsilon'^2\omega^2}{256\Gamma^2D^2(2d/c+B^2)}}} \\
&= \frac{16\Gamma D}{\beta^{bFW}\varepsilon'\omega}\sqrt{\frac{2d}{c}+B^2}.
\end{aligned}$$

Therefore,

$$\Gamma D\sqrt{\frac{2d}{n_k c}+B^2}\sqrt{1-p_g} \leq \frac{\beta^{bFW}\varepsilon'\omega p_g}{16}.$$

From (4.24), we have

$$\mathbb{E}[M_{k+1}-M_k|\mathcal{F}_k] \leq -\frac{\beta^{bFW}\varepsilon'\omega}{16}p_g + \left(\frac{L\Gamma^2D^2}{2}+\frac{\varepsilon}{2}\right)(1-p_g).$$

By the choice of p_g earlier in (4.25),

$$\mathbb{E}[M_{k+1}-M_k|\mathcal{F}_k] \leq 0.$$

Thus we have established the desired negative drift. Hence, we have the same iteration complexity as in Theorem 4.4.1. In addition, with the choice of p_g above from (4.25), we also have the same sample size complexity as in Theorem 4.4.2. □

Next, we present the convergence analysis of the boosted Frank–Wolfe algorithm without Assumption 4.3.1 (sufficiently boosted iterations). In this setting, the resulting guarantees are similar to those obtained for the standard Frank–Wolfe algorithm in Corollary 3.2.1 and Corollary 3.2.2. This is because the worst-case scenario occurs when all iterations are non-boosted, in which case the boosted Frank–Wolfe algorithm effectively reduces to the standard Frank–Wolfe method.

Corollary 4.5.2. *Suppose that Assumptions 2.2.1 (L−smoothness), and 2.2.4 (strong convexity) hold.*

1. *Under Assumption 3.2.1 (bounded variance), boosted Frank-Wolfe algorithm has the same iteration and sample-size complexities as standard Frank-Wolfe's in Corollary 3.2.1 with terminal time* $T_{\varepsilon'} = \min\{k : f(x_k) - f^* \leq \varepsilon'\}$.
2. *Assumption 3.2.2 (sub-Gaussian), boosted Frank-Wolfe algorithm has the same iteration and sample-size complexities as standard Frank-Wolfe's in Corollary 3.2.2 with terminal time* $T_{\varepsilon'} = \min\{k : f(x_k) - f^* \leq \varepsilon'\}$.

where

$$\varepsilon' = \max\left\{\varepsilon + 4BD,\ \frac{1}{2\mu}\left(\sqrt{2\mu\varepsilon} + B\left(\frac{1+\varepsilon_g\delta}{\varepsilon_g\delta}\right)\right)^2\right\}.$$

Proof. Part I: Bounded Variance Gradients

By Lemmas 2.4.1 and 2.4.2,

$$\mathbb{E}[\|g(x) - \nabla f(x)\|^2] \leq \frac{V_g}{n} + B^2 \tag{4.26}$$

and

$$\mathbb{P}(\|g(x) - \nabla f(x)\| > s + B) \leq \frac{V_g}{ns^2}. \tag{4.27}$$

Pick s such that

$$s = \min\left\{\frac{\varepsilon}{4D}, \varepsilon_g\delta\sqrt{\frac{\mu\varepsilon}{2}}\right\},$$

then,

$$\mathbb{P}(\|g(x) - \nabla f(x)\| > s + B) \leq \frac{V_g}{ns^2} = \frac{V_g}{n} \cdot \max\left\{\frac{16D^2}{\varepsilon^2}, \frac{2\varepsilon_g^2\delta^2}{\mu\varepsilon}\right\}.$$

By the definition of s,

$$\mathbb{P}(\|g(x)-\nabla f(x)\| > \frac{\varepsilon}{4D}+B) \leq \mathbb{P}(\|g(x)-\nabla f(x)\| > s+B) \leq \frac{V_g}{n}\cdot\max\left\{\frac{16D^2}{\varepsilon^2}, \frac{2\varepsilon_g^2\delta^2}{\mu\varepsilon}\right\},$$

and,

$$\mathbb{P}(\|g(x)-\nabla f(x)\| > \varepsilon_g\delta\sqrt{\frac{\mu\varepsilon}{2}}+B) \leq \mathbb{P}(\|g(x)-\nabla f(x)\| > s+B) \leq \frac{V_g}{n}\cdot\max\left\{\frac{16D^2}{\varepsilon^2}, \frac{2\varepsilon_g^2\delta^2}{\mu\varepsilon}\right\}.$$

Then, pick the sample size such that

$$n \geq \max\left\{\frac{16V_gD^2}{\varepsilon^2(1-p_g)}, \frac{2V_g}{\varepsilon_g^2\delta^2\mu\varepsilon(1-p_g)}\right\} \tag{4.28}$$

will satisfy the condition that the probabilities

$$\mathbb{P}(\|g(x) - \nabla f(x)\| \leq \frac{\varepsilon}{4D} + B),$$

and

$$\mathbb{P}(\|g(x) - \nabla f(x)\| \leq \varepsilon_g \delta \sqrt{\frac{\mu\varepsilon}{2}} + B)$$

are at least p_g. By Lemma 4.5.2, we can also ensure that the event

$$\frac{\|g(x) - \nabla f(x)\|}{\|g(x)\|} \leq \varepsilon_g \delta$$

holds with probability at least p_g. Define new errors below

$$\varepsilon_1' = \varepsilon + 4BD,\ \varepsilon_2' = \frac{1}{2\mu}\left(\sqrt{2\mu\varepsilon} + B\left(\frac{1+\varepsilon_g\delta}{\varepsilon_g\delta}\right)\right)^2.$$

Because under the good gradient estimate setting, we approximate the gradient so that $\|g(x) - \nabla f(x)\| \leq \varepsilon_1'/4D$, by Lemma 2.3.1, for non-boosted iterations and while $f(x_k) - f^* > \varepsilon_1'$,

$$f(x_{k+1}) - f(x_k) \leq -\beta^{FW}(\varepsilon_1')\varepsilon_1'.$$

Because we also approximate the gradient so that $\|g(x) - \nabla f(x)\|/\|g(x)\| \leq \varepsilon_g\delta$ where the error is ε_2', by Lemma 4.3.6 with the good gradient estimate in boosted iterations and while $f(x_k) - f^* > \varepsilon_2'$,

$$f(x_{k+1}) - f(x_k) \leq -\beta^{bFW}\varepsilon_2'.$$

Then, we can acheive improvement in both cases while $f(x_k) - f^* > \max\{\varepsilon_1', \varepsilon_2'\}$. Define a new error $\varepsilon' = \max\{\varepsilon_1', \varepsilon_2'\}$. Define another value for each iteration's

improvement, $\varepsilon_3' = \min\{\varepsilon_1', \varepsilon_2'\}$. Then with a good gradient estimate,

$$f(x_{k+1}) - f(x_k) \leq \begin{cases} -\beta^{FW}(\varepsilon_3')\varepsilon_3' \text{ for non-boosted iterations} \\ -\beta^{bFW}\varepsilon_3' \text{ for boosted iterations} \end{cases}$$

Here, ε' is the new error of $f(x_k) - f^*$ which replaces ε. Let $G^* = \|\nabla f(x^*)\|$ be the norm of the gradient at optimum. In the event of bad gradient estimate, for any $x, y \in \mathcal{X}$,

$$\begin{aligned} f(x) - f(y) &\leq f(x) - f^* \\ &\leq \nabla f(x^*)^T(x - x^*) + \frac{L}{2}\|x^* - x\|^2 \\ &\leq \|\nabla f(x^*)\| \cdot \|x - x^*\| + \frac{L}{2}\|x^* - x\|^2 \\ &\leq G^* D + \frac{LD^2}{2}. \end{aligned}$$

Define a Lyapunov function $\Phi_k^{(bFW)}$ similar to the one defined in (4.7) Then, under the event $k < T_{\varepsilon'}$,

$$\begin{aligned} \mathbb{E}[\Phi_{k+1}^{(bFW)} - \Phi_k^{(bFW)}|\mathcal{F}_k] &= \mathbb{E}[f(X_{k+1}) - f(X_k)|\mathcal{F}_k] \\ &\leq \max\{-\beta^{FW}(\varepsilon_3')\varepsilon_3', -\beta^{bFW}\varepsilon_3'\}p_g + \left(G^* D + \frac{LD^2}{2}\right)(1 - p_g) \end{aligned}$$

Select the probability p_g such that

$$p_g \geq \frac{G^* D + LD^2/2 + \min\{\beta^{FW}(\varepsilon_3')\varepsilon_3', \beta^{bFW}\varepsilon_3'\}/2}{G^* D + LD^2/2 + \min\{\beta^{FW}(\varepsilon_3')\varepsilon_3', \beta^{bFW}\varepsilon_3'\}} \tag{4.29}$$

Define a stochastic process $(M_k : k \geq 0)$ similar to (4.15) in Theorem 4.4.3. Then, $\mathbb{E}[M_{k+1} - M_k|\mathcal{F}_k] \leq 0$. On the event $k \geq T_{\varepsilon'}$, $M_{k+1} = M_k$. So, $(M_k : k \geq 0)$

is a non-negative super-martingale. By the optional stopping theorem, and since $X_0 = x_0$ is deterministic with $f(x_0) > f^* + \varepsilon'$,

$$\mathbb{E}[M_{T_{\varepsilon'}}] \leq E[M_0] = \Phi_0^{(bFW)}.$$

Then, as in Theorem 4.4.3, we then have iteration complexity of the order $\mathcal{O}\left(LD^2(f(x_0) - f^*)(\varepsilon_3')^{-2}\right)$, which is bounded above by $\mathcal{O}\left(LD^2(f(x_0) - f^*)\varepsilon^{-2}\right)$. Moreover, due to and the argument of Lemma 2.5.1, we still have the same sample size complexity as in Theorem 4.4.3.

Part II: The case of sub-Gaussian Biased Gradients

By Lemma 2.4.3 and (2.7),

$$\mathbb{E}[\|g(x) - \nabla f(x)\|^2] \leq \frac{2d}{nc} + B^2,$$

and

$$\mathbb{P}(\|g(x) - \nabla f(x)\| > s + B) \leq \mathbb{P}(\|g(x) - \mathbb{E}[g(x)]\| > s) \leq 2de^{-ncs^2}.$$

Pick the value s such that

$$s = \min\left\{\frac{\varepsilon}{4D}, \varepsilon_g D\sqrt{\frac{\mu\varepsilon}{2}}\right\}.$$

This yields the necessary sample size

$$n \geq \max\left\{\frac{16D^2}{c\varepsilon^2}, \frac{2}{c\varepsilon_g^2\delta^2\mu\varepsilon}\right\}\log\left(\frac{2d}{1-p_g}\right). \tag{4.30}$$

Define the stochastic process M_k in the same way as in (4.15). Then,

$$\begin{aligned}\mathbb{E}[\Phi_{k+1}^{(bFW)} - \Phi_k^{(bFW)}|\mathcal{F}_k] &= \mathbb{E}[f(X_{k+1}) - f(X_k)|\mathcal{F}_k] \\ &\leq \max\{-\beta^{FW}(\varepsilon_3')\varepsilon_3', -\beta^{bFW}\varepsilon_3'\}p_g + \left(G^*D + \frac{LD^2}{2}\right)(1-p_g)\end{aligned}$$

Select the probability as defined in (4.30) Then, the analogous result holds for iteration complexity. However, we require the sample size

$$\begin{aligned}n &\geq \max\left\{\frac{16D^2}{c\varepsilon^2}, \frac{2}{c\varepsilon_g^2\delta^2\mu\varepsilon}\right\}\log\left(\frac{2d}{1-p_g}\right) \\ &\geq \max\left\{\frac{16D^2}{c\varepsilon^2}, \frac{2}{c\varepsilon_g^2\delta^2\mu\varepsilon}\right\}\left(\log(2d) + \log\frac{\min\{\beta^{FW}(\varepsilon_3')\varepsilon_3', \beta^{bFW}\varepsilon_3'\}/2}{G^*D + LD^2/2 + \min\{\beta^{FW}(\varepsilon_3')\varepsilon_3', \beta^{bFW}\varepsilon_3'\}}\right) \\ &\sim \mathcal{O}\left(\frac{D^2}{\varepsilon^2}\log\frac{1}{\varepsilon_3'}\right) \\ &\sim \mathcal{O}\left(\frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon}\right).\end{aligned}$$

□

Example 4.5.1. We revisit the Example 3.2.1 from Section 3.2. Let the true function f and the bias $b(x)$ in the gradient estimators be

$$f(x_1, x_2) = \frac{1}{2}(x_1+30)^2 + \frac{1}{100}(x_2+30)^2 \text{ and } b(x) = \begin{pmatrix}1\\-20\end{pmatrix}.$$

The function f is quadratic and hence strongly convex. The bias is constant and

we can take $B = \|b(x)\| = \sqrt{401}$. Moreover $\|\nabla f(x)\| > B$ for all feasible x. The gradient estimator is

$$g(x) = \nabla f(x) + b(x) = \begin{bmatrix} x_1 + 31 \\ \frac{x_2+30}{50} - 20 \end{bmatrix}.$$

If the algorithm is initiated at the top-right corner $x_0 = (0.01, R)$ and $R < 970$ then the target vertex obtained by solving the linear program in the standard Frank-Wolfe algorithm on *all* iterations gives the vertex $s = (0, R)$. In particular, for any point $x = (t, R)$ for with $t > 0$,

$$g(x) = \begin{bmatrix} t + 31 \\ \frac{R+30}{50} - 20 \end{bmatrix}.$$

Since $R < 970$, the second component of the gradient is negative, and therefore the negative gradient points outward from the feasible region. Consequently, the Frank-Wolfe direction $s - x$, where $s = (0, R)$, is the feasible direction that is best aligned with the negative gradient. Any direction toward another vertex forms an obtuse angle with the negative gradient and thus has poorer alignment. As a result, no additional descent directions satisfy the boosting criterion, and the search direction cannot be boosted.

In Figure 4.4, the only available direction is $s - x$, where $s = (0, R)$, and this holds for all $x = (t, R)$ with $0 < t \leq 0.01$. As a result, Algorithm 4 always takes the standard Frank–Wolfe step and cannot generate a boosted direction. By Example 3.2.2, the resulting error depends on the term BD, where the diameter D is determined by R. Hence, the error bound increases as R becomes larger.

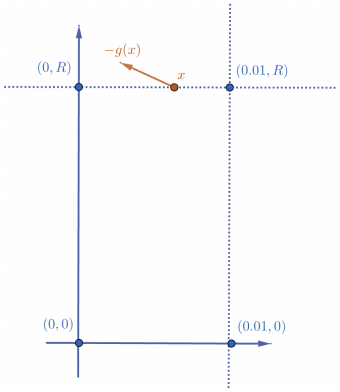


Figure 4.4: The negative gradient $g(x)$ points outward and the search direction is not boosted.

4.5.2 Time to Reach Near-Optimality Has a Light Tail

Since the non-boosted iterations of the boosted Frank–Wolfe algorithm coincide with those of the standard Frank–Wolfe method, the stopping time T_ε also satisfies a light-tail property analogous to that established for the standard Frank-Wolfe algorithm in Chapter 3, Section 3.3. All complexities will be the same as those analyzed in Section 4.4. To analyze this property, we define the stochastic process for the boosted Frank–Wolfe algorithm in the same manner as in Chapter 3, Section 3.3,

$$\Phi_k^{(bFW_m)} = \exp(f(X_k) - f^*). \tag{4.31}$$

Theorem 4.5.1. *Suppose the initial solution $X_0 = x_0$ is deterministic with $f(x_0) - f^* > \varepsilon$. Let $T_\varepsilon = \inf\{k : f(X_k) - f^* \leq \varepsilon\}$, and suppose the gradient follows Assumption 2.4.2 (sub-Gaussian). Under Assumption 2.2.1 ($L-$smoothness), Assumption 2.2.4 (strong convexity), and Assumption 4.3.1 (sufficiently boosted), boosted Frank-Wolfe has the same iteration complexities as in Theorem 4.4.1. In addition, the stopping time T_ε has a finite moment generating function in a neighborhood of 0.*

Proof. Define the stochastic process $(M_k : k \geq 0)$ where

$$M_k = \exp\left(\frac{\beta^{bFW}\varepsilon}{2}\sum_{i=0}^{(k\wedge T_\varepsilon)-1} B_i - \frac{\beta^{bFW}\omega\varepsilon}{4}\sum_{i=0}^{(k\wedge T_\varepsilon)-1}(1-B_i)\right)\Phi_{k\wedge T_\varepsilon}^{(bFW_m)}$$

where

$$B_k = \begin{cases} 1 & k^{th} \text{ iteration is boosted} \\ 0 & \text{otherwise} \end{cases}$$

By Assumption 4.3.1 (sufficiently boosted), $\sum_{i=0}^{k-1} B_i \geq \omega k$. The analogous result holds as in Theorem 4.4.1 where under the good gradient estimate $\mathcal{G}$ and under event $k < T_\varepsilon$,

$$\exp(f(x_{k+1}) - f(x_k)) \leq \begin{cases} \exp(\beta^{bFW}\omega\varepsilon/8) & \text{if non-boosted} \\ \exp(-\beta^{bFW}\varepsilon) & \text{if boosted.} \end{cases}$$

Therefore, under good gradient estimate $\mathcal{G}$,

$$\frac{M_{k+1}}{M_k} \leq \begin{cases} \exp(-\beta^{bFW}\varepsilon\omega/8) & \text{if non-boosted} \\ \exp(-\beta^{bFW}\varepsilon/2) & \text{if boosted} \end{cases}$$

Taking the maximum of the two cases, we have

$$\frac{M_{k+1}}{M_k} \leq \exp(-\beta^{bFW}\varepsilon\omega/8).$$

However, under the bad gradient estimate $\mathcal{G}'$, by (3.8) from Lemma 3.3.2, we have,

$$\begin{aligned}\mathbb{E}\left[\frac{\Phi_{k+1}^{(bFW_m)}}{\Phi_k^{(bFW_m)}}\mathbb{1}_{\mathcal{G}'}\middle|\mathcal{F}_k\right] &= \mathbb{E}\left[\exp(f(X_{k+1})-f(X_k))\mathbb{1}_{\mathcal{G}'}|\mathcal{F}_k\right] \\ &\leq C\exp\left(\frac{L\Gamma^2D^2}{2}+\frac{2\Gamma^2D^2}{c}\right)\sqrt{1-p_g}.\end{aligned}$$

Then,

$$\begin{aligned}\mathbb{E}\left[\frac{M_{k+1}}{M_k}\middle|\mathcal{F}_k\right] &= \mathbb{E}\left[\frac{M_{k+1}}{M_k}\mathbb{1}_{\mathcal{G}}\middle|\mathcal{F}_k\right]+\mathbb{E}\left[\frac{M_{k+1}}{M_k}\mathbb{1}_{\mathcal{G}'}\middle|\mathcal{F}_k\right] \\ &\leq \exp\left(-\frac{\beta^{bFW}\varepsilon\omega}{8}\right)p_g + C\exp\left(\frac{L\Gamma^2D^2}{2}+\frac{2\Gamma^2D^2}{c}+\frac{\varepsilon}{2}\right)\sqrt{1-p_g}.\end{aligned}$$

Select the probability p_g such that

$$1-p_g \leq \left\{\frac{1}{2},\ \frac{(1-\exp(-\beta^{bFW}\varepsilon\omega/8))^2}{4C^2\exp\left(L\Gamma^2D^2+4\Gamma^2D^2/c+\varepsilon\right)}\right\}. \tag{4.32}$$

Then, on the event $k<T_\varepsilon$, and choice of probability p_g from (4.32),

$$\begin{aligned}\mathbb{E}\left[\frac{M_{k+1}}{M_k}\middle|\mathcal{F}_k\right] &\leq \exp\left(-\frac{\beta^{bFW}\varepsilon\omega}{8}\right)p_g + C\exp\left(\frac{L\Gamma^2D^2}{2}+\frac{2\Gamma^2D^2}{c}+\frac{\varepsilon}{2}\right)\sqrt{1-p_g} \\ &\leq \exp\left(-\frac{\beta^{bFW}\varepsilon\omega}{8}\right)p_g + \frac{1}{2}\left(1-\exp\left(-\frac{\beta^{bFW}\varepsilon\omega}{8}\right)\right) \\ &\leq \exp\left(-\frac{\beta^{bFW}\varepsilon\omega}{8}\right)p_g + \left(1-\exp\left(-\frac{\beta^{bFW}\varepsilon\omega}{8}\right)\right)p_g \\ &= 1.\end{aligned}$$

On the event $T_\varepsilon \leq k$, $M_{k+1}=M_k$. So $(M_k : k\geq 0)$ is a non-negative super-

martingale. By the optional stopping theorem,

$$\mathbb{E}[M_{T_\varepsilon}] \leq \mathbb{E}[M_0] = \Phi_0^{(bFW_m)}.$$

Therefore,

$$\begin{aligned}
\Phi_0^{(bFW_m)} &\geq \mathbb{E}[M_{T_\varepsilon}] \\
&= \mathbb{E}\left[\exp\left(\frac{\beta^{bFW}\varepsilon}{2}\sum_{i=0}^{T_\varepsilon-1} B_i - \frac{\beta^{bFW}\omega\varepsilon}{4}\sum_{i=0}^{T_\varepsilon-1}(1-B_i)\right)\Phi_{T_\varepsilon}^{(bFW_m)}\right] \\
&\geq \mathbb{E}\left[\exp\left(\frac{\beta^{bFW}\varepsilon}{2}\sum_{i=0}^{T_\varepsilon-1} B_i - \frac{\beta^{bFW}\omega\varepsilon}{4}\sum_{i=0}^{T_\varepsilon-1}(1-B_i)\right)\right] \\
&\geq \mathbb{E}\left[\exp\left(\frac{\beta^{bFW}\omega(\omega+1)\varepsilon}{4}T_\varepsilon\right)\right],
\end{aligned}$$

implying that T_ε has a finite moment generating function in a neighborhood of 0. By concavity,

$$\begin{aligned}
\log\Phi_0^{(bFW_m)} &\geq \log\mathbb{E}[M_{T_\varepsilon}] \\
&= \log\mathbb{E}\left[\exp\left(\frac{\beta^{bFW}\varepsilon}{2}\sum_{i=0}^{T_\varepsilon-1} B_i - \frac{\beta^{bFW}\omega\varepsilon}{4}\sum_{i=0}^{T_\varepsilon-1}(1-B_i)\right)\Phi_{T_\varepsilon}^{(bFW_m)}\right] \\
&\geq \mathbb{E}\left[\frac{\beta^{bFW}\varepsilon}{2}\sum_{i=0}^{T_\varepsilon-1} B_i - \frac{\beta^{bFW}\omega\varepsilon}{4}\sum_{i=0}^{T_\varepsilon-1}(1-B_i)\right] \\
&= \frac{\beta^{bFW}\varepsilon}{2}\mathbb{E}\left[\sum_{i=0}^{T_\varepsilon-1} B_i\right] + \frac{\beta^{bFW}\omega\varepsilon}{4}\mathbb{E}\left[\sum_{i=0}^{T_\varepsilon-1}(B_i-1)\right] \\
&\geq \frac{\beta^{bFW}\varepsilon}{2}\omega\mathbb{E}[T_\varepsilon] + \frac{\beta^{bFW}\omega\varepsilon}{4}(\omega-1)\mathbb{E}[T_\varepsilon] \\
&= \frac{\beta^{bFW}\varepsilon\omega}{2}\cdot\frac{\omega+1}{2}\mathbb{E}[T_\varepsilon]
\end{aligned}$$

Hence,

$$\mathbb{E}[T_\varepsilon] \leq \frac{4 \log \Phi_0^{(bFW_m)}}{\beta^{bFW} \varepsilon \omega(\omega+1)} \sim \mathcal{O}\left(\varepsilon^{-1}(f(x_0) - f^*)\right)$$

□

Theorem 4.5.2. *The sample size in each iteration that is required to achieve a good gradient estimate under the Assumption 2.4.2 (sub-Gaussian), Assumption 2.2.1 (L−smooth), Assumption 2.2.4 (strong convexity), and Assumption 4.3.1 (sufficiently boosted) using Lyapunov functions defined in* (4.31) *is* $\mathcal{O}(\varepsilon^{-1} \log(\varepsilon^{-1}))$.

Proof. Take $s = \min\left\{\varepsilon_g \delta \sqrt{\mu\varepsilon/2}, \sqrt{\varepsilon}/(4D)\right\}$ in Lemma 2.4.3 and select the sample size n so large such that $2de^{-ncs^2}$ is a lower bound of $1 - p_g$. Thus,

$$n \geq \max\left\{\frac{2}{\varepsilon_g^2 \delta^2 \mu}, 16D^2\right\} \frac{1}{c\varepsilon} \log\left(\frac{2d}{1-p_g}\right).$$

Using probability p_g from (4.32), and the fact that for positive x, $1 - x \leq e^{-x}$,

$$\begin{aligned} n &\geq \max\left\{\frac{2}{\varepsilon_g^2 \delta^2 \mu}, 16D^2\right\} \frac{1}{c\varepsilon} \left(\log(2d) + \log \frac{4C^2 e^{L\Gamma^2 D^2 + 4\Gamma^2 D^2/c} e^{\varepsilon/2}}{(1 - e^{-\beta^{bFW} \varepsilon\omega/8})^2}\right) \\ &\geq \max\left\{\frac{2}{\varepsilon_g^2 \delta^2 \mu}, 16D^2\right\} \frac{1}{c\varepsilon} \left(\log(2d) + \log \frac{4C^2 e^{L\Gamma^2 D^2 + 4\Gamma^2 D^2/c} e^{\varepsilon/2}}{(\beta^{bFW} \varepsilon\omega/8)^2}\right) \\ &\sim \mathcal{O}\left(\max\left\{\frac{1}{\mu}, D^2\right\} \frac{1}{\varepsilon} \log \frac{LD}{\varepsilon}\right) \\ &\sim \mathcal{O}\left(\varepsilon^{-1} \log \varepsilon^{-1}\right). \end{aligned}$$

□

Theorem 4.5.3. *Suppose the initial solution $X_0 = x_0$ is deterministic with $f(x_0) - f^* > \varepsilon$. Let $T_\varepsilon = \inf\{k : f(X_k) - f^* \leq \varepsilon\}$, and suppose the gradient follows the*

Assumption 2.4.2 (sub-Gaussian). Under Assumption 2.2.1 (L−smoothness), and Assumption 2.2.4 (strong convexity) boosted Frank-Wolfe has the same iteration complexities as in Theorem 4.4.3. In addition, the stopping time T_ε has a finite moment generating function in a neighborhood of 0.

Proof. Define the stochastic process $(M_k : k \geq 0)$ where

$$M_k = \exp\left(\frac{\beta^{FW}(\varepsilon)\varepsilon}{2}\sum_{i=0}^{k\wedge T_\varepsilon - 1}(1-B_i) + \frac{\beta^{bFW}\varepsilon}{2}\sum_{i=0}^{k\wedge T_\varepsilon - 1}B_i\right)\Phi_{k\wedge T_\varepsilon}^{(bFW_m)},$$

where

$$B_k = \begin{cases} 1 & k^{th} \text{ iteration is boosted} \\ 0 & \text{otherwise} \end{cases}$$

and $\Phi^{(bFW_m)}$ is the same as the one defined in (4.31). From the definition of M_k, on the event $k < T_\varepsilon$, we have

$$\begin{aligned}
&\mathbb{E}[M_{k+1}|\mathcal{F}_k] \\
&= \mathbb{E}\left[\exp\left(\frac{\beta^{FW}(\varepsilon)\varepsilon}{2}\sum_{i=0}^{k\wedge T_\varepsilon}(1-B_i) + \frac{\beta^{bFW}\varepsilon}{2}\sum_{i=0}^{k\wedge T_\varepsilon}B_i\right)\Phi_{k+1}^{(bFW_m)}\middle|\mathcal{F}_k\right] \\
&= \exp\left(\frac{\beta^{FW}(\varepsilon)\varepsilon}{2}\sum_{i=0}^{k\wedge T_\varepsilon - 1}(1-B_i) + \frac{\beta^{bFW}\varepsilon}{2}\sum_{i=0}^{k\wedge T_\varepsilon - 1}B_i\right)\cdot \\
&\quad \mathbb{E}\left[\exp\left(\frac{\beta^{FW}(\varepsilon)\varepsilon}{2}(1-B_k) + \frac{\beta^{bFW}\varepsilon}{2}B_k\right)\Phi_{k+1}^{(bFW_m)}\middle|\mathcal{F}_k\right].
\end{aligned}$$

We analyze each component based on the event of good gradient approximation for boosted Frank-Wolfe without sufficiently boosted assumption as defined in Definition 4.4.2 $\mathcal{G}$ using Lyapunov function defined in (4.31). By Lemma 4.3.6

under the event $\mathcal{G}$, we have

$$\begin{aligned}\frac{\Phi_{k+1}^{(bFW_m)}}{\Phi_k^{(bFW_m)}}\mathbb{1}_{\mathcal{G}} &= \exp(f(X_{k+1})-f(X_k))\mathbb{1}_{\mathcal{G}} \\ &\leq \exp(-\beta^{bFW}\varepsilon)\mathbb{1}_{\mathcal{B}}+\exp(-\beta^{FW}(\varepsilon)\varepsilon)\mathbb{1}_{\mathcal{B}'},\end{aligned}$$

or, while the algorithm is not terminated,

$$\Phi_{k+1}^{(bFW_m)}\mathbb{1}_{\mathcal{G}} \leq (\exp(-\beta^{bFW}\varepsilon)\mathbb{1}_{\mathcal{B}}+\exp(-\beta^{FW}(\varepsilon)\varepsilon)\mathbb{1}_{\mathcal{B}'})\Phi_k^{(bFW_m)}.$$

Because at iteration k, we have $\mathbb{1}_{\mathcal{B}} = B_k$, then

$$\begin{aligned}&\mathbb{E}\left[\exp((\beta^{bFW}\varepsilon/2)B_k+(\beta^{FW}(\varepsilon)\varepsilon/2)(1-B_k))\Phi_{k+1}^{(bFW_m)}\mathbb{1}_{\mathcal{G}}|\mathcal{F}_k\right] \\ &= \mathbb{E}\left[\exp((\beta^{bFW}\varepsilon/2)B_k+(\beta^{FW}(\varepsilon)\varepsilon/2)(1-B_k))\Phi_{k+1}^{(bFW_m)}\mathbb{1}_{\mathcal{G}}(\mathbb{1}_{\mathcal{B}}+\mathbb{1}_{\mathcal{B}'})|\mathcal{F}_k\right] \\ &\leq \mathbb{E}\left[(\exp(-\beta^{bFW}\varepsilon/2)\mathbb{1}_{\mathcal{B}}+\exp(-\beta^{FW}(\varepsilon)\varepsilon/2)\mathbb{1}_{\mathcal{B}'})\Phi_k^{(bFW_m)}\mathbb{1}_{\mathcal{G}}|\mathcal{F}_k\right] \\ &= \left(\exp\left(-\frac{\beta^{bFW}\varepsilon}{2}\right)p_g\mathbb{P}(\mathcal{B}|\mathcal{F}_k,\mathcal{G})+\exp\left(-\frac{\beta^{FW}(\varepsilon)\varepsilon}{2}\right)p_g\mathbb{P}(\mathcal{B}'|\mathcal{F}_k,\mathcal{G})\right)\Phi_k^{(bFW_m)} \\ &= \exp\left(-\frac{m_\varepsilon}{2}\right)p_g\Phi_k^{(bFW_m)},\end{aligned}$$

where $m_\varepsilon = \min\{\beta^{bFW}\varepsilon, \beta^{FW}(\varepsilon)\varepsilon\}$ Next, under the bad gradient approximation $\mathcal{G}'$ by (3.8) from Lemma 3.3.2, we have,

$$\begin{aligned}\mathbb{E}\left[\frac{\Phi_{k+1}^{(bFW_m)}}{\Phi_k^{(bFW_m)}}\mathbb{1}_{\mathcal{G}'}\middle|\mathcal{F}_k\right] &= \mathbb{E}\left[\exp(f(X_{k+1})-f(X_k))\mathbb{1}_{\mathcal{G}'}|\mathcal{F}_k\right] \\ &\leq C\exp\left(\frac{L\Gamma^2D^2}{2}+\frac{2\Gamma^2D^2}{c}\right)\sqrt{1-p_g}.\end{aligned}$$

For simplicity, let $M_\varepsilon = \max\{\beta^{bFW}\varepsilon, \beta^{FW}(\varepsilon)\varepsilon\}$, then

$$\begin{aligned}
&\mathbb{E}\left[\exp((\beta^{bFW}\varepsilon/2)B_k + (\beta^{FW}(\varepsilon)\varepsilon/2)(1-B_k))\Phi_{k+1}^{(bFW_m)}\mathbb{1}_{\mathcal{G}'}|\mathcal{F}_k\right] \\
&\quad\leq \mathbb{E}\left[\exp(M_\varepsilon/2)\Phi_{k+1}^{(bFW_m)}\mathbb{1}_{\mathcal{G}'}|\mathcal{F}_k\right] \\
&\quad\leq \exp(M_\varepsilon/2)C\exp\left(\frac{L\Gamma^2D^2}{2}+\frac{2\Gamma^2D^2}{c}\right)\sqrt{1-p_g}\Phi_k^{(bFW_m)}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\mathbb{E}\left[\exp((\beta^{FW}(\varepsilon)\varepsilon/2)(1-B_k) + (\beta^{bFW}\varepsilon/2)B_k)\Phi_{k+1}^{(bFW_m)}|\mathcal{F}_k\right] &\leq \exp\left(-\frac{m_\varepsilon}{2}\right)p_g\Phi_k^{(bFW_m)} \\
&\quad+\exp\left(\frac{M_\varepsilon}{2}\right)C\exp\left(\frac{L\Gamma^2D^2}{2}+\frac{2\Gamma^2D^2}{c}\right)\sqrt{1-p_g}\Phi_k^{(bFW_m)}.
\end{aligned}$$

Select the probability p_g such that

$$1-p_g \leq \left\{\frac{1}{2},\ \frac{\left(1-\exp\left(-\frac{m_\varepsilon}{2}\right)\right)^2}{4C^2\exp\left(L\Gamma^2D^2+\frac{4\Gamma^2D^2}{c}\right)\exp\left(M_\varepsilon\right)}\right\}. \tag{4.33}$$

Then,

$$\begin{aligned}
&\exp\left(-\frac{m_\varepsilon}{2}\right)p_g + C\exp\left(\frac{L\Gamma^2D^2}{2}+\frac{2\Gamma^2D^2}{c}\right)\exp\left(\frac{M_\varepsilon}{2}\right)\sqrt{1-p_g} \\
&\quad\leq \exp\left(-\frac{m_\varepsilon}{2}\right)p_g + \frac{1}{2}\left(1-\exp\left(-\frac{m_\varepsilon}{2}\right)\right) \\
&\quad\leq \exp\left(-\frac{m_\varepsilon}{2}\right)p_g + \left(1-\exp\left(-\frac{m_\varepsilon}{2}\right)\right)p_g \\
&\quad\leq 1.
\end{aligned}$$

Therefore, on the event $k < T_\varepsilon$,

$$\begin{aligned}
&\mathbb{E}\left[M_{k+1}|\mathcal{F}_k\right] \\
&= \exp\left(\frac{\beta^{FW}(\varepsilon)\varepsilon}{2}\sum_{i=0}^{k\wedge T_\varepsilon-1}(1-B_i)+\frac{\beta^{bFW}\varepsilon}{2}\sum_{i=0}^{k\wedge T_\varepsilon-1}B_i\right)\cdot \\
&\quad \mathbb{E}\left[\exp\left(\frac{\beta^{FW}(\varepsilon)\varepsilon}{2}(1-B_k)+\frac{\beta^{bFW}\varepsilon}{2}B_k\right)\Phi_{k+1}^{(bFW_m)}|\mathcal{F}_k\right] \\
&\leq \exp\left(\frac{\beta^{FW}(\varepsilon)\varepsilon}{2}\sum_{i=0}^{k\wedge T_\varepsilon-1}(1-B_i)+\frac{\beta^{bFW}\varepsilon}{2}\sum_{i=0}^{k\wedge T_\varepsilon-1}B_i\right)\Phi_k^{(bFW_m)} \\
&= M_k.
\end{aligned}$$

On the event $T_\varepsilon \leq k$, $M_{k+1} = M_k$. So $(M_k : k \geq 0)$ is a non-negative supermartingale. By the optional stopping theorem,

$$\mathbb{E}[M_{T_\varepsilon}] \leq \mathbb{E}[M_0] = \Phi_0^{(bFW_m)}.$$

Let S_k be the number of boosted iterations up until iteration k, then

$$\mathbb{E}\left[\exp\left(\frac{\beta^{bFW}\varepsilon}{2}S_{T_\varepsilon}+\frac{\beta^{FW}(\varepsilon)\varepsilon}{2}(T_\varepsilon-S_{T_\varepsilon})\right)\Phi_{T_\varepsilon}^{(bFW_m)}\right]$$

Since $\Phi_k^{(bFW_m)} \geq 1$ for all non-negative integers k,

$$\mathbb{E}\left[\exp\left(\frac{\beta^{bFW}\varepsilon}{2}S_{T_\varepsilon}+\frac{\beta^{FW}(\varepsilon)\varepsilon}{2}(T_\varepsilon-S_{T_\varepsilon})\right)\right] \leq \Phi_0^{(bFW_m)} < \infty.$$

If there is not a sufficient number of boosted iterations, by concavity, then

$$\mathbb{E}[(m_\varepsilon/2)T_\varepsilon] \leq \log\mathbb{E}\left[e^{\frac{m_\varepsilon}{2}T_\varepsilon}\right] \leq \log\mathbb{E}\left[\exp\left(\frac{\beta^{bFW}\varepsilon}{2}S_{T_\varepsilon}+\frac{\beta^{FW}(\varepsilon)\varepsilon}{2}(T_\varepsilon-S_{T_\varepsilon})\right)\right] \leq \log\Phi_0^{(bFW_m)}.$$

We then have the expected terminal time

$$\begin{aligned}\mathbb{E}[T_\varepsilon] &\leq \frac{2\Phi_0^{(bFW_m)}}{m_\varepsilon} \\ &= \frac{2\log\Phi_0^{(bFW_m)}}{\min\{\beta^{FW}(\varepsilon),\beta^{bFW}\}\varepsilon} \\ &\sim \mathcal{O}\left(LD^2\frac{f(x_0)-f^*}{\varepsilon^2}\right) \\ &\sim \mathcal{O}\left(\frac{f(x_0)-f^*}{\varepsilon^2}\right),\end{aligned}$$

establishing the finiteness of the moment generating function in a neighborhood of 0 similar to standard Frank-Wolfe results.

□

Theorem 4.5.4. *The sample size in each iteration that is required to achieve a good gradient estimate under the Assumption 2.4.2 (sub-Gaussian), 2.2.1 (L−smoothness), and 2.2.4 (strong convexity) using Lyapunov functions defined in* (4.31) *is* $\mathcal{O}(\varepsilon^{-2}\log(\varepsilon^{-1}))$.

Proof. Take $s = \min\{\varepsilon_g\delta\sqrt{\mu\varepsilon/2}, \varepsilon/(4D)\}$ in Lemma 2.4.3 and select the sample size n so large such that $2de^{-ncs^2}$ is a lower bound of $1-p_g$. Thus,

$$n \geq \max\left\{\frac{2}{\varepsilon_g^2\delta^2\mu\varepsilon}, \frac{16D^2}{\varepsilon^2}\right\}\frac{1}{c}\log\left(\frac{2d}{1-p_g}\right).$$

Using probability p_g from (4.33), and the fact that for positive x, $1 - x \leq e^{-x}$,

$$\begin{aligned} n &\geq \max\left\{\frac{2}{\varepsilon_g^2\delta^2\mu\varepsilon}, \frac{16D^2}{\varepsilon^2}\right\}\frac{1}{c}\left(\log(2d) + \log\frac{4C^2e^{L\Gamma^2D^2+4\Gamma^2D^2/c}e^{M_\varepsilon}}{(1-e^{-m_\varepsilon/2})^2}\right) \\ &\geq \max\left\{\frac{2}{\varepsilon_g^2\delta^2\mu\varepsilon}, \frac{16D^2}{\varepsilon^2}\right\}\frac{1}{c}\left(\log(2d) + \log\frac{4C^2e^{L\Gamma^2D^2+4\Gamma^2D^2/c}e^{M_\varepsilon}}{(m_\varepsilon/2)^2}\right) \\ &\sim \mathcal{O}\left(\frac{1}{\varepsilon^2}\log\frac{LD}{\varepsilon}\right) \\ &\sim \mathcal{O}\left(\varepsilon^{-2}\log\varepsilon^{-1}\right). \end{aligned}$$

□

4.5.3 Improved Complexity of the boosted Frank-Wolfe algorithm with an Interior Optimum

When the global optimum x^* lies strictly in the interior of $\mathcal{X}$, the non-boosted iterations exhibit behavior analogous to that of the standard Frank–Wolfe algorithm. In particular, the results from Chapter 3, Section 3.4 apply through the constant μ_f^{FW} introduced in Definition 3.4.1. Consequently, the standard Frank–Wolfe updates used during non-boosted iterations can be replaced by the interior-optimum variant given in Algorithm 3. Motivated by this observation, we modify Algorithm 4 accordingly and introduce Algorithm 6. In the interior optimum setting, Lemma 3.4.1 establishes the decrease in $f(x_{k+1}) - f(x_k)$ under the good gradient approximation condition. In particular, the resulting improvement for non-boosted iterations is of the form $-\beta^{iFW}\varepsilon$, which matches the $\mathcal{O}(\varepsilon)$ improvement order obtained for the Away-step Frank–Wolfe algorithm during non-drop iterations in Lemma 2.3.6. Under the interior-optimum assumption, we obtain a stronger descent guarantee than that available for the boosted Frank–Wolfe algorithm without the sufficiently

Algorithm 6 Interior Boosted Frank-Wolfe (IBFW).

Initialize x_0 to be a vertex v, maximum number of oracle steps T, an improvement tolerance δ.
for $k = 0, 1, 2, ...$ **do**
  Estimate the gradient $\nabla f(x_k)$ by g_k
  $d_k, \Lambda_k \leftarrow$ Boosted Direction Oracle(T, δ, g_k)
  $d_k \leftarrow d_k / \Lambda_k$
  if d_k not boosted **then**
    $\gamma_k = \min\{1, \frac{\sqrt{2\mu_f^{FW}\varepsilon}}{2LD^2}\}$ (interior standard Frank-Wolfe step)
  else
    $\gamma_k = \min\{1, -\frac{g_k^T d_k}{L\|d_k\|^2}\}$
  end if
  $x_{k+1} \leftarrow x_k + \gamma_k d_k$
end for

boosted assumption. Although the resulting iteration and sample size complexities remain of the same order as those established for boosted Frank–Wolfe under the sufficiently boosted assumption, the interior-optimum assumption provides these guarantees without requiring any condition on the frequency of boosted iterations. Consequently, it yields stronger worst-case convergence results for boosted Frank–Wolfe, even when sufficiently boosted iterations cannot be guaranteed. Motivated by this stronger descent behavior, we next establish the preparatory results for the interior boosted Frank–Wolfe algorithm.

Definition 4.5.1. *Fix some positive number ε_g. The vector g_k is a good gradient approximation at iteration k if*

$$\|g_k - \nabla f(x_k)\| \le \varepsilon_g \|g_k\| \eta_k \quad \textit{and} \quad \|g_k - \nabla f(x_k)\| \le \frac{\sqrt{2\mu_f^{FW}\varepsilon}}{4D}$$

when η_k is the alignment between $-g_k$ and the search direction d_k, i.e.

$$\eta_k = -\frac{g_k^T d_k}{\|g_k\|\|d_k\|}.$$

This definition is referred as good gradient approximation for boosted Frank-Wolfe with interior optimum assumption.

Lemma 4.5.3. *Suppose Assumptions 2.2.1 (L−smoothness) and 2.2.4 (strong convexity), and that we have a good gradient approximation for boosted Frank-Wolfe with interior optimum assumption as defined in Definition 4.5.1 at iteration k. Then*

$$f(x_{k+1}) - f^* = \begin{cases} -\beta^{iFW}\varepsilon \ \textit{for non-boosted iterations} \\ -\beta^{bFW}\varepsilon \ \textit{for boosted iterations} \end{cases}$$

where β^{iFW} and β^{bFW} is given by

$$\beta^{iFW} = \min\left\{\frac{\mu_f^{FW}}{4LD^2}, \frac{\sqrt{2\mu_f^{FW}}}{4\sqrt{\varepsilon}}\right\}, \ \beta^{bFW} = \frac{2\mu(1/2-\varepsilon_g)\delta^2}{L(\varepsilon_g\delta+1)^2}$$

Proof. For non-boosted iterations, we apply the step size policy $\gamma_k = \min\left\{1, \sqrt{2\mu_f^{FW}\varepsilon}/(2LD^2)\right\}$ as described in Algorithm 6, and by Lemma 3.4.1, we have

$$f(x_{k+1}) - f(x_k) \leq -\beta^{iFW}\varepsilon.$$

For boosted iterations, by Lemma 4.3.5, we have

$$f(x_{k+1}) - f(x_k) \leq -\beta^{bFW}\varepsilon.$$

□

Definition 4.5.2. *We say, at iteration k, that G_k is a good gradient estimate for*

the interior standard Frank-Wolfe algorithm with at least probability p_g if the event

$$E_k^{(ibFW)} = \left\{ \|G_k - \nabla f(X_k)\| \leq \varepsilon_g \|G_k\| \eta_k \quad \textit{and} \quad \|G_k - \nabla f(X_k)\| \leq \frac{\sqrt{2\mu_f^{FW}\varepsilon}}{4D} \right\}$$

satisfies the condition $\mathbb{P}(E_k^{(ibFW)}|\mathcal{F}_k) \geq p_g$ on the event $\{k < T_\varepsilon\}$. For simplicity, define this event as good gradient approximation for boosted Frank-Wolfe with interior optimum assumption.

Lemma 4.5.4. *Under Assumption 2.4.1 (bounded variance), the sample sizes required for each iteration to achieve a good gradient estimate for the interior standard Frank-Wolfe is $\mathcal{O}(\varepsilon^{-1}(1-p_g)^{-1})$.*

Proof. By Lemma 3.4.2 and (3.13), selecting sample size

$$n \geq \frac{8V_g D^2}{\varepsilon \mu_f^{FW}(1-p_g)},$$

is sufficient to satisfy the good gradient approximation for standard Frank-Wolfe under interior optimum assumption with probability at least p_g. In addition by Lemma 4.4.2 and (4.11), selecting sample size

$$n \geq \frac{2V_g}{\varepsilon_g^2 \delta^2 \mu \varepsilon (1-p_g)},$$

is sufficient to satisfy the good gradient approximation for boosted Frank-Wolfe in boosted iterations with probability at least p_g. Therefore, selecting sample size of

the maximum of both cases, i.e.

$$n \geq \max\left\{\frac{8V_g D^2}{\varepsilon\mu_f^{FW}(1-p_g)}, \frac{2V_g}{\varepsilon_g^2\delta^2\mu\varepsilon(1-p_g)}\right\}. \tag{4.34}$$

will also satisfy the good gradient approximation event $E^{(ibFW)}$ with probability at least p_g, or satisfy the inequality of good gradient approximation for boosted Frank-Wolfe with interior optimum assumption in Definition 4.5.1. □

Define the stochastic process similar to the one with standard Frank-Wolfe algorithm from (2.13) below,

$$\Phi_k^{(ibFW)} = f(X_k) - f^* \tag{4.35}$$

We now establish the key results for complexity.

Lemma 4.5.5. *There exists $p_g \in (0,1)$ such that, with the sample size given in* (4.34) *for boosted Frank-Wolfe (under Assumptions 2.2.1 (L−smoothness) and 2.2.4 (strong convexity)), on the event $T_\varepsilon > k$,*

$$\mathbb{E}\left[\Phi_{k+1}^{(ibFW)} - \Phi_k^{(ibFW)} \middle| \mathcal{F}_k\right] \leq \begin{cases} -\beta^{iFW}\varepsilon/2 \ \textit{for non-boosted iterations} \\ -\beta^{bFW}\varepsilon/2 \ \textit{for boosted iterations} \end{cases}$$

Proof. Define Γ to be the maximum possible step size, which is 1. Similar to Theorem 2.5.1, by (2.15), we have

$$\mathbb{E}[f(X_{k+1})-f(X_k)|\mathcal{F}_k] \leq \mathbb{E}[(f(X_{k+1})-f(X_k))\mathbb{1}_{\mathcal{G}}|\mathcal{F}_k]+\Gamma D\sqrt{\frac{V_g}{n_k}}\sqrt{1-P_k}+\frac{L\Gamma^2D^2}{2}(1-P_k).$$

By Algorithm 6, each iteration's search direction is either boosted or non-boosted (standard Frank-Wolfe direction). Then, by Lemma 4.5.3

$$\mathbb{E}[f(X_{k+1}) - f(X_k)|\mathcal{F}_k] \leq -\beta\varepsilon p_g + \Gamma D\sqrt{\frac{V_g}{n_k}}\sqrt{1-p_g} + \frac{L\Gamma^2 D^2}{2}(1-p_g),$$

where

$$\beta = \begin{cases} \beta^{iFW} & \text{for non-boosted iterations} \\ \beta^{bFW} & \text{for boosted iterations} \end{cases}$$

With the sample size which is the maximum between the sample size from (3.13) and (4.12), we have

$$n_k \geq \max\left\{\frac{8V_g D^2}{\varepsilon\mu_f^{FW}(1-p_g)}, \frac{2V_g}{\varepsilon_g^2\delta^2\mu\varepsilon(1-p_g)}\right\}.$$

Pick probability p_g such that

$$p_g \geq \max\left\{\frac{\Gamma D\kappa + L\Gamma^2 D^2/2 + \beta^{iFW}\varepsilon/2}{\Gamma D\kappa + L\Gamma^2 D^2/2 + \beta^{iFW}\varepsilon}, \frac{\Gamma D\kappa + L\Gamma^2 D^2/2 + \beta^{bFW}\varepsilon/2}{\Gamma D\kappa + L\Gamma^2 D^2/2 + \beta^{bFW}\varepsilon}\right\}, \quad (4.36)$$

where

$$\kappa = \min\left\{\sqrt{\frac{\varepsilon\mu_f^{FW}}{8D^2}}, \varepsilon_g\delta\sqrt{\frac{\mu\varepsilon}{2}}\right\}.$$

Therefore,

$$\begin{aligned}\mathbb{E}[\Phi_{k+1}^{(ibFW)} - \Phi_k^{(ibFW)}|\mathcal{F}_k] &= \mathbb{E}[f(X_{k+1}) - f(X_k)|\mathcal{F}_k] \\ &\leq -\beta\varepsilon p_g + \Gamma D\kappa(1-p_g) + \frac{L\Gamma^2 D^2}{2}(1-p_g) \\ &\leq \begin{cases} -\beta^{iFW}\varepsilon/2 \text{ for non-boosted iterations} \\ -\beta^{bFW}\varepsilon/2 \text{ for boosted iterations.} \end{cases}\end{aligned}$$

Then,

$$\mathbb{E}[\Phi_{k+1}^{(ibFW)} - \Phi_k^{(ibFW)}|\mathcal{F}_k] \leq \begin{cases} -\beta^{iFW}\varepsilon/2 \text{ for non-boosted iterations} \\ -\beta^{bFW}\varepsilon/2 \text{ for boosted iterations.} \end{cases}$$

□

Theorem 4.5.5. *Suppose the initial solution $X_0 = x_0$ is deterministic with $f(x_0) - f^* > \varepsilon$. Let $T_\varepsilon = \inf\{k : f(X_k) - f^* \leq \varepsilon\}$. Using the boosted Frank-Wolfe algorithm for interior optimum case under Assumptions 2.2.1 (L−smoothness) and 2.2.4 (strong convexity) along with the sample size given in* (4.34), *$\mathbb{E}T_\varepsilon = \mathcal{O}(\varepsilon^{-1}(f(x_0) - f^*))$.*

Proof. Consider a stochastic process $\Phi^{(ibFW)} = (\Phi_k^{(ibFW)} : k \geq 0)$, adapted to $(\mathcal{F}_k : k \geq 0)$ that satisfies the expected decrease as in Lemma 4.5.5 and define a stochastic process $M_k = \delta(k \wedge T_\varepsilon) + \Phi_{k\wedge T_\varepsilon}^{(ibFW)}$ where $\delta = \min\{\beta^{iFW}\varepsilon/2, \beta^{bFW}\varepsilon/2\}$. Using the same argument on M_k from the Theorem 2.5.2, on the event $T_\varepsilon \leq k$, $M_{k+1} = M_k$, so $(M_k : k \geq 0)$ is a non-negative super-martingale. By the optional

stopping theorem, and since $X_0 = x_0$ is deterministic with $f(x_0) > f^* + \varepsilon$,

$$\mathbb{E}[M_{T_\varepsilon}] \leq \mathbb{E}[M_0] = \Phi_0^{(ibFW)}.$$

So, $\mathbb{E}[\delta T_\varepsilon + \Phi_{T_\varepsilon}^{(ibFW)}] \leq \Phi_0$, and, since $\Phi_{T_\varepsilon}^{(ibFW)} \geq 0$, $\mathbb{E}[\delta T_\varepsilon] \leq \Phi_0^{(ibFW)} < \infty$. Therefore,

$$\mathbb{E}[T_\varepsilon] \leq \frac{\Phi_0^{(ibFW)}}{\min\{\beta^{iFW}\varepsilon/2,\, \beta^{bFW}\varepsilon/2\}} \sim \mathcal{O}\left(\frac{f(x_0) - f^*}{\varepsilon}\right).$$

□

Theorem 4.5.6. *The sample sizes in each iteration that are required to achieve a good gradient estimate for interior boosted Frank-Wolfe under Assumption 2.4.1 (bounded variance) and Assumption 2.4.2 (sub-Gaussian) are* $\mathcal{O}(\varepsilon^{-2})$ *and* $\mathcal{O}(\varepsilon^{-1}\log\varepsilon^{-1})$ *respectively.*

Proof. For bounded variance assumption, pick the sample size from (4.34), and probability p_g from (4.36) where

$$n_k \geq \max\left\{\frac{8V_g D^2}{\varepsilon\mu_f^{FW}(1-p_g)},\, \frac{2V_g}{\varepsilon_g^2\delta^2\mu\varepsilon(1-p_g)}\right\},$$

and

$$p_g \geq \max\left\{\frac{\Gamma D\kappa + L\Gamma^2 D^2/2 + \beta^{iFW}\varepsilon/2}{\Gamma D\kappa + L\Gamma^2 D^2/2 + \beta^{iFW}\varepsilon},\, \frac{\Gamma D\kappa + L\Gamma^2 D^2/2 + \beta^{bFW}\varepsilon/2}{\Gamma D\kappa + L\Gamma^2 D^2/2 + \beta^{bFW}\varepsilon}\right\}.$$

where

$$\kappa = \min\left\{\sqrt{\frac{\varepsilon\mu_f^{FW}}{8D^2}}, \varepsilon_g\delta\sqrt{\frac{\mu\varepsilon}{2}}\right\}.$$

Because the constants β^{iFW} and β^{bFW} do not depend on ε, then,

$$n_k \geq \max\left\{\frac{8V_g D^2}{\varepsilon\mu_f^{FW}}, \frac{2V_g}{\varepsilon\varepsilon_g^2\delta^2\mu}\right\}\left(\frac{\Gamma D\kappa + L\Gamma^2 D^2/2}{\min\{\beta^{bFW}, \beta^{iFW}\}\varepsilon/2} + \frac{1}{2}\right) \sim \mathcal{O}(\varepsilon^{-2}).$$

With the sub-Gaussian assumption 2.4.2, we have the similar result as in the Theorem 2.5.4 from (2.21),

$$\mathbb{E}[\Phi_{k+1}^{(ibFW)} - \Phi_k^{(ibFW)}|\mathcal{F}_k] \leq -\beta\varepsilon p_g + \Gamma D\sqrt{\frac{2d}{n_k c}}\sqrt{1-p_g} + \frac{L\Gamma^2 D^2}{2}(1-p_g),$$

where $\beta = \min\{\beta^{iFW}, \beta^{bFW}\}$. Actually, we have β^{iFW} for non-boosted iterations and β^{bFW} for boosted iterations. Pick probability p_g such that

$$1 - p_g \leq \min\left\{\frac{1}{6}, \frac{\beta^2\varepsilon^2 c}{72d\Gamma^2 D^2}, \frac{\beta\varepsilon}{3L\Gamma^2 D^2}\right\} \tag{4.37}$$

so that

$$\begin{aligned}
\beta\varepsilon(1-p_g) &\leq \frac{\beta\varepsilon}{6}, \\
\Gamma D\sqrt{\frac{2d}{n_k c}}\sqrt{1-p_g} &\leq \Gamma D\sqrt{\frac{2d}{c}}\sqrt{\frac{c}{2d}}\frac{\beta\varepsilon}{6\Gamma D} = \frac{\beta\varepsilon}{6}, \text{ and} \\
\frac{L\Gamma^2 D^2}{2}(1-p_g) &\leq \frac{L\Gamma^2 D^2}{2}\cdot\frac{\beta\varepsilon}{3L\Gamma^2 D^2} \leq \frac{\beta\varepsilon}{6}.
\end{aligned}$$

Combining these three inequalities,

$$\beta\varepsilon(1-p_g) + \Gamma D\sqrt{\frac{2d}{n_k c}}\sqrt{1-p_g} + \frac{L\Gamma^2 D^2}{2}(1-p_g) \leq \frac{\beta\varepsilon}{2}.$$

Hence,

$$\mathbb{E}[\Phi_{k+1}^{(ibFW)} - \Phi_k^{(ibFW)} | \mathcal{F}_k] \leq -\beta\varepsilon p_g + \Gamma D\sqrt{\frac{2d}{n_k c}}\sqrt{1-p_g} + \frac{L\Gamma^2 D^2}{2}(1-p_g) \leq -\frac{\beta\varepsilon}{2}.$$

By sub-Gaussian assumption 2.4.2, Lemma 4.4.2, and the definition of good gradient approximation for boosted Frank-Wolfe with interior optimum assumption in Definition 4.5.1, take the error

$$s = \min\left\{\varepsilon_g\delta\sqrt{\frac{\mu\varepsilon}{2}}, \frac{\sqrt{2\mu_f^{FW}}}{4D}\right\},$$

and it suffices to take n so large that $2d\exp(-ncc'\varepsilon) \leq 1 - p_g$, where

$$c' = \min\left\{\frac{\varepsilon_g^2\delta^2\mu}{2}, \frac{\mu_f^{FW}}{8D^2}\right\}.$$

Then,

$$n \geq \frac{1}{cc'\varepsilon}\log\left(\frac{144d^2\Gamma^2D^2}{\beta^2\varepsilon^2}\right) \sim \mathcal{O}(\varepsilon^{-1}\log\varepsilon^{-1}).$$

□

In summary, from Theorem 4.5.5 and Theorem 4.5.6, the total work complexity for boosted Frank-Wolfe algorithm under Assumptions 2.2.1 ($L-$smoothness), 2.2.4 (strong convexity), and 3.4.1 (interior optimum) are

1. $\mathcal{O}(\varepsilon^{-3})$ under Assumption 2.4.1 (bounded variance), and
2. $\mathcal{O}(\varepsilon^{-2}\log\varepsilon^{-1})$ under Assumption 2.4.2 (sub-Gaussian)

The bound on the expected stopping time obtained in Theorem 4.5.5 can be expressed more explicitly as

$$\mathbb{E}[T_\varepsilon] \leq \frac{2(f(x_0) - f^*)}{\min\{\beta^{iFW}, \beta^{bFW}\}\varepsilon} \sim \mathcal{O}\left(\max\left\{\frac{D^2}{\mu_f^{FW}}, \frac{1}{\mu}\right\}\frac{L}{\varepsilon}\right).$$

Under Assumption 2.4.1 (bounded variance), Theorem 4.5.6 yields the sample size complexity

$$\mathcal{O}\left(\frac{V_g L^2 D^2}{\varepsilon^2}\max\left\{\frac{D^2}{\mu_f^{FW}}, \frac{1}{\mu}\right\}^2\right),$$

while under Assumption 2.4.2 (sub-Gaussian), the sample size complexity becomes

$$\mathcal{O}\left(\frac{1}{\varepsilon}\max\left\{\frac{D^2}{\mu_f^{FW}}, \frac{1}{\mu}\right\}\log\frac{dDL}{\varepsilon}\max\left\{\frac{D}{\mu_f^{FW}}, \frac{1}{\mu}\right\}\right).$$

Combining the iteration and sample size complexities gives the corresponding total work complexities. Under the bounded-variance assumption, the total work complexity is

$$\mathcal{O}\left(\frac{V_g L^3 D^2}{\varepsilon^3}\max\left\{\frac{D^2}{\mu_f^{FW}}, \frac{1}{\mu}\right\}^3\right),$$

whereas under the sub-Gaussian assumption, it is

$$\mathcal{O}\left(\frac{L}{\varepsilon^2}\max\left\{\frac{D^2}{\mu_f^{FW}}, \frac{1}{\mu}\right\}^2\log\frac{dDL}{\varepsilon}\max\left\{\frac{D}{\mu_f^{FW}}, \frac{1}{\mu}\right\}\right).$$

These expressions highlight the dependence of the overall complexity on the smoothness constant L, the variance parameter V_g, the strong convexity parameter μ, strong convexity interior parameter μ_f^{FW}, the feasible-region diameter D, and the dimension d.

CHAPTER 5

BOOSTED FRANK-WOLFE OVER AN UNBOUNDED FEASIBLE REGION

5.1 Introduction

Frank-Wolfe algorithms have been applied to a wide range of optimization problems, including machine learning tasks, regularized optimization models, and matrix completion problems. Most existing analyses and applications of Frank–Wolfe methods assume that the feasible region is compact or bounded. Nevertheless, several important practical problems naturally involve unbounded feasible regions while still remaining suitable for projection-free optimization methods. For example, [55] applies Frank–Wolfe methods to structural SVM training, where the constraint set can be unbounded. Other examples arise in statistical learning and signal processing, including constrained least squares problems, ℓ_1-trend filtering in [51], and basis pursuit problems in [20]. In a stochastic setting these problems are linearly constrained simulation optimization problems (LCSOPs), providing a natural motivation for studying Frank–Wolfe methods beyond the bounded-feasible-region setting.

A broad class of algorithms can be brought to bear on LCSOPs. For example, [10] explored projected gradient descent (PGD), an away-step Frank-Wolfe (AFW) algorithm and an active set method in combination with 4 different line search strategies through computational comparisons on LCSOPs. There it was found that PGD and the Frank-Wolfe algorithm typically outperformed the active set method, though not universally. One of the problems explored in that paper involved an unbounded feasible region, but the variant of the Frank-Wolfe algorithm

that was employed required a *bounded* feasible region. To handle this difficulty, a linear constraint was added that rendered the feasible region bounded and that was sufficiently "loose" that it was not expected to change the optimal solution to the problem. While this strategy allowed the Frank-Wolfe algorithm to make progress, the algorithm struggled on this problem compared to the other algorithms.

Given the strong overall performance of the Frank-Wolfe algorithm in [10], except on an unbounded problem, we are motivated to seek an improved Frank-Wolfe algorithm for LCSOPs that requires no tailoring when encountering a problem with an unbounded feasible region. It is important that no tailoring be required, to minimize the knowledge required of a user. We want to retain the strong performance of the Frank-Wolfe algorithm on bounded problems, while also ensuring competitive performance on unbounded problems. (Problems with unbounded feasible regions are common in practice, e.g., in inventory problems with unbounded order quantities, or in capacity expansion problems where capacity comes at a cost rather than being constrained.) [85] develop a Frank-Wolfe method for linearly constrained unbounded regions, but their method applies only to very specially structured feasible regions that are a direct sum of a linear subspace and a bounded constraint set.

In this chapter, we attempt to develop a Frank-Wolfe algorithm for general LCSOPs. The new algorithms we explore are not a direct extension of the standard Frank-Wolfe algorithm [26]. Rather, we first choose to extend the AFW algorithm [89] and, separately, the boosted Frank-Wolfe (BFW) algorithm [21]. These algorithms have demonstrated strong performance on bounded feasible regions and are thus natural candidates for extension. Our first extensions allow these algorithms to incorporate not just *vertices* of the feasible region in specifying the search

direction, but also *extreme directions*.

The previous chapter focused on the convergence analysis of boosted Frank–Wolfe algorithms over bounded feasible regions. Building on that foundation, this section extends the boosted Frank–Wolfe framework to linearly constrained feasible regions that may be unbounded. In Algorithm 5, the search direction is constructed iteratively by solving a sequence of linear optimization subproblems using the negative estimated gradient. For bounded feasible regions, the solutions to these subproblems correspond to vertices of the feasible region. In contrast, when the feasible region is unbounded, the linear optimization oracle may identify directions along which the objective is unbounded, corresponding to extreme rays of the feasible region. To accommodate this possibility, we modify the boosted Frank–Wolfe framework to detect such extreme directions and incorporate them into the iterative construction of the search direction. The resulting algorithm extends the boosted Frank–Wolfe methodology beyond bounded polytopes while preserving its projection-free structure.

The numerical results in this section are based on simulation-optimization problems from the open-source *SimOpt* package. The performance of the unbounded boosted Frank-Wolfe algorithm is compared against classical optimization methods, including Projected Gradient Descent (PGD) and the Away-step Frank-Wolfe algorithm. In addition, motivated by the slow-progress phenomenon commonly referred to as "plateauing," we investigate several variants of the unbounded boosted Frank–Wolfe algorithm designed to mitigate this effect and improve practical convergence behavior.

Our objective in this chapter is not to establish iteration or sample complexity

guarantees, but rather to provide an empirical assessment of the proposed algorithm and to better understand its practical behavior on simulation-optimization problems. In particular, we focus on identifying scenarios in which the unbounded boosted Frank–Wolfe algorithm performs well, examining the impact of plateauing, and evaluating the effectiveness of the proposed variants in overcoming this phenomenon. Through these experiments, we aim to gain insight into the strengths and limitations of the algorithm beyond what is captured by the theoretical analysis.

This chapter is organized as follows. Section 5.2 presents the unbounded stochastic boosted Frank–Wolfe algorithm and the unbounded stochastic Away-step Frank–Wolfe algorithm. These methods are based on Algorithms 4, 5, and 2, respectively, but are modified to handle unbounded solutions returned by the linear programming oracle. Section 5.3 introduces the simulation-optimization problems from [77] that serve as test instances throughout this chapter. Section 5.4 presents numerical results for the various algorithms and problem instances, and highlights a challenge observed in several experimental settings. Section 5.5 proposes new variants of the boosted Frank–Wolfe algorithm designed to address this challenge. Finally, Section 5.6 presents experimental results for all algorithms, including the proposed variants, and discusses their relative strengths and weaknesses. Section 5.7 summarizes the experimental results presented in the previous sections and compares the performance of all algorithms considered in this chapter.

5.2 Unbounded Boosted Frank-Wolfe Algorithm

We now present the unbounded boosted Frank–Wolfe algorithms. These algorithms follow the same overall framework as Algorithm 4 and Algorithm 5. In particular,

Algorithm 7 is analogous to Algorithm 4, with the main differences arising from the unbounded nature of the feasible region. Specifically, the search direction is normalized to control the magnitude of updates, and the step size is determined using a line search procedure rather than a predefined bounded-domain step size rule. Let LP(r) be a function that returns either a vertex solution to the linear program $\max_{x \in \mathcal{X}} r^T x$ when the solution is finite, or an extreme direction u of unit norm such that $r^T u > 0$ and such that $x + \lambda u$ is feasible for any feasible x and $\lambda > 0$.

Algorithm 7 Unbounded Boosted Frank-Wolfe (uBFW).

Initialize x_0 to be a vertex v, maximum number of oracle steps T, an improvement tolerance δ.
for $k = 0, 1, 2, ...$ **do**
  Estimate the gradient $\nabla f(x_k)$ by g_k
  $d_k, \Lambda_k \leftarrow$ Unbounded Boosted Direction Oracle(T, δ, g_k)
  $d_k \leftarrow d_k / \Lambda_k$
  $\gamma_k^{max} \leftarrow$ maximum step size computed from minimum ratio test
  $\gamma_k \leftarrow \arg\min_{\gamma \in [0, \gamma_k^{max}]} \hat{f}(x_k + \gamma d_k)$ (line search)
  $x_{k+1} \leftarrow x_k + \gamma_k d_k$
end for

Algorithm 8 extends Algorithm 5 by incorporating the possibility of extreme rays returned by the linear programming solver. The search domain consists of conic combinations of the directions $\mathcal{V} - x_k$, which correspond to the standard bounded-domain setting, together with the extreme directions of the feasible region $\mathcal{X}$, which arise specifically in the unbounded case.

The algorithm iteratively improves the alignment between the negative gradient and the constructed search direction. At each iteration, a linear program is solved to identify either a vertex or an extreme direction that most effectively reduces the discrepancy between the negative gradient and the current search direction. The procedure terminates when the alignment improvement falls below a prescribed threshold δ, which is typically chosen as 10^{-3}. In Algorithm 8, the parameter T

denotes the maximum number of alignment-improvement iterations allowed within the Boosted Direction Oracle, and is typically set to 10.

Algorithm 8 Unbounded Boosted direction oracle.

Accept as input T, the maximum number of iterations, δ, a parameter to indicate sufficient alignment, and g, the estimated gradient
Initialize d_0 to be a zero vector for the direction, $\Lambda = 0$ to be a normalizing term.
for $t = 0, 1, 2, ..., T-1$ **do**
  $r_t \leftarrow -g - d_t$; this is the running residual
  $v_t \leftarrow \text{LP}(r_t)$; solve the LP
  if LP returned a vertex solution **then**
    $w = v_t - x$
  else
    $w = v_t$
  end if
  $u_t \leftarrow \arg\max_{u \in \{w, -d_t/\|d_t\|\}} r_t^T u$; see [60] to understand this step
  $\lambda_t \leftarrow r_t^T u_t / \|u_t\|^2$
  $d' \leftarrow d_t + \lambda_t u_t$
  if $align(-g, d') - align(-g, d_t) \geq \delta$ **then**
    $d_{t+1} \leftarrow d'$
    $\Lambda = \begin{cases} \Lambda + \lambda_t & \text{if } u_t = w \\ \Lambda(1 - \lambda_t/\|d_t\|) & \text{if } u_t = -d_t/\|d_t\| \end{cases}$
  else
    break
  end if
end for
return d_t, Λ

The unbounded boosted Frank–Wolfe algorithm is also compared with an Away-step Frank–Wolfe method that has also been modified to handle unbounded solutions returned by the linear programming oracle. This modified variant extends Algorithm 2 to accommodate extreme directions arising from the unbounded feasible region. The resulting algorithm is presented in Algorithm 9 where it accepts both vertices and extreme rays detected by Linear Programming.

In each iteration, we first solve LP($-g_k$), i.e., we solve a linear program whose objective function is the linearized function. If the optimal solution is a vertex, s_k, then the potential search direction is $d_k^{FW} = s_k - x_k$, i.e., we search in the

direction of that vertex. If, instead, the linear program is unbounded and returns the (normalized) extreme direction s_k, then the potential search direction is $d_k^{FW} = s_k$.

The key difference with the usual AFW algorithm for bounded feasible regions is that the linear program can be unbounded. In that case, we use the extreme direction that is returned by the linear program solver as a search direction. The "bookkeeping" for maintaining the representation of the current point as not only a convex combination of extreme points, which is standard, but also a conic combination of extreme directions, which is new, is also slightly more complicated.

In the case that the search direction is the extreme direction, $d_k^{FW} = s_k$, obtained by solving the linear program, the step size is more complicated. We retain a parameter, γ_k^{max} that limits the maximum possible step size in Iteration k. In each iteration of this form we perform a line search for the step size $\gamma_k \in [0, \gamma_k^{max}]$. If the optimal step size, γ_k is attained at the endpoint, γ_k^{max}, then we increase the maximal step size by a (geometric) growth factor, i.e., we set $\gamma_{k+1}^{max} = \gamma_k^{max}/r$ for some $r < 1$. If, instead, the optimal step size, γ_k, is interior to the interval then we reduce the maximal step size to γ_k.

In Frank–Wolfe-type algorithms, it is possible to encounter the same unbounded solution of the linear programming subproblem, and hence the same extreme ray, over several consecutive iterations. Therefore, if the line search returns a step size satisfying $\gamma_k < \gamma_k^{\max}$, then the minimum of the objective along that ray is likely attained before $\gamma_k^{\max}$. Since the same extreme ray may be selected again in future iterations, the computed step size γ_k provides useful information about an appropriate upper bound along that direction. Consequently, we update the

maximum step size directly by setting $\gamma_{k+1}^{\max} = \gamma_k$.

In contrast, methods whose search directions are not derived from the extreme rays of the feasible region, such as projected gradient descent with search direction $-g(x_k)$, typically generate a different direction at each iteration. In this case, the condition $\gamma_k < \gamma_k^{\max}$ only indicates that the current upper bound is too large for the present search direction. Because there is no reason to expect future search directions to have a similar optimal step length, the information contained in γ_k is less reliable for subsequent iterations. As a result, the maximum step size is reduced more conservatively, for example by setting $\gamma_{k+1}^{\max} = r\gamma_k^{\max}$ for some $r \in (0, 1)$, rather than replacing it directly with γ_k.

Algorithm 9 Unbounded Away-step Frank-Wolfe (uAFW).

Initialize x_0 to be a vertex v, set $\alpha_v^{(0)} = 1$ and all other coefficients to 0, set a step size discount factor $r \in (0, 1)$
for $k = 0, 1, 2, ...$ **do**
  $s_k \leftarrow \text{LP}(-g_k)$; the usual Frank-Wolfe vertex or an extreme direction
  $v_k \leftarrow \arg\max_{v \in \mathcal{S}^{(k)}} {g_k}^T v$; a "bad" vertex
  $d_k^{FW} \leftarrow \begin{cases} s_k - x_k & \text{if the LP solution was a vertex} \\ s_k & \text{if the LP solution was an unbounded direction} \end{cases}$
  $d_k^A \leftarrow x_k - v_k$ (away direction)
  if $-g_k^T d_k^{FW} \geq -g_k^T d_k^A$ **then**
    $d_k \leftarrow d_k^{FW}$
    $\gamma_k \leftarrow \begin{cases} \arg\min_{\gamma \in [0,1]} \hat{f}(x_k + \gamma d_k) & \text{if LP gives a vertex solution} \\ \arg\min_{\gamma \in [0, \gamma_k^{max}]} \hat{f}(x_k + \gamma d_k) & \text{if LP gives an unbounded direction} \end{cases}$
    $\gamma_{k+1}^{max} \leftarrow \begin{cases} \frac{\gamma_k^{max}}{r} & \text{if LP gives an unbounded direction and } \gamma_k = \gamma_k^{max} \\ \gamma_k & \text{if LP gives an unbounded direction and } \gamma_k < \gamma_k^{max} \end{cases}$
  else
    $d_k \leftarrow d_k^A$
    $\gamma_k \leftarrow \arg\min_{\gamma \in [0, \alpha_{v_k}^{(k)}/(1-\alpha_{v_k}^{(k)})]} \hat{f}(x_k + \gamma d_k)$ (line search)
    $x_{k+1} \leftarrow x_k + \gamma_k d_k$
    Update the weights $\alpha^{(k)}$ giving the vertex representation of x_{k+1} to $\alpha^{(k+1)}$
  end if
end for

The unbounded boosted Frank–Wolfe algorithm will also be compared with another

classical optimization method, namely Projected Gradient Descent (PGD), which is presented in Algorithm 10. PGD, adapted from [46], moves in the direction of the negative gradient, uses an L_2 projection $\mathcal{P}_{\mathcal{X}}$ back to the feasible region to retain feasibility, then performs a linesearch in the direction of the projected point. The projection is obtained by solving a quadratic program using CVXPY ([24], [2]). Stepsizes are constrained to an iteration-specific maximum stepsize that is chosen adaptively.

Boosted Frank-Wolfe attempts to mimic the behavior of Projected Gradient Descent by aligning its search direction with the negative gradient using combinations of vertices and extreme rays, while avoiding the need for an explicit projection step. The numerical experiments will compare the performance of projected negative gradient directions in PGD with the inexact negative gradient directions constructed from vertices and extreme rays in the boosted Frank-Wolfe framework.

Algorithm 10 Projected gradient descent (PGD).

Initialize feasible solution $x_0 \in \mathcal{X}$; maximum step size $\gamma_0^{max} \in [1, \infty)$; step size discount factor $r \in (0, 1)$
for $k = 0, 1, 2, ...$ **do**
  $d_k \leftarrow -g_k$
  $y_{k+1} \leftarrow x_k + \gamma_k^{max} d_k$
  if y_{k+1} is NOT feasible **then**
    $y_{k+1} \leftarrow \mathcal{P}_{\mathcal{X}}(y_{k+1})$
    $d_k \leftarrow \frac{(y_{k+1} - x_k)}{\gamma_k^{max}}$
  end if
  Let $\gamma_k = \arg\min_{\gamma \in [0, \gamma_k^{max}]} \hat{f}(x_k + \gamma d_k)$ (line search)
  if $\gamma_k = \gamma_k^{max}$ **then**
    $\gamma_k^{max} \leftarrow \frac{\gamma_k^{max}}{r}$
  else
    $\gamma_k^{max} \leftarrow r\gamma_k^{max}$
  end if
  $x_{k+1} \leftarrow x_k + \gamma_k d_k$
end for

5.3 Problems

Here we provide only brief remarks on the choices of test problems. For full details on the test problems, see the [77].

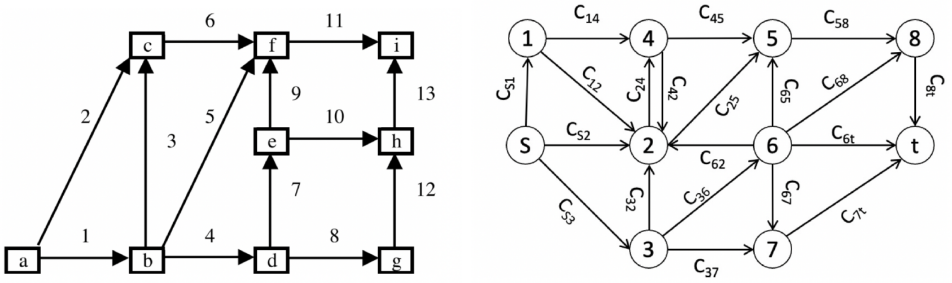


Figure 5.1: An example of the SAN problem consisting of 9 nodes and 13 edges (left), and an example of the SMFCVX problem with 10 nodes.

Our test problems consist of 10 random instances of each of the following:

1. Constrained Stochastic Activity Network (SAN)
2. Stochastic Max Flow (SMFCVX)
3. Open Jackson Network (OJN)

for a total of 30 test problems. All problems are convex, and we obtain gradient estimates using infinitesimal perturbation analysis. Both SAN and OJN have unbounded feasible regions, while SMFCVX has a bounded feasible region. The SAN problem involves minimizing the maximum-length path from a source node to a sink node in a feed-forward network. The task durations (arc lengths) are random, though controlled through decision variables specifying their means. The SMFCVX problem entails maximizing the flow through a capacitated network, the capacities of which are random but partially controlled through decision variables. The OJN problem entails minimizing the mean steady-state number of jobs in an open Jackson queueing network, with decision variables giving the mean service rates.

5.4 Initial Experiments

All of our experiments use *Simopt*[77]. In reporting the results, we compare the various algorithms based only on the simulation replications used; we do not compare the algorithms on their *processing time*. This is deliberate; see [25] for a detailed justification of this approach. In short, the reason is that simulation models can be computationally expensive to run compared with deterministic algorithmic elements such as computing projections in PGD or solving linear programs in the Frank-Wolfe algorithm variants.

All problems are run using common random numbers across all solvers, to ensure a fair comparison; see [25]. We ensure different random problem instances are generated using different substreams, and each distinct problem factor and model factor has its own individual subsubstream. We name the problems we generated by the names introduced in Section 5.3 with the instance's label. For example, the problem "SAN-R10" is the 10^{th} instance of the SAN problem.

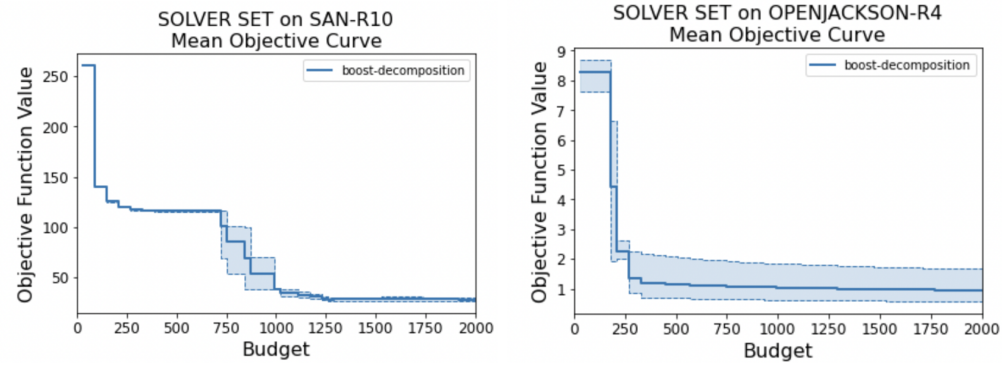


Figure 5.2: An instance of SAN tackled using Algorithm 7 exhibits a plateau in the budget interval $250 - 750$ (left), and an instance of OJN has a plateau in the budget interval $500 - 2000$.

Algorithm 4, which we name *Decomposition*, attempts to align the negative gradient with a conic combination of Frank-Wolfe and extreme directions. The use of a combination of vertices as well as extreme directions can improve the align-

ment with the negative gradient compared to the standard Frank-Wolfe algorithm. However, poor alignment is still possible when the feasible region is unbounded. Poor alignment is evident in progress plots. These plots give the estimated objective function value of the best solution found to date on the vertical axis and the expended budget, as measured in simulation replications, on the horizontal axis. Plateaus are evident in Figure 5.2, in which the objective function is not improved for an extended period.

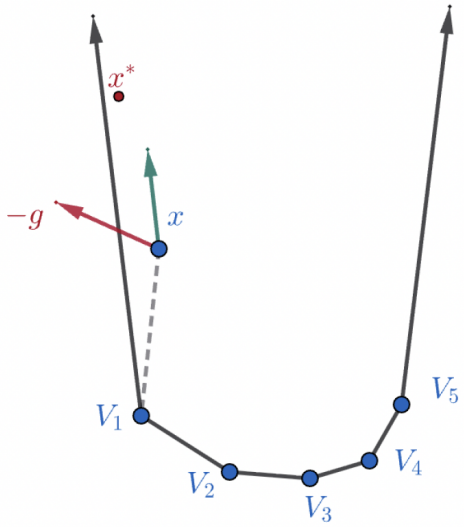


Figure 5.3: An example of the plateau effect with a 2D problem. The optimal solution x^* lies far away from the vertices below. The search direction will not be boosted and only follows an extreme ray from Linear Programming .

5.5 Additional Algorithms

To mitigate the plateau effect observed in Figure 5.2, we propose several variants of the BFW algorithm that incorporate a few key modifications. These modifications are mostly intended to enhance the set of search directions available to the BFW algorithm, thereby better aligning the search direction to the negative gradient.

1. **Negative Gradient**:

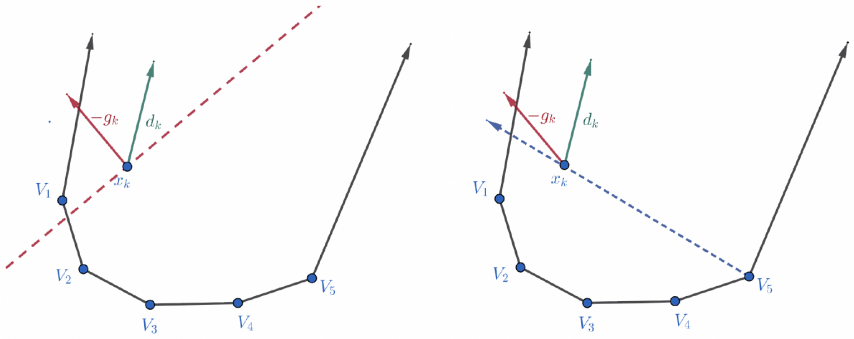


Figure 5.4: A plane (dashed line) perpendicular to the negative gradient is added as a new constraint in an unbounded boosted Frank-Wolfe (left), and away-steps are added to the components of final search direction (right).

We compute the search direction d_k as described in Algorithm 4. If the maximum step size γ_k^{max} computed from the minimum ratio test is infinite, we recompute γ_k^{max} using the direction of the negative gradient. If this recomputed γ_k^{max} is non-zero, we replace the search direction, d_k, with the negative gradient; otherwise we keep the original d_k.

2. **Outward Cutting**: When the search direction is unbounded, we add a new constraint (cut), hoping that the cut creates new vertices. The new cut at such an iteration, k, is ${g_k}^T x \leq {g_k}^T x_k$ where x_k is the current solution at iteration k. This cut is valid if we use sample-average approximation with a fixed sample size and the gradient estimate, g_k, is obtained using infinitesimal perturbation analysis.

3. **Cutting**: This is similar to the previous variation, but a new cut is added in every iteration.

4. **Away**: In addition to directions that are towards vertices and extreme directions, we also allow the algorithm to consider away directions of the form $x_k - v_k$ where x_k is the solution at iteration k and v_k is the bad vertex that is computed in the same manner as in the AFW algorithm. In this variant, we add vertices to the active set if they are selected in the Boosted Direc-

tion Oracle and remove the "bad" vertex from the active set when its weight reaches 0.

5.6 Additional Experiments

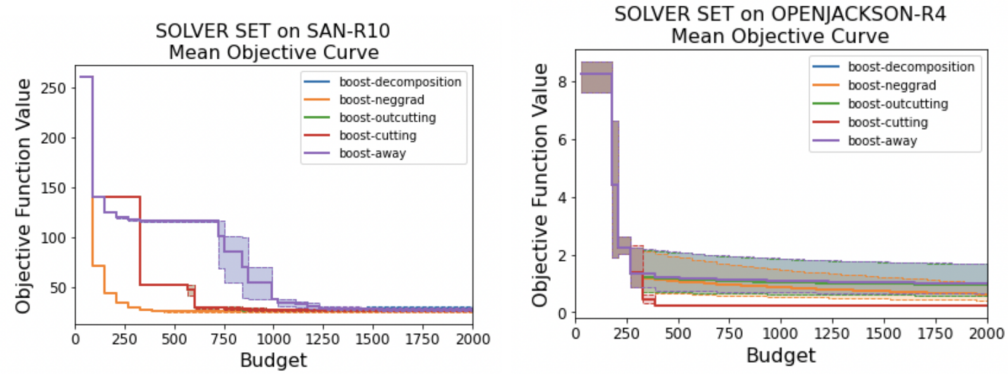


Figure 5.5: An instance of SAN (left), and OJN (right), tackled using variants of Algorithm 4.

Figure 5.5 illustrates the performance of the BFW variants on the same problems shown in Figure 5.2, to demonstrate that plateauing has been reduced. In the SAN-R10 problem, both the away-step and decomposition methods exhibit plateau behavior (their curves lie on top of one another), whereas the negative gradient and cutting-plane variants significantly reduce this effect. While the decomposition method tends to move steadily toward the interior of the feasible region (not shown in the plots, but observed in the experiment), the negative gradient method follows the negative gradient direction more closely, leading to improved performance. The cutting-plane methods (both outward cutting and cutting) alleviate the plateau by introducing additional constraints. These constraints generate new vertices, which enrich the set of directions available for constructing search directions. In the OPENJACKSON-R4 problem, the cutting-plane method outperforms the other variants. In this instance, *cutting* has a better performance than *outward cutting*. The additional constraints and resulting vertices from adding cuts

at each step appear to enhance the search direction, accelerating convergence and helping to avoid the plateau. This improvement is particularly notable when the search direction leads to a bounded maximum step size, (not shown), allowing the algorithm to make meaningful progress. In comparison, the *decomposition* method often moves deeper into the interior without such bounded steps, resulting in slower convergence.

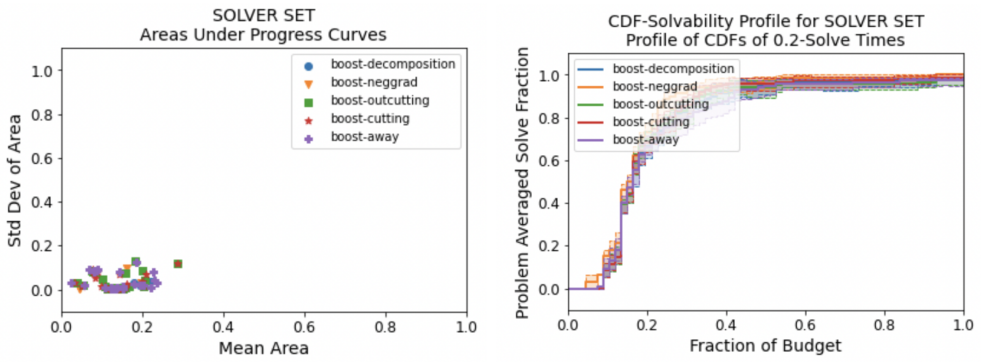


Figure 5.6: Area scatter plot of different variants of BFW solvers (left) and CDF solvability profile at level 0.2 (right) on all test problems.

Figure 5.6 summarizes the performance of all BFW variants on all test problems. To recap, [25] where the curves in Figure 5.6 are introduced, the area-under-curve plot computes the mean and standard deviation of the area under the progress curve on different macroreplications, while the CDF solvability profile gives the fraction of problems solved to within 20% of the initial optimality gap. The scatter plot suggests similar overall performance among the variants. However, both the *cutting* and *outward cutting* methods exhibit higher standard deviations and significantly larger mean areas in one of the SAN problem instances. While these methods aim to enrich the search space by introducing additional constraints, the added cuts may inadvertently remove useful vertices, thereby impairing rather than enhancing the available search directions as shown in the area under progress curves where cutting-plane methods have high mean areas on one problem instance. The CDF solvability profile indicates that all variants are able to solve most prob-

lems within the given budget, with the negative gradient variant showing a slight performance advantage over the others.

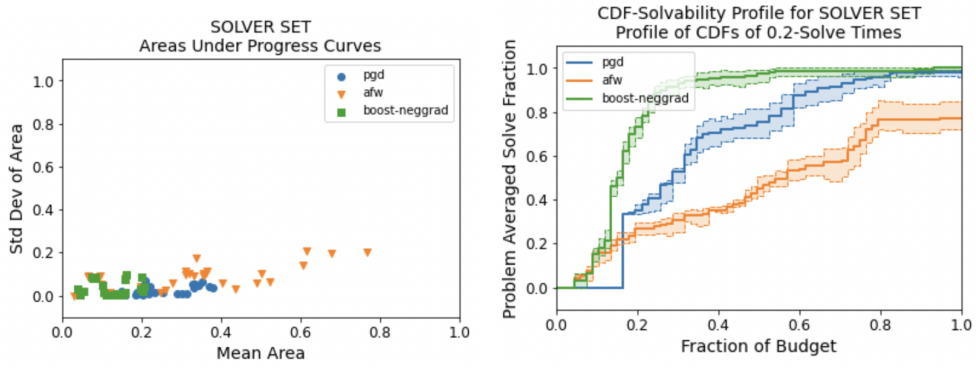


Figure 5.7: Area scatter plot on different solvers (left) and CDF solvability profile at level 0.2 (right).

Figure 5.7 compares PGD, AFW, and BFW with negative gradient. The area scatter plot indicates that BFW generally outperforms both PGD and AFW. While PGD exhibits a lower standard deviation in area, it also shows a larger mean area under the progress curve, suggesting a slower convergence toward the optimal solution. AFW, on the other hand, has both higher standard deviation and mean area than the other two algorithms, indicating more variability in performance, though it may perform well on some instances. In the solvability profile, both BFW and PGD solve nearly all instances, whereas AFW solves only approximately 75% of the problems.

Figure 5.8 shows the performance of the three algorithms—PGD, AFW, and BFW with negative gradient—on the SMFCVX-R5 and OPENJACKSON-R6 problems. On SMFCVX-R5, both AFW and BFW reach good-quality solutions very quickly, but eventually PGD finds a better solution. In this example, the optimal solution lies on or near a vertex, and AFW can quickly move toward it, resulting in fast convergence. In contrast, BFW prioritizes search directions that are closely aligned with the negative gradient, which may not point directly toward a vertex, thereby

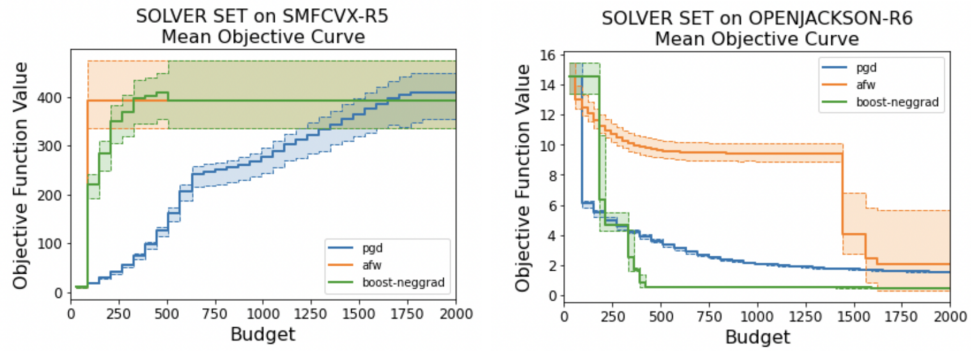


Figure 5.8: Mean progress curves of SMFCVX-R5 (left) and OPENJACKSON-R6 (right).

slowing its performance. Still in the SMFCVX-R5 plot, PGD exhibits two distinct phases in its progress curve: it initially moves quickly along the negative gradient with increasing step sizes, but after reaching the boundary around budget 750 its progress slows considerably. In the OPENJACKSON-R6 problem, AFW performs poorly with the unbounded feasible region, particularly in the budget range of 500−1500, where it repeatedly selects an unbounded direction with little improvement. In contrast, PGD steadily follows the negative gradient, and BFW outperforms both alternatives.

5.7 Conclusions

When the feasible region is bounded, the algorithms we developed all work relatively well. The key challenge is to also handle unbounded feasible regions. Our first algorithms exhibit a plateauing effect, in which the algorithms appear to stall for some period of time before once again making rapid progress. This plateauing effect seems to arise when searching in unbounded directions, i.e., with search directions that permit unbounded step sizes while remaining feasible.

Plateauing arises when there are limited resources for choosing a direction. The

AFW algorithm exhibits a more pronounced plateauing effect because it is restricted to moving toward or away from vertices and along extreme directions. In contrast, BFW forms a combination of directions toward both vertices and extreme points, resulting in directions that more closely align with the negative gradient. This allows greater flexibility and reduces plateauing. While PGD does not exhibit this effect, similar behavior can occur when the algorithm repeatedly hits the boundary and projects back into the feasible region.

We proposed several algorithmic modifications to attempt to obviate plateauing in the BFW algorithm. All of the new variants perform better; performance is significantly improved, though still not completely satisfactory, with some residual plateauing. Of our proposed modifications, the variant of BFW that permits a search in the direction of the negative gradient performed the best in both speed in making progress towards an optimal solution and in the quality of the solution attained once the simulation budget was expended. Other variants do better in selected problem instances, but BFW with negative gradient searches is reliable and fast. Cutting plane algorithms may be worth further study, especially since they can render the feasible region bounded, but they can also limit the search direction options due to removing vertices.

Our extension of the AFW algorithm for unbounded regions has mostly disappointing performance, which arises primarily due to the plateauing effect. We have attempted to limit that effect through various algorithmic innovations, but the effect persists in experiments. We conjecture that there is an interaction between the choices of search directions and the step sizes for the line search that, with further study, might improve the performance. The algorithm does, however, perform well compared to the other algorithms when the optimal solution lies near

a vertex.

Both PGD and BFW with negative gradient searches perform reasonably well, but the latter was the clear winner in our experiments.

In conclusion, we believe that our newly developed BFW algorithm (with negative gradient searches) shows strong promise, whether the feasible region is bounded or unbounded, and that we have been successful in our goal of delivering a Frank-Wolfe style algorithm that can handle unbounded feasible regions. The algorithm still exhibits some plateauing effects that merit further study. On a perhaps related note, we believe there is the potential for "scaling" issues, in which the algorithm may not adapt when we change the scale of the decision variables; the algorithm is not "scale free."

As a final comment, this article showcases the utility of SimOpt in algorithm development.

CHAPTER 6

DIALYSIS SCHEDULING

6.1 Introduction

Hemodialysis is a life-sustaining treatment for patients with kidney failure, serving as a substitute for essential kidney functions. According to [65], more than 400,000 Americans receive hemodialysis treatment. Patients typically require treatment three times per week, with each session lasting approximately three to four hours. Given the large number of patients and the frequency of treatment, efficient operation of dialysis clinics is crucial for both patient care and resource utilization. A typical dialysis session involves several operational tasks, including scheduling patients, connecting patients to dialysis machines (*put-on*), and disconnecting them after treatment (*take-off*). These tasks are primarily performed by healthcare technicians. According to [78], in one shift a technician is often responsible for approximately 12 patients served by 4 dialysis stations or chairs, each of which can accommodate only one patient at a time. Clinic operations commonly begin as early as 4 or 5 AM and continue until all patients have completed treatment and been disconnected from the machines.

Because dialysis resources are limited and treatment sessions are lengthy, appointment scheduling and technician decision-making policies play an important role in clinic performance. Poor scheduling can lead to substantial chair idle time, resulting in inefficient utilization of equipment, staff, and clinic capacity [43]. At the same time, excessive waiting can negatively affect patient experience and increase the workload of healthcare staff. Furthermore, patient arrivals, treatment durations, and task completion times are inherently uncertain and vary across in-

dividuals. Consequently, both appointment schedules and technician policies must be designed to operate effectively under uncertainty while maintaining efficient patient flow and resource utilization.

In this chapter, we examine daily planning in a dialysis clinic with the objective of improving patient flow, reducing total waiting time, and decreasing the clinic makespan, or completion time. The first stage of this process is *patient scheduling*, defined as the process of assigning appointment times to all patients. Several studies, including [87] and [37], provide foundational reviews of appointment scheduling methods. [87] analyzed scheduling policies designed to reduce patient waiting times and clinic congestion by scheduling two patients at the beginning of a session. Later works such as [36], [50], and [71] identified efficient scheduling structures with a "dome-shaped" pattern, in which consecutive arrivals are scheduled more closely together at the beginning and toward the end of the clinic session.

More recently, [78] proposed push and interlaced scheduling methods for dialysis clinics. These approaches either divide patients into blocks or introduce strategic pauses between consecutive patients, allowing technicians to complete future tasks when earlier patients finish treatment. In addition, [37] highlights several factors that can improve waiting times and resource utilization under uncertainty, such as dividing treatment sessions into blocks and incorporating service-time information into scheduling decisions. One particularly important strategy is to use service times to assign priorities to patients. In the context of dialysis clinics, treatment time serves as a natural measure for determining patient priority.

In this chapter, we adopt the scheduling methods proposed by [78] and further investigate the effect of prioritizing patients according to their treatment times. Ef-

fective scheduling, if possible (patients may choose their own appointment times), enables the clinic to control patient arrivals to some extent; however, uncertainty remains unavoidable. Patients may arrive late, treatment durations may vary, and task completion times may differ from expectations. Consequently, technicians must also follow effective operational policies to manage patient flow and utilize clinic resources efficiently throughout the day. While [78] demonstrates the benefits of improved scheduling methods, the impact of technician policies on clinic performance has received comparatively less attention. This chapter therefore focuses on both scheduling decisions and technician policies, and studies how their interaction affects waiting times, makespan, and overall clinic efficiency.

According to [37], although First-In-First-Out (FIFO), or First-Come-First-Served (FCFS), remains common in practice, more informed scheduling and sequencing rules are often needed to effectively reduce waiting times and alleviate congestion. In particular, patient service times can provide valuable information for prioritizing patients and improving system performance. This observation is consistent with the job scheduling literature, including [73], [86], [17], and [35], which study rules such as First-In-First-Out (FIFO), Shortest Processing Time (SPT), and Longest Processing Time (LPT).

Unlike FIFO, both SPT and LPT explicitly incorporate processing-time information when determining priorities. SPT prioritizes jobs with shorter processing times, whereas LPT prioritizes jobs with longer processing times. These rules generally have different effects on system performance. As summarized in [73], SPT is commonly associated with reducing average waiting and completion times, while LPT is often employed to improve makespan-related objectives and workload balancing. Although these rules were originally developed for job scheduling

environments, they can also be adapted as operational policies for technicians in dialysis clinics, where treatment time serves as a natural analogue of job processing time.

However, as technician policies, FIFO, SPT, and LPT make decisions only among patients who have already arrived and are waiting for service. Consequently, they do not account for future patient arrivals, future treatment completions, or the downstream effects of current decisions on subsequent clinic operations. These limitations motivate the development of more sophisticated policies that incorporate additional information about future tasks and patient flow in order to further reduce total waiting time and clinic makespan.

Several studies in healthcare operations, such as [76], and [82], formulate mixed-integer programming models to determine appointment schedules using patient-specific information. In particular, [82] focuses on minimizing operational costs subject to scheduling constraints, whereas [76] seek to minimize patient waiting times together with other operational costs under appointment scheduling constraints.

Several healthcare treatments that share operational characteristics with dialysis, such as chemotherapy, also require effective patient scheduling and resource management. Chemotherapy treatment is similar to dialysis in that patients typically require long treatment sessions. According to [74], chemotherapy treatment durations can range from 120 to 270 minutes. In that study, patient information was used to improve the utilization of resources, such as nursing staff, and discrete-event simulation was employed to evaluate the performance of different operational strategies.

Other studies have focused on optimization-based appointment scheduling. For example, [44] developed a mixed-integer programming model to balance patient flow time and clinic makespan under nurse-capacity constraints by assigning patients to discrete appointment slots. Building upon optimization-based scheduling, [40] incorporated machine learning techniques to predict treatment durations and no-show probabilities from historical patient data. These predictions were then integrated into the scheduling model, allowing decision makers to leverage patient-specific information when constructing appointment schedules. In contrast, [23] formulated a stochastic optimization model with continuous appointment times to minimize patient waiting times and operational costs under uncertain treatment durations.

These studies demonstrate that optimization is a powerful tool for improving patient flow and resource utilization in healthcare systems. However, patient arrivals, treatment durations, and operational conditions are inherently uncertain. As a result, schedules generated before the start of the day may become suboptimal as new information is revealed. This motivates the need for adaptive decision-making approaches that can update operational decisions throughout the day in response to uncertainty.

The approaches differ in how appointment times are represented and updated. [76] employs a rolling optimization framework in which patients are assigned to discrete time slots. The optimization problem is repeatedly re-solved whenever new information becomes available, using an updated list of patients at each decision epoch. Rolling optimization provides a systematic way to incorporate current system information into decision making. By updating the patient list and re-solving the optimization problem as the system evolves, it can continuously adapt deci-

sions using available information, such as expected processing times and current resource availability.

Motivated by these ideas, we incorporate rolling optimization as a technician policy within the dialysis clinic. In addition to the FIFO, SPT, and LPT policies discussed earlier, we propose rolling optimization as a decision-making framework that provides technicians with dynamic instructions based on the current state of the clinic. This allows decisions to account not only for the patients currently waiting but also for the anticipated impact of those decisions on future patient flow and resource utilization.

[4] provides a comprehensive review of appointment scheduling research in healthcare. The review highlights a substantial increase in the number of studies published since 2001, reflecting the growing importance of appointment scheduling and the need to address variability and uncertainty in healthcare operations. Alongside classical scheduling approaches from the job scheduling literature and optimization-based healthcare scheduling methods, researchers also require tools that can accurately represent the dynamic behavior of healthcare systems.

In addition to optimization, simulation has become a widely used analytical tool in this area. According to [4] and [74], discrete-event simulation is the most commonly adopted simulation approach for healthcare appointment scheduling problems. This is because healthcare systems naturally evolve through a sequence of events, such as patient arrivals, queue formation, service initiation, and service completion. By explicitly tracking these events, discrete-event simulation enables researchers to evaluate the effects of different scheduling methods and operational policies on patient waiting times, resource utilization, and overall patient flow.

Furthermore, discrete-event simulation can capture the decisions made by healthcare staff throughout the operation of the system. These decision records provide valuable information for subsequent analysis and can potentially be used to develop improved scheduling policies or data-driven decision-support tools. As a result, simulation serves not only as an evaluation framework but also as a platform for learning and improving healthcare operations.

Finally, with the rapid advancement of artificial intelligence and machine learning, an increasing number of studies have explored their applications in healthcare systems with the goal of improving service quality and resource management. For example, [16] and [79] incorporate classification models based on patient records to classify patient types and predict patient behavior before making scheduling decisions. By leveraging historical patient information, these approaches provide deeper insights into patient characteristics and arrival patterns, leading to more informed scheduling decisions and improved operational performance.

Motivated by these developments, we investigate the use of machine learning for policy selection within a dialysis clinic. Rather than predicting patient arrivals, patient treatment duration or no-show behavior as in [40], [79] and [16], we use historical patient information and the performance of existing policies to train a classification model that recommends the most suitable policy for a given set of patients. A similar simulation-based policy learning framework was proposed in [83] for dispatching volunteers to Out-of-Hospital Cardiac Arrests (OHCA). Using incident-specific features, including response times, the authors generate labeled data through simulation by identifying the best dispatch policy for each scenario and subsequently train a machine learning model to predict the most suitable policy. This approach is closely related to our policy building framework. The re-

sulting policy, which we refer to as a *super policy*, learns from past operational data and combines the strengths of multiple existing policies. The objective is to create a policy that can adapt to different patient characteristics and scheduling environments, thereby improving patient flow, reducing waiting times, and enhancing overall clinic performance.

This chapter is organized as follows. Section 6.2 describes the dialysis clinic simulation model, including the key parameters, assumptions, events, and operational behaviors captured by the model. It also presents the performance measures and statistics used throughout the study. Section 6.3 presents several technician policies for selecting patients to be connected to and disconnected from dialysis machines after appointment schedules have been determined. Section 6.4 introduces the scheduling methods implemented in the simulation model. These methods are primarily adapted from [78] and are extended with variants inspired by classical job scheduling rules discussed in [73], [17], and [35]. Section 6.5 reports computational results and discusses the performance of different scheduling methods and policies under a variety of patient scenarios. Section 6.6 explores the use of classification techniques to improve policy selection when there is limited flexibility in choosing the scheduling method, leading to the development of a data-driven super policy. Finally, Section 6.7 summarizes the performance of the proposed policies across different scheduling methods, reviews the framework for constructing the Super Policy using a classification model and its resulting performance, and discusses potential directions for future research, including extensions to larger dialysis clinic systems with additional patients, chairs, and technicians.

6.2 Simulation Framework for Dialysis Clinic Operations

This section presents the simulation framework for the dialysis clinic model. The model simulates the operation of a dialysis clinic over a single day. Patients arrive at the clinic, are assigned to dialysis machines by technicians, undergo treatment, and are subsequently disconnected from the machines upon completion of their treatments. Technicians perform two primary tasks: *put-on*, which initiates a patient's treatment by connecting the patient to a dialysis machine, and *take-off*, which concludes the treatment by disconnecting the patient from the machine and preparing the chair for the next patient. The simulation terminates when all patients have completed their treatments and have been taken off their machines. The clinic consists of multiple chairs (or treatment stations), each of which can serve at most one patient at a time.

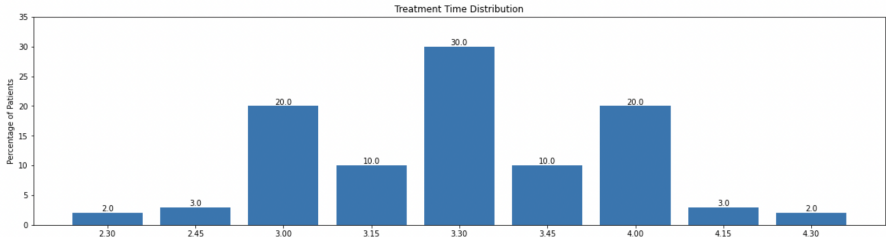


Figure 6.1: Dialysis Treatment Time Distribution

Let N denote the number of patients scheduled in a day, and let C denote the number of chairs. Define the set of chairs by $\mathcal{C} = 1, 2, \ldots, C$. In practice, the put-on task typically requires more time than the take-off task, while dialysis treatment times are substantially longer than both. In [78], it is assumed that the put-on processing time ($p^{\downarrow}$) exceeds the take-off processing time ($p^{\uparrow}$), and that the treatment time t is sufficiently large relative to these tasks. In particular, the following condition is assumed:

$$t \geq (C-1)(p^{\downarrow} + p^{\uparrow}). \tag{6.1}$$

Similarly, [43] reports that put-on tasks generally require longer processing times than take-off tasks and that treatment durations are significantly longer than either task. [78] analyzes patient flow under the additional assumption that the treatment time t, put-on time $p^{\downarrow}$, and take-off time $p^{\uparrow}$ are constant across all patients. Under inequality (6.1), patients can be partitioned into blocks, each containing C patients. In particular, this inequality implies the following properties:

1. After the last patient in a block has been connected to a dialysis machine, all patients in that block are still undergoing treatment. Consequently, the treatment periods of all patients within the same block overlap.
2. While the first patient in a block is receiving treatment, no patient in that block will have completed treatment and be ready for disconnection.
3. The first patient in the next block can begin treatment only after the first $C-1$ patients in the previous block have been disconnected.

These properties motivate the *push* and *interlaced* scheduling methods proposed by [78], which are discussed in Section 6.4. These scheduling methods organize patients into blocks and simplify the analysis under the assumption of constant treatment, put-on, and take-off times. In contrast, our simulation model allows treatment times (t), put-on times ($p^{\downarrow}$), and take-off times ($p^{\uparrow}$) to vary across patients, providing a more realistic representation of clinic operations.

Although the inequality above may not hold for every patient in our simulation, since treatment times can be as short as 2.5 or 2.75 hours, it is nevertheless repre-

sentative of typical operating conditions in dialysis clinics where treatment times dominate technician processing times. For each of the N patients, the following attributes are generated according to the specified distributions and assumptions:

1. **Treatment time.** Treatment times are generated based on data reported in [70] and [78]. The resulting distribution is shown in Figure 6.1. The mean treatment time is 3.5 hours, with treatment durations taking values in 15-minute increments ranging from 2.5 to 4.5 hours.
2. **Put-on time.** Put-on times are modeled using data from [43] and [78]. We assume a Gamma distribution with mean 24.25 minutes and standard deviation 7.13 minutes.
3. **Take-off time.** Take-off times are also modeled using data from [43] and [78]. We assume a Gamma distribution with mean 17.5 minutes and standard deviation 9.22 minutes.
4. **Patient punctuality.** Patient punctuality is modeled based on observations reported in the healthcare scheduling literature. For example, [88] found that patient arrival times are approximately normally distributed with a mean of 20.5 minutes early and a standard deviation of 17.9 minutes. Similarly, [52] reported arrival times that are approximately normally distributed with a mean of 9.39 minutes early and a standard deviation of 13.7 minutes, with approximately 24.65% of patients arriving late. Motivated by these findings, we assume that each patient arrives late independently with probability 0.2.

   Conditional on being late, the lateness amount is modeled as a non-negative random variable following a truncated normal distribution. The use of normal distributions to represent patient unpunctuality is common in the appointment scheduling literature; see, for example, [92], [66], [18], and [15].

In particular, [15] assumes truncated normally distributed patient unpunctuality, while [92] reports an average patient lateness of approximately 9.67 minutes (580 seconds) based on outpatient data.

Furthermore, the arrival-time distributions reported by [88] and [52] suggest that the vast majority of late arrivals are concentrated within approximately $0 - 20$ minutes after the scheduled appointment time. Motivated by these observations, we model the lateness amount using a truncated normal distribution with mean 10 minutes and standard deviation 3 minutes. This choice represents a moderate lateness scenario that is consistent with values reported in the literature while ensuring that nearly all simulated late arrivals fall within the $0 - 20$ minute range.

Each patient is assigned a scheduled arrival time according to the scheduling heuristics described in Section 6.4. Most patients arrive on time, while approximately 20% experience a random delay generated according to the lateness model described above. After generating all patient attributes, including treatment times, put-on times, take-off times, and lateness, the clinic simulation begins by processing patient arrivals.

At any point in time, the simulation maintains two separate lists: a put-on queue $\mathcal{P}$ and a take-off queue $\mathcal{T}$. These queues contain patients waiting to be put on and taken off dialysis machines, respectively, and are ordered according to the times at which the corresponding tasks become available. When a patient arrives at the clinic, the patient is added to the put-on queue $\mathcal{P}$. Once the patient has been put on a dialysis machine, the patient is subsequently added to the take-off queue $\mathcal{T}$, where the patient remains until treatment is completed and a technician

is available to perform the take-off task.

Initially, the first arriving patient is immediately assigned to the first available chair and undergoes the put-on procedure. If additional patients arrive before this put-on task is completed, they are added to the put-on queue and must wait for service. As the day progresses, both queues may accumulate patients waiting for put-on and take-off tasks. Consequently, when a technician becomes available, a decision must be made regarding which task should be performed next.

To model this decision-making process, let P denote a policy function that takes as input the technician's current time, the put-on queue $\mathcal{P}$, and the take-off queue $\mathcal{T}$, and returns the next task to be performed. The policies used to prioritize put-on and take-off tasks are described in Section 6.3.

In this simulation model, we define the following events, which are events that all patients have to go through.

1. **Arrival**. A patient arrives to the clinic and will be added to the system. In particular he will be added to the put-on list $\mathcal{P}$. If there is at least one chair available, and a technician is available, then the technician will make a decision based on the policy of the clinic.
2. **End Put-on**. The technician just finished putting on a patient. This patient will be added to the take-off list $\mathcal{T}$. After that the technician will make a decision based on a given policy on the next task to process from both $\mathcal{P}$ and $\mathcal{T}$.
3. **End Treatment**. A patient just finished treatment and needs to be taken

off the machine. The technician will now make a decision to complete a task from the lists if he is available.

4. **End Take-off**. A patient is taken-off and the treatment process is completed. This patient will leave the system. The technician will also decide the next task to process.

All **arrival** events are scheduled at the beginning of the simulation according to the scheduling methods described in Section 6.4. For patients who arrive late, the scheduled arrival time is adjusted by adding a randomly generated lateness. Once a patient has arrived and a technician becomes available, the patient is eligible for a put-on task.

When a technician performs the put-on task, the corresponding **end put-on** event is scheduled using the patient's generated put-on time. At the same time, since the patient is connected to a dialysis machine and treatment begins immediately, the **end treatment** event is also scheduled based on the patient's treatment duration. After treatment is completed, the patient remains in the system until a technician becomes available and selects the corresponding take-off task according to the policy P. Once the take-off task begins, the **end take-off** event is scheduled using the patient's generated take-off time. Upon completion of this event, the patient leaves the system.

All event times are represented as nonnegative real numbers, with the simulation clock initialized at time 0. For each patient, the simulation records the arrival time, put-on start time, treatment start time, take-off start time, and completion time. These timestamps are subsequently used to compute performance measures such

as patient waiting times and clinic makespan. In addition, the chair assignment for each patient is recorded when the patient is put on a dialysis machine.

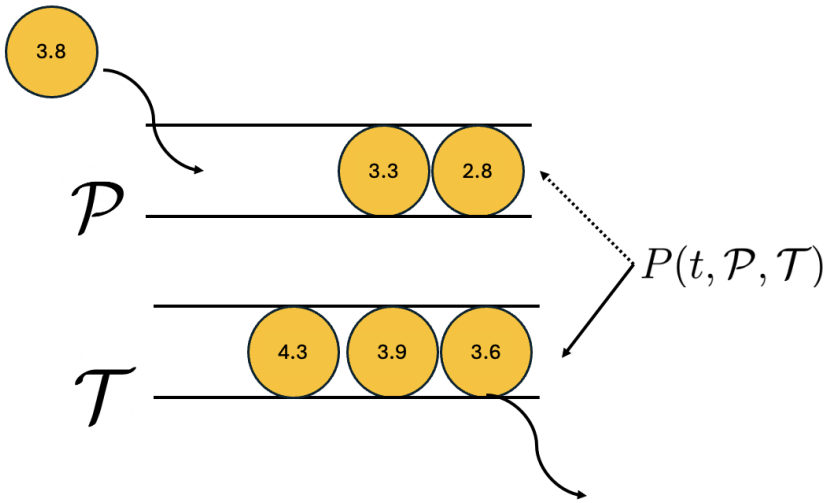


Figure 6.2: An example of simulation model

Figure 6.2 illustrates the simulation model. In this example, the system maintains two queues: the put-on queue $\mathcal{P}$ and the take-off queue $\mathcal{T}$. A technician makes decisions according to the policy function P, which takes as input the technician's current available time t together with the contents of both queues.

The put-on queue $\mathcal{P}$ contains patients waiting to be connected to a dialysis machine, where the numbers shown in the figure represent their arrival times. The take-off queue $\mathcal{T}$ contains patients whose treatments have been completed and who are waiting to be disconnected from their machines; the numbers shown in this queue correspond to treatment completion times. The figure also shows a newly arriving patient who will be added to the put-on queue $\mathcal{P}$. Based on the information available in both queues and the policy function P, the technician decides to select a patient from the take-off queue $\mathcal{T}$. After the take-off task is completed, the patient leaves the system. According to [78], most dialysis clinics

have a number of treatment stations that is a multiple of four. Therefore, in this chapter, we simulate the operation of a dialysis clinic with 4 chairs ($C = 4$), 1 technician, and assume that 12 patients ($N = 12$) are scheduled each day.

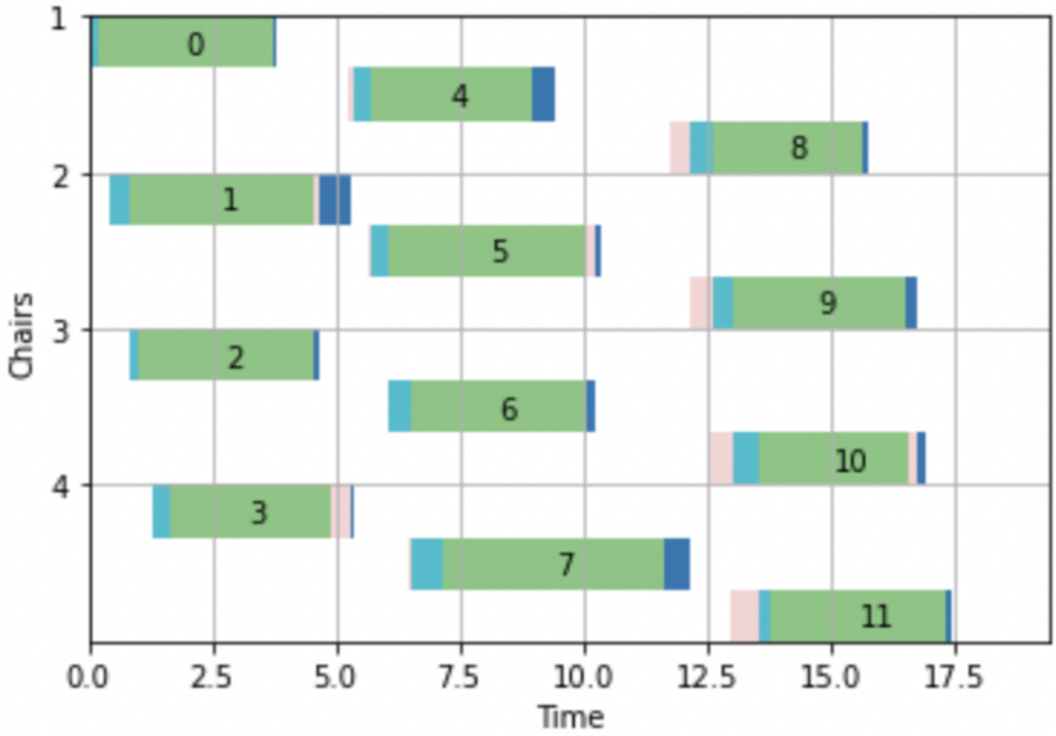


Figure 6.3: An example of simulation results of all patients

Figure 6.3 illustrates the results of a single-day dialysis clinic simulation. All 12 patients are assigned to 4 chairs. The green bars represent patient treatment durations, the light blue bars represent put-on tasks, and the dark blue bars represent take-off tasks. The durations of treatment, put-on, and take-off tasks vary across patients according to the probability distributions described earlier. The red bars represent periods during which patients are waiting for a technician to either initiate treatment (put-on) or disconnect them from treatment (take-off).

6.3 Operating Policies

The policy function P accepts the technician's current available time, the put-on queue $\mathcal{P}$, and the take-off queue $\mathcal{T}$, and returns the next task that the technician should perform. This task may be either a put-on or a take-off task selected from the corresponding queue or a decision to idle the technician if no task can be performed at that time. In practice, the technician's available time is equivalent to the current simulation time. For **arrival** and **end treatment** events, a decision is made using policy P only if a technician is available; otherwise, no task can be initiated. For **end put-on** and **end take-off** events, the technician becomes available immediately after completing the task, so the technician's available time coincides with the current simulation time. The purpose of the policy is to provide clear operational guidelines for technicians, promote smooth patient flow, and, where possible, improve performance measures such as patient waiting times and clinic makespan. We consider four policies: First-In-First-Out (FIFO), Shortest Processing Time (SPT), Longest Processing Time (LPT), and Rolling Optimization.

The first policy is First-In-First-Out (FIFO), in which the technician serves requests in the order in which they enter the system, including any delays caused by patient lateness. Under this policy, the technician simply selects the task that has been waiting the longest, regardless of whether it is a put-on or take-off task. For example, a patient waiting for take-off will be served before a patient who arrived after the first patient completed treatment and is waiting for put-on. FIFO is simple to implement and easy for technicians to follow. In addition, it is often perceived as fair because tasks are processed according to their arrival order. However, FIFO does not account for treatment durations and may perform poorly

when long treatments occur early in the day. Such patients can delay many subsequent tasks, causing queues to accumulate and increasing both waiting times and makespan. Consequently, FIFO may be outperformed by policies that explicitly account for treatment durations and re-prioritize patients accordingly.

The second and third policies are based on treatment duration. The Shortest Processing Time (SPT) policy prioritizes patients with the shortest treatment times, whereas the Longest Processing Time (LPT) policy prioritizes patients with the longest treatment times. Since technicians do not know the exact durations of individual put-on and take-off tasks, the patient's scheduled treatment time is used as a proxy for processing time. Under the SPT policy, a patient who arrives later may be selected before an earlier arrival if the patient's treatment time is shorter. Conversely, under the LPT policy, a later-arriving patient with a longer treatment time may be prioritized. The survey by [73] reviews a variety of scheduling rules, including SPT and LPT, and notes that SPT is commonly associated with reducing average waiting times, whereas LPT is often used to improve completion-related objectives. Furthermore, [86] provides theoretical support for SPT-type rules by showing that, although SPT may not be optimal at every decision point, its deviation from the optimal policy remains bounded and the weighted SPT rule is optimal for most scheduling decisions. For LPT, [35] shows that prioritizing long jobs can be an effective strategy for reducing makespan in parallel-machine scheduling problems. The performance of both policies will be evaluated and compared in Section 6.5.

Another method is rolling optimization, in which, at any time a technician becomes available, a scheduling problem is solved to minimize a weighted objective function consisting of total patient waiting time and makespan. This optimization

is performed subject to constraints on job starting times, task–chair assignments, and task ordering. The resulting problem is formulated in (6.2), where the decision variables x represent job starting times. In addition, we introduce auxiliary decision variables to capture assignment and precedence structure. Let $A \in \{0,1\}^{T\times C}$ denote the task–chair assignment matrix, defined as

$$A_{ic} = \begin{cases} 1 \;\; \text{if task } i \text{ is positioned at chair } c, \\ 0 \text{ otherwise.} \end{cases}$$

To encode ordering constraints, we define two binary precedence matrices. The first captures precedence relations between tasks assigned to the same chair, while the second represents global precedence relations across all chairs. Specifically, let $P \in \{0,1\}^{T\times T}$ denote the chair-specific precedence matrix and $P' \in \{0,1\}^{T\times T}$ the global precedence matrix, where

$$P_{ij} = \begin{cases} 1 \;\; \text{if task } i \text{ strictly comes before task } j \text{ in the same chair} \\ 0 \text{ otherwise} \end{cases}$$

and,

$$P'_{ij} = \begin{cases} 1 \;\; \text{if task } i \text{ strictly comes before task } j \\ 0 \text{ otherwise} \end{cases}$$

Note that, since there is only one technician serving patients, tasks cannot overlap in time. Therefore, if task i precedes task j, the start time of task j must be at least the processing time of task i later than the start time of task i. In other words, the precedence relationship between tasks must be strict, ensuring that the technician completes task i before beginning task j. We denote by t the vector of

task processing times, where t_i corresponds to the processing time of task i. For put-on task i, t_i is the expected put-on task using patient's average put-on time. For take-off task i, t_i is the expected take-off task using patient's average take-off time. We define these processing times using the respective average durations of put-on and take-off operations. The release time vector r depends on the time at which decisions are made within the rolling optimization framework. Because we maintain a put-on task list $\mathcal{P}$ and a take-off task list $\mathcal{T}$, the release times for put-on tasks correspond to patient arrival times, which are known upon arrival. In contrast, the release time of a take-off task corresponds to the completion time of the associated treatment, and is therefore necessarily greater than the starting time x of the corresponding put-on task.

For the ordering constraints, we use the big-M method in the formulation. Constraint (5e) enforces $x_j \geq x_i + t_i$ only when $P_{ij} = 1$ and $A_{ic} = A_{jc} = 1$. This implies that task j is processed after task i if task i precedes task j and both tasks are assigned to the same chair ($P_{ij} = 1$), namely chair c, ($A_{ic} = A_{jc} = 1$). Conversely, if $P_{ij} = 0$, this precedence constraint is not enforced. However, the matrix P only enforces ordering between tasks assigned to the same chair and does not specify the ordering of tasks assigned to different chairs. Therefore, we introduce another binary matrix P' to indicate whether task i is processed before task j. Constraint (5f) is analogous to Constraint (5e), but without the matrix A for task-chair assignments.

Finally, we enforce that each task starting time x_i must satisfy the release-time constraint $x_i \geq r_i$, i.e., a task can only start after it is released. However, since there are two types of tasks—put-on and take-off—the release times differ depending on the task type. We therefore define r_i as a set of parameters, where some values

are already known. Specifically, the release times for put-on tasks correspond to patient arrival times, which are known for all arrived patients. The release times for take-off tasks correspond to treatment completion times, which are also known since these tasks are associated with patients who have already been put on treatment and processed. For tasks not yet included in the lists $\mathcal{P}$ and $\mathcal{T}$, we do not have information about their true put-on and take-off times. In such cases, we use average put-on and take-off durations, denoted by ($\bar{p}^{\downarrow}$) and ($\bar{p}^{\uparrow}$), respectively, computed as the means of the distributions generated in Section (6.2). These values are treated as historical averages of patient service times. By the definition of processing time t, $t_i = \bar{p}^{\downarrow}$ for put-on task and $t_i = \bar{p}^{\uparrow}$ for take-off task. Then, we have the following constraints on the release times r_i:

1. For put-on tasks i that are in $\mathcal{P}$,

$$r_i = \text{arrival time.}$$

   Since these tasks are in $\mathcal{P}$, all arrival times are known.
2. For take-off tasks i that are in $\mathcal{T}$,

$$r_i = \text{treatment end time.}$$

   For tasks in $\mathcal{T}$, the corresponding patient has already been put on treatment, and their treatment information is known. Hence, the treatment end time is also known.
3. For take-off tasks i that are not in $\mathcal{T}$ but correspond to put-on tasks in $\mathcal{P}$, we use information from the corresponding put-on task. Let i' denote the corresponding put-on task, whose starting time variable is $x_{i'}$. Let j be the

patient index. Then,

$$r_i = x_{i'} + \bar{p}^{\downarrow} + \text{treatment time of patient } j.$$

The release time corresponds to the expected treatment completion time. We use $\bar{p}^{\downarrow}$ since patient j has been assigned but has not yet been put on treatment and is still in $\mathcal{P}$.

4. For non-arrived patients and put-on tasks i, we assume the release time equals the scheduled arrival time. Let j be the patient index of this task, then

$$r_i = \text{scheduled time of patient } j.$$

We assume that the patient arrives on time.

5. For non-arrived patients and take-off tasks i, we apply the same logic as in the previous case for take-off tasks corresponding to put-on tasks in $\mathcal{P}$. Let i' denote the corresponding put-on task, whose starting time variable is $x_{i'}$. Let j be the patient index. Then,

$$r_i = x_{i'} + \bar{p}^{\downarrow} + \text{treatment time of patient } j.$$

The optimization problem is formulated below:

$$\min_{x,A,P,P',r} \lambda \sum (x_i - r_i) + \zeta \tag{5a}$$

$$s.t.\ \zeta \geq x_i + t_i \ \forall i \tag{5b}$$

$$x_i \geq r_i \ \forall i \tag{5c}$$

$$\sum_j A_{ij} = 1 \ \forall i, A_{ic} = A_{i'c} \ \forall \text{ put-on and take-off pair } i, i' \tag{5d}$$

$$\begin{cases} x_j \geq x_i + t_i - M(1 - P_{ij}) - M(2 - A_{ic} - A_{jc}) \\ x_i \geq x_j + t_j - M(P_{ij}) - M(2 - A_{ic} - A_{jc}) \end{cases} \quad \forall i < j \ \forall c \in \mathcal{C} \tag{5e}$$

$$\begin{cases} x_j \geq x_i + t_i - M(1 - P'_{ij}) \\ x_i \geq x_j + t_j - M(P'_{ij}) \end{cases} \quad \forall i < j \tag{5f}$$

$$P' \geq P \tag{5g}$$

$$P'_{ij} + P'_{ji} = 1 \ \forall\, i \neq j \tag{5h}$$

$$P'_{ii'} = 1, P_{ii'} = 1 \ \forall \ \text{ put-on and take-off pair } i, i' \tag{5i}$$

$$x \geq 0, A_{ij}, P_{ij}, P'_{ij} \in \{0, 1\} \tag{5j}$$

(6.2)

Inequality (5b) enforces the makespan ζ to be at least the completion time of every task and, consequently, equal to the completion time of the last processed task. Inequality (5c) ensures that the starting time x_i is no earlier than the corresponding release time r_i. Constraint (5d) requires each task to be assigned to exactly one chair where corresponding put-on and take-off must be on a same chair. Constraints (5e) and (5f) enforce task ordering. Constraint (5e) ensures that if task i is processed before task j on the same chair, then the starting time of task j must be

at least t_i units later than the starting time of task i. In contrast, Constraint (5f) enforces precedence relationships regardless of whether the two tasks are assigned to the same chair. Constraint (5g) ensures consistency between chair-specific and global task orderings: if task i precedes task j on the same chair, then task i must also precede task j in the overall schedule. Constraint (5h) ensures that every pair of tasks is assigned an order and that no two tasks overlap, reflecting the assumption that only one technician is available. Constraint (5i) ensures that the put-on task of each patient is completed before the corresponding take-off task. Constraint (5j) guarantees non-negativity and binary values.

Since our experiments involve 12 patients and 4 chairs, the patients can be partitioned into at most 3 groups of 4 patients, where each group occupies all available chairs. The maximum treatment time of a patient is 4.5 hours. Including put-on and take-off operations, we conservatively estimate that a patient may require up to 6 hours from the start of the put-on task to the completion of the take-off task. To obtain an upper bound on the clinic makespan, consider the worst-case scenario in which the three groups of patients are processed sequentially. In this case, the first patient of each group contributes at most 6 hours to the overall completion time, resulting in a total of 18 hours. To further ensure that the bound remains valid under all possible task orderings and delays, we add an additional 6-hour buffer corresponding to one more patient process. This yields an upper bound of 24 hours on the makespan. Therefore, throughout the formulation, we set $M = 25$, which is sufficiently large to exceed the makespan ζ for all instances involving 12 patients.

In each simulated day, there are 12 patients, resulting in a total of 24 tasks, consisting of one put-on task and one take-off task for each patient. Consequently, the

rolling optimization problem may need to be solved up to 24 times per day. However, solving an optimization problem whenever a technician becomes available is considerably more computationally expensive than implementing FIFO, SPT, or LPT policies, which simply select tasks directly from the corresponding queues.

To mitigate this computational burden, we propose two variants of the rolling optimization policy: quick rolling optimization and greedy rolling optimization. Under the standard rolling optimization policy, the optimization problem is solved whenever both the put-on and take-off lists contain at least one task, with the objective of minimizing a weighted combination of total waiting time and makespan.

Quick rolling optimization reduces computational effort by reverting to FIFO whenever only one type of task can be processed. For example, if all chairs are currently occupied, no put-on task can be performed. In this case, the policy simply selects the first task from the take-off list according to FIFO. Conversely, if no patient is ready for take-off, the technician selects the first task from the put-on list according to FIFO. The optimization problem is solved only when both put-on and take-off tasks are simultaneously available for processing.

Greedy rolling optimization, on the other hand, solves Optimization Problem (6.2) at every decision epoch whenever both the put-on and take-off lists are nonempty. Unlike quick rolling optimization, it does not revert to a simpler dispatching rule and therefore incurs a higher computational cost in exchange for potentially improved scheduling decisions.

In summary, we consider the following five policies:

1. First-in-First-out (FIFO)
2. Shortest Processing Time (SPT)
3. Longest Processing Time (LPT)
4. Quick Rolling Optimization (QRoll)
5. Greedy Rolling Optimization (GRoll)

Figure 6.3 shows an example of simulation results obtained using the FIFO policy. The following figures present example simulation results for the remaining technician policies.

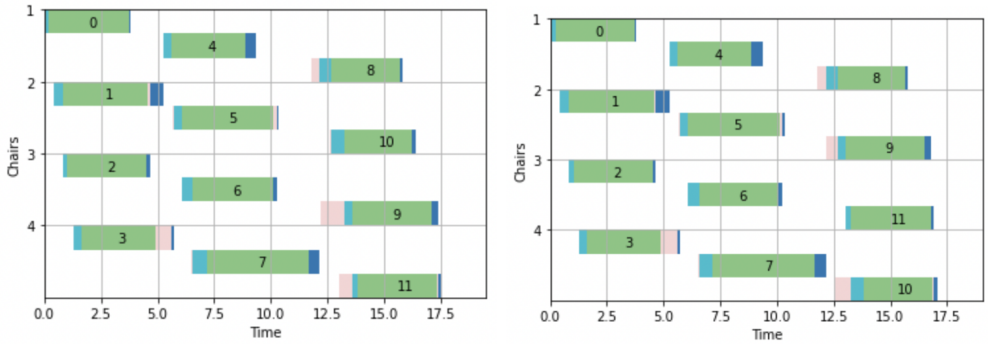


Figure 6.4: Simulation results with SPT (left) and LPT (right)

Each policy produces a different sequence of put-on and take-off tasks depending on how tasks are selected from the lists $\mathcal{P}$ and $\mathcal{T}$.

The policies described above determine how technicians prioritize tasks once put-on and take-off requests become available. However, the arrival pattern of these requests is largely governed by the appointment schedule. As a result, technician dispatching policies and appointment scheduling policies interact to shape patient flow and system performance throughout the clinic. The next section introduces the scheduling heuristics used to determine patient appointment times. These

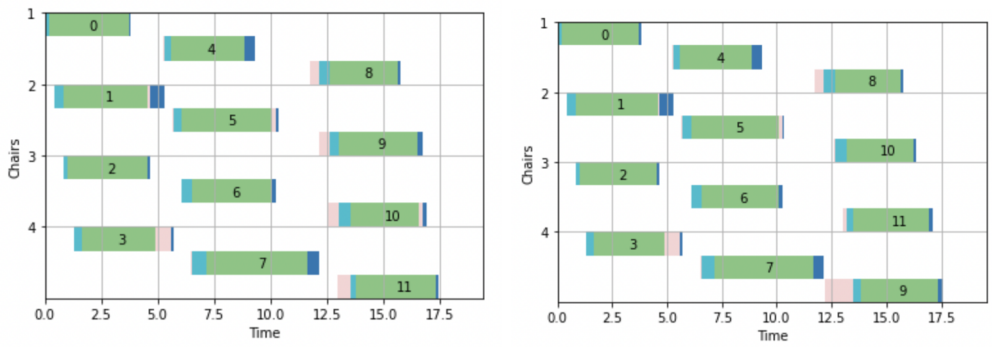


Figure 6.5: Simulation results with Quick Rolling Optimization (left) and Greedy Rolling Optimization (right)

scheduling heuristics are then combined with the technician policies described above, and the resulting policy combinations are evaluated through simulation.

6.4 Heuristics for Scheduling

Sections 6.2 and 6.3 present the simulation framework in which patient data, including treatment, put-on, and take-off times, are generated and technicians can follow the policies to process all tasks. However, before the simulation begins, patients must first be assigned appointment times. This section introduces two main scheduling methods—push and interlaced—and their variants. These scheduling methods, proposed and analyzed in [78], are applied prior to the Dialysis Clinic Simulation described in Section 6.2, where the technician policies from Section 6.3 are subsequently implemented.

Both scheduling methods require estimates of patients' put-on and take-off times. In practice, however, the actual durations of these tasks are unknown when appointments are scheduled. Following [78], we therefore use designated put-on and take-off times of 25 and 20 minutes, respectively, when constructing the schedule. These values provide approximate estimates that allow appointment times to be

assigned in advance, under the assumption that patients arrive on time and that the realized task durations do not differ substantially from their expected values.

The push scheduling method divides patients into blocks based on the number of patients scheduled for the day and the number of available chairs, C. In the case of 12 patients and 4 chairs, the patients are divided into 3 blocks. Under this method, all patients in one block must complete their take-off tasks before the first patient in the next block can begin the put-on process. Within each block, the C patients are assigned appointment times according to a specified ordering, which may be random or based on ascending or descending treatment durations. According to [78], this method is straightforward to implement in dialysis clinics; however, it may lead to operational inefficiencies because chairs can remain idle while waiting for an entire block to finish.

To address some of the inefficiencies associated with push scheduling, [78] also proposed the interlaced scheduling method. This approach introduces a strategically placed pause of fixed duration among the first C put-on tasks, allowing put-on and take-off tasks to overlap more effectively throughout the day. As a result, technicians tend to alternate between put-on and take-off tasks rather than processing them in large blocks. In addition, the interlaced schedule reduces the time gap between a patient's take-off task and the subsequent put-on task on the same chair, thereby improving chair utilization.

In addition to the push and interlaced scheduling methods, we consider different ways of ordering patients according to their treatment durations. Given a set of patients with known treatment times, either scheduling method may be applied after arranging patients randomly, in ascending order, or in descending order of

treatment duration. The latter two orderings are conceptually similar to scheduling short treatments first or long treatments first. Although these ideas resemble the SPT and LPT policies introduced in the previous section, the scheduling methods considered here differ in an important way. Specifically, the patient ordering is determined once before the simulation begins and is used to construct appointment times. In contrast, the SPT and LPT policies dynamically select patients from the put-on and take-off queues during the simulation. Consequently, patients who have not yet arrived at the clinic cannot be considered by the SPT and LPT policies. While [73] also uses the terms SPT and LPT, the scheduling approaches considered here are distinct from the technician-dispatching policies defined earlier.

In the experiments presented in Section 6.5, we refer to the random, ascending, and descending orderings of patients as asis, asc, and dsc, respectively. These terms describe the ordering of patients in the schedule. In contrast, the policy names SPT and LPT are used directly, as these policies select the next task to process according to treatment times in ascending and descending order, respectively. Therefore, combining the push and interlaced scheduling methods with random, ascending, and descending treatment-time orderings yields a total of six scheduling methods, described below.

1. Push scheduling with the given random order of patients' treatment times (push_asis)
2. Push scheduling with ascending patients' treatment times (push_asc)
3. Push scheduling with descending patients' treatment times (push_dsc)
4. Interlaced scheduling with the given random order of patients' treatment times (inter_asis)

5. Interlaced scheduling with ascending patients' treatment times (inter_asc)
6. Interlaced scheduling with descending patients' treatment times (inter_dsc)

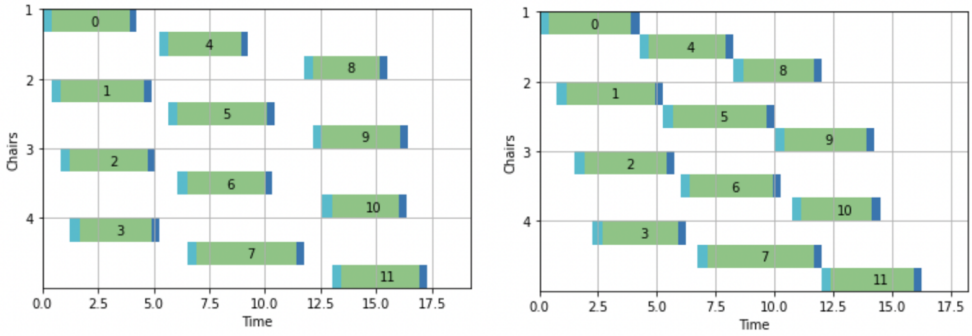


Figure 6.6: An example of push (left) and interlaced (right) schedule

Figure 6.6 illustrates the push and interlaced scheduling methods (not the simulation results) applied to the same set of patients, where the patients are ordered according to their original (random) treatment-time sequence, and the numbers indicate patients' labels. The push schedule groups patients into blocks, whereas the interlaced schedule introduces a planned pause among the early put-on tasks to better coordinate subsequent take-off and put-on activities. In the figure, the blue bars represent the assumed put-on and take-off durations of 25 and 20 minutes, respectively, used when constructing the schedules. In practice, however, the realized put-on and take-off times may differ from these assumed values once patients arrive at the clinic. Such variability can affect patient waiting times, chair utilization, and the overall makespan of the clinic schedule.

Using the same patient instance, Figure 6.7 illustrates examples of the push ascending and push descending scheduling policies. The bars have the same interpretation as those in Figure 6.6; however, the sequence in which patients are scheduled differs between the two policies, resulting in different patient orderings throughout the day.

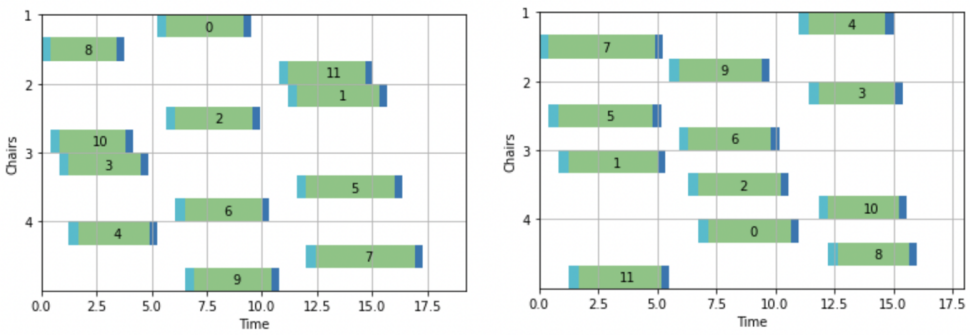


Figure 6.7: An example of push ascending scheduling (left) and push descending scheduling (right) schedule

6.5 Numerical Experiments

We generated 500 daily patient schedules, each consisting of 12 patients, and simulated 30 replications for each day. For a given day, all replications share the same set of patients and treatment times. However, the put-on times, take-off times, and patient lateness are generated independently across replications. In particular, both the selection of patients who arrive late and the amount of lateness are random. These random variables are generated according to the distributions described in Section 6.2.

We evaluate the policies presented in Section 6.3 together with the scheduling methods introduced in Section 6.4 across all 500 simulated days. For each combination of policy and scheduling method, we compute average performance measures, including patient waiting times and clinic makespan. To ensure a fair comparison, common random numbers are used across all policies and scheduling methods within each simulated day, so that differences in performance can be attributed solely to the scheduling and policy decisions rather than to random variation.

We compare all policies against FIFO, which serves as the benchmark policy. FIFO

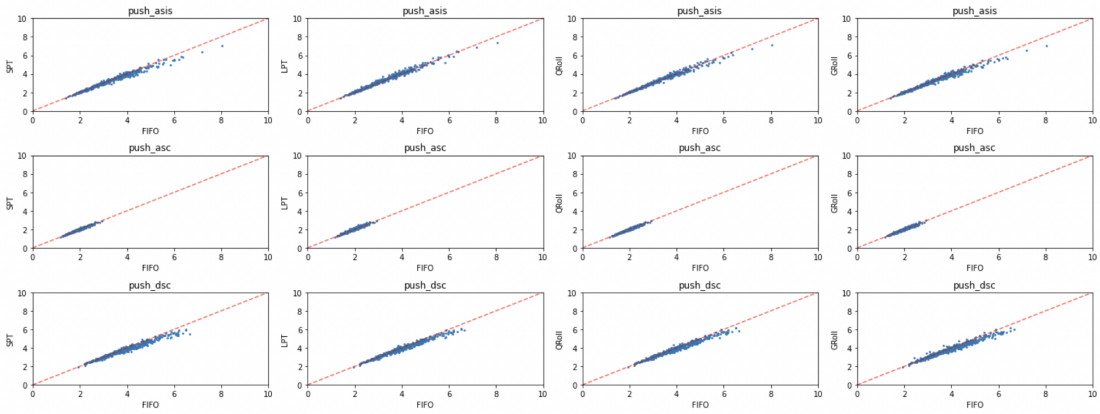


Figure 6.8: Waiting Time under Push scheduling

is straightforward to implement, as the technician simply processes whichever put-on or take-off task becomes available first, without using any information about treatment times or the designated put-on (25 minutes) and take-off (20 minutes) durations used by the Rolling Optimization policies. All values reported in the figures are measured in hours. Figure 6.8 presents the total waiting-time results under the push scheduling method. Comparing the three push variants, the asis variant generally produces larger waiting times, whereas the ascending variant achieves the lowest average waiting time and exhibits the smallest variability. The results also show that SPT, QRoll, and GRoll consistently outperform FIFO in terms of total waiting time.

Because SPT already prioritizes patients with shorter treatment times, its performance is less sensitive to the scheduling order. In contrast, FIFO struggles when long-treatment patients are scheduled early in the day. These long treatments contribute to congestion in both the put-on and take-off queues, causing subsequent patients to experience longer waiting times. For LPT, the asis and ascending variants do not provide a noticeable improvement over FIFO. However, when combined with the descending variant, LPT performs substantially better,

indicating that prioritizing longer treatments can be beneficial when the schedule is already arranged to place such patients earlier in the day.

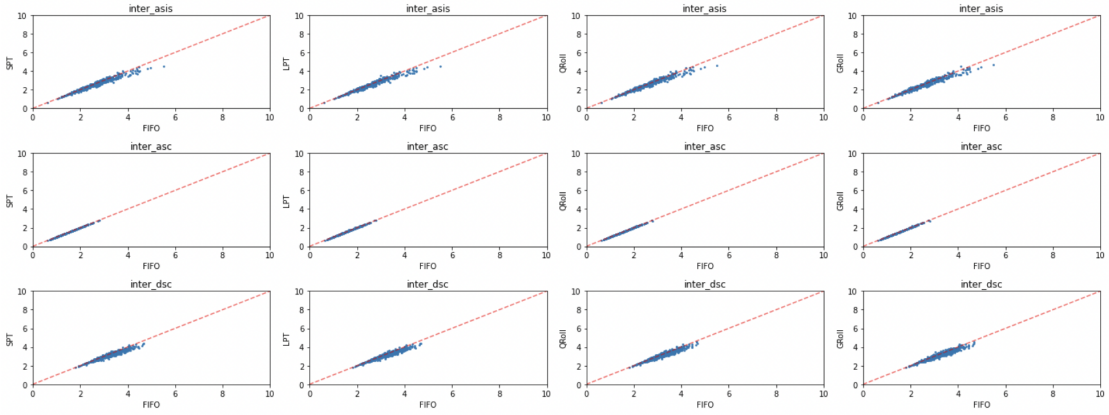


Figure 6.9: Waiting Time under Interlaced scheduling

Figure 6.9 presents the total waiting time results under the interlaced scheduling method. The overall pattern is similar to that observed with push scheduling: the ascending variant generally achieves the best performance, while the asis variant tends to produce the largest waiting times. However, under interlaced scheduling, the differences in waiting time between FIFO and the other policies become substantially smaller than those observed under push scheduling. This effect is particularly evident in the ascending variant, where all policies perform nearly identically to FIFO. Interlaced scheduling also improves the performance of LPT. In particular, LPT achieves lower total waiting times than under push scheduling and is able to outperform FIFO in the interlaced asis variant. These results suggest that the scheduling method itself has a significant influence on system performance. More broadly, interlaced scheduling consistently yields lower total waiting times than push scheduling across most policy combinations. This indicates that the structure of the appointment schedule may have a greater impact on waiting-time performance than the choice of technician policy, and that interlacing put-on and take-off activities can substantially improve patient flow throughout the clinic.

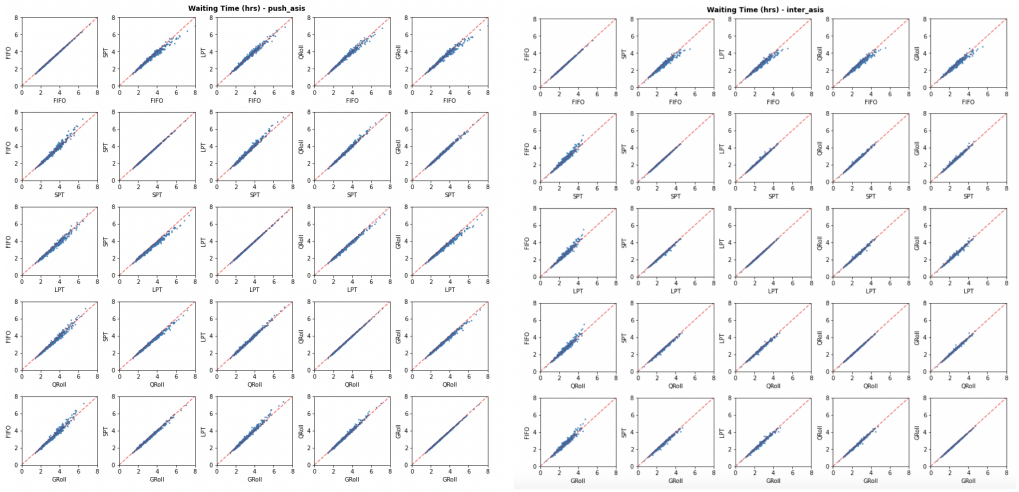


Figure 6.10: Waiting time on all policies of push asis (left) and interlaced asis (right)

Figure 6.10 presents pairwise comparisons of all policies under the asis scheduling variant. Among the three scheduling variants, the asis case exhibits the most pronounced differences between policies, making it easier to distinguish their relative performance. Overall, SPT achieves the lowest total waiting time, while Greedy Rolling performs nearly as well, with only a very small gap separating the two methods. These are followed by QRoll and LPT, respectively. Although GRoll attains performance comparable to SPT, it is considerably more complex to implement, as it requires solving an optimization problem whenever a technician must make a scheduling decision. In contrast, SPT is straightforward to implement: the technician simply examines the put-on list and selects the patient with the shortest treatment time. Given its simplicity and strong performance, SPT appears to offer an attractive balance between implementation effort and waiting-time reduction.

The remaining scheduling methods exhibit similar trends to those observed in Figure 6.10. While the magnitude of the differences varies across scheduling variants, the relative ranking of the policies remains largely unchanged in the pairwise com-

parisons.

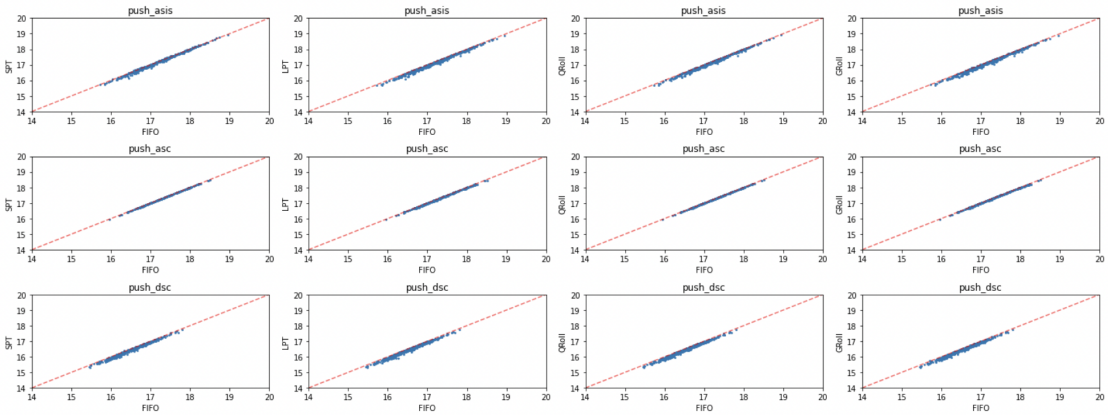


Figure 6.11: Makespan under Push scheduling

In the previous figures, substantial differences can be observed in total waiting time across policies and scheduling methods. In contrast, the makespan results exhibit much smaller performance differences between policies. Figures 6.11 and 6.12 show that all policies generally achieve slightly better makespan performance than FIFO, although the differences are relatively modest. In particular, under the ascending scheduling variant, the makespan values of all policies are nearly identical.

Unlike the total waiting-time results, the makespan results in Figure 6.11 indicate that the descending scheduling variant provides the best performance. Under this variant, the clinic can complete all treatments in as little as 15.5 hours, whereas the ascending and asis variants rarely achieve makespans below 16 hours. Furthermore, the asis variant exhibits the greatest variability, with some instances producing makespans exceeding 19 hours, a level not observed under either the ascending or descending variants.

These results suggest that the choice of scheduling method has a stronger influence on makespan than the choice of technician policy. In particular, scheduling longer-

treatment patients earlier in the day appears to improve overall clinic completion time, while the technician's task-selection policy mainly affects patient waiting times rather than the final completion time of the clinic.

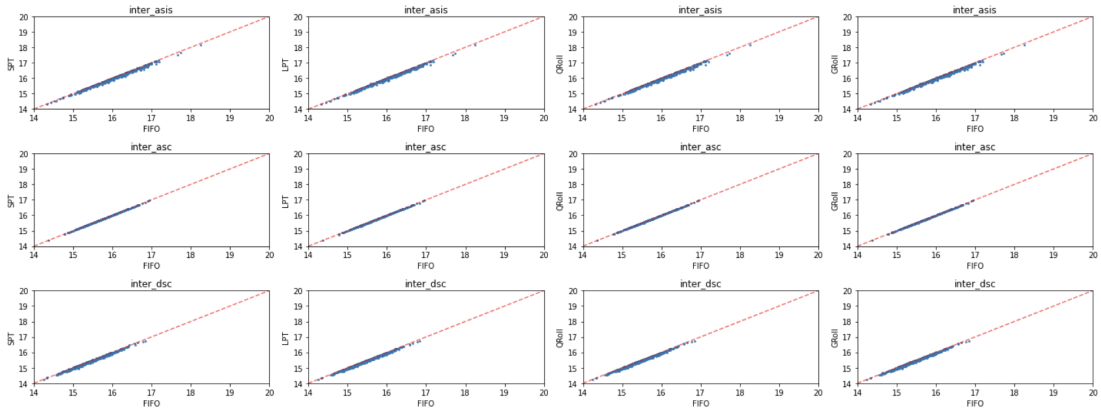


Figure 6.12: Makespan under Interlaced scheduling

Figure 6.12 shows that interlaced scheduling generally achieves better makespan performance than push scheduling. In addition, the performance differences between FIFO and the other policies are smaller under interlaced scheduling, indicating that the choice of technician policy has less influence on makespan when an interlaced schedule is used. Figure 6.13 presents pairwise makespan comparisons under the push and interlaced asis scheduling variants. The asis variant is shown because it exhibits the largest performance differences among the policies. For the ascending and descending variants, the pairwise differences are substantially smaller, and the performance of most policy pairs is nearly indistinguishable. Among all scenarios, FIFO performs the worst under the push descending variant, while the remaining policies achieve modest but consistent improvements in makespan.

Figure 6.13 also illustrates that LPT becomes more effective when the objective is to minimize makespan. Under the push asis scheduling variant, LPT achieves

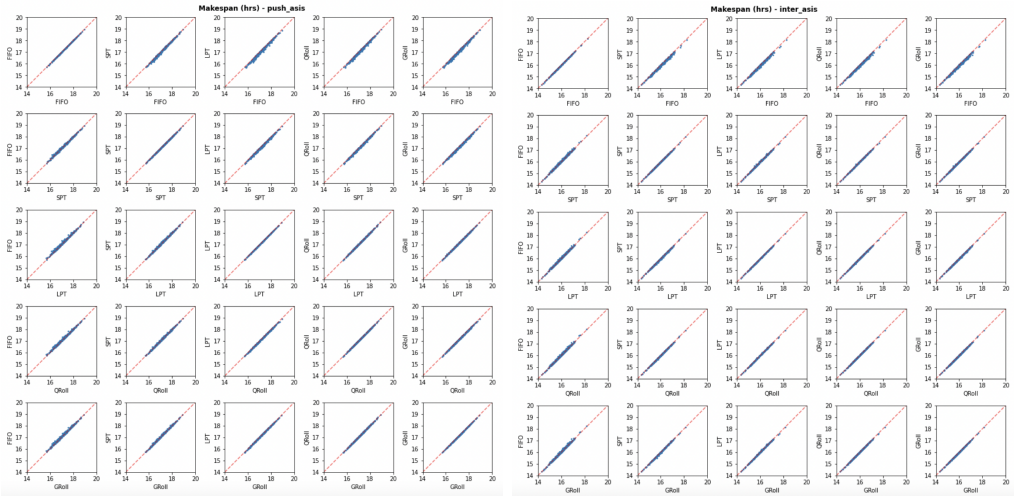


Figure 6.13: Makespan on all policies of push asis (left) and interlaced asis (right)

slightly better makespan performance than SPT, and a similar pattern can be observed under the interlaced asis variant. This result is consistent with the intuition that prioritizing longer-treatment patients earlier in the day can help reduce the overall clinic completion time.

Under the push asis scheduling variant, SPT exhibits makespan performance comparable to that of QRoll and GRoll. A similar trend is observed under the interlaced asis variant, where SPT, QRoll, and GRoll achieve nearly identical makespan values. Although small differences exist among these policies, they are minor compared with the performance gap between any of these policies and FIFO. Overall, the choice among SPT, QRoll, GRoll, and LPT has a relatively limited effect on makespan, whereas FIFO consistently produces the least favorable results.

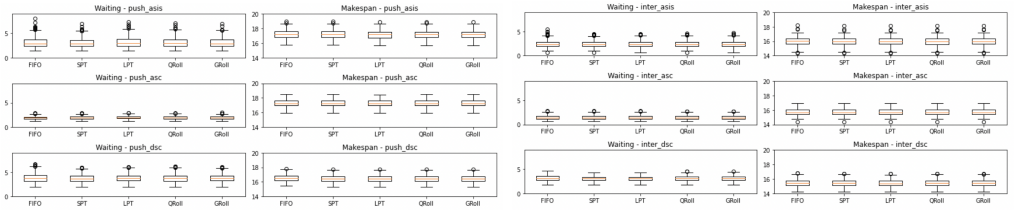


Figure 6.14: Box plots on all policies of push (left) and interlaced (right) scheduling

Figure 6.14 presents box plots of the total waiting time and makespan for all policy and scheduling-method combinations. Although these plots do not display the results of every simulated day individually, as in the previous scatter plots, they provide a clear summary of the distributions of total waiting time and makespan across the different scheduling methods. The box plots show that the interlaced scheduling method consistently achieves lower total waiting times and makespans than the push scheduling method. In addition, interlaced scheduling exhibits less variability in total waiting time, as reflected by its smaller interquartile ranges. Comparing the three ordering variants within each scheduling method, the ascending variant generally produces the best overall performance, followed by the descending variant, while the asis variant tends to perform the worst. Looking specifically at FIFO, several extreme waiting-time observations can be seen under the push asis scheduling variant. These instances occur when patients with long treatment times are scheduled early in the day, causing congestion in both the put-on and take-off queues. As a result, later patients experience substantially longer waits. In contrast, the other policies are better able to manage these situations by prioritizing patients based on treatment-time information or optimization-based decision rules, thereby reducing the impact of early long-treatment patients on the overall system performance.

Finally, we compare all policies using a combined objective function that assigns a weight of 5 to total waiting time and a weight of 1 to makespan. The overall trends are similar to those observed in the total waiting-time results. In particular, SPT, QRoll, and GRoll consistently outperform FIFO under this combined performance measure. Among these policies, SPT and GRoll achieve very similar results, with SPT exhibiting a slight advantage. However, from an implementation perspective, SPT is considerably simpler, as the technician only needs to select the patient with

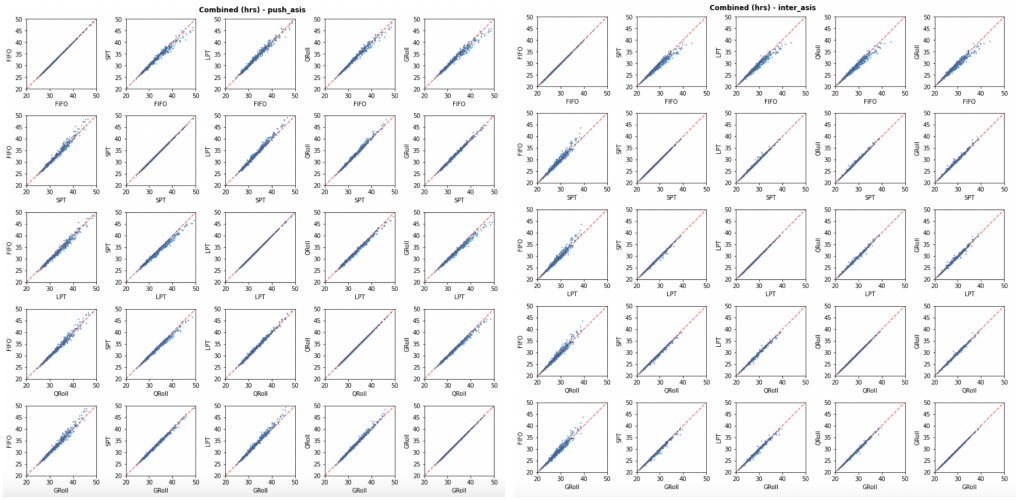


Figure 6.15: Combined objective function on all policies of push asis (left) and interlaced asis (right)

the shortest treatment time from the put-on list rather than repeatedly solving an optimization problem.

Overall, the experimental results suggest that, although FIFO is easy to implement, it is generally not the most effective policy. Several alternative policies achieve substantially better performance while remaining relatively simple to apply in practice. The results also highlight the importance of the scheduling method itself. A well-designed schedule can significantly reduce both waiting times and makespan, and under effective scheduling methods the performance differences among policies become much smaller.

Among all scheduling methods considered, the **interlaced ascending** variant provides the best overall performance. If a single technician policy must be selected for implementation, **SPT** appears to be the most attractive choice. It combines strong performance with a simple decision rule, making it easy for technicians to follow in practice. Although GRoll occasionally achieves comparable results, its

reliance on repeated optimization makes it more complex to implement. In contrast, SPT performs consistently well across a wide range of scheduling methods, making it a more robust and practical policy overall.

6.6 Building the Super Policy via Classification

In Section 6.5, we evaluated the performance of different technician policies under various scheduling methods. Based on those results, one possible approach is to select the policy and scheduling-method combination that yields the best overall performance. However, in practice, the scheduling method may not always be under our control due to operational constraints, institutional policies, or pre-existing appointment schedules. In such situations, it becomes important to develop a technician policy that can perform well regardless of the scheduling method and further improve the combined objective of patient waiting time and clinic makespan.

In this section, we consider the following setting. Suppose we are given a set of patients together with their treatment times and a predetermined scheduling method. In our experiments, each day consists of 12 patients, but the methodology can be extended to other problem sizes. The question we seek to answer is whether it is possible to construct a *super policy* that, on average, outperforms all policies introduced in Section 6.3.

To achieve this goal, we leverage information about patient treatment times, the scheduling method, and historical simulation results. Since several policies demonstrated strong performance in Section 6.5, we propose using a classification model to select among these policies adaptively. The raw input consists of the treatment times of the 12 patients scheduled for the day. We then enrich this information

by constructing additional features derived from the treatment times and the appointment schedule. The simplest extension is to include the scheduled arrival times of the patients, which provides 12 additional features whose values depend on the scheduling method being used. Finally, we construct a collection of aggregate features based on the treatment times and scheduled times, as described below.

1. **Clustering**: This feature measures the degree to which the scheduled times are clustered together. Given a schedule array s, the clustering measure is defined by

$$\frac{\sigma_s - \mu_s}{\sigma_s + \mu_s}$$

where σ_s and μ_s denote the standard deviation and mean of s, respectively. Since the scheduled times are nonnegative, we have $\mu_s \geq 0$ and $\sigma_s \geq 0$. Moreover, because the schedule array contains at least one positive entry, $\mu_s > 0$, which implies that $\sigma_s + \mu_s > 0$ and the denominator is always well defined. The clusterness measure takes values in the interval $[-1, 1)$. Values close to -1 indicate that the scheduled times are highly concentrated, as the standard deviation is small relative to the mean. In contrast, values close to 1 indicate a more dispersed schedule, where the variability of scheduled times is large relative to their average value. This feature therefore provides a normalized measure of how tightly or loosely patients are scheduled throughout the day

2. **Extremeness**: This feature measures the extent to which a single value dominates the remaining entries in an array. For a given array t, the ex-

tremeness measure is defined as

$$\frac{\max_i t_i}{\text{mean}(t)}.$$

Since $\max_i t_i \geq \text{mean}(t)$ for any nonnegative array t, this quantity is always at least 1. Values close to 1 indicate that the entries are relatively uniform, whereas larger values indicate the presence of one or more unusually large entries compared to the average level of the array. In our setting, a large extremeness value suggests that at least one patient has a treatment time or scheduled time that is substantially longer than that of a typical patient.

3. **Ranking Coefficient**: We use Kendall's Tau rank correlation coefficient to measure the degree of monotonic ordering in the treatment times. Kendall's Tau takes values in the interval $[-1, 1]$, where a value of 1 indicates that the treatment times are perfectly sorted in ascending order, a value of -1 indicates that they are perfectly sorted in descending order, and a value near 0 indicates little or no monotonic ordering. In our feature set, Kendall's Tau is computed using the treatment time array and its corresponding position indices, providing a measure of how closely the patients' treatment times follow an increasing or decreasing pattern within the schedule.

4. **Correlation**: This feature measures the linear relationship between patients' treatment times and their scheduled times. Specifically, we compute the Pearson correlation coefficient between the treatment-time array and the corresponding scheduled-time array. The correlation coefficient takes values in the interval $[-1, 1]$, where a value close to 1 indicates that patients with longer treatment times tend to be scheduled later in the day, a value close to -1 indicates that patients with longer treatment times tend to be scheduled

earlier, and a value near 0 indicates little or no linear relationship between treatment times and scheduled times. This feature captures how strongly the scheduling method is associated with the ordering of treatment durations.

5. **Weighted Position Index (WPI)**: This feature measures the average position of the mass of a given array. For an array t of length n, the weighted position index is defined as

$$\mathrm{WPI}(t) = \frac{\sum_{i=1}^{n} t_i i}{\sum_{i=1}^{n} t_i}.$$

   Intuitively, WPI represents the average position of the values in the array when weighted by their magnitudes. A large WPI indicates that larger values tend to appear in later positions of the array, whereas a small WPI indicates that larger values tend to appear in earlier positions. In our setting, we apply WPI to the treatment time array. Consequently, a large WPI suggests that patients with longer treatment times are scheduled toward the end of the day, while a small WPI indicates that longer-treatment patients are scheduled earlier. This feature therefore captures whether the schedule follows a pattern similar to ascending or descending treatment-time ordering. We also consider a modified version of WPI in which the position index i in the numerator is replaced by the scheduled time of patient i. This variant measures the average scheduled time weighted by treatment duration and provides a more direct characterization of how treatment workload is distributed throughout the day.

6. **Coefficient of Variation**: This feature measures the relative variability of a given array. For an array t with mean μ_t and standard deviation σ_t, the

coefficient of variation is defined as

$$\mathrm{CV}(t) = \frac{\sigma_t}{\mu_t}.$$

The coefficient of variation is a dimensionless measure of dispersion that quantifies the magnitude of variability relative to the average value of the data. A small coefficient of variation indicates that the values are relatively homogeneous, whereas a large coefficient of variation indicates greater variability and dispersion. In our setting, we compute the coefficient of variation using the treatment time array. A large coefficient of variation suggests that treatment durations vary substantially across patients, while a small coefficient of variation indicates that most patients have similar treatment times. This feature therefore captures the degree of heterogeneity in patient workloads and may help explain differences in policy performance across scheduling instances.

We generated 500 days of patient schedules and computed the corresponding total waiting time and makespan for each day. For every day, we constructed a feature vector consisting of patient treatment times, scheduled times determined by the scheduling method, and the additional aggregate features described above. The dataset was then divided into training and testing sets using a 4:1 ratio, resulting in 400 training instances and 100 testing instances.

For each day in the training set, we identified the best-performing policy using the combined objective function of expected total waiting time and makespan. The policy achieving the lowest objective value was assigned as the label for that in-

stance. Consequently, the classification problem consists of predicting which policy is expected to perform best on average for a given set of patient and scheduling characteristics.

We implemented the classifier using the `scikit-learn` package. Specifically, we trained a Random Forest classifier with 300 decision trees. The resulting decision rule is referred to as the *Super Policy*, as it learns from historical performance data and adaptively selects among the candidate policies based on the characteristics of a given day. Rather than following a fixed policy, the Super Policy attempts to combine the strengths of multiple policies through data-driven policy selection.

The performance of the Super Policy is evaluated on the testing set and compared with the existing policies using total waiting time, makespan, and the combined objective function.

Figure 6.16 presents the total waiting time performance of the Super Policy compared with the existing policies. For the interlaced ascending scheduling variant, the Super Policy performs similarly to the other policies, with only negligible differences. This is consistent with the results discussed in Section 6.5, where the interlaced ascending schedule leads to nearly identical performance across all policies.

For the remaining scheduling methods, the Super Policy consistently outperforms or matches FIFO, LPT, and QRoll in terms of total waiting time. Its performance is often comparable to that of SPT and GRoll, particularly under the push descending, interlaced asis, and interlaced ascending variants. In some cases, such as push ascending and interlaced descending scheduling, the Super Policy achieves

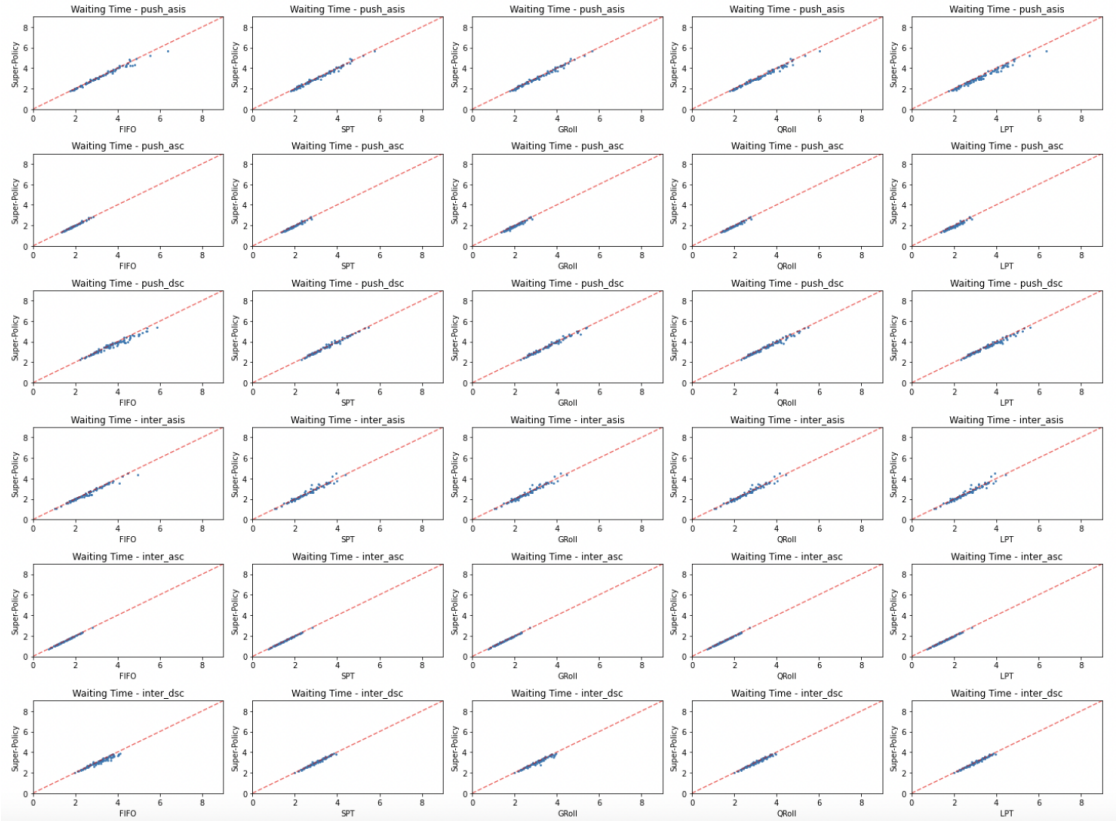


Figure 6.16: Waiting time of Super Policy and other policies

slightly lower total waiting times than GRoll, although the improvement is relatively small. Overall, the results indicate that the Super Policy is able to match the strongest existing policies while maintaining robust performance across different scheduling methods.

Most of the differences in makespan among the policies are relatively small. Figure 6.17 shows that the Super Policy achieves makespan results that are comparable to those of SPT, LPT, QRoll, and GRoll across all scheduling methods. Although the Super Policy generally outperforms FIFO, the improvement in makespan is modest compared with the differences observed in total waiting time.

Compared to the waiting-time results discussed earlier, the primary advantage

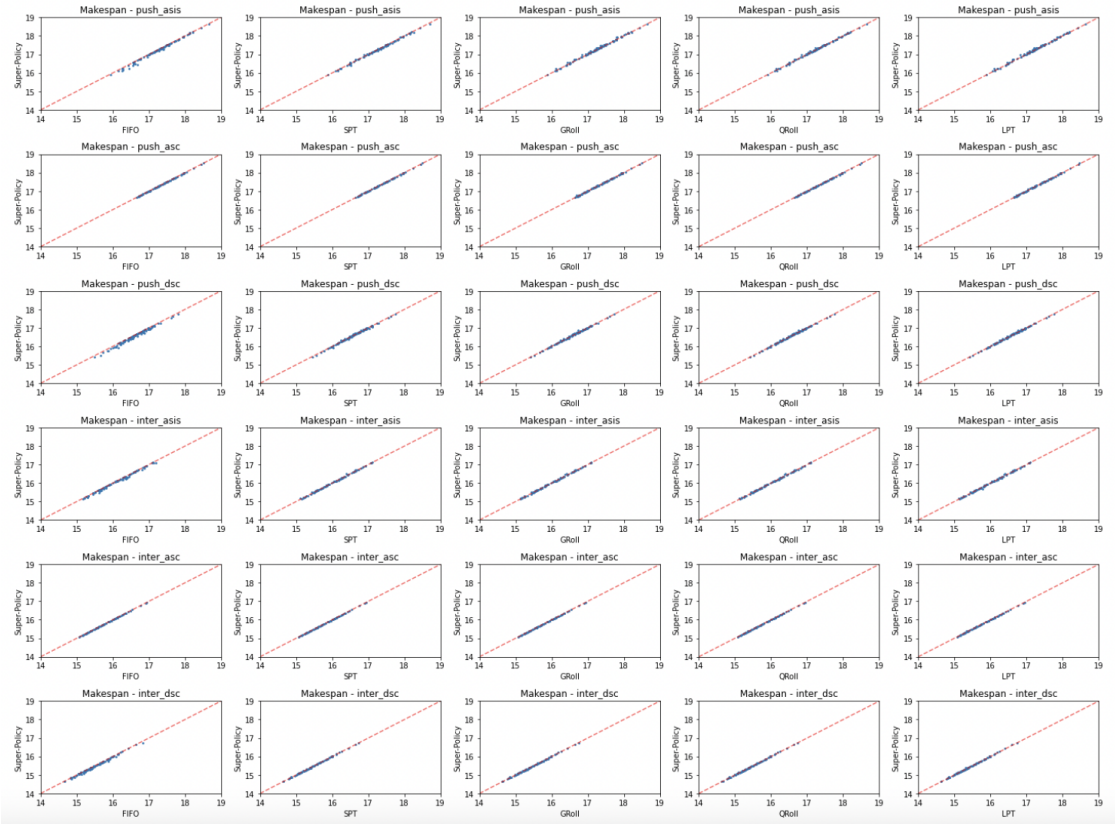


Figure 6.17: Makespan of Super Policy and other policies

of the Super Policy is that it achieves improved total waiting-time performance while maintaining makespan performance that is competitive with the best existing policies. Consequently, the Super Policy provides a favorable balance between the two objectives, offering consistent improvements in waiting time without sacrificing makespan.

We obtain similar results for the combined objective as those observed for total waiting time. The Super Policy consistently outperforms or matches FIFO, LPT, and QRoll across all scheduling methods.

Figure 6.18 further shows that, compared to SPT and GRoll, the Super Policy achieves better performance under the push asis, push ascending, interlaced asis,

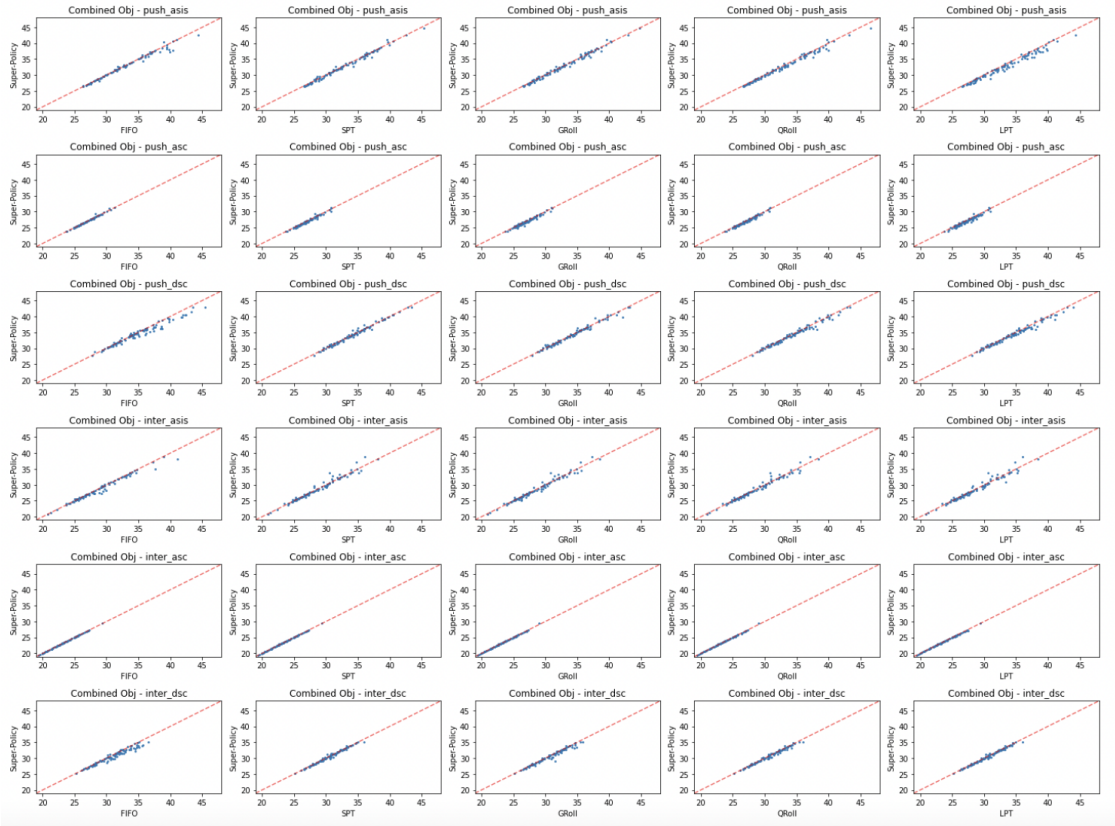


Figure 6.18: Combined Objective of Super Policy and other policies

and interlaced descending scheduling methods. For the push descending and interlaced ascending variants, its performance is comparable to both SPT and GRoll. In the push descending case, the Super Policy exhibits higher variability across instances, whereas under interlaced ascending scheduling, its performance closely matches both SPT and GRoll.

Figure 6.19 presents examples of feature importance for the classifier trained under the push asis and push descending scheduling methods. In both cases, the plots display the top ten most influential features used by the classification model.

Across both settings, the weighted position index (WPI) and its variant (WsPI)—which replaces the positional index with scheduled times—consistently

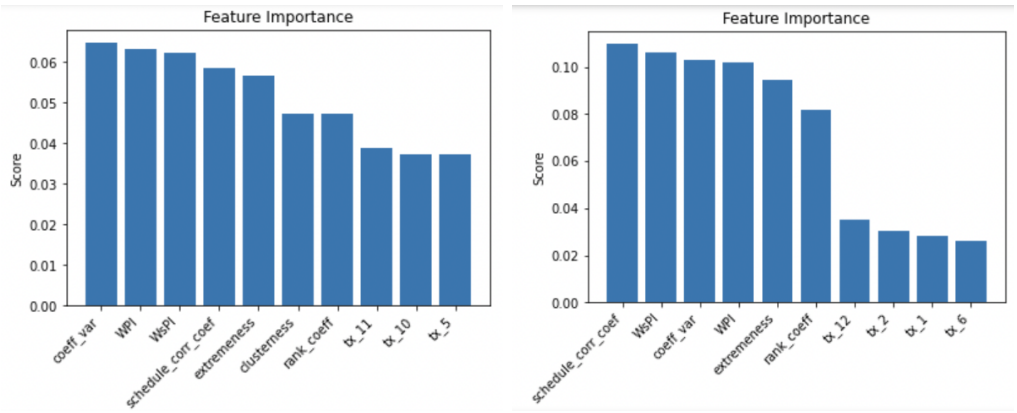


Figure 6.19: Feature importance from the classifier under push asis (left) and push descending (right)

appear among the top five most important features. This indicates that features capturing the distribution of treatment times and their temporal placement throughout the day play a central role in the classifier's decision-making process. In particular, these measures significantly influence the selection of the technician policy.

In summary, we demonstrate that a Super Policy can be constructed using patient treatment times, scheduled times, and other global features as inputs to a classification model. Overall, this Super Policy achieves better performance in terms of total waiting time, makespan, and the combined objective compared to FIFO, LPT, and QRoll. However, its performance is comparable to that of SPT and GRoll under the push descending and interlaced ascending scheduling methods. In the remaining scheduling settings, it still produces consistently strong and competitive results across all objectives.

6.7 Conclusions

Without control over the scheduling method, SPT provides the best overall performance in terms of total waiting time, makespan, and the combined objective across most scheduling methods. The exception is the ascending scheduling variant, where all policies tend to exhibit similar performance. Although SPT often produces results comparable to those of GRoll, it is considerably simpler to implement and easier for technicians to follow in practice. When control over the scheduling method is limited, we propose the Super Policy, which is constructed through a machine-learning classification model using patient-specific features. The goal is to create a policy that performs well regardless of the scheduling method being used.

To train the Super Policy, we use patients' treatment times, scheduled times, and several additional features derived from these quantities. The labels for the classifier are generated by selecting the policy that achieves the best value of the combined objective for each simulated day. While the process of generating labels is straightforward, careful feature engineering is essential. In particular, the features must capture multiple aspects of patient information, including treatment durations, scheduled times, and the ordering of treatments throughout the day. The resulting Super Policy is also easy to implement in practice. Once the classification model has been trained offline, the clinic only needs to compute the required features for the current set of patients and evaluate the classifier, which immediately recommends a policy for the technician to follow. No additional optimization problem needs to be solved during clinic operations, making the approach computationally inexpensive and easy to integrate into existing workflows. The Super Policy is especially valuable under the as-is scheduling variant, where patients are

scheduled without any deliberate ordering based on treatment times. In contrast, the ascending scheduling variant already yields similar performance across most policies, while the descending variant offers less opportunity for improvement than the as-is variant.

All analyses and experiments in this chapter focus on a dialysis clinic with a single technician who is responsible for all operational tasks and decision making. An important direction for future research is to extend the framework to larger clinics with more chairs, more patients, and multiple technicians. One straightforward extension of the simulation model in Section 6.2 is to assign each incoming task to the first available technician, treating technicians as parallel resources with similar responsibilities. Another approach is to organize the clinic into multiple pods and allocate different task types to different technicians. For example, [78] considers a clinic with two technicians by combining two pods containing a total of $2C$ chairs and assigning all put-on tasks to one technician and all take-off tasks to the other. Alternatively, technicians may be assigned responsibility for different subsets of patients rather than different task types. For instance, in a clinic with 24 patients, one technician could manage the first 12 patients while the second technician manages the remaining 12. The machine-learning framework proposed in this chapter could also be extended to this setting by training technician-specific Super Policies based on each technician's historical experience and decision-making patterns.

The improvements achieved by the proposed scheduling methods and policies relative to the baseline FIFO policy are generally modest. Likewise, the performance differences among SPT, LPT, QRoll, and GRoll are often small. Nevertheless, these improvements come at virtually no additional operational cost. The proposed

policies simply alter the order in which patients are processed, while the scheduling methods modify the timing and ordering of patient appointments. Consequently, meaningful improvements in waiting time and makespan can be achieved without requiring additional staff, equipment, or infrastructure. Although the gains are modest, they represent practical and low-cost enhancements to patient flow and clinic resource utilization, ultimately contributing to more efficient healthcare delivery and an improved patient experience.

BIBLIOGRAPHY

[1] Alekh Agarwal, Martin J. Wainwright, Peter Bartlett, and Pradeep Ravikumar. Information-theoretic lower bounds on the oracle complexity of convex optimization. *Advances in Neural Information Processing Systems*, 22, 2009.

[2] Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

[3] Ahmad Ajalloeian and Sebastian U. Stich. On the convergence of SGD with biased gradients. *arXiv:2008.00051*, 2020.

[4] Ali Ala and Feng Chen. Appointment scheduling problem in complexity systems of the healthcare services: A comprehensive review. *Journal of Healthcare Engineering*, 2022(1):5819813, 2022.

[5] Zeyuan Allen-Zhu, Elad Hazan, Wei Hu, and Yuanzhi Li. Linear convergence of a frank-wolfe type algorithm over trace-norm balls. *Advances in neural information processing systems*, 30, 2017.

[6] Mokhtari Aryan, Hamed Hassani, and Amin Karbasi. Stochastic conditional gradient methods: From convex minimization to submodular maximization. *Journal of machine learning research*, 21(105):1–49, 2020.

[7] Amir Beck and Shimrit Shtern. Linearly convergent away-step conditional gradient for non-strongly convex functions. *Mathematical Programming*, 164:1–27, 2017.

[8] Martin Beckmann, Charles B. McGuire, and Christopher B. Winsten. *Studies in the Economics of Transportation.* Yale University Press, 1956.

[9] Jose Blanchet, Coralia Cartis, Matt Menickelly, and Katya Scheinberg. Convergence rate analysis of a stochastic trust region method via supermartingales. *INFORMS Journal on Optimization*, 1(2):92–119, 2019.

[10] Natthawut Boonsiriphatthanajaroen, Rongyi He, Litong Liu, Tinghan Ye, and Shane G. Henderson. Evaluating solvers for linearly constrained simulation optimization. In H. Lam, E. Azar, D. Batur, S. Gao, W. Xie, S.R. Hunter, and M. D. Rossetti, editors, *Proceedings of the 2024 Winter Simulation Conference*, pages 3482–3493, Piscataway, New Jersey, 2024. IEEE.

[11] Natthawut Boonsiriphatthanajaroen and Shane G. Henderson. Designing a Frank-Wolfe algorithm for simulation optimization over unbounded linearly constrained feasible regions. In *2025 Winter Simulation Conference (WSC)*, pages 3346–3357, 2025.

[12] Natthawut Boonsiriphatthanajaroen and Shane G. Henderson. A new convergence analysis of two stochastic Frank-Wolfe algorithms, 2025.

[13] Stephen Boyd and Lieven Vandenberghe. *Convex optimization.* Cambridge university press, 2004.

[14] Gábor Braun, Alejandro Carderera, Cyrille W. Combettes, Hamed Hassani, Amin Karbasi, Aryan Mokhtari, and Sebastian Pokutta. Conditional gradient methods. *arXiv preprint arXiv:2211.14103*, 2022.

[15] Tugba Cayirli and Emre Veral. Outpatient scheduling in health care: a review of literature. *Production and operations management*, 12(4):519–549, 2003.

[16] Tugba Cayirli and Kum Khiong Yang. A universal appointment rule with patient classification for service times, no-shows, and walk-ins. *Service Science*, 6(4):274–295, 2014.

[17] Bo Chen. A note on LPT scheduling. *Operations Research Letters*, 14(3):139–142, 1993.

[18] Ping-Shun Chen, Hsiu-Wen Chen, Marielle Donice M. Abiog, Roxanne Mae B. Guerrero, and Christine Grace E. Latina. Patient unpunctuality's effect on appointment scheduling: A scenario-based analysis. *Healthcare*, 11(2):231, 2023.

[19] Ruobing Chen, Matt Menickelly, and Katya Scheinberg. Stochastic optimization using a trust-region method and random models. *Mathematical Programming*, 169:447–487, 2018.

[20] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001.

[21] Cyrille Combettes and Sebastian Pokutta. Boosting Frank-Wolfe by chasing gradients. *International conference on machine learning*, pages 2111–2121, 2020.

[22] Fang Cong, Chris J. Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path integrated differential estimator. *Advances in neural information processing systems*, 31, 2018.

[23] Nur Banu Demir, Serhat Gul, and Melih Çelik. A stochastic programming approach for chemotherapy appointment scheduling. *Naval Research Logistics*, 68(1):112–133, 2021.

[24] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[25] David J. Eckman, Shane G. Henderson, and Sara Shashaani. Diagnostic tools for evaluating and comparing simulation-optimization algorithms. *INFORMS Journal on Computing*, 35(2):350–367, 2023.

[26] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3, 1956.

[27] Peter I. Frazier, J. Massey Cashore, Ning Duan, Shane G. Henderson, Alyf Janmohamed, Brian Liu, David B. Shmoys, Jiayue Wan, and Yujia Zhang. Modeling for COVID-19 college reopening decisions: Cornell, a case study. *Proceedings of the National Academy of Sciences*, 119(2):e2112532119, 2022.

[28] Michael C. Fu. Stochastic gradient estimation. In Michael C. Fu, editor, *Handbook of Simulation Optimization*, volume 216 of *International Series in Operations Research & Management Science*, chapter 5, pages 105–148. Springer New York, 2015.

[29] Dan Garber. Faster projection-free convex optimization over the spectrahedron. *Advances in Neural Information Processing Systems*, 29, 2016.

[30] Dan Garber and Elad Hazan. Faster rates for the frank-wolfe method over strongly-convex sets. In *Proceedings of the 32nd International Conference on Machine Learning - Volume 37*, pages 541–549. JMLR.org, 2015.

[31] Dan Garber and Elad Hazan. A linearly convergent variant of the conditional gradient algorithm under strong convexity, with applications to online and stochastic optimization. *SIAM Journal on Optimization*, 26(3):1493–1528, 2016.

[32] Guillaume Garrigos and Robert M. Gower. Handbook of convergence theorems for (stochastic) gradient methods. *arXiv:2301.11235*, 2013.

[33] P. W. Glynn. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33:75–84, 1990.

[34] Donald Goldfarb, Garud Iyengar, and Chaoxu Zhou. Linear convergence of stochastic Frank-Wolfe variants. *Artificial Intelligence and Statistics*, pages 1066–1074, 2017.

[35] Ronald L. Graham. Bounds on multiprocessing timing anomalies. *SIAM journal on Applied Mathematics*, 17(2):416–429, 1969.

[36] Diwakar Gupta and Brian Denton. A sequential bounding approach for optimal appointment scheduling. *IIE transactions*, 35(11):1003–1016, 2003.

[37] Diwakar Gupta and Brian Denton. Appointment scheduling in health care: Challenges and opportunities. *IIE transactions*, 40(9):800–819, 2008.

[38] Isabelle Guyon, Steve Gunn, Asa Ben-Hur, and Gideon Dror. Result analysis of the nips 2003 feature selection challenge. In *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2004.

[39] Hassani Hamed, Amin Karbasi, Aryan Mokhtari, and Zebang Shen. Stochastic conditional gradient++:(non) convex minimization and continuous submodular maximization. *SIAM Journal on Optimization*, 30(4):3315–3344, 2020.

[40] Ying Han, Marina E. Johnson, Xiaojun Shan, and Mohammad Khasawneh. A multi-appointment patient scheduling system with machine learning and optimization. *Decision Analytics Journal*, page 100392, 2024.

[41] Elad Hazan and Satyen Kale. Projection-free online learning. *arXiv:1206.4657*, 2012.

[42] Elad Hazan and Haipeng Luo. Variance-reduced and projection-free stochastic optimization. *International Conference on Machine Learning*, 2016.

[43] Parker Sarah Henrickson, Matthew N. Jesso, Laurie D. Wolf, Kerry Avondet Leigh, Stephanie Booth, Nicole Gualandi, Renee E. Garrick, Alan S. Kliger, and Priti R. Patel. Human factors contributing to infection prevention in outpatient hemodialysis centers: a mixed methods study. *American Journal of Kidney Diseases*, 84(1):18–27, 2024.

[44] Alireza F. Hesaraki, Nico P. Dellaert, and Ton de Kok. Generating outpatient chemotherapy appointment templates with balanced flowtime and makespan. *European Journal of Operational Research*, 275(1):304–318, 2019.

[45] Frederick S. Hillier. *Introduction to operations research.* McGrawHill, 2005.

[46] Alfredo N. Iusem. On the convergence properties of the projected gradient method for convex optimization. *Computational & Applied Mathematics*, 22:37–52, 2003.

[47] Martin Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. *Proceedings of Machine Learning Research*, 2013.

[48] Chi Jin, Praneeth Netrapalli, Rong Ge, Sham M. Kakade, and Michael I. Jordan. A short note on concentration inequalities for random vectors with sub-Gaussian norm. *arXiv:1902.03736*, 2019.

[49] Anatoli Juditsky, Arkadi Nemirovski, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19:1574–1609, 2009.

[50] Guido C. Kaandorp and Ger Koole. Optimal outpatient appointment scheduling. *Health care management science*, 10(3):217–229, 2007.

[51] Seung-Jean Kim, Kwangmoo Koh, Stephen Boyd, and Dimitry Gorinevsky. ℓ_1 trend filtering. *SIAM review*, 51(2):339–360, 2009.

[52] Kenneth J. Klassen and Reena Yoogalingam. Strategies for appointment policy design with patient unpunctuality. *Decision Sciences*, 45(5):881–911, 2014.

[53] Simon Lacoste-Julien and Martin Jaggi. An affine invariant linear convergence analysis for Frank-Wolfe algorithms. *arXiv:1312.7864*, 2013.

[54] Simon Lacoste-Julien and Martin Jaggi. On the global linear convergence of Frank-Wolfe optimization variants. *Neural Information Processing Systems*, 2015.

[55] Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher. Block-coordinate Frank-Wolfe optimization for structural SVMs. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 53–61, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[56] Simon Lacoste-Julien, Mark Schmidt, and Francis Bach. A simpler approach to obtaining an $\mathcal{O}(1/t)$ convergence rate for projected subgradient method. *arXiv:1212.2002*, 2012.

[57] Jean Lafond, Hai-To Wai, and É. Moulines. On the online frank-wolfe algorithms for convex and non-convex optimizations. *arXiv:1510.01171*, 2015.

[58] Guanghui Lan and Yi Zhou. Conditional gradient sliding for convex optimization. *SIAM Journal on Optimization*, 26(2):1379–1409., 2016.

[59] E.S. Levitin and B.T. Polyak. Constrained minimization methods. *USSR Computational Mathematics and Mathematical Physics*, pages 1–50, 1996.

[60] Francesco Locatello, Michael Tschannen, Gunnar Rätsch, and Martin Jaggi. Greedy algorithms for cone constrained optimization with convergence guarantees. *Advances in Neural Information Processing Systems*, 30:773–784, 2017.

[61] Haihao Lu and Robert M. Freund. Generalized stochastic Frank-Wolfe algorithm with stochastic substitute gradient for structured convex optimization. *Mathematical Programming*, 187:317–349, 2021.

[62] Rualan Nazykov, Aleksandar Shestakov, Vladimir Solodkin, Aleksandar Beznosikov, Gauthier Gidel, and Alexandar Gasnikov. Stochastic Frank-Wolfe: Unified analysis and zoo of special cases. *International Conference on Machine Learning*, pages 4870–4878, 2024.

[63] Lam M. Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. Sarah: A novel method for machine learning problems using stochastic recursive gradient. *International conference on machine learning*, pages 2613–2621, 2017.

[64] Geoffrey Négiar, Gideon Dresdner, Alicia Tsai, Laurent El Ghaoui, Francesco Locatello, Robert M. Freund, and Fabian Pedregosa. Stochastic Frank-Wolfe for constrained finite-sum minimization. *International Conference on Machine Learning*, pages 7253–7262, 2020.

[65] National Forum of ESRD Networks. Quarterly national ESRD census., 2025. `https://esrdnetworks.org/resources-news/national-esrd-census-data/`.

[66] Xingwei Pan, Na Geng, and Xiaolan Xie. Appointment scheduling and real-time sequencing strategies for patient unpunctuality. *European Journal of Operational Research*, 295(1):246–260, 2021.

[67] Courtney Paquette and Katya Scheinberg. A stochastic line search method with convergence rate analysis. *SIAM Journal on Optimization*, 30(1):349–376, 2020.

[68] Yijie Peng, Michael C. Fu, Jiaqiao Hu, Pierre L'Ecuyer, and Bruno Tuffin. Generalized likelihood ratio method for stochastic models with uniform random numbers as inputs. *European Journal of Operational Research*, 321:493–502, 2025.

[69] Sashank J. Reddi, Suvrit Sra, Barnabas Poczos, and Alex Smola. Stochastic Frank-Wolfe methods for nonconvex optimization. *54th annual Allerton conference on communication, control, and computing (Allerton)*, pages 1244–1251, 2016.

[70] Matthew B. Rivara, Scott V. Adams, Sooraj Kuttykrishnan, Kamyar Kalantar-Zadeh, Onyebuchi A. Arah, Alfred K. Cheung, Ronit Katz, Miklos Z Molnar, Vanessa Ravel, Melissa Soohoo, Elani Streja, Jonathan Himmelfarb, and Rajnish Mehrotra. Extended-hours hemodialysis is associated with lower mortality risk in patients with end-stage renal disease. *Kidney International*, 90(6):1312–1320, 2016.

[71] Lawrence W. Robinson and Rachel R. Chen. Scheduling doctors' appointments: optimal and empirically-based heuristic policies. *IIE Transactions*, 35(3):295–307, 2003.

[72] Ralph T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

[73] Panwalkar Shrikant S. and Wafik Iskander. A survey of scheduling rules. *Operations research*, 25(1):45–61, 1977.

[74] Pablo Santibáñez, Ruben Aristizabal, Martin L. Puterman, Vincent S. Chow, Wenhai Huang, Christian Kollmannsberger, Travis Nordin, Nancy Runzer, and Scott Tyldesley. Operations research methods improve chemotherapy patient appointment scheduling. *The Joint Commission Journal on Quality and Patient Safety*, 38(12):541–AP2, 2012.

[75] Zebang Shen, Cong Fang, Peilin Zhao, Junzhou Huang, and Hui Qian. Complexities in projection-free stochastic non-convex minimization. *The 22nd international conference on artificial intelligence and statistics*, pages 2868–2876, 2019.

[76] Yuan Shi, Saied Mahdian, Jose Blanchet, Peter Glynn, Andrew Y. Shin, and David Scheinker. Surgical scheduling via optimization and machine learning with long-tailed data. *Health Care Management Science*, 26(4):692–718, 2023.

[77] SimOpt Library. Simulation optimization (simopt) library. `https://github.com/simopt-admin/simopt/tree/python_dev_boom`, 2025. Accessed Aug 19, 2025.

[78] Vincent Slaugh and Andre Augusto. Dialysis clinic patient flow and work design. *SSRN*, 2025.

[79] Sharan Srinivas and A. Ravi Ravindran. Optimizing outpatient appointment system using machine learning algorithms and scheduling rules: A prescriptive analytics framework. *Expert Systems with Applications*, 102:245–261, 2018.

[80] Tongyi Tang, Krishna Balasubramanian, and Thomas Chun Man Lee. High-probability bounds for robust stochastic frank-wolfe algorithm. *Uncertainty in Artificial Intelligence*, pages 1917–1927, 2022.

[81] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.

[82] Man Yiu Tsang, Karmel S. Shehadeh, Frank E. Curtis, Beth R. Hochman, and Tricia E. Brentjens. Stochastic optimization approaches for an operating room and anesthesiologist scheduling problem. *Operations research*, 73(3):1430–1458, 2025.

[83] Pieter L. van den Berg, Océane Fourmentraux, Shane G. Henderson, Caroline J. Jagtenberg, and Hemeng Li. Learning to dispatch volunteers to out-of-hospital cardiac arrests. *Health Care Management Science*, 29(2):21, 2026.

[84] Roman Vershynin. *Introduction to the non-asymptotic analysis of random matrices*. Cambridge University Press, 2012.

[85] Haoyue Wang, Haihao Lu, and Rahul Mazumder. Frank–wolfe methods with an unbounded feasible region and applications to structured learning. *SIAM Journal on Optimization*, 32(4):2938–2968, 2022.

[86] Gideon Weiss. Turnpike optimality of Smith's rule in parallel machines stochastic scheduling. *Mathematics of Operations Research*, 17(2):255–270, 1992.

[87] J. D. Welch and Normal T. J. Bailey. Appointment systems in hospital outpatient departments. *The lancet*, 259(6718):1105–1108, 1952.

[88] Kayode A. Williams, Chester G. Chambers, Maqbool Dada, Julia C. McLeod, and John A. Ulatowski. Patient punctuality and clinic performance: observations from an academic-based private practice pain centre: a prospective quality improvement study. *BMJ Open*, 4(5):e004679, 2014.

[89] Philip Wolfe. Convergence theory in nonlinear programming. *Integer and nonlinear programming*, 3:1–36, 1970.

[90] Alp Yurtsever, Suvrit Sra, and Volkan Cevher. Conditional gradient methods via stochastic path-integrated differential estimator. *International conference on machine learning*, pages 7282–7291, 2019.

[91] Mingrui Zhang, Zebang Shen, Aryan Mokhtari, Hamed Hassani, and Amin Karbasi. One sample stochastic Frank-Wolfe. *International Conference on Artificial Intelligence and Statistics*, pages 4012–4023, 2020.

[92] Han Zhu, Youhua Chen, Eman Leung, and Xing Liu. Outpatient appointment scheduling with unpunctual patients. *International Journal of Production Research*, 56(5):1982–2002, 2018.