UNIVERSITÀ DI PISA

DIPARTIMENTO DI MATEMATICA

Corso di Laurea Triennale in Matematica

Tesi di Laurea

Improvement of Frank-Wolfe Methods via Bundle-inspired Directions

RELATORE:
Prof. Antonio Frangioni

CANDIDATA:
Silvia Calabretta

CORRELATORE:
Dott. Gabriele Iommazzo

ANNO ACCADEMICO 2023 - 2024

Alla mia famiglia,
e a tutte le amicizie nate lungo questo percorso.

Contents

Abstract

This thesis presents a modified version of the Frank-Wolfe method with an alternative approach to selecting the search direction. The Frank-Wolfe algorithm is known to have slow convergence due to its tendency to zigzag. Inspired by Bundle methods, this issue is addressed by developing a regularized piecewise-linear approximation of the objective function, exploiting the information derived from the previous iterates. Compared to the bare gradient, this model provides a more informed direction to be passed to the Linear Minimization Oracle, that defines the search direction, and therefore enhances convergence. The method was implemented to validate the proposed modification and assess its performance. We provide experiments on various step size rules and Linear Minimization Oracles, and perform a thorough tuning of the algorithmic parameters of the new approach, in order to analyze under which conditions it is competitive with the Frank-Wolfe method.

Introduction

Optimization problems play a crucial role in numerous scientific and engineering fields, with applications ranging widely and gaining renewed interest due to machine learning. A prominent approach to solving constrained optimization problems is the Frank-Wolfe (F-W) method, also known as conditional gradient method. Its main appeal lies in its simplicity, making it especially suitable for problems with constraint structures that allow efficient linear programming. However, a well-documented limitation of the F-W method is its slow convergence, particularly in the presence of zigzagging behavior.

To address this limitation, various enhancements and modifications of the F-W method have been proposed over the years. Regularization and stabilization techniques, often employed in bundle methods, offer a promising direction for improving performance. Bundle methods are designed to approximate the objective function using piecewise-linear models that incorporate information from past iterations.

This thesis introduces a modified version of the F-W method that incorporates ideas from bundle methods to mitigate the zigzagging behavior and improve convergence rates. Specifically, the proposed approach leverages a regularized piecewise-linear approximation of the translated objective function to derive more informed search directions, which replace the bare gradient used in the vanilla F-W method. The new method seeks to achieve better performance while preserving the computational efficiency and simplicity that characterize the F-W framework.

The proposed method has been implemented and evaluated through extensive numerical experiments. These experiments explore the impact of various step size rules and linear minimization oracles (LMOs). Particular attention is given to identifying the conditions under which the new method outperforms the vanilla F-W method.

The thesis is organized as follows. Chapter 1 provides a comprehensive contextual framework, an overview of F-W algorithms, and key concepts from bundle methods. Chapter 2 details the development of the bundle-enhanced F-W method, including parameter tuning, convergence analysis, and dual problem considerations. Chapter 3 focuses on the implementation aspects, addressing subproblem formulations, stopping criteria, and the handling of numerical errors. Finally, Chapter 4 presents the results of the experimentation, including an evaluation of the proposed method, comparisons with the F-W method, and an analysis of the conditions under which the new approach demonstrates its competitive advantages. The Appendix A includes part of the implemented code for reproducibility and further exploration.

Chapter 1

Contextual Framework

1.1 Preliminaries

This section introduces definitions useful for understanding the problem context.

Definition 1 (Convex set)**.** A set $\mathcal{X} \subseteq \mathbb{R}^n$ is *convex* if $\forall x, y \in \mathcal{X}$

$$\lambda x + (1-\lambda)y \in \mathcal{X} \quad \text{for all } 0 \leq \lambda \leq 1.$$

Definition 2 (Convex function)**.** A function $f : \mathcal{X} \to \mathbb{R}$ is *convex* if its domain $\mathcal{X}$ is convex and

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y) \quad \text{for all } x, y \in \mathcal{X},\ 0 \leq \lambda \leq 1.$$

If f is differentiable over $\mathcal{X}$, then

$$f \quad \text{convex} \iff f(y) - f(x) \geq \langle \nabla f(x), y - x \rangle \quad \text{for all } x, y \in \mathcal{X}. \tag{1.1}$$

A valuable property of convex functions is that local minima are always global minima.

Definition 3 (Subgradient)**.** If $f : \mathcal{X} \to \mathbb{R}$ is convex, a vector $z \in \mathbb{R}^n$ is called a *subgradient* of f at $x \in \mathcal{X}$ if

$$f(y) - f(x) \geq \langle z, y - x \rangle \quad \text{for all } y \in \mathcal{X}. \tag{1.2}$$

The set of all subgradients at x is called *subdifferential* at x and is denoted as $\partial f(x)$. The subdifferential is always a non empty convex compact set.

Definition 4 (ϵ-Subgradient)**.** If $f : \mathcal{X} \to \mathbb{R}$ is convex, a vector $z \in \mathbb{R}^n$ is called an *ϵ-subgradient* of f at $x \in \mathcal{X}$ if

$$f(y) - f(x) \geq \langle z, y - x \rangle - \epsilon \quad \text{for all } y \in \mathcal{X}. \tag{1.3}$$

The set of all ϵ-subgradients at x is denoted as $\partial_\epsilon f(x)$.

Iterative optimization algorithms generally depend on a reliable local approximation of the objective function. The following two properties has proven effective in constructing quadratic approximations.

Definition 5 (Strongly convex function)**.** A function $f : \mathcal{X} \to \mathbb{R}$, differentiable over $\mathcal{X}$, is *μ-strongly smooth* over $\mathcal{X}$ if $\exists\ \mu > 0$ such that

$$f(y) - f(x) \geq \langle \nabla f(x), y - x \rangle + \frac{\mu}{2}\|y - x\|^2 \quad \text{for all } x, y \in \mathcal{X}.$$

Definition 6 (Smooth function)**.** A function $f : \mathcal{X} \to \mathbb{R}$, differentiable over $\mathcal{X}$, is *L-smooth* over $\mathcal{X}$ if $\exists\ L > 0$ such that

$$f(y) - f(x) \leq \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|y - x\|^2 \quad \text{for all } x, y \in \mathcal{X}.$$

In particular, smoothness is fundamental as it allows to estimate the decrease in the value of f when moving to a different point of $\mathcal{X}$, as it is shown in the following [1, Lemma 1.5].

Lemma 7 (Progress lemma for smoothness)**.** Let f be an L-smooth function and let $y = x - \gamma d$, where d is an arbitrary vector. If y is in the domain, then

$$f(x) - f(y) \geq \frac{\langle \nabla f(x), y - x \rangle}{2} \cdot \gamma \quad \text{for } 0 \leq \gamma \leq \frac{\langle \nabla f(x), d \rangle}{L\|d\|^2} \tag{1.4}$$

1.2 Frank-Wolfe algorithms

F-W algorithms [1, 2] are a class of first-order iterative algorithms for constrained convex optimization problems, although they can also be used for unconstrained optimization. Given a differentiable function $f : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X} \subseteq \mathbb{R}^n$ is a compact convex set, the goal is to find a point x^* in the domain that minimizes the value of f:

$$f_* = f(x^*) = \min_{x \in \mathcal{X}} f(x). \tag{1.5}$$

In general the minimizer is not unique, we look for a point $x^* \in \Omega^*$, the set of minima of f.

In order to decrease the value of the objective function, many first-order algorithms take a step along properly chosen directions and then perform projection to maintain feasibility of the iterates. Unfortunately the rise of large data and complicated constraints in recent years has made the computational cost of the projection operator unmanageably high. F-W algorithms are *projection-free methods*: they avoid projection by moving towards an extreme point via linear minimization. This task is managed through the following two oracles, which are in fact the only methods with direct access to the objective function f and the feasible region $\mathcal{X}$.

The *First-Order Oracle* (FOO) is queried with a point $x \in \mathcal{X}$ and returns the function value $f(x)$ and the gradient $\nabla f(x)$ of f at x.

Oracle 1: First-Order Oracle (FOO)
Input: Point $x \in \mathcal{X}$
Output: $f(x)$ and $\nabla f(x)$

The LMO is queried with a linear function c, and returns an extreme point $v \in \arg\min_{x \in \mathcal{X}} \langle c, x \rangle$. It is important to observe that v is not necessarily unique, but selecting one is a task performed by the oracle itself.

Oracle 2: Linear Minimization Oracle (LMO)
Input: Linear objective c
Output: $v \in \arg\min_{x \in \mathcal{X}} \langle c, x \rangle$

The original algorithm of this class is due to Frank and Wolfe [3]. It generates a sequence of feasible points x_i by optimizing first-order approximations of the objective function f, that we are assuming smooth and convex. At each iteration, the LMO is queried with $\nabla f(x_i)$ and provides an extreme point v_i, which is used to construct the *descent direction* $v_i - x_i$. The next iterate x_{i+1} is obtained by proceeding along this direction, i.e. $x_{i+1} = x_i + \gamma_i(v_i - x_i)$ in $[x_i, v_i]$, the segment between x_i and v_i, where γ_i is the *step size*.

One possible choice is *line search*, that consists in selecting γ such that x_{i+1} is the point with the minimum value function along the descent direction:

$$\gamma_i := \arg\min_{\gamma \in [0,1]} f(\gamma v + (1-\gamma)x).$$

This method guarantees the quickest progress and monotone decreasing function values, but typically has a high computational cost. That is why the next methods are generally preferred.

The *short step rule* exploits the smoothness of the objective function [4, pp. 18-19], leading to

$$\gamma_i := \min\{\frac{\langle \nabla f(x), x_i - v_i \rangle}{LD^2}, 1\},$$

where $D := \sup_{x,y \in \mathcal{X}} \|x - y\|$ is the *diameter* of $\mathcal{X}$ and L is the smoothness constant of f. This exact formula has the quality of being easily computable, and guarantees monotone decreasing function values. Unfortunately, it requires a good approximation of L, that is not necessarily easy to estimate. That is why some adaptive versions of this method have been developed, which dynamically approximate L adjusting the rule based on local changes.

The *function-agnostic step size rule* finally chooses γ_i based solely on the iterate number i, regardless of the objective function f. This avoids the necessity of approximating the smoothness constant L, and is highly beneficial when computing the values of f is expensive. An example is the rule $\gamma_i = 2/(i+2)$, which gained popularity thanks to Jaggi [5]. The main drawback of any function-agnostic step size rule is that the convergence rate of the algorithm is constrained to a predetermined value, limiting the algorithm's potential to adapt.

1.2.1 Indicators of performance

To assess the quality of a proposed solution x of the minimization problem, several measures can be used. A natural measure could be the *distance to the set of minima* $\operatorname{dist}(x, \Omega^*)$, but, unfortunately, little is known about the convergence behavior of the iterates x_i. Therefore, the F-W algorithms are evaluated via the function value $f(x_i)$ and in particular its distance from $f(x^*)$. The *primal gap* is the most commonly used measure for conditional gradient algorithms.

Definition 8 (Primal gap)**.** The *primal gap* of a function $f : \mathcal{X} \to \mathbb{R}$ at a point $x \in \mathcal{X}$ is

$$p(x) := \max_{y \in \mathcal{X}} \{f(x) - f(y)\} = f(x) - f(x^*).$$

However, without knowing an optimum x^* or the objective function value $f(x^*)$, the primal gap is not directly computable. In order to avoid this issue, the next measure is used.

Definition 9 (F-W gap)**.** The *F-W gap* (or *dual gap*) of a function $f : \mathcal{X} \to \mathbb{R}$ at a point x is

$$q(x) := \max_{y \in \mathcal{X}} \langle \nabla f(x), x - y \rangle.$$

The F-W gap represents the difference between the gradient at the current point and the best descent direction available within the feasible region. Clearly $p(x) \geq 0$ and $q(x) \geq 0$ for all $x \in \mathcal{X}$. The advantage in using the F-W gap is that it is computable without knowing x^* and it provides an upper bound on the primal gap [1, Equation 1.5]:

$$0 \leq p(x) = f(x) - f(x^*) \leq \langle \nabla f(x), x - x^* \rangle \leq \max_{v \in \mathcal{X}} \langle \nabla f(x), x - v \rangle = q(x). \tag{1.6}$$

It directly follows that $q(x)$ serves as a stopping criterion for the algorithm when it comes to find a minimum, as $q(x) < \epsilon$ guarantees a precision of at least ϵ on the estimate of f_*. Moreover, when f is convex and differentiable, $q(x)$ provide a *first-order optimality condition*: the point $x^* \in \mathcal{X}$ is optimal if and only if

$$\langle \nabla f(x^*), x^* - x \rangle \leq 0 \quad \text{for all } x \in \mathcal{X};$$

this in particular holds in (1.6) for the maximum of the scalar product over all $x \in \mathcal{X}$, which therefore gives

$$q(x) = 0 \iff x = \arg\min_{y \in \mathcal{X}} f(y).$$

However, when f is non-convex, this property is no longer valid: $q(x) = 0$ is a necessary condition but is not sufficient, even when considering local optimality.

The main steps performed by the vanilla F-W algorithm are summarized in Algorithm 3.

Algorithm 3: Vanilla F-W Algorithm

Input: f, x_0, LMO, step_size, max_iter, ϵ
Output: x , v

Initialization: $x \leftarrow x_0$, FW_gap $\leftarrow$ Inf.
while $k = 1 \dots$ *max_ iter* $\wedge$ *FW_ gap* $\geq \epsilon$ **do**
  $v_k \leftarrow \text{LMO}(\nabla f(x_k))$
  $d^k \leftarrow x_k - v_k$
  FW_gap $\leftarrow \langle \nabla f(x_k), d^k \rangle$
  $\gamma_k \leftarrow \text{step\_size}(x_k, v_k)$
  $k \leftarrow k + 1$
  $x_{k+1} \leftarrow x_k - \gamma_k \cdot d^k$
end

1.3 Bundle Methods

The Bundle Methods (BM) [6, 2] are a large class of algorithms for the constrained convex minimization problem (1.5), where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is proper and convex, but not necessarily differentiable. A different approach from that of the F-W algorithms is then required.

Firstly, the gradient is no longer available, so subgradients are used instead. The FOO is replaced with the *Subgradient First-Order Oracle* (SFOO): when queried with a point $x \in \mathcal{X}$, it returns the function value $f(x)$ and a subgradient in $\partial f(x)$ of f at x.

Oracle 4: Subgradient First-Order Oracle (SFOO)

Input: Point $x \in \mathcal{X}$
Output: $f(x)$ and $z \in \partial f(x)$

The basic idea of BM is to approximate the subdifferential of the objective function by gathering subgradients from previous iterations into a bundle $\mathcal{B}$. This approach provides more information about the local behavior of the function than a single arbitrary subgradient can offer.

At each iteration i, we consider some auxiliary points $x_j \in \mathcal{X}$ and subgradients $z_j \in \partial f(x_j)$ for $j \in \mathcal{J}_i$, where $\emptyset \neq \mathcal{J}_i \subset \{1, \dots, i\}$. The bundle $\mathcal{B}_i$ keeps track of the auxiliary points and their subgradients storing their values. The objective function f is now approximated with a convex model $\check{f}^i$ that relies on the bundle, so that the next iterate can be selected:

$$x_{i+1} \in \arg\min \{\check{f}^i(x) \mid x \in \mathcal{X}\}. \tag{1.7}$$

The *Master Problem* (MP) (1.7) is a convex optimization problem and in general a manageable problem to solve. The last step of the iteration consists in updating the bundle, and therefore the model, with the new trial point.

Algorithms in BM class are distinguished, among other things, by the model used

for the objective function. The most common choice is the *Cutting Plane Model* (CPM), which follows.

As shown in [2, Equation 11.1], a convex function f can be represented as

$$f(x) = \max\{f(y) + \langle z, x - y\rangle \mid z \in \partial f(y), y \in \mathbb{R}^n\} \quad \text{for all } x \in \mathbb{R}^n. \tag{1.8}$$

Since this representation would need the whole subdifferential $\partial f(y)$, which is an excessive requirement, the model is developed through approximation.

Definition 10 (Cutting Plane Model)**.** Given $\{(x_j, z_j)\}_{j\in\mathcal{J}_i}$, the *CPM* of f at the current iteration i is the finite piecewise approximation

$$\check{f}^i(x) := \max\{f(x_j) + \langle z_j, x - x_j\rangle \mid j \in \mathcal{J}_i\} \quad \text{for all } x \in \mathbb{R}^n. \tag{1.9}$$

What makes CPM a model of f is $z_j \in \partial f(x_j)$, as (1.2) directly implies $\check{f}^i(x) \leq f(x)$. By definition of the model, for all i it holds $\check{f}^i(x_j) = f(x_j)$. Moreover, as the iterations increase, the CPM becomes more and more refined, i.e. $\check{f}^i(x) \leq \check{f}^{i+1}(x)$.

When the CPM is used in BM, it is more efficient to define the bundle as $\mathcal{B}_i = \{(z_j, \alpha_j)\}_{j\in\mathcal{J}_i}$, where $\alpha_j := \langle z_j, x_j\rangle - f(x_j)$. The reason lies in the fact that (1.9) can be reformulated as

$$\check{f}^i(x) = \max\{\langle z_j, x\rangle - \alpha_j \mid j \in \mathcal{J}_i\}, \tag{1.10}$$

thus allowing to use and store the linearization errors α_i, without requiring direct access to the trial points.
An interesting feature of this method is that the next iterate can be written as

$$x_{i+1} \in \arg\min\{y \mid y \geq \langle z_j, x\rangle - \alpha_j,\ j \in \mathcal{J}_i, y \in \mathbb{R}, x \in \mathcal{X}\}. \tag{1.11}$$

This formulation brings out that the natural space for the MP in (1.11) is the epigraphical space of the objective function, where the variable y represents the values of $\check{f}^i$.

One of the most problematic aspects of the CPM is that the convergence is by nature prone to instability, potentially resulting in consecutive iterations far apart from one another, which also produce slow convergence. This leads to the necessity to stabilize the model, which can occur in different ways. The discussion can be applied to a generic model, but we will be referring to the CPM in particular. The following section closely follows [6, paras. 2.1-2.2].

1.3.1 Stabilization

Stabilizing the CPM consists in ensuring that the iterates do not deviate excessively from a properly chosen point. However the proper point - ideally x^* - is generally unknown, and thus must be estimated and updated iteratively. Therefore a sequence $\{\overline{x}_i\}$ of *stability centers* is considered. It is reasonable to assume that $\{\overline{x}_i\} \subseteq \{x_i\}$, with typically the useful consequence $\check{f}^i(\overline{x}_i) = f(\overline{x}_i)$ when they result to be part of

the bundle. Different variants of BM may respond differently to various stabilization strategies, and the *stabilization parameter* helps to regulate the process.

The *Proximal Bundle Method* (PBM) retrace the idea behind the proximal operator, considering the MP

$$x_{i+1} \in \arg\min \{\check{f}^i(x) + \frac{\mu_i}{2}\|x - \overline{x}_i\|_2^2\}, \tag{1.12}$$

where the stabilization parameter is μ_i. In this context, it is interesting to consider the Moreau-Yosida regularization ϕ_μ of f in terms of the *displacement* d from $\overline{x}$:

$$\phi_\mu(\overline{x}) = \min \{f(\overline{x} + d) + \frac{\mu}{2}\|d\|_2^2 \mid d \in \mathbb{R}^n\}. \tag{1.13}$$

Since $d = 0$ is feasible, it holds $\phi_\mu \leq f$. Moreover, given d_* the unique optimal solution of (1.13), $d_* = 0$ implies that $\overline{x}$ is a minimum both of f and ϕ_μ, hence minimizing ϕ_μ is equivalent to (1.5). Therefore, in the case where (1.13) is easily solvable, it is possible to determine the next stability center as $\overline{x}_{i+1} = \overline{x}_i + d_*^i = x_{i+1}$, which results in the usually called *Proximal Point Algorithm* (PPA).

1.4 Duality

When a mathematical optimization problem is challenging to solve, a classical approach is to consider the associated dual problem, with the hope that it may be more tractable. This section briefly expose the theory on the subject, and presents the dual problems of the models introduced in the previous section.
Let's consider the constrained minimization problem

$$\begin{aligned} f_* &= \min \{f(x) \mid x \in \mathcal{X}\} \\ \mathcal{X} &= \{x \in \mathbb{R}^n \mid g_i(x) \leq 0 \quad i \in \mathcal{I}, \\ &\qquad\qquad h_j(x) = 0 \quad j \in \mathcal{J}\}, \end{aligned} \tag{1.14}$$

where $\mathcal{I}$ and $\mathcal{J}$ are sets of indices. We assume the domain $\mathcal{D} = \bigcap_{i\in\mathcal{I}} \operatorname{dom} g_i \cap \bigcap_{j\in\mathcal{J}} \operatorname{dom} h_j$ non empty. The *Lagrangian* associated with the problem (1.14) is the function

$$L(x, \lambda, \mu) = f(x) + \sum_{i\in\mathcal{I}} \lambda_i g_i(x) + \sum_{j\in\mathcal{J}} \mu_j h_j(x). \tag{1.15}$$

The vectors $\lambda \geq 0$ and μ are the *dual variables* associated with the problem (1.14), and their components, λ_i and μ_j, are the *Lagrangian multipliers* respectively associated with the ith inequality and jth equality constraints. The *Lagrange dual function* is defined as the minimum value of the Lagrangian over x:

$$g(\lambda, \mu) = \inf_{x\in\mathcal{D}} L(x, \lambda, \mu) = \inf_{x\in\mathcal{D}} \Big(f(x) + \sum_{i\in\mathcal{I}} \lambda_i g_i(x) + \sum_{j\in\mathcal{J}} \mu_j h_j(x)\Big). \tag{1.16}$$

It is interesting to observe that since the dual function is the point-wise infimum of a family of affine functions of (λ, μ), it is always concave.

As shown in [7, para. 5.1.3], for any $\lambda \geq 0$ and any μ, the dual function provide lower bounds to the optimal value f_*, i.e. $g(\lambda, \mu) \leq f_*$. Therefore, the best lower bound yielded by the dual function is provided by the *Lagrange dual problem* associated with (1.14):

$$g_* = \max \{g(\lambda, \mu) \mid \lambda \geq 0\} \tag{1.17}$$

In this context, we may refer to (1.14) as the *primal problem*. Optimal values (λ^*, μ^*) for (1.17) are called *dual optimal* or *optimal Lagrange multipliers*.

The dual problem is a convex optimization problem, since we are maximizing the concave function $g(\lambda, \mu)$ on a convex set $(\lambda \geq 0)$, and this happens whether or not the primal problem is convex. Clearly, the inequality $g_* \leq f_*$ holds, which is referred to as *weak duality*. *Strong duality* holds when the inequality holds as an equality.

We now proceed to explicitly address the dual problem in the case of PBM [6, para. 3.1].

1.4.1 PBM dual problem

In this context, it is useful to consider the MP obtained through *translation* of (1.12):

$$\min_{d \in \mathbb{R}^n} \check{f}_{\overline{x}}(d), \quad \text{where} \quad \check{f}_{\overline{x}}(d) := \check{f}(\overline{x} + d) - f(\overline{x}) + \frac{\mu}{2}\|d\|_2^2. \tag{1.18}$$

Typically $\overline{x}$ is the current stability center and d is the displacement $x - \overline{x}$. Note that this model also depends on the bundle; however, we will avoid complicating the notation. Actually the translation affects the bundle itself, since α_i is replaced with the linearization error

$$\overline{\alpha}_i(\overline{x}) := f(\overline{x}) - f(x_i) - \langle z_i, \overline{x} - x_i \rangle = \alpha_i - \langle z_i, \overline{x} \rangle + f(\overline{x}). \tag{1.19}$$

For sake of simplicity we will be using $\overline{\alpha}_i$ as much as possible, since usually $\overline{x}$ is clear from the context.

This interpretation allows to rewrite (1.18) as follows, which represents the primal problem:

$$\min \{r + \frac{\mu_i}{2}\|d\|_2^2 \mid r \geq \langle z_j, d \rangle - \overline{\alpha}_j,\ j \in \mathcal{J}_i,\ d \in \mathbb{R}^n\}. \tag{1.20}$$

Since $\frac{\mu_i}{2}\|d\|_2^2 = \frac{1}{2} d^T Q d$ with $Q = \mu I$, (1.20) is a quadratic programming problem with some quadratic and some linear variables. This leads to the dual problem

$$[-] \min \{\sum_{j \in \mathcal{J}_i} \overline{\alpha}_j \theta^j + \frac{1}{2\mu_i} \| \sum_{j \in \mathcal{J}_i} z_j \theta^j \|_2^2 \mid \theta \in \Theta\}, \tag{1.21}$$

where $\Theta = \{\theta \in \mathbb{R}_+^{|\mathcal{J}_i|} \mid \sum_{j \in \mathcal{J}_i} \theta^j = 1\}$. Given θ_* the optimal solution of (1.21), $z_*^i = \sum_{j \in \mathcal{J}_i} z_j \theta_*^j$ and $\overline{\alpha}_*^i = \sum_{j \in \mathcal{J}_i} \overline{\alpha}_j \theta_*^j$ the *aggregated subgradient* and *linearization*, such that $v_*^i \in \partial_{\overline{\alpha}_*^i} f(\overline{x}_i)$ [6, (25)]; the KKT conditions allow us to compute the optimal solution of (1.20):

$$d_*^i = -\frac{1}{\mu_i} z_*^i, \quad r_*^i = -\frac{1}{\mu_i}\|z_*^i\|_2^2 - \overline{\alpha}_*^i. \tag{1.22}$$

Chapter 2

Bundle-Enhanced Frank-Wolfe Method

This chapter introduces the proposed bundle F-W method, which integrates elements of BM into the classical F-W framework. The chapter begins by outlining the general methodology, emphasizing how the regularized piecewise-linear approximation of the objective function informs the search direction. The second section extends the model presented in the first section, aiming to incorporate additional information for for more precise results.

2.1 Method Overview

Let $f : \mathcal{X} \to \mathbb{R}$ be a differentiable function, where $\mathcal{X} \subseteq \mathbb{R}^n$ is a compact convex set. The goal is to find a point $x^* \in \mathcal{X}$ that minimizes f, i.e.

$$f_* = f(x^*) = \min_{x \in \mathcal{X}} f(x), \tag{2.1}$$

by running the F-W algorithm. Assume the point x_{k-1} has been computed during the $(k-1)$-th iteration, we now proceed with the k-th iteration of the vanilla F-W algorithm. In this step, the LMO is queried with the direction $d^{k-1} = \bar{g}^{k-1}$, where $\bar{g}^{k-1} := \nabla f(x_{k-1})$ is the gradient of f at x_{k-1}. The LMO returns v_k, an extreme point of the feasible region $\mathcal{X}$. The descent direction $\xi^k = v_k - x_{k-1}$ is then defined, and the next iterate x_k is computed as

$$x_k = x_{k-1} + \gamma_k \xi^k, \tag{2.2}$$

where the step size $\gamma^k \in [0, 1]$ is chosen to maximize the reduction in the objective function along the direction ξ^k. Typically, the *line search* procedure begins with $\gamma^k = 1$ and iteratively refines this value to optimize the decrement. Hence, from a computational standpoint, it is then reasonable to assume that the gradient $g^k := \nabla f(v_k)$, evaluated at the extreme point v_k, is available.

In the subsequent iteration, the LMO would again be queried with the direction $d^k = \bar{g}^k$, , where $\bar{g}^k := \nabla f(x_k)$, to determine the next vertex v_{k+1}. This approach involves approximating the objective function f using its linear first-order expansion at x_k. While this approach is computationally efficient, it does not leverage additional information about the curvature or higher-order behavior of f.

To address this limitation, the aim is to construct a more accurate model of the objective function by incorporating information gathered in previous iterations. Specifically, consider the translated function $h^k(d) = f(x_k + d) - f(x_k)$. The two functions

$$\begin{aligned} &\bar{l}^{k-1}(d) := \bar{g}^{k-1}d - \bar{\alpha}^{k-1} \quad \text{where} \quad \bar{\alpha}^{k-1} := f(x_k) - f(x_{k-1}) - \bar{g}^{k-1}(x_k - x_{k-1}) \geq 0, \\ &l^{k-1}(d) := g^{k-1}d - \alpha^k \quad \text{where} \quad \alpha^k := f(x_k) - f(v_k) - g^{k-1}(x_k - v_k) \geq 0, \end{aligned} \tag{2.3}$$

correspond to the linear first-order expansion of h^k in x_{k-1} and v_k, from which clearly follows $h^k \geq \bar{l}^{k-1}$ and $h^k \geq l^{k-1}$. From convexity it also holds that $h^k(d) \geq \bar{l}^k(d) := \bar{g}^k d$, the first-order expansion of f (hence of h^k) previously mentioned. As a consequence,

$$\underline{h}^k(d) = \max\{\bar{l}^k(d), \bar{l}^{k-1}(d), l^{k-1}(d)\} \tag{2.4}$$

is still a model of h^k, i.e. $h^k(d) \geq \underline{h}^k(d)$ for all $d \in \mathbb{R}^n$. In particular, (2.4) exactly corresponds to the CPM of h^k built with the bundle $\mathcal{B}_k = \{(\bar{g}^{k-1}, \bar{\alpha}^{k-1}), (g^{k-1}, \alpha^k), (\bar{g}^k, 0)\}$.

Based on the classic technics of BM, since (2.4) is a piecewise-linear model and, therefore, generally not differentiable, an additional stabilization term is introduced (see (1.3.1)), resulting in the model

$$\underline{h}_t^k(d) = \underline{h}^k(d) + \frac{1}{2t}\|d\|_2^2, \tag{2.5}$$

where t is the stabilization parameter. In order to determine d^k, we now proceed to solve the problem $\min \underline{h}_t^k(d)$, which can be rephrased as

$$\min\{r + \tfrac{1}{2t}\|d\|_2^2 \quad | \; r \geq \bar{g}^k d, \quad r \geq \bar{g}^{k-1}d - \bar{\alpha}^{k-1}, \quad r \geq g^{k-1}d - \alpha^k\}. \tag{2.6}$$

Given (r_*^k, d_*^k) as the optimal solution of (2.6), the value of d_*^k can be explicitly computed. This is achieved by applying duality theory: the dual problem of (2.6) is

$$\min\{\bar{\alpha}^{k-1}\bar{\theta}^{k-1} + \alpha^k\theta^k + \tfrac{1}{2}t\|\bar{g}^{k-1}\bar{\theta}^{k-1} + g^k\theta^k + \bar{g}^k\bar{\theta}^k\|_2^2 \mid \bar{\theta}^{k-1} + \theta^k + \bar{\theta}^k = 1, \theta \geq 0\}, \tag{2.7}$$

which from (1.22) returns

$$d_*^k = -tz_*^k \quad \text{where} \quad z_*^k = \bar{g}^{k-1}\bar{\theta}^{k-1} + g^k\theta^k + \bar{g}^k\bar{\theta}^k, \tag{2.8}$$

and, given $\alpha_*^k = \bar{\alpha}^{k-1}\bar{\theta}^{k-1} + \alpha^k\theta^k$, it holds $z_*^k \in \partial_{\alpha_*^k} f(x_k)$, i.e.

$$f(y) - f(x_k) \geq \langle z_*^k, y - x_k\rangle - \alpha_*^k \quad \forall y \in \mathbb{R}^n. \tag{2.9}$$

It is important to note that the optimal value of (2.6) is non positive, as the pair $(0, 0)$ is feasible. Furthermore if $d_*^k = 0$, then necessarily $r_*^k = 0$, which implies $z_*^k = 0$ and

$\alpha_*^k = 0$. This is the only possible scenario for $\underline{h}_k^t(d_*^k) = 0$, which ultimately leads to $0 \in \partial f(x_k)$, i.e. x_k is the optimal point.

Since $\bar{g}^k d_*^k = \nabla f(x_k) d_*^k \leq r_*^k < 0$, d_*^k is a descent direction, hence $d^k = z_*^k = -(1/t)d_*^k$ is a valid choice as objective function for the LMO. It is exactly the choice we make in our method, so that the LMO computes

$$v_{k+1} \in \arg\min \{z_*^k x \mid x \in \mathcal{X}\}, \tag{2.10}$$

hence $z_*^k v_{k+1} \leq z_*^k x_k$, which implies $d_*^k \xi^{k+1} \geq 0$. Since $\bar{g}^k d_*^k < 0$, there are good probabilities that also $\nabla f(x_k)\xi^{k+1} < 0$, which would mean that ξ^{k+1} is a descent direction.

We now proceed as in a classical iteration of the F-W method, by calculating the step γ_{k+1} and determining the new iteration $x_{k+1} = x_k + \gamma_{k+1}\xi^{k+1}$. The main steps of the proposed method are summarized in Algorithm 5.

2.1.1 Stabilization parameter tuning

To ensure that ξ^k is a descent direction, it is necessary to carefully adjust the value of t. Specifically, as t increases, the norm term becomes progressively more dominant in the objective function of (2.7). For sufficiently large t, the resulting direction corresponds to the vector with the minimum norm in $conv(\bar{g}^{k-1}, g^{k-1}, \bar{g}^k)$, regardless of the magnitudes of $\bar{\alpha}^{k-1}$ and α^k. In other words, the direction choice becomes independent of the quality of the first-order information provided by $\bar{g}^{k-1}$ and g^{k-1}. Conversely, for smaller values of t, the impact of the linearization errors becomes more pronounced: as $t \to 0$, it is expected that $z_*^k = \bar{g}^k$. Therefore, if ξ^{k+1} does not result in a descent direction, the parameter t can be reduced, and the process can be repeated. This procedure is reminiscent of a NS, where the stability center, x_k in this scenario, does not change. Typically in BM a NS is used to enrich the model with the new information produced, in particular the bundle is informed with the new gradient g^k.

Another issue to consider is that the bundle subproblem does not explicitly include the constraints that define the feasible region. One way to provide the master problem (2.7) with information about how close the current iterate x_k is to the boundary of the feasible region, is to incorporate this information setting the parameter t such that the norm of d_*^k and the norm of ξ^k are comparable. In fact, they respectively represent the step size deemed necessary by the master problem and the step size actually feasible. Consequently, gradually reducing the value of t as the iterate approaches the boundary of the feasible region can be considered an effective strategy to adopt.

2.1.2 Convergence measure

In this context, the dual gap no longer provides an upper bound on the primal gap, as the inequality (1.6) relies on using the gradient as objective function for the LMO. However, it is possible to leverage (2.9) to derive a new bound on the primal gap. Specifically, by taking $y = x^*$, we obtain

$$0 \leq p(x_k) = f(x_k) - f(x^*) \leq \langle z_*^k, x_k - x^* \rangle + \alpha_*^k \leq \langle z_*^k, x_k - v_{k+1} \rangle + \alpha_*^k =: \mathbf{q}(x_k). \tag{2.11}$$

Actually, a tighter estimate of the gap can be achieved. Consider the quantity

$$q^k := f(x_k) + \langle z_*^k, v_{k+1} - x_k \rangle - \alpha_*^k = f(x_k) - \mathbf{q}(x_k). \tag{2.12}$$

From (2.11), it follows that $f(x^*) \geq q^k$ for all k, and thus

$$\underline{\mathbf{f}}^k := \max_{i \leq k} q^i \leq f(x^*) \tag{2.13}$$

represents the best available lower bound on $f(x^*)$ at the k-th iteration. Conversely, the best available upper bound is clearly given by

$$\bar{f}^k := \min_{i \leq k} f(x_i) \geq f(x^*). \tag{2.14}$$

It follows that the best estimate of the gap available at the k-th iteration is

$$w^k = \bar{f}^k - \underline{\mathbf{f}}^k, \tag{2.15}$$

This value serves as stopping criteria for the algorithm: the inequality $w^k \geq \bar{f}^k - f(x^*)$ holds and $\bar{f}^k - f(x^*)$ gives a better estimate of the precision rather the $p(x_k)$, as $\bar{f}^k - f(x^*) \leq p(x_k)$. It follows that when $w^k \leq \epsilon$, where ϵ is the desired precision for the approximation, the optimal solution of (2.1) is given by $\bar{f}^k$, and a minimizer is $x^* \in \arg\min_{i \leq k} f(x_i)$. This approach leverages the best upper and lower bounds obtained across all iterations, which may have been computed in different iterations due to the zigzag phenomenon, rather than relying solely on the values calculated in a single iteration, and thus may be satisfied in less iterations.

Actually, the same concepts can be applied to vanilla F-W. The best upper bound available at the k-th iteration is still $\bar{f}^k$. Similarly as before, leveraging (1.6) the best available lower bound is

$$\underline{f}^k := \max_{i \leq k} \{f(x_i) - q(x_i)\}. \tag{2.16}$$

Consequently, the most accurate approximation of the gap at the k-th iteration is given by $\bar{f}^k - \underline{f}^k$. This value reduces to $q(x_k)$, and therefore induces the classic stopping criteria $q(x_k) < \epsilon$, only when both $\bar{f}^k$ and $\underline{f}^k$ are monotones, which is not necessarily the case, e.g. when a function-agnostic stepsize is used. Therefore, it would be more reasonable to adopt $\bar{f}^k - \underline{f}^k < \epsilon$ as the stopping criteria, as this approach could save unnecessary iterations. Moreover, computing and storing the value of $\bar{f}^k$ and $\underline{f}^k$ have a constant computational cost, since primal and dual gaps are already being computed.

2.1.3 Dual problem solution

Since (2.7) is a quadratic problem in two free variables, it can be solved quite easily. An active-set method only has to enumerate all possible 7 patterns of non-zeroes of the variables (all-zero is non feasible). If only one variable is nonzero, the problem is trivial. If one variable is set to zero, we come down to a problem with only one free variable,

which can be solved with a closed formula. Suppose $\bar{\theta}^{k-1} = 0$ ($\theta^k = 0$ is analogous), the problem becomes

$$\min \{\alpha^k \theta^k + \tfrac{1}{2} t \|g^k \theta^k + \bar{g}^k (1 - \theta^k)\|_2^2 \mid \theta^k \in [0, 1]\}, \tag{2.17}$$

whose optimal solution has the following closed-form expression:

$$\theta_*^k = \min \left\{1, \max \left\{0, \tfrac{-\alpha^k - t\langle g^k - \bar{g}^k, \bar{g}^k \rangle}{t\|g^k - \bar{g}^k\|_2^2}\right\}\right\}. \tag{2.18}$$

Similarly if $\bar{\theta}^k = 0$, the problem reduces to

$$\min \{\bar{\alpha}^{k-1}(1 - \theta^k) + \alpha^k \theta^k + \tfrac{1}{2} t \|\bar{g}^{k-1}(1 - \theta^k) + g^k \theta^k\|_2^2 \mid \theta^k \in [0, 1]\}, \tag{2.19}$$

and the optimal solution is:

$$\theta_*^k = \min \left\{1, \max \left\{0, \tfrac{\bar{\alpha}^{k-1} - \alpha^k - t\langle g^k - \bar{g}^{k-1}, \bar{g}^{k-1} \rangle}{t\|g^k - \bar{g}^{k-1}\|_2^2}\right\}\right\}. \tag{2.20}$$

Lastly, if all the three variables are different from zero, we can consider the problem without the constraint $\theta \geq 0$, reformulated in its vectorized structure:

$$\min \{\alpha^T \theta + \frac{1}{2} t\, \theta^T G \theta \mid e^T \theta = 1\}, \tag{2.21}$$

where $\alpha = [\bar{\alpha}^{k-1}, \alpha^k, 0]$, $\theta = [\bar{\theta}^{k-1}, \theta^k, \bar{\theta}^k]$, $e = [1, 1, 1]$ and G is the *Gram matrix*[1] of the vectors $\{\bar{g}^{k-1}, g^{k-1}, \bar{g}^k\}$. We now consider the 4×4 system given by the KKT conditions

$$\begin{cases} G\theta = -\frac{1}{t}(\alpha + \mu e) \\ e^T \theta = 1 \end{cases}, \tag{2.22}$$

where μ is the forth variable. If the solution θ_* of this system satisfies $\theta_* \geq 0$, then it is the optimal value of (2.7), otherwise it is to be found in the previous cases.

2.2 Generalization of the Model

In the previous section, we present a model that employs three gradients to effectively approximate the behavior of the objective function. While this approach is capable of delivering accurate results, it is possible to further enhance the model's precision and adaptability by incorporating more gradients, potentially all those available. Expanding the number of gradients provides additional directional information, enabling a more refined representation of the function.

Consider x^h for $h \in \mathcal{I}^k$, where $\emptyset \neq \mathcal{I}^k \subseteq \{1, \ldots, k\}$. These represent the iterations used to build the CPM of $h^k(d)$, by considering the linear first-order expansion of $h^k(d)$ in x^h. Actually, given the form of the linear expansions (2.3), rather than the iterates

[1]The Gram matrix of a set of vectors $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ in an inner product space is the $n \times n$ Hermitian matrix of inner products, whose entries are given by $G_{ij} = \langle \mathbf{v}_i, \mathbf{v}_j \rangle$.

x^h, we are interested in a set of pairs (g^i, α^i) such that $g^i \in \partial_{\alpha^i} f(x^k)$, which constitute the bundle $\mathcal{B}^k$. It should be noted that the indices i may not be related with the h (cf. (2.4): some of the approximate subgradients correspond to some x^h, some do not).

Building on the framework introduced in the previous section, we consider the bundle $\mathcal{B}^k = \{(\bar{g}^k, 0)\} \cup \{(\bar{g}^{i-1}, \bar{\alpha}^{i-1}), (g^{i-1}, \alpha^i) \mid i \leq k\}$. This means that the CPM is built using the linear expansion of $h^k(d)$ in all the previous iterates and vertices, other than the current one.
That is, $\underline{h}^k(d)$ is better defined in the general way

$$\underline{h}^k(d) = \max\{g^i d - \alpha^i \mid i \in \mathcal{B}^k\}, \tag{2.23}$$

where the notation $i \in \mathcal{B}^k$ stands for $(g^i, \alpha^i) \in \mathcal{B}^k$. The corresponding primal and dual problems then are

$$\min\{r + \tfrac{1}{2t}\|d\|_2^2 \mid r \geq g^i d - \alpha^i, i \in \mathcal{B}^k\} \tag{2.24}$$

and

$$\min\Big\{\sum_{i\in\mathcal{B}^k} \alpha^i\theta^i + \tfrac{1}{2}t\|\sum_{i\in\mathcal{B}^k} g^i\theta^i\|_2^2 \mid \theta \in \Theta\Big\}, \tag{2.25}$$

with solution

$$d_*^k = -tz_*^k \quad \text{where} \quad z_*^k = \sum_{i\in\mathcal{B}^k} g^i\theta_*^i. \tag{2.26}$$

Given $\alpha_*^k = \sum_{i\in\mathcal{B}^k} \alpha^i\theta_*^i$, z_*^k still have the crucial property $z_*^k \in \partial_{\alpha_*^k} f(x^k)$. Thus, it follows that all considerations about the method presented in the previous section remain applicable using this model, as the properties of the elements are entirely preserved: d_*^k is still a descent direction, and therefore the stabilization parameter tuning and convergence measures remain valid.

2.2.1 Bundle management

Assuming the property $g^i \in \partial_{\alpha^i} f(x^k)$ holds, the selection of the elements in the bundle offers flexibility. From a computational perspective, it is clearly advantageous to choose elements that are inexpensive to evaluate.

A standard strategy is to ensure that $i \in \mathcal{B}^{k+1}$ for all i such that $\theta_*^i > 0$, a procedure known as *selection*. It is important to note that, according to Carathéodory's theorem, there always exists a vector θ_*^i with at most $n+1$ nonzero components. This establishes a finite upper bound on the cardinality of the bundle, ensuring its size remains manageable.

Another technique is suggested by the property $z_*^k \in \partial_{\alpha_*^k} f(x^k)$: this means that the pair (z_*^k, α_*^k) can be inserted into $\mathcal{B}^k$. The aggregated pair (z_*^k, α_*^k) has not been obtained at any iterate x^h, but this is not an issue. The important result [6, sec. 3.2] is that it can actually substitute all the pairs currently into the bundle: if one were to set $\mathcal{B}^{k+1} = \{(z_*^k, \alpha_*^k)\}$, which means forgetting all the other information and just keeping the aggregated pair, then the problem

$$\min\{v + \tfrac{1}{2t}\|d\|_2^2 \mid v \geq z_*^k d - \alpha_*^k\} \tag{2.27}$$

has precisely the same solution d_*^k as (2.24). This allows, at each iteration, to start from an arbitrarily large bundle and *compress* it down to a single element without losing some important properties. Of course, one does not actually want the solution to remain unchanged, but this would only be the case if x^k and t would stay the same, which we do not expect to happen.
Unfortunately, although using this bundle update technique results in a convergent method, its advantage is only theoretical, as the convergence is so slow that the only feasible stopping criterion is a limit on the maximum number of iterations.

Algorithm 5: Bundle F-W Algorithm

Input: f, x_0, t, LMO, step_size, max_iter, ϵ
Output: x^* , v^*

Initialization: x $\leftarrow x_0$, primal $\leftarrow f(x_0)$, FW_gap $\leftarrow$ Inf, upper $\leftarrow$ Inf, lower $\leftarrow$ -Inf.
while $(k = 1 \dots$ *max_ iter*$) \wedge$ (*upper* - *lower* $\geq \epsilon$) **do**
    $z_k, \alpha_k \leftarrow$ bundle(x_k, t)
    $v_k \leftarrow$ LMO(z_k)
    $d^k \leftarrow x_k - v_k$
    if $\langle \nabla f(x), d^k \rangle > 0$ **then**
        $\gamma_k \leftarrow$ step_size(x_k, v_k)
    else
        t tuning
        $\gamma_k \leftarrow 0$
    end
    primal $\leftarrow f(x_k)$
    if *primal* $<$ *upper* **then**
        upper $\leftarrow$ primal
        $x^* \leftarrow x_k$
    end
    FW_gap $\leftarrow \langle z_k, d^k \rangle + \alpha_k$
    dual $\leftarrow$ primal - FW_gap
    if *dual* $>$ *lower* **then**
        lower $\leftarrow$ dual
    end
    $k \leftarrow k + 1$
    $x_{k+1} \leftarrow x_k - \gamma_k \cdot d^k$
end
$z^*, \_ \leftarrow$ bundle(x^*, t)
$v^* \leftarrow$ LMO(z^*)

Chapter 3

Method Implementation

To conduct the experimentation and evaluate both the convergence properties of the proposed method and its competitiveness relative to the vanilla F-W algorithm, the implementation was carried out using the Julia programming language (version v1.11.2). This choice was driven primarily by Julia's high execution speed, achieved through just-in-time compilation, which ensures reliable and efficient performance. Additionally, the availability of specialized optimization libraries, such as `FrankWolfe.jl` and `JuMP.jl`, provided essential tools for the development process.

The primary reference for the implementation was the `FrankWolfe.jl` package, specifically its `frank_wolfe()` function, which serves as the core algorithmic component. Although, theoretically, the only required modification to the F-W algorithm concerned the direction provided to the LMO, direct changes to the function `frank_wolfe`() were necessary. These modifications will be discussed in detail in the following chapter.

Algorithms (3) and (5) present the main steps performed, respectively, by the `frank_wolfe`() function and its modified version, `bundle_frank_wolfe`().

3.0.1 Bundle subproblem

The core component of this algorithm is the implementation of the bundle subproblem. It is passed to the function `bundle_frank_wolfe()` as an argument, so that different models may be implemented. Initially, an alternative approach was explored, in which the bundle function was passed directly as an argument to `frank_wolfe()`, in place of the gradient function. However, this approach proved unsuitable, as the exact gradient is necessary to verify whether the direction d^k is increasing. This lead to the development of a modified version of the original function.

The resolution of the bundle subproblem was implemented through the function `bundle_grad!(storage, x, h, trajectory_arr, t, alpha; use_vertices, tol)` (refer to the full code in (A)). When evaluated at x_k, this function returns the gradient $\nabla f(x_0)$ for the first iteration; otherwise, it proceeds with the following steps.

1. Extracts $x_{k-1}, \ldots, x_{k-h}$ (and $v_{k-1}, \ldots, v_{k-h}$ if `use_vertices=true`) from the vector `trajectory_arr`, which stores information about previous iterations.

2. Computes the relative gradients g^i and linearization errors α^i.

3. Solves the dual problem (2.7) by computing the optimal dual variables θ_*^i.

4. Stores the value of z_*^k in `storage` and append the value of α_*^k to `alpha`.

In particular, step 3 was implemented using JuMP, a domain-specific modeling language in Julia designed for mathematical programming and optimization problem formulation. The specific solver used for this step was `Ipopt` (Interior Point OPTimizer), which is well-suited for solving large-scale nonlinear optimization problems.

To enhance computational efficiency, the problem in (2.7) was formulated in its vectorized form, as shown in (2.21). This formulation significantly accelerates execution by leveraging vector operations, which are more efficient than element-wise operations.

To enable the function `bundle_grad!()` to access information from previous iterations, the *callback* function plays a crucial role. This function tracks the values computed at each iteration and stores them in a vector, making data accessible for subsequent analysis. The use of callback function has already been implemented in `frank_wolfe()`. Moreover, it is possible to pass as argument an external callback function, which stores specific information in a global variable previously defined. In this case, when the callback function is queried, it performs both the storage operations on the global variable and a defined vector within the function scope. We preserved this feature and leveraged it to store the iterates x_i and the extreme points v_i in a global variable, which was passed as the `trajectory_arr` argument to the function `bundle_grad!()`. This approach was made feasible by Julia's support for anonymous functions.

3.0.2 Stopping criteria

As discussed in Section 2.1.2, in the proposed method the dual gap does not provide an upper bound on the primal gap, and therefore cannot serve as stopping criteria. Instead, it is replaced by $\mathbf{q}(x)$. To compute this quantity, the additional term α_*^k is evaluated by the `bundle_grad!()` function and stored in the vector `alpha`, which maintains a record of these values across iterations.

The value $\mathbf{q}(x)$ may be used as stopping criteria, as the standard approach uses the $dual\_gap < \epsilon$ technique. However, this can be improved, since it relies solely on data from the current iteration, while better upper and lower bounds might have been identified during previous iterations.

To address this, the `bundle_frank_wolfe()` algorithm introduces the variables `upper` and `lower`, which store $\bar{f}^k$ and $\underline{\mathbf{f}}^k$, respectively. These represent the best upper and lower bounds found up to the current iteration. The stopping criterion is thus reformulated as $w^k = \bar{f}^k - \underline{\mathbf{f}}^k < \epsilon$. This adjustment prevents unnecessary iterations when the best bounds are identified in different iterations rather than in the same one.

Additionally, a maximum iteration limit, passed as an argument to the function, is implemented. This provides greater control over the algorithm and avoids infinite loops in challenging problems where convergence is poor.

3.0.3 Numerical errors management

During the implementation, it became necessary to address numerical errors that arose in variables theoretically constrained to be non-negative but which, due to computational approximations, occasionally assumed slightly negative values.

Specifically, this issue was observed in the linearization errors α^i, the scalar product $\langle z_*^k, d^k \rangle$ and the dual solutions θ_*^i. The errors in the former two variables can be attributed to numerical inaccuracies in the computation of dot products, particularly when working with small numbers. Instead, errors in the dual solutions are linked to the behavior of the solver, which employs slightly relaxed constraints compared to the initial specifications to improve convergence.

These numerical discrepancies were managed by defining a tolerance threshold $\eta = 10^{-12} * \max(|f_*|, 1)$. Values falling within the range $[-\eta, 0)$ were manually set to zero. Furthermore, an error message is generated if values below $-\eta$ are encountered, as this could indicate underlying issues in the algorithm. An additional restriction was imposed on the `Ipopt` solver, responsible for computing the values of θ_*^i: the parameter `bound_relax_factor`, the factor used for bounds relaxation, was set to 10^{-12} (its default value is 10^{-8}).

Chapter 4

Computational results

This chapter presents the results of the experimentation conducted to analyze the performance of the proposed method. The metrics used for evaluation primarily include the number of iterations required for convergence, the progression of the gap $\mathbf{q}(x_k)$ compared to the dual gap $q(x_k)$ of the F-W method, and the corresponding values of the running gaps.

4.1 Experimental setup

The experiments were conducted on the objective function $f(x) = \|x - x_p\|_2^2$, where x_p is a fixed point generated according to specific criteria. In particular, the study focused on scenarios where x_p is infeasible. This function was chosen due to its simplicity: it is non-negative, its unconstrained minimum is at x_p, and its level sets form concentric circles centered at x_p. These features facilitated precise control over the iteration behavior, enabling the straightforward identification of potential errors or issues — an especially advantageous characteristic for a preliminary exploration of the method.

To control the function values during the iterations and at the optimum, the feasible region was generated within a Euclidean ball of radius 5, and the point x_p was randomly generated with a norm in the interval $(5, 7.5)$. The maximum iterates number was set to 200 and the required precision was $\epsilon = 10^{-8}$.

Two types of feasible sets were considered: an l_2-norm ball centered at the origin and a polytope defined by randomly generated vertices. The implementation of each configuration was simplified by the `FrankWolfe.jl` library, where the abstract type `LinearMinimizationOracle` is defined. Specifically, the library provides the structures `LpNormLMO{p}(radius)`, which implements the LMO for a p-norm ball of radius `radius`, and `ConvexHullOracle(vertices)`, which implements the LMO for a polytope constructed from the vertices provided in the `vertices` array.

For each feasible set, three types of step sizes were tested: short step, function-agnostic and line search. These methods are implemented in `FrankWolfe.jl` via the abstract type `LineSearchMethod`. Specifically, the library includes `Shortstep(L)`, where `L` is the Lipschitz constant of the gradient, `Agnostic()`, which computes the simple

rule $\gamma_k = 2/(k+2)$, and `Goldenratio(tol)`. The latter implements the golden-section search based on [8], where `tol` represents the precision used to compute the minimum on the considered segment. The default value of `tol` is 10^{-7}, which, for the standards of this work, was found to be too high. Therefore, `tol` was set to 10^{-10}.

4.2 Performance Evaluation

The first parameter considered in the experiments observations is the problem size. Although this dimension influenced the absolute execution time of the methods, it did not significantly affect their comparative performance. This suggests that the relative efficiency of the methods remains consistent across different problem dimensions. Furthermore, the execution time is not a focus of this study, so all results presented correspond to problems with a fixed size of $n = 100$, allowing for a clearer focus on other factors.

A more pronounced impact was observed concerning the choice of the stabilization parameter. This parameter plays a pivotal role in shaping the performance outcomes, underscoring its importance in the analysis. The details of its influence and the insights derived are explored in the subsequent sections, highlighting its contribution to the observed results.

The values displayed in the graphs throughout this chapter will illustrate four quantities as they vary with the iterations. Specifically, *Normalized Primal* and *Running Primal Gap* refer, respectively, to the quantities

$$\frac{p(x_k)}{\max\{|f_*|,1\}}, \quad \frac{\bar{f}^k - f_*}{\max\{|f_*|,1\}},$$

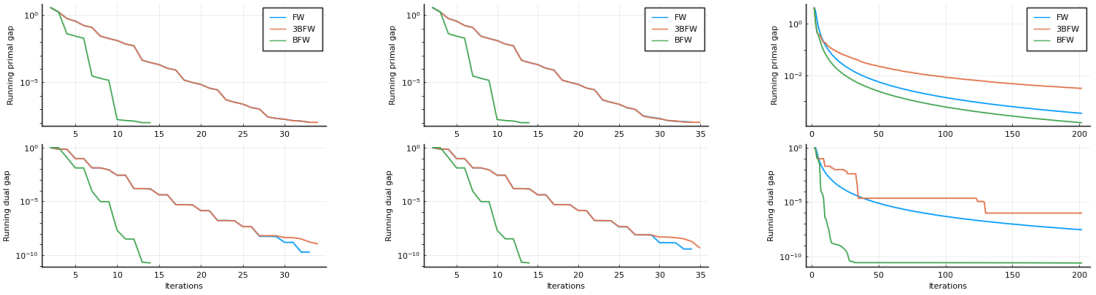
while *Normalized FW Gap* and *Running Dual Gap* refer to

$$\begin{aligned} &\frac{q(x_k)}{\max\{|f_*|,1\}}, \quad \frac{f_* - \underline{f}^k}{\max\{|f_*|,1\}} && \text{for the vanilla F-W method,} \\ &\frac{\mathbf{q}(x_k)}{\max\{|f_*|,1\}}, \quad \frac{f_* - \underline{\mathbf{f}}^k}{\max\{|f_*|,1\}} && \text{for the two bundle F-W methods.} \end{aligned}$$

We will refer to the vanilla F-W method as FW, to the bundle F-W method introduced in Section 2.1 as 3BFW and to the bundle F-W method with the model proposed in Section 2.2 as BFW.

4.2.1 Fixed stabilization parameter

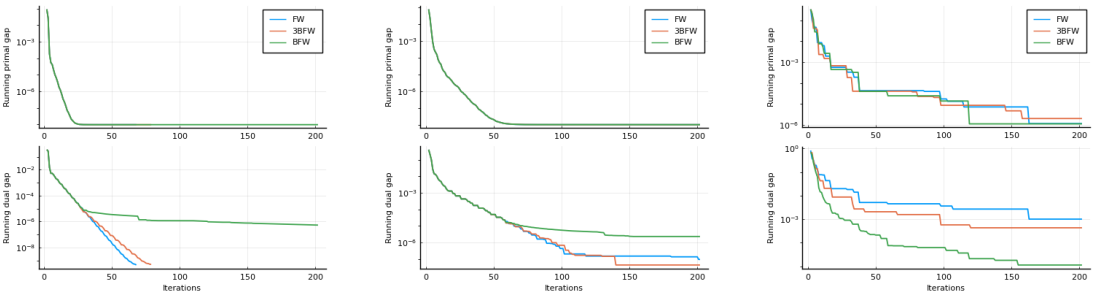
A preliminary phase of research was conducted on a fixed stabilization parameter value. Although no online tuning was performed of t, excellent results were observed on the Euclidean ball domain for values of $t = 10^2$ or slightly higher. As shown in the instance in Figure 4.1, for all three types of step sizes BFW produced both primal and dual gaps equal or smaller than those of FW. Moreover, using line search and short step methods, the stopping criteria was satisfied in less than half the iterations compared to FW.



Short step | Line search | Function-agnostic

Figure 4.1: Normalized running gaps on Euclidean ball with fixed $t = 10^3$.

Not as optimal results were obtained on polytope domain, where the behavior highly depended on the specific instance. As the number of vertices increases, the three algorithms tend to follow the same pattern, eventually coinciding. However, generally the three step size methods lead to significantly distinct behaviors, as shown on the instance in Figure 4.2.



Short step, $t = 10^{-1}$ | Line search, $t = 10^{-3}$ | Function-agnostic, $t = 10^3$

Figure 4.2: Normalized running gaps on a polytope with 7 random vertices.

Short step is the only method for which no instances were found where one between 3BFW and BFW manages to improve the gaps provided by FW, even with significant variations in the initial value of t. While the primal gap generally aligns, the dual gap tends to be relatively larger, especially for BFW.

Line search instead leads to highly variable outcomes. Even for the same value of t, three distinct behaviors of BFW were observed: in some instances, all three algorithms produce the same output. In others, BFW exhibits a much larger dual gap, despite the primal being identical to others. Finally, in certain cases both gaps produced by BFW were higher then the ones of the other methods. In this configuration, 3BFW tends to perform a lot better better. Generally, its behavior closely mirrors that of FW; however, in instances where FW stagnates in a continuous oscillation without converging, 3BFW provides a lower dual gap, like in the instance represented in Figure 4.2.

The only step size strategy to generally outperform FW on polytope domain is the function-agnostic step size. With an initial value of $t = 10^2$, 3BFW, and even more so BFW, achieve lower dual gaps compared to FW. It should be noted that these values

are not directly comparable to those obtained using different step sizes with the same number of iterations, as the latter lead to faster convergence. Regardless, for instances where the previous step sizes are too costly to employ, using BFW with a higher number of iterations seems to be a promising solution.

4.2.2 Tuning of t

As is well known in BM theory, the online tuning of t is crucial. However, for the same reason, adopting the wrong strategy can make things a lot worse.

For the experimentation we mainly focused on adapting the value of t when a NS occurred. An additional rule was also tested, in order to keep the norms of d_*^k and ξ^k comparable, as discussed in Section 2.1.1. The experimentation was conducted on the following strategies:

(a) NS: $t \leftarrow \min(10t, 10^8)$;

(b) NS: $t \leftarrow \min(10t, 10^{12})$;

(c) NS: $t \leftarrow \min(10t, 10^{12})$

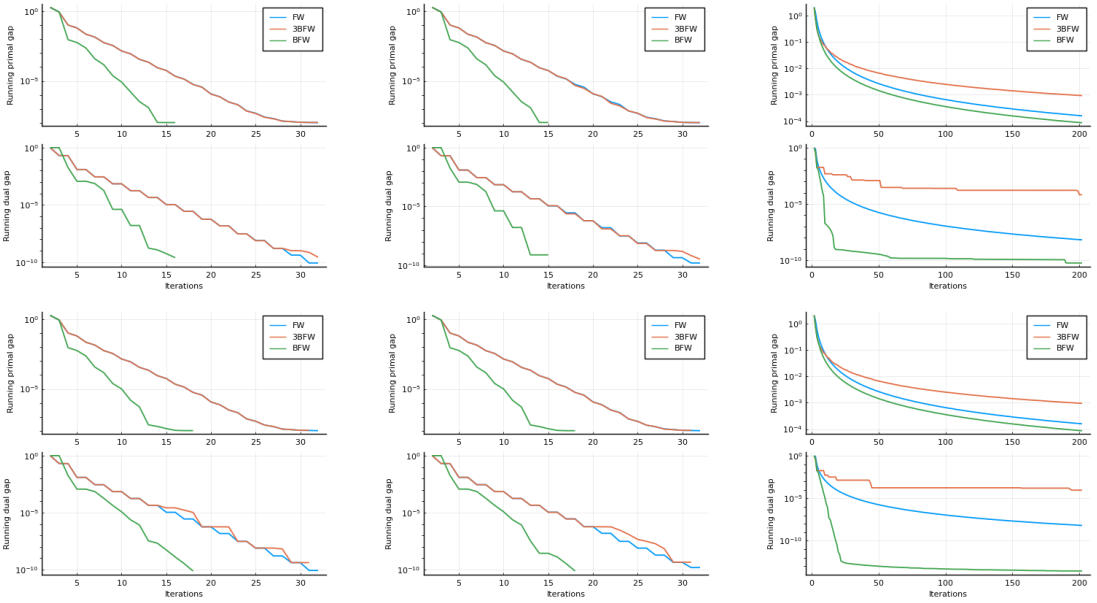
$$\text{SS: } \begin{cases} \text{if } \frac{\|d^k\|}{\|z_k\|} < 10^{-1} \text{ then } t \leftarrow \min(10t, 10^{12}) \\ \text{if } \frac{\|d^k\|}{\|z_k\|} > 10 \quad \text{then } t \leftarrow \max(t/10, 10^{-12}) \end{cases},$$

where d^k and z_k are referred to the notation used in Algorithm 5.

Regarding the Euclidean ball domain, the promising results obtained with a fixed parameter value were preserved. It should be noted, however, that due to convergence occurring in only a few iterations and the limited number of NS, parameter tuning through methods (a) and (b) does not significantly impact the progression of the iterations. Consequently, starting with an initial value of $t = 10^2$, the gaps values obtained with these tuning strategies result equal to the ones produced with a fixed t.

Conversely, improvements were observed with the use of method (c): incorporating tuning during the SS allows the parameter t to be continuously informed on the distance of the current iterate from the boundary of the feasible region. For BFW, this resulted in lower dual gaps, particularly when using function-agnostic step size. Figure 4.3 compares the results on the same instance of the algorithms without tuning and with the application of method (c). Unfortunately, 3BFW does not result competitive in this scenario.

Regarding the polytope domain, the tuning methods did not generally lead to 2BFW and BFW algorithms outperforming FW. However, improvements were observed compared to the use of a fixed t when using short step and line search step sizes. Specifically, with all three tuning strategies considered, initializing $t = 10$ or lower the gap progression of the proposed methods closely resembled that of FW, while in the previous case, particularly for BFW, the dual gap often remained significantly higher due to the presence of NSs. Nonetheless, these results are encouraging. For example, Figure 4.4 illustrates an instance where methods (a) and (b) enabled both 2BFW and BFW to

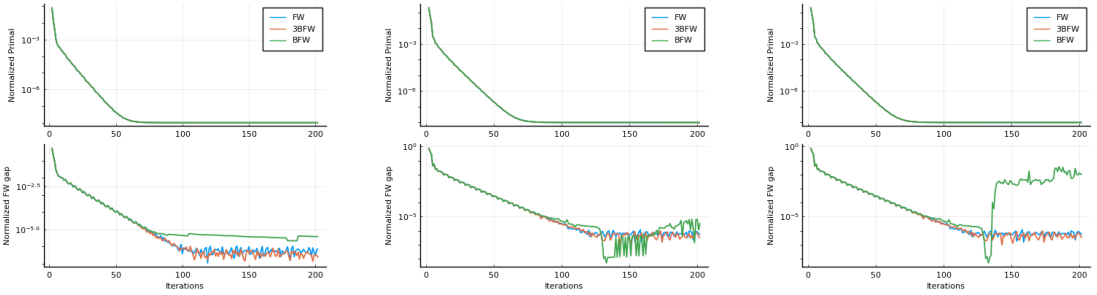


Short step Line search Function-agnostic

Figure 4.3: Normalized running gaps on Euclidean ball with starting value $t = 10^2$. On the first row: fixed t value. On the second row: (c) tuning strategy.

achieve a lower dual gap than FW when using a line search step size - a behavior not observed in the same instance with a fixed t.

Notably, the smaller bound on the value of t used in method (a) effectively mitigated the oscillations in the dual gap, justifying the investigation of the two methods separately.



Fixed t Method (a) Method (b)

Figure 4.4: Normalized primal and dual gaps on polytope domain, line search step size with different t tuning strategies.

The performance of the BFW and 3BFW algorithms with the function-agnostic step size remains effective, as they consistently provide lower dual gap values compared to those achieved by FW. However, on the same instance, the application of the proposed tuning strategies generally leads to a worse performance, resulting in a slower convergence.

Conclusions and Future Work

We have developed a new variant of the vanilla F-W method with a proximal bundle approach. Its main characteristic is to exploit additional information retrieved during previous iterations in order to determine a better search direction to use as objective of the LMO, rather than the bare gradient. This is obtained by developing a piecewise-linear model of the objective function, that becomes more and more refined as the iterations increase. The possibility to choose which information are used to build the model leaves space for numerous variants, two of which were considered in this work.

In order to validate the proposed method and evaluate its performance, the bundle F-W algorithm was implemented and tested in a controlled setup. The objective was to start identifying the scenarios in which it performs better and the observed results are promising. Remarkably, a significant improvement was noted in comparison to the classical F-W method when applied on a Euclidean ball as the domain. Although the polytope domain proved to be more challenging, certain findings suggest that a more in-depth investigation is warranted.

Undoubtedly, extending the experimentation to more complex functions and to other domains would provide greater clarity regarding the contexts in which the bundle F-W method performs optimally. There is considerable room for exploration in terms of potential tuning methods for the stabilization parameter. More sophisticated update rules could be attempted, particularly in distinguishing between NS and SS. It is also worth investigating whether introducing a counter to mitigate the variation of the parameter value might prove to be a useful technique, allowing the method to adapt.

A theoretical study about convergence would be highly beneficial in this regard. This not only would serve to unequivocally validate the method, but it would probably yield inequalities that constrain the possible values of the stabilization parameter, potentially providing direct methods for its update.

Another potential area for development concerns the model employed in the bundle subproblem. In this work, the proposed model uses proximal stabilization, but other techniques could be employed. One example is the trust region stabilization, whose theoretical framework is introduced in Appendix B.

In conclusion, this work has presented an exploratory research on the proposed bundle F-W method, demonstrating the potential it holds. The results obtained paves the way for future research. A deeper exploration will be key to fully establishing the method's efficacy and broadening its impact.

Appendix A

Implemented code

bundle_grad!()

```
function bundle_grad!(storage, x, h::Int, trajectory_arr, t, alpha;
↪  use_vertices=true,tol=1)
#first iteration
if isempty(trajectory_arr)
    gradient!(storage, x)
    push!(alpha, 0)

#bundle subproblem
else
    #extraction of past h iterates and h vertices (if the case)
    h = min(length(trajectory_arr),h)
    h_iters = trajectory_arr[end-h+1:end]
    bundle_x = getindex.(h_iters,1)
    if use_vertices
        bundle_v = getindex.(h_iters,2)
    end
    #computation of gradients and linear errors
    bundle_grad = []
    bundle_alpha = []
    obj = objective(x)
    for i in 1:h
        push!(bundle_grad, 2 * (bundle_x[i] - xp))
        push!(bundle_alpha, obj - objective(bundle_x[i]) - dot(bundle_grad[end],
        ↪  x-bundle_x[i]))
        if use_vertices
            push!(bundle_grad, 2 * (bundle_v[i] - xp))
            push!(bundle_alpha, obj - objective(bundle_v[i]) - dot(bundle_grad[end],
            ↪  x-bundle_v[i]))
        end
    end
    push!(bundle_grad, 2 * (x - xp))    # adding gradient(x) at the end
    push!(bundle_alpha, 0)
    bundle_dim = length(bundle_grad)
    tolerance = 1e-12 * tol
    #check on negative values of alpha
```

```
33      for i in 1:bundle_dim
34          if bundle_alpha[i] < -tolerance        # small negative tolerance
35              @warn("[$h-bundle] alpha[$i]: $(bundle_alpha[i]) < 0 (violates tolerance
                ↪  -$tolerance)")
36          elseif bundle_alpha[i] < 0
37              bundle_alpha[i] = 0                # clip small negatives to zero
38          end
39      end
40
41      #Gram matrix
42      G = zeros(bundle_dim,bundle_dim)
43      for i in 1:bundle_dim
44          for j in 1:bundle_dim
45              G[i,j] = dot(bundle_grad[i], bundle_grad[j])
46              if i != j
47                  G[j,i] = G[i,j]
48              end
49          end
50      end
51
52      #bundle model
53      model = Model(Ipopt.Optimizer)
54      set_optimizer_attribute(model, "bound_relax_factor", 1e-12)
55      set_silent(model)
56      @variable(model, theta[1:bundle_dim] .>= 0)
57      @constraint(model, sum(theta) == 1)
58      @objective(model, Min, dot(bundle_alpha,theta) + 0.5*(t[end])*dot(theta, G *
        ↪  theta))
59      optimize!(model)
60      Theta = value.(theta)       #solutions
61
62      #check on negative values of theta
63      for i in 1:bundle_dim
64          if Theta[i] < -tolerance     # small negative tolerance
65              @warn("[$h-bundle] theta[$i]: $(Theta[i]) < 0 (violates tolerance
                ↪  $tolerance)")
66          elseif Theta[i] < 0
67              Theta[i] = 0  # Clip small negatives to zero
68          end
69      end
70      if sum(Theta) < 1-tolerance || sum(Theta) > 1+tolerance
71          @warn("[$h-bundle] theta does not sum up to 1: $(sum(Theta)) (tolerance=
            ↪  $tolerance)")
72      end
73
74      #store values of alpha
75      push!(alpha, sum(bundle_alpha[i] * Theta[i] for i in 1:bundle_dim))
76
77      #final direction
78      z = sum(bundle_grad[i] * Theta[i] for i in 1:bundle_dim)
79      @. storage = z
80  end
81  return nothing
```

Appendix B

Trust region stabilization

A very intuitive approach for stabilization is to constrain the iterate to a *Trust Region* (TR), by solving the *stabilized* MP

$$x_{i+1} \in \arg\min \{\check{f}^i(x) \mid \|x - \overline{x}_i\| \leq \delta_i\}, \tag{B.1}$$

where δ_i is the stabilization parameter, that regulate the radius of the trust region. Usually the infinity norm is used, because rewriting explicitly the model $\check{f}^i$, the MP (B.1) results being an LP.

In order to update $\overline{x}_i$ and δ_i correctly, rules need to be defined. For the stabilization parameters, a boundedness condition $0 < \underline{\delta} \leq \delta_i \leq \overline{\delta} < \infty$ is sufficient. For the stability centers instead, we consider the condition

$$f(x_{i+1}) \leq f(\overline{x}_i) + m(\check{f}^i(x_{i+1}) - f(\overline{x}_i)) \quad \equiv \quad \Delta f^i \leq m\Delta^i \tag{B.2}$$

where $m \in (0,1)$ is fixed, $\Delta f^i := f(x_{i+1}) - f(\overline{x}_i)$ and $\Delta^i := \check{f}^i(x_{i+1}) - f(\overline{x}_i) \leq 0$. If $\Delta^i = 0$, then $\overline{x}_i$ is optimal for the MP. The condition (B.2) regulate the choice of $\overline{x}_{i+1}$: if the inequality holds, it is reasonable to set $\overline{x}_{i+1} = x_{i+1}$, which is usually called *Serious Step* (SS). Alternatively, the stability center is kept unchanged, i.e. $\overline{x}_{i+1} = \overline{x}_i$, which is called *Null Step* (NS). This results in $\{f(\overline{x}_i)\}$ being a decreasing sequence.

TR dual problem

Shen and Gao examined the TR Lagrange dual problem in [9], making the assumption that the norm in (B.1) is induced by a scalar product. However, it is possible to obtain a higher level of generalization by introducing the following operator.

Definition 11 (Conjugate function)**.** Given $f : \mathbb{R}^n \to \mathbb{R}$, the *conjugate function* $f^* : \mathbb{R}^n \to \mathbb{R}$ is defined as

$$f^*(y) = \sup_{x \in \mathrm{dom} f} (\langle y, x \rangle - f(x)).$$

The domain of the conjugate function consists of all $y \in \mathbb{R}^n$ for which the supremum is finite, i.e. for which the difference $\langle y, x\rangle - f(x)$ is upper bounded. The function f^* is always convex, since it is the point-wise supremum of a family of convex functions of y, and this is true whether or not f is convex.

The conjugate operator allows to generalize Lagrangian duality as it follows. Given two functions $f_1(x)$ and $f_2(x)$ closed convex such that the intersection of their domains is nonempty, the *Fenchel's duality* property holds:

$$\inf_x \{f_1(x) + f_2(x)\} = \inf_z \{f_1^*(z) + f_2^*(-z)\}. \tag{B.3}$$

Thus, when considering the *Generalized BM* (GBM) [6, para. 3.4], i.e. the MP

$$d^i \in \arg\min \{\underline{f}^i(\overline{x} + d) + D_{\mu^i}(d)\}, \tag{B.4}$$

where $D_{\mu^i}(d)$ is referred to as *generalized stabilization term*, it is possible to immediately obtain its Fenchel's dual

$$z^i \in \arg\min \{(\underline{f}^i)^*(z) + \langle z, \overline{x}\rangle + D^*_{\mu^i}(-z)\}. \tag{B.5}$$

In particular, when $\underline{f} = \check{f}$

$$(\check{f}^i)^*(z) = \min \{\sum_{j\in\mathcal{J}_i} \alpha_j\theta_j \mid \sum_{j\in\mathcal{J}_i} z_j\theta_j = z, \theta \in \Theta\}.$$

In this way, a proper choice of D_μ allows to derive the Fenchel's dual for the TR MP (B.1):

$$D_\delta(d) = 1_{\{d\,:\,\|d\|\le\delta\}} \quad D^*_\delta(z) = \delta\|z\|_*,$$

where

$$1_C(d) = \begin{cases} 0 & \text{if } d \in C \\ \infty & \text{if } d \notin C \end{cases}$$

is the indicator function of the set C and $\|.\|_*$ is the dual norm of $\|.\|$, i.e. $\|z\|_* = \sup\{\langle z, x\rangle \mid \|x\| \le 1\}$. Note also that $D_\mu(d) = \frac{\mu}{2}\|d\|_2^2$ and $D^*_\mu(v) = \frac{1}{2\mu}\|z\|_2^2$ immediately reproduces the PBM Lagrange dual problem (1.21), paying attention to switch from α_j to $\overline{\alpha}_j$ since the model is translated.

Bibliography

[1] Gábor Braun et al. "Conditional Gradient Methods". In: 2022. URL: https://api.semanticscholar.org/CorpusID:254018302.

[2] A.M. Bagirov, Napsu Karmitsa, and Marko Mäkelä. *Introduction to Nonsmooth Optimization: Theory, Practice and Software*. Springer Cham, Aug. 2014, pp. 299–308. ISBN: 978-3-319-08113-7. DOI: 10.1007/978-3-319-08114-4.

[3] Marguerite Frank and Philip Wolfe. "An algorithm for quadratic programming". In: *Naval Research Logistics Quarterly* 3 (1956), pp. 95–110. URL: https://api.semanticscholar.org/CorpusID:122654717.

[4] Sebastian Pokutta. "The Frank-Wolfe Algorithm: A Short Introduction". In: *Jahresbericht der Deutschen Mathematiker-Vereinigung* 126.1 (Mar. 2024), pp. 3–35. ISSN: 1869-7135. DOI: 10.1365/s13291-023-00275-x. URL: https://doi.org/10.1365/s13291-023-00275-x.

[5] Martin Jaggi. "Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization". In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 1. Atlanta, Georgia, USA: PMLR, June 2013, pp. 427–435. URL: https://proceedings.mlr.press/v28/jaggi13.html.

[6] Antonio Frangioni. *Standard Bundle Methods: Untrusted Models and Duality*. Tech. rep. Dipartimento di Informatica, Università di Pisa, 2019.

[7] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[8] Cyrille W. Combettes and Sebastian Pokutta. *Boosting Frank-Wolfe by Chasing Gradients*. 2020. arXiv: 2003.06369 [math.OC]. URL: https://arxiv.org/abs/2003.06369.

[9] Jie Shen and Ya-Li Gao. "Research on the Dual Problem of Trust Region Bundle Method". In: *Journal of Advances in Mathematics and Computer Science* 22.3 (May 2017), pp. 1–6. DOI: 10.9734/BJMCS/2017/33880.

[10] E.S. Levitin and B.T. Polyak. "Constrained minimization methods". In: *USSR Computational Mathematics and Mathematical Physics* 6.5 (1966), pp. 1–50. ISSN: 0041-5553. DOI: https://doi.org/10.1016/0041-5553(66)90114-5. URL: https://www.sciencedirect.com/science/article/pii/0041555366901145.