On Variants of Sliding and Frank-Wolfe Type Methods and Their Applications in Video Co-localization

Seyed Hamid Nazari
Clemson University

On Variants of Sliding and Frank-Wolfe Type Methods and Their Applications in Video Co-localization

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Mathematics

by
Hamid Nazari
December 2022

Abstract

In this dissertation, our main focus is to design and analyze first-order methods for computing approximate solutions to convex smooth optimization problems over certain feasible sets. Specifically, our goal in this dissertation is to explore some variants of sliding and Frank-Wolfe (FW) type algorithms, analyze their convergence complexity, and examine their performance in numerical experiments.

We achieve three accomplishments in our research results throughout this dissertation. First, we incorporate a linesearch technique to a well-known projection-free sliding algorithm, namely the conditional gradient sliding (CGS) method. Our proposed algorithm, called the conditional gradient sliding with linesearch (CGSls), does not require the knowledge of Lipschitz constant of the gradient of objective function, which is critical in the numerical implementation of the CGS method. Second, we explore the possibility of designing a bundle level type version of the CGS method, which to the best of our knowledge has not yet appeared in the literature. Our proposed sliding APL (SAPL) method achieves the same complexity to the CGS method. Third, we study numerical algorithms for solving the image co-localization problem. For this problem, we propose new variants of the Frank-Wolfe (FW) method and compare their empirical performance with other existing methods.

The dissertation is organized as follows. In the first chapter, we review some projection-based and projection-free algorithms, their variants, and their respective advantages and disadvantages. Several useful definitions, theorems, and lemmas are also introduced in this chapter that will be utilized throughout the dissertation. For completeness, we prove most of the known results listed in this chapter (proof deferred to the appendix). In the second chapter, we incorporate a linesearch technique to the well-known CGS method and propose the CGSls method. We show that the proposed CGSls method converges with similar complexity to the CGS method. We also examine the performance of the proposed algorithm by comparing it to the CGS method and other projection-free algorithms. In the third chapter, we explore the possibility of designing a bundle level type variant of the CGS method. The proposed SAPL method is inspired by previous literature on bundle level type method. Such bundle level type method has not yet appeared in any literature on sliding algorithms. We show that the proposed SAPL method converge with the same order of complexity as the CGS and CGSls methods. In the fourth chapter, we apply the algorithms studied in previous chapters to the well-known video co-localization problem. We also propose new variants of the FW method and compare their empirical performance with other numerical methods.

Acknowledgments

I would like to close this chapter of my academic life by expressing my profound appreciation to the people I owe this happy ending, those who have made this journey and its book possible.

I want to give my warmest thanks to my adviser Dr. Yuyun "Lance" Ouyang, who was not only my supervisor but also my leader and my close friend. His dominance and the vast range of knowledge in various areas provided the best possible guidance in my Ph.D. career. I want to thank him for all his superior and kind support. I will never forget the favors he did for me, which were beyond his responsibilities.

I also want to thank the support, help, and guidance of my committee members Dr. Billy Bridges, Dr. Kai Liu, and especially Dr. Boshi Yang, for his availability and generosity in sharing his knowledge. Dr. Yang is truly an asset to the school of math. Many thanks to the School of Mathematical and Statistical Sciences for providing such high-quality faculty and staff group. To Dr. Akshay Gupte, who taught me my first-ever optimization course and supported me as a research adviser for more than two years. I also took advantage of outstanding professors in our department Dr. Margaret Wiecek, Dr. Mathew Saltzman, Dr. Fei Xue, Dr. Brian Fralix, Dr. Leo Rebholz, Dr. Shitao Liu, Dr. Kevin James, Dr. Keri Sather Wagstaff, Dr. Chriss McMahan, Dr. Brook Russel, and other professors in school of math.

I have been pleased to have friends and colleagues in graduate school with whom I spent most of my time on campus. Special thanks to my best friends, Amy Burton Gore, for all homework problem-solving we did together and, more importantly, for inviting me to her wedding, Brandon Lumsden and Yuan Yang, for the exceptional and memorable times I spent beside them. Many thanks to Trevor Squires (for all his help in research), Pitter Westerban (for his loud laughter), Luke Duncan (for being a very kind friend, and also for his Keurig coffee maker), James Gossel (for all the hiking he managed), Cameron Arnold (for his intelligence and our problem-solving), Alan Hahn (for being the nicest guy in our department), and Andrew Panagia (for being the smartest person I've ever met), and tens of other indelible friends.

Finally, I thank my family for their extraordinary support. To my brother Hadi Nazari who is, in fact, the main reason I started graduate school. To my mother, who taught me to be strong, patient, and to love. To my wife, Nafis Ebrahimi, for all her support and understanding when undertaking my research. Without her, this could not start and end.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

With the significant increase in the application of data science in real-world problems, optimization, as one of the critical components of machine learning techniques, becomes increasingly important. Mathematical modeling of any research problem, choosing the proper method to solve the mathematical model, and comparing different models and their convergence plays an essential role in optimization. Although various problems would have different objectives and constraints, understanding the performance of possible methods to solve problems with specific objectives and constraints and improving them in their convergence rate and ion practice is as vital as designing new algorithms.

Among all of the existing methods to solve optimization problems, first-order methods, because of their particular characteristics, have been of interest to many researchers. Generally, first-order methods require relatively small amount of information to converge and have cheap iteration costs. Although higher-order methods, such as Newton's method, converge in less number of iterations to the optimal solution or an approximately optimal solution, the cost per iteration of such algorithms could increase drastically as the dimension of the problem increases.

In this dissertation, our focus will be on some of the most well-known first-order methods to find an approximate solution for special kinds of constraint optimization problems. More precisely, our ultimate goal in this document is to find a solution x^* for the convex optimization problem

$$f^* := \min_{x \in \mathcal{X}} f(x). \tag{1.1}$$

Accepting that we may only numerically compute approximately optimal solutions, with accuracy threshold ϵ we would like to compute an ϵ-approximate solution $\hat{x}$ such that $f(\hat{x}) - f^* \leq \epsilon$. In this setting the feasible region $\mathcal{X} \subseteq \mathbb{R}^n$ is a convex compact set and $f : \mathbb{R}^n \to \mathbb{R}$ is a smooth convex function. We assume that the gradient function $\nabla f(\cdot)$ is Lipschitz continuous (with respect to the norm $\|\cdot\|$) with Lipschitz constant $L > 0$, namely

$$\|\nabla f(x) - \nabla f(y)\|_* \leq L \|x - y\|, \ \forall x, y \in \mathbb{R}^n, \tag{1.2}$$

where $\|\cdot\|$ is any norm and $\|\cdot\|_*$ is its dual norm, defined by

$$\|y\|_* := \sup_{\|x\| \leq 1} \langle x, y \rangle$$

for $y \in \mathbb{R}^n$. Also, we define the diameter of the set $\mathcal{X}$ as

$$D_{\mathcal{X}} \equiv D_{\mathcal{X}, \|\cdot\|} := \max_{x, y \in \mathcal{X}} \|x - y\|. \tag{1.3}$$

In this chapter, as an introduction, we will introduce and review some of the most well-known methods to solve problem (1.1). These methods are the foundation of future chapters, and reading through the algorithms and their properties would help grasping the ideas of the proposed methods. This chapter consists of three major sections. In Section 1.1, we will briefly discuss projection-based methods and their pros and cons to solve problem (1.1). In Section 1.2, we will introduce and discuss some of the most famous projection-free methods and their properties that are the basis of our proposed methods appearing later in Chapters 2 and 4. Finally, in Section 1.3, we will introduce bundle-level type methods that are foundations for our proposed method appearing later in Chapter 3. Note that the details about the convergence analysis of the algorithms discussed in this chapter are provided in the Appendix A.

The rest of this dissertation is organized as follows. In chapter 2 of this dissertation, inspired by a sliding type method introduced in [30], we will propose an algorithm that addresses an issue with the existing method. We will also discuss in detail the convergence analysis of our proposed algorithm. Moroever, we will run numerical experiments to compare the performance of the proposed algorithm to some similar algorithms.

In Chapter 3, inspired by a bundle-level type method in [26], we will propose a bundle-level variant of our proposed algorithm in Chapter 2. We will also provide detailed proof of the convergence of our proposed algorithm. We will show that the convergence rate is maintained and is the same as that of Chapter 2.

Finally, we will dedicate chapter 4 to a practical problem. In this chapter, we will introduce the mathematical programming model of the problem and will discuss how the methods we have learned in the first three chapters apply to this model. We will also propose two new algorithms to solve this model inspired by the method we proposed in chapter 2. We will show in numerical results the emperical advantage of the two proposed algorithms.

1.1 Projection-Based Algorithms

Suppose that we are trying to find an approximate solution for an instance of the problem (1.1) using the gradient descent method. Recall that in each iteration of the gradient method the current point moves toward the negative gradient with some step size. There will not be any issue in the case where the feasible region $\mathcal{X}$ is $\mathbb{R}^n$, i.e., when our problem is unconstrained. However, if $\mathcal{X}$ is a subset of $\mathbb{R}^n$ but not equal to it, there is no guarantee that in each iteration the gradient descent method maintains feasibility. To resolve this issue the natural concept is to apply a projection to $\mathcal{X}$ after each iteration to maintain feasibility and then continue with the new projected point. Projection-based algorithms are of the type that need to solve a projection problem as their subproblems. The projections that appear in these algorithms might be different depending on the problem structure. Of most well-known projection based algorithms are *projected gradient descent* (PGD) and *Nesterov's accelerated gradient descent* (NAGD) [35] methods. Here we describe these algorithms and their convergence complexity.

1.1.1 Projected Gradient Descent Method

Projected gradient descent (PGD) is one of the most straight forward projection-based algorithms that follows exactly the above mentioned natural concept: In each of its iteration from the current point, it moves toward the negative gradient direction of function f, and then projected back to $\mathcal{X}$ to maintain feasibility. A graphical representation of this algorithm is given in Figure 1.1. The general scheme of the PGD method is described in Algorithm 1.

Algorithm 1 Projected gradient descent (PGD)

```
Choose $x_0 \in \mathcal{X}$.
for $k = 1, \ldots, N$ do
    $x_k = \mathrm{Proj}_{\mathcal{X}}\left(x_{k-1} - \eta_k \nabla f(x_{k-1})\right)$    (1.4)
end for
Output $x_N$.
```

Here in Algorithm 1, the projection function might be defined in different ways. A common definition of such projection is

$$x_k = \operatorname*{arg\,min}_{x \in \mathcal{X}} \langle \nabla f(x_{k-1}), x \rangle + \frac{1}{2\eta_k} \|x - x_{k-1}\|^2 . \tag{1.5}$$

To see that the above is equivalent to (1.4), note that it can be written as

$$\begin{aligned} x_k &= \operatorname*{arg\,min}_{x \in \mathcal{X}} \langle \nabla f(x_{k-1}), x \rangle + \frac{1}{2\eta_k} \|x - x_{k-1}\|_2^2 \\ &= \operatorname*{arg\,min}_{x \in \mathcal{X}} \frac{1}{2\eta_k} \|x - (x_{k-1} - \eta_k \nabla f(x_{k-1}))\|^2 \\ &= \mathrm{Proj}_{\mathcal{X}}\left(x_{k-1} - \eta_k \nabla f(x_{k-1})\right). \end{aligned}$$

There is a very interesting and subtle idea behind the definition (1.5). First-order or linear approximation at the iterate in a typical iteration is the most straightforward approximation that we might have for the objective function f. Although minimizing f might be complicated, minimizing its linear approximation is a much simpler task. However, since minimizing the linear approximation over $\mathcal{X}$ may be unbounded (e.g., when $\mathcal{X} = \mathbb{R}^n$), we add an additional term that prevent us from moving too far away. This additional term is the quadratic term in (1.5) that penalizes new point being too far away from current point. We may observe that when the step-size η_k is very small, the multiplier of the quadratic term becomes very large.

We have the following convergence result concerning Algorithm 1: the average of iterates $\bar{x}_N := \frac{1}{N}\sum_{k=1}^{N} x_k$ is an approximate solution. In order to guarantee that $f(\bar{x}_N) - f^* \le \varepsilon$, the total

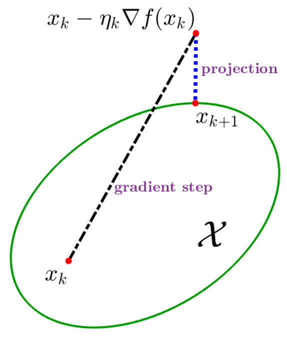

Figure 1.1: PGD algorithm project back the point from the gradient step onto the feasible region $\mathcal{X}$.

number of required iterations is bounded by $\mathcal{O}\left({}^{L\|x_0-x^*\|^2}/_{\epsilon}\right)$. A detailed proof of the convergence results of PGD method is provided in the Appendix A.1.

1.1.2 Nesterov's accelerated gradient descent method

As described above, the PGD requires at most $\mathcal{O}\left({}^1/_{\epsilon}\right)$ number of iterations to compute an approximate solution of problem (1.1). Is it possible to improve this upper bound with some modification in Algorithm 1? Nesterov introduced such a modification in [37] (see also [35]) and proposed an algorithm that we now call the *Nesterov's accelerated gradient descent* (NAGD) method. The general scheme of the PAGD algorithm is presented in Algorithm 2.

Algorithm 2 Nesterov's Accelerated Gradient Descent (NAGD)

```
Choose $x_0 \in \mathcal{X}$ and set $y_0 = x_0$.
for $k = 1, \cdots, N$ do
    $$\begin{aligned} z_k &= (1-\gamma_k)y_{k-1} + \gamma_k x_{k-1} \\ x_k &= \arg\min_{x\in\mathcal{X}} \langle \nabla f(z_k), x\rangle + \frac{1}{2\beta_k}\|x - x_{k-1}\|^2 \\ y_k &= (1-\gamma_k)y_{k-1} + \gamma_k x_k \end{aligned}$$
end for
Output $y_N$.
```

The idea behind the NAGD method is that in each iteration, unlike PGD, instead of moving directly toward the negative gradient of f at the current point, the algorithm includes a multiple of the difference between the current point and the point from previous iteration, and then moves toward the negative gradient and updates the point. See Figure 1.2 for a geometrical interpretation. This idea of adding multiple values is sometimes called the momentum idea.

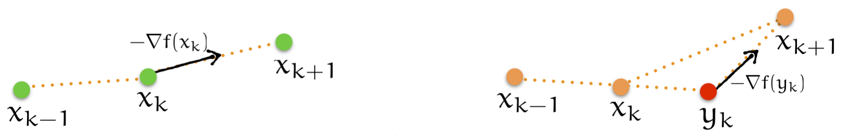

Figure 1.2: A typical iteration of PGD (on the left) and a NAGD iterate (on the right). Image is from [47].

Similar to PGD, NAGD is a projection-based algorithm due to the projection subproblem in (1.7). Note that if $\gamma_k \equiv 1$ then x_k updates as an step in PGD. In other words, with $\gamma_k \equiv 1$, NAGD is equivalent to PGD with step size $\frac{1}{\beta_k}$.

Note that the iteration complexity upper bound for the NAGD is $\mathcal{O}(\sqrt{1/\varepsilon})$, which is optimal for solving smooth convex optimization with Lipschitz constant L [35]. Details on the analysis of the convergence of Algorithm 2 is deferred to Appendix A.2.

1.2 Projection-Free Algorithms

While NAGD method has optimal complexity bound $\mathcal{O}(\sqrt{1/\epsilon})$, the projection subproblems of such a projection-based algorithm might be problematic in many cases since they are not always efficiently computable. For example, if $\mathcal{X}$ is a general polyhedron (i.e. polyhedron $Ax \leq b$ for general matrix A and vector b), then the projection on to the polyhedron might have expensive computational costs. A different class of algorithms, called projection-free algorithms, are useful in such expensive cases.

1.2.1 Frank-Wolfe Algorithm

The Frank-Wolfe (FW) algorithm [15], also known as the conditional gradient method (due to [31]), is one of the earliest projection-free first-order algorithms for solving convex programming problems. It was initially developed by Frank and Wolfe in 1956 and is an alternative to PGD/NAGD method.

Different from PGD and NAGD, the FW method does not require the projection as a subproblem, but performs updates in each iteration in a way that they are always feasible. More precisely, in each iteration, instead of solving the projection subproblem, FW solves a linear minimization subprolem over the feasible region $\mathcal{X}$. The objective of such subproblem is the linear approximation of function f at the point obtained in previous iteration. Recall from Section 1.1.1 that we might be concerned on the boundedness of minimizing linear approximation over $\mathcal{X}$ (e.g., when $\mathcal{X} = \mathbb{R}^n$). Therefore, to maintain the boundness of linear minimization subproblems, $\mathcal{X}$ is required to be a convex and compact set in (1.1). The FW method is described in Algorithm 3.

Algorithm 3 Frank-Wolfe (FW)

```
Choose $z_0 \in \mathcal{X}$ and set $x_0 = z_0$.
for $k = 1, \cdots, N$ do
    $z_k = (1-\gamma_k)y_{k-1} + \gamma_k x_{k-1}$    (1.9)
    $x_k \in \operatorname*{Argmin}_{x\in\mathcal{X}} \langle \nabla f(z_k), x \rangle$    (1.10)
    $y_k = y_{k-1} + \gamma_k(x_k - y_{k-1})$    (1.11)
end for
Output $x_N$.
```

A few remarks can be made regarding FW. First, (1.10) in Algorithm 3 is equivalent to minimizing the linear approximation of function f at point z_k, i.e.

$$\begin{aligned} x_k &\in \operatorname*{Arg\,min}_{x\in\mathcal{X}} f(z_k) + \langle \nabla f(z_k), x - z_k \rangle \\ &= \operatorname*{Arg\,min}_{x\in\mathcal{X}} \langle \nabla f(z_k), x - z_k \rangle \\ &= \operatorname*{Arg\,min}_{x\in\mathcal{X}} \langle \nabla f(z_k), x \rangle . \end{aligned} \tag{1.12}$$

Note that the solution to these subproblems is not necessarily unique. Second, the FW algorithm has a nice feature that distinguishes it from PGD and NAGD: in each iteration FW admits a vary natural duality gap from optimizing the linear approximation over the constraint set. More precisely, the quantity $\langle \nabla f(z_k), x - z_k \rangle$ that appears in (1.12) and is computed in the k-th iteration k of FW provides an upper bound on $f(z_k) - f^*$. The reason is that, by the definition of the convexity of function f, for all $x \in \mathcal{X}$ we have

$$f(x) \geq f(z_k) + \langle \nabla f(z_k), x - z_k \rangle . \tag{1.13}$$

Now if we minimize both sides of (1.13) over all $x \in \mathcal{X}$ we get

$$\begin{aligned} f^* &\geq f(z_k) + \min_{x \in \mathcal{X}} \langle \nabla f(z_k), x - z_k \rangle \\ &= f(z_k) + \langle \nabla f(z_k), x_k - z_k \rangle, \end{aligned}$$

which implies that

$$\langle \nabla f(z_k), z_k - x_k \rangle \geq f(z_k) - f^*.$$

This gap is called the Wolfe-gap or the duality gap between the current iterate and the optimal solution. In practice, Wolfe gap can be used as a stopping criterion of the FW for a desired pre-defined accuracy threshold of ϵ; we conclude that z_k is an ϵ-approximate solution whenever

$$f(z_k) - f^* \leq \langle \nabla f(z_k), z_k - x_k \rangle \leq \epsilon.$$

Finally, a graphical description of the behavior of an iteration of FW algorithm is given in Figure 1.3. This image appeared in Jaggi's publication [21] on FW methods.

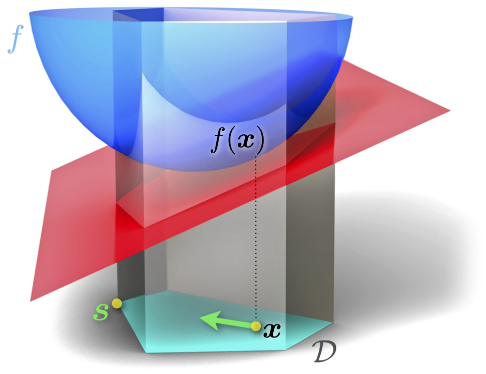

Figure 1.3: A geometrical interpretation of FW [21]. Here $\mathcal{D}$ is the feasible region and the red hyperplane is the tangent space of f at point x. The solution to the subproblem of FW at current iteration is shown by s.

When designing numerical methods that uses first-order information, the structures of the objective function and the compact feasible set $\mathcal{X}$ have significant impact on the theoretical convergence theory and practical computing performance. As an example, consider the performance of two classical iterative numerical methods on a simple instance of problem (2.1) in which $\mathcal{X} := \{(x^{(1)}, x^{(2)}) \in \mathbb{R}^2 | x^{(1)} + x^{(2)} = 1, x^{(1)}, x^{(2)} \geq 0\}$ is a line segment in $\mathbb{R}^2$ and $f(x) = \|x\|^2/2$. The first numerical method is the projected gradient method with iterates

$$x_k = y_{k-1} - (1/\beta_k)\nabla f(y_{k-1}),\ y_k = \mathrm{Proj}_{\mathcal{X}}(x_k).$$

In each iteration, the projected gradient method moves along the negative gradient direction with a pre-specified stepsize $(1/\beta_k)$ to obtain an updated iterate x_k. Such update x_k may fall outside of the feasible region, so we maintain feasibility by projecting x_k onto the feasible set to obtain a new approximate solution y_k. The second numerical method is the Frank-Wolfe method [15]

$$x_k = \min_{x \in \mathcal{X}} \langle \nabla f(x_{k-1}), x \rangle,\ y_k = \sum_{i=1}^{k} \lambda_k^i x_i. \tag{1.14}$$

Here we compute an update x_k from a linear optimization over the line segment $\mathcal{X}$. Note that x_k will always be selected from the two extreme points $(1, 0)^\top$ and $(0, 1)^\top$. To avoid oscillating between the extreme points, we select the new approximate solution y_k to a convex combination of all previous updates x_i's. Both the aforementioned methods have been extensively studied in the literature. See, e.g., the monograph [9] or the book [28] for the survey of both methods.

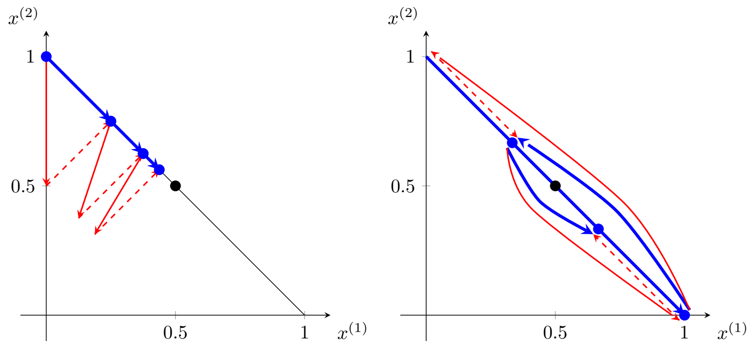

Figure 1.4: Left: Project gradient method iterations. Blue arrows describe update of approximate solutions, red solid arrows describe gradient descent moves, and red dashed arrows describe projections onto $\mathcal{X}$. Right: Frank-Wolfe method iterations. Blue arrows describe update of approximate solutions, red solid arrows describe the updates x_k (always extreme points), and red dashed arrows describe the convex combination weighting process.

The performance of the projected gradient and Frank-Wolfe type methods are illustrated in Figure 1.4. Both methods start at point $(0, 1)^\top$. In the projected gradient method we select the constant stepsize $1/\beta_k \equiv 0.5$ and in the Frank-Wolfe method we select the weights $\lambda_i = 2i/(k(k+1))$. From Figure 1.4 we can observe that approximate solutions of both methods are gradually approaching the optimal solution. The ones produced by the projected gradient method converge faster comparing to the "back and forth" behavior (see the blue arrows in the right plot) from the Frank-Wolfe method; this is because the projected gradient method benefits from the structure of the objective function (i.e., smoothness and strong convexity). However, in the Frank-Wolfe method iterations we only need to solve a linear program, while the projected gradient method needs to compute projections onto $\mathcal{X}$. If we extend the aforementioned simple example to the general case in where $\mathcal{X}$ is a high dimensional general convex polytope, then solving linear programs would likely be more preferable than solving projections onto general convex polytopes.

The FW method computes an ϵ-solution to the problem (1.1) in at most $\mathcal{O}(1/\epsilon)$ iterations. These methods are still preferred in many practices due to its advantage of not requiring the projection computation, even though its requirement of $\mathcal{O}(L/\varepsilon)$ gradient computations is not as good as the theoretical limit $\mathcal{O}(\sqrt{L/\varepsilon})$. For problems with sophisticated feasible set $\mathcal{X}$, the possibly expensive computational time of projection operator can significantly outweigh the theoretical advantage of the smaller $\mathcal{O}(\sqrt{L/\varepsilon})$ gradient evaluation of any projection-based methods. Also, we will discuss later in Section 1.2.4 how this drawback of FW is resolved by the *conditional gradient sliding* method proposed in [30].

The convergence analysis of the FW algorithm is well studied in [10, 13, 18, 21]. For completeness, we provide the proof of the convergence of FW in detail in Appendix A.3. Note that, under extra conditions, FW would converge faster than the sub-linear convergence rates mentioned above. In the following section we will discuss a plausible explanation of the sublinear convergence rate of FW and two variants that converge faster under stronger conditions.

1.2.2 The Zig-Zagging Phenomenon of the Frank-Wolfe Method

As we discussed in the previous section, when f in (1.1) is smooth and $\mathcal{X}$ is a convex compact set, FW converges at a sub-linear rate of $\mathcal{O}(1/\epsilon)$. However, under more assumptions, possibly more than strongly convexity of f, FW can converge at faster rates. These assumptions rely on the geometry of the feasible region $\mathcal{X}$ and the position of the optimal solution x^*. For example, authors in [16] showed that when $\mathcal{X}$ is a polytope and none of the constraints are active at the solution x^* to the (1.1) (i.e., x^* is in the relative interior of $\mathcal{X}$), the FW algorithm converges at a faster rate.

However, these conditions are very strong and not quite interesting in practice. In fact, it possible that the sublinear convergence rate of the FW is due to the fact the x^* is located on the boundary of $\mathcal{X}$. In such cases, the iterates of the FW algorithm start to zig-zag between the vertices defining the face containing the x^*. The reason for this is that the current iterate of a typical iteration of FW moves only towards vertices obtained by the linear minimzation subproblems. Figure 1.5 illustrates this phenomenon.

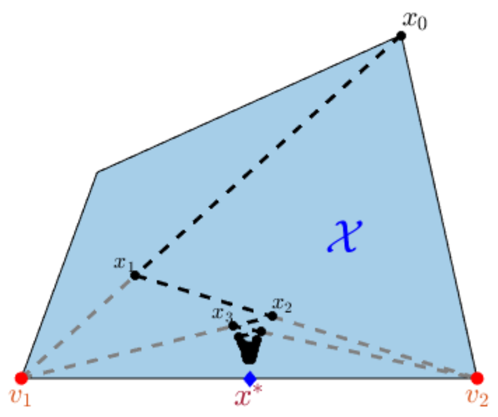

Figure 1.5: zig-zagging trajectory of FW towards the optimal solution. Here FW alternatively iterates toward the vertices v_1 and v_2 where the optimal solution is located on the face generated by these vertices.

1.2.3 Variants of Frank-Wolfe

As discussed before, under stronger conditions on f and the geometry of $\mathcal{X}$, there are variants of FW that converge faster than than the classical FW Algorithm 3. More precisely, these variants are guaranteed to converge linearly when in (1.1) the objective f is strongly convex and $\mathcal{X}$ is the polytope obtained from the convex hull of a vertex set X (i.e., X is a finite set of vectors and $\mathcal{X} = \mathrm{conv}(X)$).

1.2.3.1 Away-Steps Frank-Wolfe

The zig-zagging issue in FW was first studied by Wolfe [48]. In 1970 he proposed a variant of FW which is recently called away-steps FW (AFW). The idea behind the AFW is that in each iteration it either moves towards a vertex or moves away from a vertex depending on how the duality gap changes. Note that the modification designed by Wolfe in [48] was not convergent. Guélet and Marcotte in [16] corrected that version and proposed the modified Frank-Wolfe (MFW) in the case when $\mathcal{X}$ can be represented by linear inequalities. Here we introduce the general scheme of the AFW algorithm, as proposed in [24], in Algorithm 4.

Algorithm 4 Away-steps Frank-Wolfe (AFW)

```
Let $x^{(0)} \in X$ and $\mathcal{S}^{(0)} := \{x^{(0)}\}$
for $k = 1, \cdots, N$ do
        $s_k = \arg\min_{s \in \mathcal{X}} \langle \nabla f(x_k), s \rangle$    (1.15)
        $d_k^{FW} := s_k - x_k$    (1.16)
        $v_k = \arg\max_{v \in \mathcal{S}} \langle \nabla f(x_k), v \rangle$    (1.17)
        $d_k^A := x_k - v_k$    (1.18)
        if $\left\langle -\nabla f(x_k), d_k^{FW} \right\rangle \leq \epsilon$ then return $x_k$    (1.19)
        if $\left\langle -\nabla f(x_k), d_k^{FW} \right\rangle \geq \left\langle -\nabla f(x_k), d_k^A \right\rangle$ then    (1.20)
            $d_k := d_k^{FW}$
            $\gamma_{\max} := 1$    (1.21)
        else
            $d_k := d_k^A$
            $\gamma_{\max} := \alpha_{v_k}/(1 - \alpha_{v_k})$    (1.22)
        end if
        $\gamma_k \in \arg\min_{\gamma \in [0, \gamma_{\max}]} f(x_k + \gamma d_k)$    (1.23)
        $x_{k+1} := x_k + \gamma_k d_k$    (1.24)
        $\mathcal{S}_{k+1} := \{v \in X \ : \ \alpha_v^{k+1} > 0\}$
end for
Output $x_N$.
```

A few remarks can be made regarding Algorithm 4. First, due to the origin of the modification of FW, at each iteration, AFW requires two calls for the linear optimization oracle. In fact, at step (1.15) of Algorithm 4, a linear minimization oracle returns a FW vertex, and at its step (1.17), a linear maximization oracle returns an away vertex. Note that the maximization is over the set of candidate vertices and so it is fundamentally easier than the linear optimization oracle in (1.15).

Second, steps (1.16) and (1.18) correspond to a potential FW or away direction. A FW direction is the one taken in an iteration of classic FW, and an away direction is the one taken in which AFW moves away from a vertex that and makes the best progress.

Third, step (1.20) checks on the Wolfe gaps corresponding to the calculated FW and away directions. Among the two directions, we pick the direction that yields the higher gap value. Note that the AFW algorithm incorporates the set $\mathcal{S}^{(k)}$ that is called active set, where k corresponds to the k-th iteration of the algorithm. This set stores the vertices that the algorithm has visited until the current iteration and (if required) also the vertices that potentially will be used for the away direction.

Fourth, the step-size γ_k in the k-iteration k at step (1.23) is achieved using a line-search method. Also, the maximum step-size $\gamma_{\max}$ computed at steps (1.21) and (1.22) ensures that the new iterate update at (1.24) is feasible and stays in $\mathcal{X} = \operatorname{conv}(X)$. In fact, the $\gamma_{\max}$ given in (1.22) guarantees that the convex representation remains feasible. For a general polytope $\mathcal{X}$, it is not possible to to compute the maximum feasible step-size, as it requires the ability to know when we cross the boundary of $\mathcal{X}$ along a specific line. Hence, the maximum step-size given in (1.22) is a conservative representation which ensures that we do not need this more powerful oracle. In fact, this is the reason that Algorithm 4 requires to keep record of the set $\mathcal{S}^{(t)}$. From memory usage point of view, this is a drawback of the AFW algorithm as it requires to store their convex combinations onto the vertices of $\mathcal{X}$ which can become very expensive in memory usage.

Fifth, the Wolfe gap $\left\langle -\nabla f(x_k), d_k^{FW} \right\rangle$ in (1.19) is used as a stopping criterion of the algorithm 4 and is an upper bound on the unknown suboptimality. This is since

$$f(x_k) - f(x^*) \leq \langle -\nabla f(x_k), x^* - x_k \rangle \leq \left\langle -\nabla f(x_k), d_k^{FW} \right\rangle \leq \epsilon.$$

Here we use the convexity of f, and ϵ is the threshold for the stopping criterion. Note that when the optimal step-size at iteration t in (1.23) equals $\gamma_{\max}$, then v_t is removed from the current active set of vertices. Such an step is called a drop step.

Finally, the weights α_v^{k+1} and the set $\mathcal{S}^{(k+1)}$ are updated in the following form: If a FW step is taken, then

$$\mathcal{S}^{(k+1)} = \begin{cases} \{x_k\} & \text{if } \gamma_k = 1 \\ \mathcal{S}^{(k)} \cup \{s_k\} & \text{otherwise.} \end{cases}$$

Also, we have

$$\alpha_{s_k}^{(k+1)} := (1-\gamma_k)\alpha_{s_k}^{(k)} + \gamma_k$$
$$\alpha_v^{(k+1)} := (1-\gamma_k)\alpha_v^{(k)} \text{ for } v \in \mathcal{S}^{(k)} \setminus \{s_k\}$$

On the other hand, if an away step is taken, then we have

$$\mathcal{S}^{(k+1)} = \begin{cases} \mathcal{S}^{(k)} \setminus \{v_k\} & \text{if } \gamma_k = \gamma_{\max} \text{ (a drop step)} \\ \mathcal{S}^{(k)} & \text{otherwise.} \end{cases}$$

In such case, we have

$$\alpha_{v_t}^{(k+1)} := (1+\gamma_k)\alpha_{v_t}^{(k)} - \gamma_k$$
$$\alpha_v^{(k+1)} := (1+\gamma_k)\alpha_v^{(k)} \text{ for } v \in \mathcal{S}^{(k)} \setminus \{v_k\}$$

1.2.3.2 Pairwise Frank-Wolfe

Another variant of the FW algorithm that we introduce here is studied by Lacoste and Jaggi in [24]. As the authors mention in their paper, this method is inspired by an algorithm called MDM algorithm that is studied by Mitchell et al. [33]. MDM algorithm was originally invented for the polytope distance problem. The idea of the new variant of FW, that is called pairwise Frank-Wolfe (PFW), is to only move weight mass between two vertices in each step. A graphical representation to compare the behavior of the variants of FW is shown in Figure 1.6. The algorithm of PFW that is described in Algorithm 5 is very similar to the previously introduced Algorithm 4 of AFW. The geometric interpretations of FW, AFW, and PFW are illustrated in Figure 1.6 (figure is from [24]).

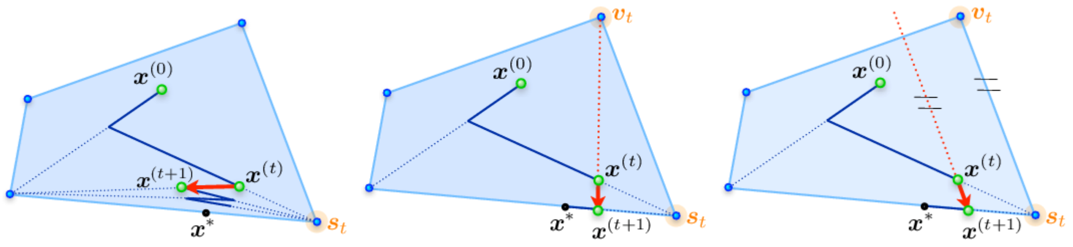

Figure 1.6: Left: the classical FW zig-zagging trajectory. Middle: the AFW trajectory. Right: the PFW trajectory. Image is from [24].

Algorithm 5 Pairwise Frank-Wolfe (PFW)

In Algorithm 4, replace the descriptions of d_k and $\gamma_{\max}$ in step (1.20) to the following: $d_k = d_k^{PFW} := s_k - v_k$ and $\gamma_{\max} := \alpha_{v_t}$.

Note that in the k-th iteration of the PFW method, the weights are moved from the away vertex v_k to the FW vertex s_k, and all other α weights are kept unchanged. The weights are defined as

$$\alpha_{v_k}^{(k+1)} = \alpha_{v_k}^{(k)} - \gamma \quad \text{and} \quad \alpha_{s_k}^{(k+1)} = \alpha_{s_k}^{(k)} + \gamma,$$

for some $\gamma \leq \gamma_{\max} := \alpha_{v_k}^{(k)}$.

Lacoste and Jaggi in [24] proved for the first time that the variants of FW converge with linear rate when $\mathcal{X}$ is the convex hull of a finite set of vectors X, and f is strongly convex with Lipschitz continuous gradient over $\mathcal{X}$. For a detailed analysis of the convergence of the AFW and PFW along with two more variants, namely, the fully-corrective FW and Wolfe's minimum norm point algorithm, see [24].

1.2.4 Conditional Gradient Sliding Algorithm

In Section 1.1.2, we discussed the NAGD method achieves the optimal complexity bound $\mathcal{O}(\sqrt{1/\epsilon})$ for solving convex optimization problem (1.1). As mentioned, the drawback of this method is that it requires to solve a projection subproblem in each iteration. In certain cases, solving this subprolem could be as difficult as the orginal problem itself. In contrast, FW and its variants are projection-free methods that require calling linear optimization (LO) oracles in their iterations. However, as previously mentioned, they gradient evaluation complexity bound may not be as good as that of NAGD.

The goal of the *conditional gradient sliding* (CGS) [30] method is to present a new LO based convex programming method which can skip the computation for the gradient of f from time to time when performing linear optimization over the feasible region $\mathcal{X}$. The basic scheme of this method is obtained by applying the FW method to approximately solve the projection subproblems existing in the PAGD iterations. By properly specifying the accuracy for solving these subproblems, it can be shown that the resulting CGS method can achieve the optimal bounds on the number of calls to the first-order and linear optimization oracles for solving problem (1.1). The development of CGS method, in spirit, is similar to the gradient sliding algorithm developed by Lan in [27] for solving a class of composite optimization problems. However, the gradient sliding algorithm in [27] requires us to perform projection over the feasible set $\mathcal{X}$ and targets to solve convex programming problems with a general nonsmooth term in objective function. The CGS method is formally described in Algorithm 6.

Algorithm 6 Conditional gradient sliding (CGS)

```
Initial point $x_0 \in \mathcal{X}$ and iteration limit $N$.
Let $\beta_k \in \mathbb{R}^n_{++}$, $\gamma_k \in [0,1]$, and $\eta_k \in \mathbb{R}_+$, $k = 1, 3\cdots$, be given and set $y_0 = x_0$.
for $k = 1, \ldots, N$ do
    $z_k = (1-\gamma_k)y_{k-1} - \gamma_k x_{k-1}$    (1.25)
    $x_k = \mathcal{FW}(f'(z_k), x_{k-1}, \beta_k, \eta_k),$    (1.26)
    $y_k = (1-\gamma_k)y_{k-1} + \gamma_k x_k$    (1.27)
end for
procedure $u^+ = \mathcal{FW}(g, u, \beta, \eta)$
    1. Set $u_1 = u$ and $t = 1$.
    2. Let $v_t$ be the optimal solution for the subproblem of
       $V_{g,u,\beta}(u_t) := \max_{x\in\mathcal{X}} \langle g + \beta(u_t - u), u_t - x\rangle$    (1.28)
    3. If $V_{g,u,\beta}(u_t) \le \eta$, set $u^+ = u_t$ and terminate the procedure.
    4. Set $u_{t+1} = (1-\alpha_t)u_t + \alpha_t v_t$, with
       $\alpha_t = \min\left\{1, \frac{\langle \beta(u-u_t) - g, v_t - u_t\rangle}{\beta\|v_t - u_t\|^2}\right\}$
    5. Set $t \leftarrow t+1$ and go to step 2.
end procedure
```

Recall that we defined the projection subproblem in the NAGD method as the minimization of the following function:

$$\phi_k(x) := \langle \nabla f(z_k), x\rangle + \frac{\beta_k}{2}\|x - x_{k-1}\|^2. \tag{1.29}$$

over the convex and compact set $\mathcal{X}$. The CGS algorithm solves the minimization of the above function, a projection subproblem, using the FW method. In fact, CGS applies the FW to (1.29) but until the η_k accuracy threshold is achieved. This happens through the $\mathcal{FW}$ procedure at step (1.26) in Algorithm 6. Note that within this application, a call of a LO oracle in FW method is given by

$$\operatorname*{Arg\,min}_{x\in\mathcal{X}} \langle \nabla\phi_k(x), x\rangle .$$

Using the above oracle and also the benefit of the Wolfe gap that can be obtained in each iteration of FW (as fully discussed in Section 1.2.1), the $\mathcal{FW}$ procedure continues until the gap shrinks upto a threshold. More precisely, x_k in (1.26) is an approximate solution to the projection subproblem $min_{x\in\mathcal{X}}\phi_k(x)$ in the $\mathcal{FW}$ procedure until the Wolfe gap $V(\cdot)$ in step 2

$$\max_{x\in\mathcal{X}} \langle \nabla\phi_k(x_k), x_k - x\rangle = \max_{x\in\mathcal{X}} \langle \nabla f(z_k) + \beta_k(x_k - x_{k-1}), x_k - x\rangle$$

is smaller that η_k for some $\eta_k \geq 0$. It should also be pointed out that the step-size in the $\mathcal{FW}$ procedure at step 4 is in fact the closed form of a line-search calculation given by

$$\operatorname*{arg\,min}_{\alpha\in[0,1]} \phi(u_t + \alpha(v_t - u_t)).$$

In Appendix A.4 we will discuss a detailed analysis of the convergence of CGS. We will show that CGS method requires at most $\mathcal{O}\left(1/\sqrt{\epsilon}\right)$ gradient evaluation and $\mathcal{O}\left(1/\epsilon\right)$ LO oracle calls. We will also discuss the appropriate settings for $\{\gamma_k\}, \{\beta_k\}$ and $\{\eta_k\}$ that conclude such convergence results.

As an special case of CGS, if we limit the number of inner iterations to one iteration in the FW procedure of CGS we get Algorithm 7 below. Note that (1.31) in Algorithm 7 is equivalent to

$$x_k \in \operatorname*{Argmin}_{x\in\mathcal{X}} \langle \nabla f(z_k), x\rangle \tag{1.30}$$

and (1.30) is exactly the subprobem of FW algorithm. Therefore, limiting the number of inner iterations of CGS to one iteration leads to the FW algorithm.

At this step we are ready to summarize and compare the convergence result and also requirements of NAGD, FW and CGS. In order to compute an ϵ-solution NAGD requires $\mathcal{O}(\sqrt{L(D_{\mathcal{X}})^2/\epsilon})$ gradient evaluations where L and $D_{\mathcal{X}}$ are the true values of Lipschitz constant and the diameter of $\mathcal{X}$, respectively. This number of evaluations is significantly smaller than the $\mathcal{O}(LD^2_{\mathcal{X}}/\epsilon)$ evaluations of FW. However, NAGD requires the solution to the projection subproblem in each iteration of

Algorithm 7 The Conditional Gradient Sliding Algorithm with One Inner Iteration

```
Initial point $x_0 \in \mathcal{X}$ and iteration limit $N$.
Let $\beta_k \in \mathbb{R}^n_{++}$, $\gamma_k \in [0,1]$, and $\eta_k \in \mathbb{R}_+$, $k = 1, 3 \cdots$, be given and set $y_0 = x_0$.
for $k = 1, \ldots, N$ do
    $$\begin{aligned} z_k &= (1-\gamma_k) y_{k-1} - \gamma_k x_{k-1} \\ x_k &\in \underset{x \in \mathcal{X}}{\text{Argmax}} \ \langle \nabla f(z_k), x_{k-1} - x \rangle, \\ y_k &= (1-\gamma_k) y_{k-1} + \gamma_k x_k \end{aligned} \tag{1.31}$$
end for
Output $y_N$.
```

its algorithm which is not always efficiently solvable. This can be a drawback for NAGD method. CGS, on the other hand, resolves the requirement of projection calculation in NAGD and also the complexity of total number of gradient evaluations in FW. This observation is summarized in Table 1.1. Note that $\text{Proj}_{\mathcal{X}}(\cdot)$ in Table 1.1 is the projection function where (1.7) and (1.4) the subproblem of NAGD and GD, respectively, are examples of this function.

	NAGD	FW	CGS
Subproblem	$\text{Proj}_{\mathcal{X}}(\cdot)$	$\min_{x\in\mathcal{X}}\langle \cdot, x\rangle$	$\min_{x\in\mathcal{X}}\langle \cdot, x\rangle$
Number of subproblem computations	$\sqrt{LD^2_{\mathcal{X}}/\varepsilon}$	$L(D_{\mathcal{X}})^2/\varepsilon$	$L(D_{\mathcal{X}})^2/\varepsilon$
Number of gradient evaluations	$\sqrt{LD^2_{\mathcal{X}}/\varepsilon}$	$L(D^*_{\mathcal{X}})^2/\varepsilon$	$\sqrt{L(D_{\mathcal{X}})^2/\varepsilon}$

Table 1.1: Comparing the complexity of algorithms PAGD, FW and CGS

However, CGS still requires $L(D_{\mathcal{X}})^2/\varepsilon$ number of solutions to linear optimization problem that cannot be improved according to [25, 36]. Another drawback of CGS is its requirement to the parameter L. This drawback is resolved in a *CGS with line search* (CGS-ls) method that we will propose it in next chapter.

In the original CGS method in [30], the knowledge of the Lipschitz constant L and the number of gradient evaluations N are required for implementation[1]. Such requirements lead to two disadvantages in practice. First, in order to make sure that a constant L satisfies the Lipschitz condition (1.2), we will need to choose a constant L that satisfies the Lipschitz condition (1.2) for all pairs x and y in $\mathcal{X}$. Computing such L can be difficult; the computed L can also be too conservative

[1]Some parameter settings of the CGS method does not require N; see, e.g., Corollary 2.3 in [30]. However, since no termination criterion is proposed in [30], to terminate we still need to specify the total number of iterations N. It should be noted that we can use some termination criterion for the CGS method, e.g., the Wolfe gap, which we will use for CGS in the numerical experiments of this chapter; however, the theoretical convergence property in terms of the number of iterations needed to achieve small Wolfe gap is different from the properties of the CGS method, and may deteriorate its practical performance significantly.

and lead to worse practical performance. Second, in order to compute an ε-approximate solution, we need to either tune the number of gradient evaluations N in practice or follow its theoretical property and specify a possibly conservative $N = \mathcal{O}(L/\sqrt{\varepsilon})$. While the CGS method reaches the theoretical performance limits, such disadvantages may deteriorate its practical performance significantly.

1.3 Bundle-Level methods

In this section we will discuss some variants of the *bundle-level* (BL) type methods. Although these methods belong to the category of the projection-based methods, we describe their variants specifically in this section as they will be our inspiration for the algorithm we proposed in chapter 3.

We start this discussion with a short review on BL method. Consider the convex programming

$$f^* := \min_{x \in X} f(x) \tag{1.32}$$

where X is a convex compact set and $f : X \to \mathbb{R}$ is a closed convex function and satisfies

$$f(y) - f(x) - \langle f'(x), y - x \rangle \leq \frac{M}{1+\nu} \|y - x\|^{1+\nu}, \quad \forall x, y \in X, \tag{1.33}$$

for some $M > 0,\ \nu \in [0, 1]$ and $f' \in \partial f(x)$. This class of problems cover nonsmooth ($\nu = 0$), smooth ($\nu = 1$), and weakly smooth ($\nu \in (0, 1)$) CP problems. However, our focus in our proposed algorithm is on the class of functions smooth f where $\nu = 1$ in (1.33). In that case, we will use the notation L instead of M as the Lipschitz constant of ∇f. Also, let

$$h(z, x) := f(z) + \langle f'(z), x - z \rangle, \tag{1.34}$$

then at iteration k of BL method, for the given sequence of points $x_1, \cdots, x_k \in X$, the algorithm updates the new point x_{k+1} as

$$x_{k+1} \in \operatorname*{Arg\,min}_{x \in X} \ m_k(x) := \max\{h(x_i, x) \ : \ 1 \leq i \leq k\}. \tag{1.35}$$

Indeed, the updated point in each iteration of BL is the minimizer of lower envelopes of f, that is the maximum of linear approximations of function f at previous bundle of points from the first iteration.

While BL methods is slow in convergence both theoretically and practically, modifications have been proposed to improve the algorithm. For example, Lemaréchal et al. introduced a prox-term into the objective function of (1.35) and relaxed the linear approximations $m_k(x)$ into the constraints by incorporating level sets. More specifically, the general scheme if of this modified BL method is

a) Update $\bar{f}^k$ to be the best objective value found so far and compute a lower bound on f^* by $\underline{f}_k = \min_{x \in X} m_k(x)$;

b) Set $l_k = \lambda \bar{f}^k + (1-\lambda)\underline{f}_k$ for some $\lambda \in (0,1)$;

c) Set $x_{k+1} = \arg\min_{x \in X} \left\{ \|x - x_k\|^2 \ : \ m_k(x) \leq l_k \right\}$.

Note that in each iteration of the this modified BL method the feasible set of the subproblem shrinks to the level set bounded by l_k. This algorithm for a general nonsmooth convex function can find an ϵ-solution in at most

$$\mathcal{O}\left(C(\lambda) \frac{M^2 D_X^2}{\epsilon^2} \right)$$

iterations, where $C(\lambda)$ is a constant depending on λ and D_X is the diameter of the set X.

1.3.1 Accelerated bundle-level algorithm

Inspired by Nesterov's accelerated gradient descent method, G. Lan proposed a new bundle level type method, called *accelerated bundle-level* (ABL) method. In this procedure, instead of using a single sequence $\{x_k\}$, three sequences $\{x_k^\ell\}$, $\{x_k\}$ and $\{x_k^u\}$ are being used to update the linear approximations $m_k(x)$ in each iteration. Also, iterations of ABL are grouped into phases and it continues phases until a specific termination criterion holds. In each phase, the algorithm of ABL goes through a procedure called *gap reduction*. Gap reduction procedure continues until the gap between the lower nad upper bound of f^* is reduced by a certain constant factor. The general scheme of this method is presented in Algorithm 8.

Algorithm 8 Accelerate bundle-level algorithm (ABL)

```
Choose initial point $p_0 \in X$, tolerance $\epsilon > 0$ and parameters $\lambda \in (0,1)$
  0. Set $p_1 \in \operatorname{Arg\,min}_{x\in X} h(p_0, x)$, $lb_1 = h(p_0, p_1)$, $ub_1 = f(p_1)$ and $s = 1$.
  1. If $ub_s - lb_s \le \epsilon$ terminate the algorithm with output $p_s$.
  2. Set $(p_{s+1}, lb_{s+1}) = \mathcal{G}_{ABL}(p_s, lb_s, \lambda)$ and $ub_{s+1} = f(p_{s+1})$.
  3. $s \leftarrow s+1$ and go to 1.

gap reduction procedure $(p^+, lb^+) = \mathcal{G}_{ABL}(p, lb, \lambda)$
  0. Set $x_0^u = p$, $\bar{f}_0 = f(x_0^u)$, $\underline{f}_0 = lb$, $x_0 = x_0^u$, $m_0(x) = h(x_0, x)$, $k = 1$
  1. lower bound update:
       $x_k^\ell = (1-\alpha_k)x_{k-1}^u + \alpha_k x_{k-1}$,
       $m_k(x) = \max\{m_{k-1}(x), h(x_k^\ell, x)\}$                        (1.36)
       $h_k^* := \min_{x\in X} m_k(x)$,   $\underline{f}_k := \max\{\underline{f}_{k-1}, h_k^*\}$          (1.37)
  2. prox-center update: Set $\ell_k = \lambda \underline{f}_k + (1-\lambda)\bar{f}_{k-1}$ and
       $x_k = \arg\min\{\|x - x_{k-1}\|^2 \ : \ m_k(x) \le \ell_k,\ x \in X\}$        (1.38)
  3. upper bound update:
       $\bar{f}_k = \min\{\bar{f}_{k-1}, f(\alpha_k x_k + (1-\alpha_k)x_{k-1}^u)\}$          (1.39)
       choose $x_k^u \in X$ so that $f(x_k^u) = \bar{f}_k$
  4. If $\bar{f}_k - \underline{f}_k \le \lambda(\bar{f}_0 - \underline{f}_0)$ terminate the procedure with $p^+ = x_k^u$, $lb^+ = \underline{f}_k$
  5. Set $k \leftarrow k+1$ and go to step 1.
end procedure
```

A few remarks are in place for the ABL algorithm. First, the initial phase of the algorithm uses an initial arbitrary point from the feasible set X and using the linear approximation of function f at this point, defines initial lower and upper bounds for f^* and also the initial gap between these two bounds. Then through the gap reduction procedure in afterwards phases, the algorithm reduces this gap until the threshold ϵ is satisfied. Second, at each iteration of the gap reduction procedure for a value of α_k, that is the major difference of ABL method from other bundle level type methods when $\alpha_k \neq 1$, the algorithm has three separate updates; lower and upper bounds, and prox-center updates. More specifically, in its lower bound update steps 1 and for $k \ge 1$, the $\mathcal{G}_{ABL}$ procedure finds $\underline{f}_k \le f^*$ by solving a subproblem in (1.37) corresponding to the updated value of x_k^ℓ and the updated set of linear constraints $m_k(x)$. The update of prox-center in step 2 of $\mathcal{G}_{ABL}$ procedure also requires solving the subproblem (1.38) that is the same as the update in prox-center in iterations of previously mentioned BL type method and the bound for the level set is updated in the same manner. Also, note that when $\alpha_k = 1$ for $k \geq 1$, the iterations of $\mathcal{G}_{ABL}$ will be exactly the same as of BL method. Third, the $\mathcal{G}_{ABL}$ procedure ends when the gap between upper bound $\bar{f}_k$ and lower bound $\underline{f}_k$ of f^* is less than the gap from initial step 0 of this procedure with a fixed and prespecified multiple $\lambda \in (0, 1)$. Forth, from the definition of $m_k(x)$ in 1.35 and the convexity of f, it is easy to check the for any $x \in X$ the sequences $\{m_k(x)\}_{k\geq1}$ and $\{\underline{f}_k\}_{k\geq1}$ are decreasing and $\{\bar{f}_k\}_{k\geq1}$ is an increasing sequence. Also, for any $k \geq 1$ we have $\underline{f}_k \leq f^* \leq \bar{f}_k$ that is an straight forward result from (1.37) and (1.39). Therefore, if we denote

$$\Delta_k := \bar{f}_k - \underline{f}_k, \quad k \geq 0,$$

then $\{\Delta_k\}_{k\geq0}$ is a decreasing sequence of non-negative values.

1.3.2 Accelerate prox-level method

Despite the fact that ABL method performs better than other BL type algorithm, it still has draw backs. With a deeper look at the ABL algorithm and specially in (1.36), it can be noted that in each iteration of $\mathcal{G}_{ABL}$ a linear constraint is being added to the $m_k(x)$. This accumulation of linear constraints makes the subproblems (1.37) and (1.38) more difficult after some iterations of $\mathcal{G}_{ABL}$. Motivated from this issue, the *accelerated prox-level* (APL) method incorporates a relaxation with level sets in updates of lower bounds in ABL. Note that APL method is originally designed regardless of the smoothness of objective function f in (1.32) and for any $\nu \in [0, 1]$. However, we concentrate only on the analysis and calculations for the specific case where f is smooth or $\nu = 1$. In this case we denote the parameter M in (1.33) by L, that is the Lipschitz constant of the ∇f.

Lemma 5 shows the computation of the lower bound on f^* in APL method. In fact, this method solves the issue by incorporating a convex compact set, namely X' as the *localizer* of X_ℓ, to compute a lower bound on f^* where $X_\ell := \{x \in X \ : \ f(x) \leq \ell\}$ and

$$X_\ell \subseteq X' \subseteq X. \tag{1.40}$$

A few remarks are in place for the APL method. Clearly the main difference of APL method from ABL is in its gap reduction procedure. First, unlike the ABL that updates the bound for level

Algorithm 9 Accelerate prox-level algorithm (APL)

```
Choose initial point $p_0 \in X$, tolerance $\epsilon > 0$ and parameters $\beta, \theta \in (0,1)$
  0. Set $p_1 \in \operatorname{Arg\,min}_{x \in X} h(p_0, x)$, $lb_1 = h(p_0, p_1)$, $ub_1 = f(p_1)$ and $s = 1$.
  1. If $ub_s - lb_s \leq \epsilon$ terminate the algorithm.
  2. Set $(p_{s+1}, lb_{s+1}) = \mathcal{G}_{APL}(p_s, lb_s, \beta, \theta)$ and $ub_{s+1} = f(p_{s+1})$.
  3. $s \leftarrow s + 1$ and go to 1.

gap reduction procedure $(p^+, lb^+) = \mathcal{G}_{APL}(p, lb, \beta, \theta)$
  0. Set
       $x_0^u = p, \quad \bar{f}_0 = f(x_0^u), \quad \underline{f}_0 = lb, \quad \ell = \beta \underline{f}_0 + (1-\beta)\bar{f}_0$    (1.41)
       $x_0 = p, \quad X'_0 = X, \quad d_\omega(x) = \frac{1}{2}\|x - x_0\|_2^2, \quad k = 1$
  1. lower bound update:
       $x_k^\ell = (1-\alpha_k)x_{k-1}^u + \alpha_k x_{k-1},$    (1.42)
       $\underline{h}_k := \min_{x \in X'_{k-1}} h(x_k^\ell, x)$    (1.43)
       $\underline{f}_k := \max\{\underline{f}_{k-1}, \min\{\ell, \underline{h}_k\}\}$
     If $\underline{f}_k \geq (1-\theta)\ell + \theta \underline{f}_0$ terminate the procedure with $p^+ = x_{k-1}^u, \quad lb^+ = \underline{f}_k$.
  2. prox-center update: Set
       $x_k = \operatorname*{arg\,min}_{x \in X'_{k-1}} \left\{ d_\omega(x) \ : \ h(x_k^\ell, x) \leq \ell \right\}$    (1.44)
  3. upper bound update:
       $\bar{f}_k = \min\{\bar{f}_{k-1}, f(\alpha_k x_k + (1-\alpha_k)x_{k-1}^u)\}$
       choose $x_k^u$ so that $f(x_k^u) = \bar{f}_k$
     If $\bar{f}_k \leq (1-\theta)\ell + \theta \bar{f}_0$ terminate the procedure with $p^+ = x_k^u$, $lb^+ = \underline{f}_k$
  4. localizer update: Choose $X'_k$ so that $\underline{X}_k \subseteq X'_k \subseteq \bar{X}_k$ where
       $\underline{X}_k := \{x \in X'_{k-1} \ : \ h(x_k^\ell, x) \leq \ell\}, \qquad \bar{X}_k := \{x \in X \ : \ \langle \nabla d_\omega(x_k), x - x_k \rangle \geq 0\}$
end procedure
```

sets in prox-center update of each iteration of $\mathcal{G}_{ABL}$, APL uses a universal bound for all iteration of $\mathcal{G}_{APL}$ in (1.41) for each phase. This bound ℓ is updated at initial step 1.41 of the $\mathcal{G}_{APL}$ and by the fixed and prespecified convex combination parameter β. Second, in updates of localizer in step 4 of $\mathcal{G}_{APL}$, X'_k can be any set between the sets $\underline{X}_k$ and $\bar{X}_k$. Note that a linear constraint is still being added to the $\underline{X}_k$ per each increase in k and so APL has the accumulation of the linear constraints for this set, while $\bar{X}_k$ is the set X with only one extra linear constraint for any k. Indeed, one can control the number of linear constraint in localizer X' by choosing the appropriate set between $\underline{X}_k$ and X'_k where the simplest way to choose such localizer is to set $X'_k = \underline{X}_k$ or $X'_k = \bar{X}_k$. This flexibility makes the subproblems (1.43) and (1.44) easier to solve in practice. Finally, the $\mathcal{G}_{APL}$ procedure has two separate stopping criterion. One at its lower bound update step 1 and the other at its upper bound update step 3.

In chapter three, we propose a modification of APL method for solving (1.1) with assumptions stated afterward. Although APL method does not require the input of any problem parameters and still achieves the best possible iteration complexity bounds, but it requires projection as a subproblem in each iteration. Projection problems similar the ones appear in Projected Accelerated Gradient Descent (PAGD) like algorithms are not always efficiently solvable and in many cases are costly and expensive and sometimes as difficult as the problem itself. Our proposed method, Sliding Accelerated Prox Level (SAPL) addresses this issue by applying Conditional Gradient (CG) method (aka. Frank-Wolfe method) in each iteration to solve the projection subproblem approximately with some threshold. Besides, SAPL is theoretically as efficient as APL. I am working on some numerical experiments to show the efficiency of the SAPL in practice.

Chapter 2

Conditional Gradient Sliding with Linesearch

2.1 Introduction

Recall that our problem of interest is the convex optimization problem

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

where $\mathcal{X} \in \mathbb{R}^n$ is a convex compact set and $f : \mathbb{R}^n \to \mathbb{R}$ is a convex differentiable function such that

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \ \forall x, y \in \mathcal{X}. \tag{2.2}$$

Here $\|\cdot\|$ is the Euclidean norm. Our goal is to compute an ε-approximate solution y such that $f(y) - f^* \leq \varepsilon$ using first-order information, namely, the function and gradient values f and ∇f.

In this chapter, we propose a modification of the CGS method that allows for its practical implementation. Our proposed method, called the CGS with linesearch (CGS-ls), performs a backtracking linesearch strategy to gradually increase the initial guess of Lipschitz constant L_0 to values that satisfy the convergence condition. The initial guess L_0 does not need to satisfy the Lipschitz condition (2.2) and can be significantly smaller than the actual Lipschitz constant. We also maintains the estimate of a lower bound of f^* that certificates the achievement of an ε-approximate solution.

Consequently, our propose method does not require the knowledge of either L and N and is able to stop before the theoretical $\mathcal{O}(\sqrt{L/\varepsilon})$ bound of required gradient evaluations or the $\mathcal{O}(L/\varepsilon)$ bound of linear objective optimization subproblems. It should be noted that our theoretical analysis of the backtracking linesearch is non-trivial. In order to improve the practical implementation, add proper termination criterion, and maintain the same theoretical convergence properties as the CGS method, we need to modify some theoretical analysis in the original CGS results in [30]. We demonstrate through numerical experiments the advantages of our proposed CGS-ls method in implementation.

2.2 Algorithm

In this section, we describe our proposed conditional gradient sliding method with linesearch (CGS-ls) in Algorithm 10.

A few remarks are in place for the proposed CGS-ls algorithm. First, the CndG procedure is exactly the same as the one described in the CGS method in [30]. Noting the termination criterion of the CndG procedure, we can observe that the update x_k computed by the CndG procedure satisfies

$$\langle \nabla f(z_k) + \beta_k(x_k - x_{k-1}), x_k - x \rangle \le \eta_k, \ \forall x \in \mathcal{X}, \tag{2.11}$$

where η_k is an accuracy parameter and the β_k is a stepsize parameter whose values will be described in the sequel. Note that when the accuracy $\eta_k \equiv 0$ (this is only the ideal case; in practice the CndG procedure will never terminate when η_k is set to 0), then x_k is the exact optimal solution to the problem

$$\min_{x\in\mathcal{X}} \langle \nabla f(z_k), x \rangle + \frac{\beta_k}{2}\|x - x_{k-1}\|^2. \tag{2.12}$$

In such case, the iterates (2.6), (2.7), and (2.8) becomes the iterates for the accelerated gradient method (see, e.g., [38]). Consequently, CGS-ls reduces to the accelerated gradient method with backtracking linesearch.

Second, when $L_k \equiv L$ where L satisfies condition (2.2), then CGS-ls reduces to a version of CGS method in [30] with $\Gamma_k = L\gamma_k^3$. The concept behind the CGS method is to use a version of conditional gradient method to solve the possibly sophisticated projection subproblem described in (2.12). The theoretical performance limit is achieved through proper choice of the accuracy

```
Algorithm 10 A conditional gradient sliding algorithm with backtracking linesearch (CGS-ls)
  Choose initial point $y_0 \in \mathcal{X}$ and initial guess of Lipschitz constant $L_0 > 0$. Set $x_0 = y_0$. Define
  function $\xi_0(x) \equiv 0$.
  for $k = 1, 2, \ldots, N$ do
      Find the smallest integer $j \geq 0$ such that the estimated local Lipschitz constant $L_k = L_{k-1} \cdot 2^j$
      satisfies
          $f(y_k) \leq f(z_k) + \langle \nabla f(z_k), y_k - z_k \rangle + \frac{L_k}{2} \|y_k - z_k\|^2 + \frac{\varepsilon}{2}\gamma_k$, where   (2.3)
          $\gamma_k = \begin{cases} 1 & \text{if } k = 1 \\ \text{Positive solution to } L_k\gamma_k^3 = \Gamma_{k-1}(1-\gamma_k) & \text{if } k \geq 2 \end{cases}$   (2.4)
          $\Gamma_k = L_k\gamma_k^3$   (2.5)
          $z_k = (1-\gamma_k)y_{k-1} + \gamma_k x_{k-1}$   (2.6)
          $x_k = \text{CndG}(\nabla f(z_k), x_{k-1}, \beta_k, \eta_k)$   (2.7)
          $y_k = (1-\gamma_k)y_{k-1} + \gamma_k x_k$   (2.8)
      Terminate the loop if
          $f(y_k) - \min_{x\in\mathcal{X}} \xi_k(x) \leq \varepsilon$ where function $\xi_k(\cdot)$ is defined by
          $\xi_k(x) := (1-\gamma_k)\xi_{k-1}(x) + \gamma_k(f(z_k) + \langle \nabla f(z_k), x - z_k \rangle)$.   (2.9)
  end for
  Output approximate solution $y_k$ at the termination of the above for-loop.

  procedure $u^+ = \text{CndG}(g, u, \beta, \eta)$
    1. Set $u_1 = u$ and $t = 1$.
    2. Let $v_t$ be the optimal solution for the subproblem of
          $V_{g,u,\beta}(u_t) := \max_{x\in\mathcal{X}} \langle g + \beta(u_t - u), u_t - x \rangle$   (2.10)
    3. If $V_{g,u,\beta}(u_t) \leq \eta$, set $u^+ = u_t$ and terminate the procedure.
    4. Set $u_{t+1} = (1-\alpha_t)u_t + \alpha_t v_t$ where $\alpha_t = \min\{1, \langle \beta(u - u_t) - g, v_t - u_t \rangle\}/(\beta\|v_t - u_t\|^2)$
    5. Set $t \leftarrow t + 1$ and go to step 2.
  end procedure
```

parameter η_k. Note that the convergence analysis of the choice of Γ_k and γ_k in our proposed CGS-ls method is not discussed in [30]. In the sequel, we will show that the choice $\Gamma_k = L_k\gamma_k^3$ with a backtracking strategy for L_k yields our desired convergence result.

Third, the termination criterion (2.9) of the CGS-ls method is based on the linear lower bound function $\xi_k(x)$. In the sequel, we will prove that $\xi_k(x) \leq f(x)$ for all $x \in \mathcal{X}$. Consequently, whenever the termination criterion is satisfied, we have

$$f(y_k) - f^* = f(y_k) - f(x^*) \leq f(y_k) - \xi_k(x^*) \leq f(y_k) - \min_{x\in\mathcal{X}} \xi_k(x) \leq \varepsilon \tag{2.13}$$

where x^* is an optimal solution to problem (2.1). The above relation certificates that y_k is an approximate solution to problem (2.1). Such certification strategy for approximate solutions is previously discussed in [34] for convex optimization problems and implemented in accelerated gradient methods (see, e.g., [39]). In the sequel, we will prove that the termination criterion is satisfied with at most $\mathcal{O}(\sqrt{L/\varepsilon})$ gradient evaluations. One alternative termination criterion is the Wolfe gap

$$\max_{x\in\mathcal{X}} \langle \nabla f(y_k), y_k - x\rangle \leq \varepsilon$$

which is widely employed in the literature of conditional gradient methods. When the Wolfe gap termination criterion is satisfied, we also have $f(y_k) - f^* \leq \varepsilon$ due to the convexity of f:

$$f(y_k) - f^* = f(y_k) - f(x^*) \leq -\langle \nabla f(y_k), x^* - y_k\rangle \leq \max_{x\in\mathcal{X}} \langle \nabla f(y_k), y_k - x\rangle \leq \varepsilon. \tag{2.14}$$

However, one can only show that the termination criterion through Wolfe gap is satisfied with significantly worse number $\mathcal{O}(L/\varepsilon)$ of gradient evaluations (see, e.g., [21]).

Fourth, it is necessary to point out the backtracking strategy we implement in the proposed CGS-ls method. During implementation, we compute the estimated L_k in the following way: we start with $L_k = L_{k-1}$ and compute γ_k, Γ_k, z_k, x_k, and y_k in (2.4), (2.5), (2.6), (2.7), and (2.8). After the computation, we verify whether condition (2.3) is satisfied. If not, we will multiply L_k by 2 and backtrack all the values again. Such backtracking procedure stops when the Lipschitz condition (2.3) is satisfied. It should be noted that we can derive from the convexity of f and the Lipschitz condition (2.2) that

$$f(y) \leq f(z) + \langle \nabla f(z), y - z\rangle + \frac{L}{2}\|y - z\|^2,\ \forall y, z \in \mathcal{X}.$$

Therefore, if $L_0 \geq L_{min}$, where L_{min} is the smallest Lipschitz constant that satisfies the Lipschitz condition (2.2), then we have $L_k \equiv L_0 \geq L_{min}$, and CGS-ls reduces to CGS with Lipschitz constant L_0 and parameter $\Gamma_k = L_0\gamma_k^3$. If $L_0 < L_{min}$, it is straightforward to observe that the number of backtracking required throughout the entire iterates of the CGS-ls method is $\lceil \log(2L_{min}/L_0)\rceil$. This is because that whenever $L_k \geq 2L_{min}$, then the condition (2.3) is always satisfied. Summarizing the above description of L_k's and accounting for both the cases $L_0 \geq L_{min}$ and $L_0 < L_{min}$, we have the following relation:

$$L_0 \leq L_1 \leq \cdots \leq L_k \leq \max\{2L_{min}, L_0\}. \tag{2.15}$$

Finally, in order to compute γ_k when $k \geq 2$ it suffices to solve the positive root to a cubic polynomial equation $L_k\gamma_k^3 = \Gamma_{k-1}(1-\gamma_k)$. It is easy to verify that

$$\gamma_k = \sqrt[3]{\frac{\Gamma_{k-1}}{2L_k} + \frac{\Gamma_{k-1}}{L_k}\sqrt{\frac{1}{4} + \frac{\Gamma_{k-1}}{27L_k}}} + \sqrt[3]{\frac{\Gamma_{k-1}}{2L_k} - \frac{\Gamma_{k-1}}{L_k}\sqrt{\frac{1}{4} + \frac{\Gamma_{k-1}}{27L_k}}} \in (0,1),\ \forall k \geq 2. \tag{2.16}$$

is the unique positive real root we are looking for through the cubic formula. Here, to prove that $\gamma_k \in (0,1)$ for all $k \geq 2$, note that when $\Gamma_{k-1}/L_k \in (0,1)$, from the above cubic formula description of γ_k we have $\gamma_k > 0$. Also, applying $\Gamma_{k-1}/L_k \in (0,1)$ to the relation $L_k\gamma_k^3 = \Gamma_{k-1}(1-\gamma_k)$ we have $1-\gamma_k > \gamma_k^3 > 0$. Therefore $\Gamma_{k-1}/L_k \in (0,1)$ leads to $\gamma_k \in (0,1)$. Moreover, noting from the description of L_k in Algorithm 10 we have $L_k \geq L_{k-1}$, hence $\gamma_k \in (0,1)$ implies that $\Gamma_k/L_{k+1} \in (0,1)$. Therefore, applying induction we can prove that $\gamma_k \in (0,1)$ for all $k \geq 2$. Also, note that $\gamma_1 = 1$ in (2.4). Consequently, y_k in (2.8) is the convex combination of y_{k-1} and x_k; noting that $x_k \in \mathcal{X}$ and that $y_0 = x_0 \in \mathcal{X}$ we can have that the approximation solution $y_k \in \mathcal{X}$.

2.3 Theoretical Analysis

In this section we perform the convergence analysis of the proposed CGS-ls algorithm in Algorithm 10. We begin with two technical results that will be used in the analysis.

Lemma 1. *Suppose that $\{\lambda_i\}_{i\geq 1}$ and $\{a_i\}_{i\geq 0}$ are two sequences of nonnegative real numbers, in which the sequence $\{\lambda_i\}_{i\geq 1}$ is non-decreasing. For any fixed k, we have*

$$\sum_{i=1}^{k} \lambda_i(a_{i-1} - a_i) \leq \lambda_k \max_{0\leq t\leq k} a_t.$$

Proof. Since $\lambda_i \geq \lambda_{i-1} \geq 0$ for all $i = 2, 3, \ldots$, we have immediately that

$$\begin{aligned}\sum_{i=1}^{k} \lambda_i (a_{i-1} - a_i) &= \lambda_1 a_0 + \sum_{i=2}^{k} (\lambda_i - \lambda_{i-1}) a_{i-1} - \lambda_k a_k \\ &\leq \lambda_1 \max_{0 \leq t \leq k} a_t + \sum_{i=2}^{k} (\lambda_i - \lambda_{i-1}) \max_{0 \leq t \leq k} a_t \\ &= \lambda_k \max_{0 \leq t \leq k} a_t.\end{aligned}$$

□

Lemma 2. *In Algorithm 10, suppose that $\gamma_1 = 1$, $\gamma_k \in (0, 1)$, $k = 2, 3, \ldots$, and the value of Γ_k satisfies*

$$\Gamma_k := \begin{cases} 1, & k = 1, \\ \Gamma_{k-1}(1 - \gamma_k), & k \geq 2. \end{cases} \tag{2.17}$$

If the sequence $\{\delta_k\}_{k \geq 1}$ satisfies

$$\delta_k \leq (1 - \gamma_k)\delta_{k-1} + B_k, \; k = 1, 2, \ldots, \tag{2.18}$$

then for any $k \geq 1$ we have

$$\delta_k \leq \Gamma_k \sum_{i=1}^{k} \frac{B_i}{\Gamma_i}.$$

In particular, the above inequality becomes equality when the relations in (2.18) *are all equality relations.*

Proof. The result follows from dividing both sides of (2.18) by Γ_k and then summing up the resulting inequalities or equalities. □

A few remarks are in place regarding the above lemma. First, by the descriptions of γ_k and Γ_k in Algorithm 10, the condition (2.17) is clearly satisfied. Second, applying the above lemma with

$\delta_k \equiv 1$ and $B_k = \gamma_k$ we have the following equality:

$$\Gamma_k \sum_{i=1}^{k} \frac{\gamma_i}{\Gamma_i} = 1. \tag{2.19}$$

Similarly, applying the above lemma to the definition of the lower bound function $\xi_k(\cdot)$ in (2.9) (with $\delta_k = \xi_k(x)$ and $B_k = \gamma_k(f(z_k) + \langle \nabla f(z_k), x - z_k\rangle))$ we also have

$$\xi_k(x) = \Gamma_k \sum_{i=1}^{k} \frac{\gamma_i}{\Gamma_i}(f(z_i) + \langle \nabla f(z_i), x - z_i\rangle). \tag{2.20}$$

We are now ready to analyze the convergence of the proposed CGS-ls algorithm. Theorem 1 below describes the main convergence property of Algorithm 10.

Theorem 1. *Suppose that the parameters in Algorithm 10 satisfy $\beta_k \geq L_k\gamma_k$ for all k. Then we have*

$$f(y_k) - \xi_k(x) \leq \frac{\varepsilon}{2} + \Gamma_k \sum_{i=1}^{k} \frac{\gamma_i\beta_i}{2\Gamma_i}\left(\|x - x_{i-1}\|^2 - \|x - x_i\|^2\right) + \Gamma_k \sum_{i=1}^{k} \frac{\gamma_i\eta_i}{\Gamma_i},\ \forall x \in \mathcal{X}.$$

Proof. Let us fix any $x \in \mathcal{X}$. In order to prove the result, we will estimate a lower bound of $\xi_k(x)$. From the description of y_k in (2.8) we observe that $\gamma_k(x_k - z_k) = (y_k - z_k) - (1 - \gamma_k)(y_{k-1} - z_k)$. Applying such observation to the description of $\xi_k(x)$ in (2.20) we have

$$\begin{aligned}\frac{1}{\Gamma_k}\xi_k(x) &= \sum_{i=1}^{k} \frac{1}{\Gamma_i}\left[\gamma_i f(z_i) + \gamma_i \langle \nabla f(z_i), x - x_i\rangle + \langle \nabla f(z_i), \gamma_i(x_i - z_i)\rangle\right].\\ &= \sum_{i=1}^{k} \frac{1}{\Gamma_i}[f(z_i) + \langle \nabla f(z_i), y_i - z_i\rangle - (1 - \gamma_i)\left(f(z_i) + \langle \nabla f(z_i), y_{i-1} - z_i\rangle\right)\\ &\qquad + \gamma_i \langle \nabla f(z_i), x - x_i\rangle].\end{aligned}$$

We make three observations in the above equation. First, by (2.3) we have

$$f(z_i) + \langle \nabla f(z_i), y_i - z_i\rangle \geq f(y_i) - \frac{L_i}{2}\|y_i - z_i\|^2 - \frac{\varepsilon}{2}\gamma_i = f(y_i) - \frac{L_i\gamma_i^2}{2}\|x_i - x_{i-1}\|^2 - \frac{\varepsilon}{2}\gamma_i.$$

Here the last equality is from the descriptions of z_k and y_k in (2.6) and (2.8) respectively. Second, by the convexity of f we have

$$-\left(f(z_i) + \langle \nabla f(z_i), y_{i-1} - z_i\rangle\right) \geq -f(y_{i-1}).$$

Third, by the stopping criterion of the CndG procedure in (2.11) and our assumption that $\beta_k \geq L_k\gamma_k$ for all k, we have

$$\begin{aligned}\gamma_i \langle \nabla f(z_i), x - x_i\rangle \geq{}& \gamma_i\beta_i\langle x_i - x_{i-1}, x_i - x\rangle - \gamma_i\eta_i \\ ={}& -\frac{\gamma_i\beta_i}{2}\left(\|x - x_{i-1}\|^2 - \|x_i - x_{i-1}\|^2 - \|x - x_i\|^2\right) - \gamma_i\eta_i \\ \geq{}& -\frac{\gamma_i\beta_i}{2}\left(\|x - x_{i-1}\|^2 - \|x - x_i\|^2\right) - \gamma_i\eta_i + \frac{L_i\gamma_i^2}{2}\|x_i - x_{i-1}\|^2.\end{aligned}$$

Applying the above three observations and recalling that $\gamma_1 = 1$ and $\gamma_k \in (0,1)$ for all $k \geq 2$ in (2.4) and (2.16) respectively, we obtain that

$$\frac{1}{\Gamma_k}\xi_k(x) \geq \sum_{i=1}^{k}\frac{1}{\Gamma_i}\left[f(y_i) - (1-\gamma_i)f(y_{i-1}) - \frac{\gamma_i\beta_i}{2}\left(\|x - x_{i-1}\|^2 - \|x - x_i\|^2\right) - \frac{\varepsilon}{2}\gamma_i - \gamma_i\eta_i\right].$$

In the above result, noting from the relation (2.17) between γ_k and Γ_k and the fact that $\gamma_1 = 1$, we have

$$\sum_{i=1}^{k}\frac{1}{\Gamma_i}f(y_i) - \frac{1-\gamma_i}{\Gamma_i}f(y_{i-1}) = \frac{f(y_k)}{\Gamma_k}.$$

We conclude the theorem immediately by combining the above two equations and using the relation (2.19). $\square$

Corollary 1. *Suppose that the parameters of Algorithm 10 are set to*

$$\beta_k = L_k\gamma_k \text{ and } \eta_k = \frac{L_k\gamma_kD^2}{k}, \tag{2.21}$$

where D is any constant that estimates the diameter $D_{\mathcal{X}} := \max_{x,y\in\mathcal{X}}\|x-y\|$ of $\mathcal{X}$. Then Algorithm 10 terminates with an ε-approximate solution y_k after $k \geq N_{grad}$ gradient evaluations, in which

$$N_{grad} := C\sqrt{\frac{\max\{2L_{min}, L_0\}D_{\mathcal{X}}^2}{\varepsilon}}, \text{ where } C = \sqrt{\frac{27}{2} + \frac{27D^2}{D_{\mathcal{X}}^2}\sqrt[6]{\frac{\max\{2L_{min}, L_0\}}{L_0}}}. \tag{2.22}$$

At termination, the total number of linear objective optimization (the problem in (2.10)*) is bounded by*

$$N_{lin} := \frac{6D_{\mathcal{X}}^2}{D^2}\frac{C^2\max\{2L_{min}, L_0\}D_{\mathcal{X}}}{\varepsilon} + C\sqrt{\frac{\max\{2L_{min}, L_0\}D_{\mathcal{X}}^2}{\varepsilon}}. \tag{2.23}$$

Here L_{min} is the smallest Lipschitz constant that satisfies the Lipschitz condition (2.2) *of the gradient ∇f.*

Proof. Applying Theorem 1 with the parameters described in (2.21), and noting from the description of γ_k and Γ_k in Algorithm 10 that $\Gamma_k = L_k\gamma_k^3 = (1-\gamma_k)\Gamma_{k-1}$, we have

$$f(y_k) - \xi_k(x) \le \frac{\varepsilon}{2} + \Gamma_k\sum_{i=1}^{k}\frac{1}{2\gamma_i}\left(\|x - x_{i-1}\|^2 - \|x - x_i\|^2\right) + \Gamma_k\sum_{i=1}^{k}\frac{D^2}{i\gamma_i},\ \forall x \in \mathcal{X}$$

Since $\gamma_k \in (0,1)$ for all $k \ge 2$ (see (2.16)), we observe that $L_k\gamma_k^3 = \Gamma_k = (1-\gamma_k)\Gamma_{k-1} < \Gamma_{k-1} = L_{k-1}\gamma_{k-1}^3$ for all $k \ge 2$. using this observation and noting from (2.15) that $L_k \ge L_{k-1}$, we have $\gamma_k^3 < \gamma_{k-1}^3$. Consequently, the sequence $\{1/\gamma_i\}_{i\ge 1}$ at the right hand side of the above estimate of $f(y_k) - \xi_k(x)$ is an increasing sequence. Applying Lemma 1 we have

$$f(y_k) - \xi_k(x) \le \frac{\varepsilon}{2} + \frac{\Gamma_k}{2\gamma_k}\max_{0\le i\le k}\|x - x_i\|^2 + \Gamma_k\sum_{i=1}^{k}\frac{D^2}{i\gamma_i} \le \frac{\varepsilon}{2} + \frac{\Gamma_k}{2\gamma_k}D_{\mathcal{X}}^2 + \Gamma_k\sum_{i=1}^{k}\frac{D^2}{i\gamma_i}. \tag{2.24}$$

Here in the last inequality we use the definition of diameter $D_{\mathcal{X}}$. We will estimate the right most side of the above relation.

Using the relation $\Gamma_k = L_k\gamma_k^3 = (1-\gamma_k)\Gamma_{k-1}$ again, we have

$$\sqrt[3]{\frac{1}{\Gamma_k}} - \sqrt[3]{\frac{1}{\Gamma_{k-1}}} = \frac{\frac{1}{\Gamma_k} - \frac{1}{\Gamma_{k-1}}}{\sqrt[3]{\frac{1}{\Gamma_k^2}} + \sqrt[3]{\frac{1}{\Gamma_k\Gamma_{k-1}}} + \sqrt[3]{\frac{1}{\Gamma_{k-1}^2}}} = \frac{\frac{\gamma_k}{\Gamma_k}}{\sqrt[3]{\frac{1}{\Gamma_k^2}} + \sqrt[3]{\frac{1}{\Gamma_k\Gamma_{k-1}}} + \sqrt[3]{\frac{1}{\Gamma_{k-1}^2}}}.$$

Noting that $\gamma_k \in (0,1)$ (see (2.16)) and recalling that $\Gamma_k = (1-\gamma_k)\Gamma_{k-1}$ we have $\Gamma_k \le \Gamma_{k-1}$. Therefore, we have

$$\sqrt[3]{\frac{1}{\Gamma_k^2}} \le \sqrt[3]{\frac{1}{\Gamma_k^2}} + \sqrt[3]{\frac{1}{\Gamma_k\Gamma_{k-1}}} + \sqrt[3]{\frac{1}{\Gamma_{k-1}^2}} \le 3\sqrt[3]{\frac{1}{\Gamma_k^2}}$$

Recalling that $\Gamma_k = L_k\gamma_k^3$, the above two relations imply that

$$\frac{1}{\sqrt[3]{L_k}} \geq \sqrt[3]{\frac{1}{\Gamma_k}} - \sqrt[3]{\frac{1}{\Gamma_{k-1}}} \geq \frac{1}{3\sqrt[3]{L_k}}.$$

Here, recalling the relations of L_k's in (2.15), we have

$$\frac{1}{\sqrt[3]{L_0}} \geq \sqrt[3]{\frac{1}{\Gamma_k}} - \sqrt[3]{\frac{1}{\Gamma_{k-1}}} \geq \frac{1}{3\sqrt[3]{\max\{2L_{min}, L_0\}}}.$$

Summing the above relation from 1 to k we obtain that

$$\frac{k-1}{\sqrt[3]{L_0}} \geq \frac{1}{\sqrt[3]{\Gamma_k}} - \frac{1}{\sqrt[3]{\Gamma_1}} \geq \frac{k-1}{3\sqrt[3]{\max\{2L_{min}, L_0\}}}.$$

Recalling from (2.15) that $L_0 \leq L_1 \leq \max\{2L_{min}, L_0\}$ and noting from (2.4) and (2.5) that $\Gamma_1 = L_1$, the above becomes

$$\frac{k}{\sqrt[3]{L_0}} \geq \frac{1}{\sqrt[3]{\Gamma_k}} \geq \frac{k-1}{3\sqrt[3]{\max\{2L_{min}, L_0\}}} + \frac{1}{\sqrt[3]{\max\{2L_{min}, L_0\}}} > \frac{k}{3\sqrt[3]{\max\{2L_{min}, L_0\}}},$$

i.e.,

$$\frac{L_0}{k^3} \leq \Gamma_k \leq \frac{27\max\{2L_{min}, L_0\}}{k^3}.$$

Using the first inequality above and recalling the relations $\Gamma_k = L_k\gamma_k^3$ and $L_0 \leq L_k$ we have

$$\gamma_k \geq \sqrt[3]{\frac{L_0}{L_k k^3}} \geq \frac{1}{k}\sqrt[3]{\frac{L_0}{\max\{2L_{min}, L_0\}}}.$$

Applying the above two results to (2.24), we conclude that

$$\begin{aligned} & f(y_k) - \xi_k(x) \\ \leq & \frac{\varepsilon}{2} + \frac{27\max\{2L_{min}, L_0\}D_{\mathcal{X}}^2}{2k^2} \cdot \sqrt[3]{\frac{\max\{2L_{min}, L_0\}}{L_0}} + \frac{27\max\{2L_{min}, L_0\}D^2}{k^3}\sum_{i=1}^{k}\sqrt[3]{\frac{\max\{2L_{min}, L_0\}}{L_0}} \\ = & \frac{\varepsilon}{2} + \frac{\max\{2L_{min}, L_0\}D_{\mathcal{X}}^2}{k^2}\left[\frac{27}{2} + \frac{27D^2}{D_{\mathcal{X}}^2}\right]\sqrt[3]{\frac{\max\{2L_{min}, L_0\}}{L_0}}, \ \forall x \in \mathcal{X}. \end{aligned}$$

Noting the above result and (2.13), we conclude that the proposed CGS-ls algorithm will terminate with an ε-approximate solution after $k \geq N_{grad}$ iterations, where N_{grad} is defined in (2.22). Also, by Theorem 2.2(c) in [30] and our parameter setting (2.21), the total number of linear objective optimization that is performed in the k-th call to the CndG procedure is bounded by

$$T_k := \left\lceil \frac{6\beta_k D_{\mathcal{X}}^2}{\eta_k} \right\rceil = \left\lceil \frac{6D_{\mathcal{X}}^2}{D^2} k \right\rceil$$

Therefore, at termination the total number of linear objective optimization that is performed by Algorithm 10 is bounded by

$$N_{lin} := \sum_{i=1}^{N_{grad}} T_i \leq \sum_{i=1}^{N_{grad}} T_i \left(\frac{6D_{\mathcal{X}}^2}{D^2} i + 1 \right) = \frac{6D_{\mathcal{X}}^2}{D^2} N_{grad}^2 + N_{grad}.$$

Substituting the value of N_{grad} in (2.22) we obtain (2.23). $\square$

A few remarks are in place for the above corollary. First, from (2.22) and (2.23) we conclude that the proposed CGS-ls method has the same theoretical convergence property as that of the CGS method in [30]. Specifically, to compute an ε-approximate solution, the CGS-ls method reaches the theoretical performance limit by requiring at most $\mathcal{O}(\sqrt{L/\varepsilon})$ gradient evaluations and $\mathcal{O}(L/\varepsilon)$ linear objective optimizations, where $L := \max\{2L_{min}, L_0\}$ is a Lipschitz constant that satisfies the Lipschitz condition (2.2). Second, in our parameter setting (2.21) we need to choose an estimate D for the exact diameter $D_{\mathcal{X}}$. However, as long as D is relatively close to $D_{\mathcal{X}}$ (e.g., smaller or larger than $D_{\mathcal{X}}$ but within an order of $\mathcal{O}(1/\sqrt{\varepsilon})$), our convergence properties will not be affected. Finally, when the initial guess of Lipschitz constant L_0 is larger than L_{min}, no backtracking linesearch will be performed, and the proposed CGS-ls method becomes a version of the CGS method in [30] that is equipped with a proper termination criterion. However, unlike the CGS method, when the initial guess L_0 is significantly smaller than L_{min}, the convergence property of the CGS-ls method will not be affected significantly. As an example, if we choose $L_0 = 0.001L_{min}$, then in the convergence result of the above corollary we have $\sqrt[6]{\max\{2L_{min}, L_0\}/L_0} = \sqrt[6]{2000} \approx 3.5$, namely, the number of gradient evaluations will be enlarged by a constant of approximately 3.5. In terms of theoretical convergence, such enlargement will not change the order of the gradient evaluations in terms of its dependence on $1/\varepsilon$. However, in terms of numerical implementation, we have the flexibility of choosing much smaller choice of L_0. Moreover, if the smaller choice of L_0 is satisfied along the iterates of the CGS-ls method, then we can expect that the practical performance of CGS-ls is much faster than algorithms that use a conservative choice of global Lipschitz constant estimate.

2.4 Numerical Results

In this section we present the results from our numerical experiments. We will compare the performance of the proposed CGS-ls method with that of the Frank-Wolfe (FW) method (described in (1.14) with weights $\lambda_i = 2i/(k(k+1))$) and the CGS method in [30] (parameters follow Corollary 2.3 in the paper). We consider two quadratic optimization problems with different subsets; the first is over the standard spectrahedron and the second is over the convex hull of all Hamiltonian cycles. In all of the experiments, the coefficient matrix of decision variable in the objective is generated in a way to that the objective function is not a strongly convex function. To this end, we generated rectangular and sparse matrices so that half of the singular values are zero and the other half are uniformly randomly selected values. To generate such a matrix we used the fact that for given matrices $M_{m\times n}$, $U_{p\times m}$, and $V_{n\times q}$ where $p \geq m$ and $q \geq n$ and U has orthonormal columns, then the largest $\min\{m, n\}$ singular values of UMV are equal to the singular values of M. Also, all numerical experiments are performed on a compute with Intel Core i5 2.7 GHz CPU.

In the first numerical experiment, we consider the optimization problem over the standard spectrahedron:

$$\min_{X\in \mathrm{Spe}_n} f(X) := \frac{1}{2}\left\|\mathcal{A}X - B\right\|_F^2 \quad \text{where} \quad \mathrm{Spe}_n := \{X \in \mathbb{R}^{n\times n} : \mathrm{Tr}(X) = 1,\ X \succeq 0\}.$$

Here $\mathcal{A} : \mathbb{R}^{n\times n} \to \mathbb{R}^m$ is a linear operator, $\|\cdot\|_F$ is the Frobenius norm, and the feasible set Spe_n is a standard spectrahedron. Note that the linear objective optimization over the spectrahedron can be solved by computing a maximum eigenvalue problem (see, e.g., [18]). For each random instance in this numerical experiment, we generate $\mathcal{A}$ first by equivalently generating a $m \times n^2$ matrix with 20%, 60% or 80% nonzero entries. We then generate $B = \mathcal{A}U\Sigma U^\top$ where U is a random orthogonal matrix and Σ is a diagonal matrix with uniformly random entries between 0 and 1 (normalized afterwards so that they sum to 1). Therefore, the optimal value of all generated instances are 0.

In the second numerical experiment, we consider the optimization problem over the convex hull of Hamiltonian cycles in [29]:

$$\min_{x\in\mathcal{H}} f(x) := \frac{1}{2}\left\|Ax-b\right\|_2^2 \quad \text{where} \quad \mathcal{H} = \text{conv}\{x \in \mathbb{R}^{n(n-1)/2} : x \text{ is a Hamiltonian cycle}\}.$$

Here $A : \mathbb{R}^{m\times[n(n-1)/2]}$, $b \in \mathbb{R}^m$, and the feasible set $\mathcal{H}$ is the convex hull of all Hamiltonian cycles in a complete graph with n nodes. We describe any Hamiltonian cycle through a vector of dimension $n(n-1)/2$ (the lower triagular part of the adjacency matrix). Note that the linear objective optimization over $\mathcal{H}$ can be solved by computing the solution to a traveling salesman problem (solved through Gurobi [17]). For each random instance, A is randomly generated with 60% nonzero entries that follow i.i.d. standard uniform distribution. We then generate $b = A(0.8v_1 + 0.2v_2)$ where v_1 and v_2 are two Hamiltonian cycles that are generated from random permutations of all nodes. The optimal value of all generated instances are 0.

Table 2.1: Comparison of FW, CGS and CGS-ls on the first numerical experiment (minimization over standard spectrahedron).

Ins. info.		FW			CGS				CGS-ls			
i	d	iter.	time	obj.	outer	inner	time	obj.	outer	inner	time	obj.
1	.2	188453	1800	1e-2	194883	517477	1800	4e-2	43234	86585	418	1e-4
2	.2	62861	1800	1e-1	64972	146850	1800	1e-2	62899	126219	1800	2e-10
3	.2	141408	1800	4e-2	158672	327107	1800	4e-2	60033	120192	756	2e-8
4	.2	35458	1800	1e-1	43618	87893	1800	1e-1	43471	95072	1800	3e-10
5	.2	79828	1800	9e-2	95879	192622	1800	1e-1	75597	186038	1161	6e-10
6	.2	20883	1800	5e-1	25917	51953	1800	5e-1	29863	59817	1513	1e-8
7	.6	98389	1800	5e-2	108327	218164	1800	7e-2	31733	87003	1623	4e-9
8	.6	26998	1800	3e-1	33899	68454	1800	4e-1	39386	78888	1800	1e-7
9	.6	55859	1800	9e-2	68422	137002	1800	9e-2	41865	83839	865	4e-9
10	.6	15235	1800	3e-1	19328	38836	1800	3e-1	24512	49186	1800	5e-7
11	.6	29364	1800	6e-1	37746	75519	1800	9e-1	51114	124652	1800	1e-9
12	.6	7824	1800	1.74	10199	20440	1800	1.51	14314	28884	1800	3e-6
13	.8	79549	1800	3e-2	94297	189371	1800	7e-2	47435	94966	1800	1e-8
14	.8	21604	1800	5e-1	26884	54544	1800	5e-1	32702	65585	1800	1e-7
15	.8	42895	1800	1e-1	53966	108086	1800	1e-1	46977	94078	1196	2e-8
16	.8	11624	1800	1.08	15008	30131	1800	1.49	19661	39437	1800	4e-8
17	.8	22481	1800	8e-1	27159	54326	1800	1.27	35653	86930	1800	6e-9
18	.8	5930	1800	1.50	7805	15639	1800	2.50	11095	22415	1800	1e-6

We report the performance of FW, CGS, and CGS-ls in the above two numerical experiments in Table 2.1 and Table 2.2 respectively. For FW and CGS, we terminate when the Wolfe gap described in (2.14) is smaller than 0.01; for CGS-ls, we terminate when the gap describe in (2.9) is smaller than 0.01. Consequently, all algorithms terminate either when an approximate solution is certified with $\varepsilon = 0.01$ (or when the algorithm runs over the given time limit). Note that CGS

Table 2.2: Comparison of FW, CGS and CGS-ls on the second numerical experiment (minimization over the convex hull of Hamiltonian cycles).

Ins. info.		FW			CGS				CGS-ls			
j	d	iter.	time	obj.	outer	inner	time	obj.	outer	inner	time	obj.
1	.6	119493	1800	1e-3	3572	73088	1800	4.97	14883	41018	697	1e-3
2	.6	66270	1800	1e-3	3045	50207	1800	2.64	16648	40847	960	1e-3
3	.6	137959	1800	4e-1	84146	169969	1800	2e-1	67407	170967	1800	2e-2
4	.6	65314	1800	7e-1	41322	86060	1800	3.61	33539	75476	1800	8e-2
5	.6	46395	1800	2.11	32210	66686	1800	0.35	22472	62433	1800	1e-1
6	.6	36408	1800	1.51	16152	46154	1800	33.8	21217	43854	1800	1e-1
7	.6	269203	3600	1e-1	187308	375625	3600	1e-1	126262	273593	3600	5e-2
8	.6	196834	3600	1e-2	118145	238476	3600	4e-1	122067	256582	3600	3e-2
9	.6	129473	3600	2e-1	90814	194371	3600	2e-1	66579	169998	3600	1e-2
10	.6	78726	3600	2e-1	40957	91986	3600	1.82	39437	101188	3600	3e-2

Table 2.3: Indices and their corresponding size of instances in Tables 2.1 and 2.2

.

i in Table 2.1				j in Table 2.2			
i	m x n	i	m x n	j	m x n(n-1)/2	j	m x n(n-1)/2
1	500x10000	10	1000x20000	1	500x300	10	10000x595
2	500x20000	11	2000x10000	2	500x435		
3	1000x10000	12	2000x20000	3	1000x190		
4	1000x20000	13	500x10000	4	1000x435		
5	2000x10000	14	500x20000	5	1000x595		
6	2000x20000	15	1000x10000	6	1000x780		
7	500x10000	16	1000x20000	7	10000x105		
8	500x20000	17	2000x10000	8	10000x190		
9	1000x10000	18	2000x20000	9	10000x300		

requires the Lipschitz constant of the objective function; we compute them through the maximum eigenvalue of the Hessian of the objective function and rescaling to get the desired large value (the time for computing maximum eigenvalue is not counted towards CGS' computation time). In the first experiment $\mathcal{A}$ is generated so that $L = 1.5e4$ and in the second experiment A is generated so that $L = 1e3$. For CGS-ls, we set $L_0 = .001L$ for all instances, and $D_{\text{Spe}_n} = 0.005\sqrt{2}$ and $D_{\mathcal{H}} = 0.05\sqrt{n(n-1)/2}$ for the first and second numerical experiments, respectively. We report the running time (in seconds) and the objective value of the approximate solution at termination for all algorithms. For FW, we report its total number of iterations, which is the same as the number of gradient evaluations and linear optimization subproblems. For CGS and CGS-ls, we report the total number of gradient evaluations (denoted "outer") and linear optimization subproblems (denoted as "inner"). We also compared the performance of the three algorithms with PAGD and for a small instance of size 100×45 the improvement in objective value is negligible after 60 minutes and performing only 4 iteration due to expensive calculation of the projection onto the feasible set.

We make a few remarks from Tables 2.1 and 2.2. First, among the three algorithms, FW is the simplest to implement but has the worst performance; in most of the instances it could not obtain an approximate solution with Wolfe gap smaller than 0.01 within the required 30 or 60 minute computation time limit. Such behavior is consistent with its theoretical complexity $\mathcal{O}(L/\varepsilon)$, which is the worst among the three algorithms. Second, CGS-ls has better practical performance than CGS in most instances, although both algorithms have the same theoretical convergence properties. The better practical performance is most likely due to the adaptive estimate of Lipschitz constant, which avoids the potentially conservative Lipschitz constant that CGS may suffer throughout the computation. Finally, it is interesting to observe that most objective values of the approximate solutions computed by all algorithm at termination are much better than the accuracy setup $\varepsilon = 0.01$. To the best of our knowledge, it is still unclear in the literature whether there exists termination criterion other than the ones we use in this paper ((2.9) and (2.14)) that could guarantee that the approximate solution at termination is an ε-solution while achieving good practical performance. We leave the study of better termination criterion as a future work.

Chapter 3

Sliding Accelerated Bundle-Level Method

In the introduction chapter, we introduced ABL and APL as two bundle-level type methods. Also, recall that we classified these methods as projection-based methods as they require solving projection subproblems throughout their iterations. We also discussed in the same chapter that CGS algorithm utilizes a FW-type method to solve the projection subproblem in each iteration of the NAGD method approximately and iteratively. Knowing these, can we design a CGS counterpart for the projection subproblem of the ABL/APL, namely, a bundle-level type CGS? Addressing this question motivates us to propose the *sliding APL* (SAPL) method. In this chapter, we will introduce the SAPL algorithm and will also discuss its convergence analysis. Note that using SAPL, we aim to solve the convex programming (CP)

$$f^* := \min_{x \in X} f(x) \tag{3.1}$$

where X is a convex compact set and $f : X \to \mathbb{R}$ is a closed convex function and satisfies

$$f(y) - f(x) - \langle f'(x), y - x \rangle \le \frac{M}{1+\nu} \|y - x\|^{1+\nu}, \quad \forall x, y \in X, \tag{3.2}$$

for some $M > 0$, $\nu \in [0, 1]$ and $f' \in \partial f(x)$. This class of problems cover nonsmooth ($\nu = 0$), smooth ($\nu = 1$), and weakly smooth ($\nu \in (0, 1)$) CP problems. However, our focus in our proposed algorithm is on the class of functions smooth f where $\nu = 1$ in (3.2). In that case, we will use the notation L instead of M as the Lipschitz constant of ∇f.

3.1 Proposed Algorithm

The basic scheme of this method is obtained by applying Frank-Wolfe (FW) method to solve the projection subproblem existing in APL approximately. We will show that if this accuracy is specified properly, SAPL can achieve the optimal bounds on the number of calls to the first-order and linear optimization oracles for solving (3.1) when f is smooth. The general scheme of this method is presented in Algorithm 11.

A few remarks are in place for SAPL method. First, similar to the APL method, the SAPL method runs through phases in its outer layers of iterations and in each phase through the gap reduction procedure it updates the desired bound ℓ at initial step of $\mathcal{G}_{SAPL}$ and in (3.3) with a fixed and prespecified convex combination parameter β. Second, unlike the APL method, the SAPL method updates the prox-center in Step 3, differently. We can observe that the $\mathcal{FW}$ procedure in the algorithm can be view as a specialized version of the the FW method that is applied to the projection subproblem in (1.44) of the APL. In particular, if we let $f := \langle g, x\rangle + \beta \left\|x - u\right\|^2/2$ then $V_{g,u,\beta}(u_t)$ in step 3 is equivalent to $\max_{x\in X'_{k-1}} \langle \nabla f(u_t), u_t - x\rangle$ which is know is also the Wolfe gap. The $\mathcal{FW}$ procedure terminates when $V_{g,u,\beta}(u_t)$ is smaller than the specified tolerance η. This procedure is the simplified version of FW method in the way that it updates α_t in Step 4 explicitly. This was initially suggested by Frank and Wolfe to specify the stepsize for the FW method through the minimization of an upper quadratic approximation of $f(\cdot)$ at x_k [6,8,15]. Note that FW method updates α_t by solving

$$\alpha_t = \operatorname*{arg\,min}_{\alpha\in[0,1]} f((1-\alpha)u_t + \alpha v_t).$$

Third, the localizer in step 4 of $\mathcal{G}_{SAPL}$ can be any set between $\underline{X}_k$ and $\bar{X}_k$. While constraints accumulate in $\underline{X}_k$, the set $\bar{X}_k$ is just a half-space. This flexibility simplifies the subproblem (3.4) both theoretically and practically. Finally, similar to the APL method, the gap reduction procedure of SAPL algorithm terminates in either Step 1 or Step 3 of $\mathcal{G}_{SAPL}$.

Algorithm 11 Sliding accelerate prox-level algorithm (SAPL)

```
Choose initial point $p_0 \in X$, tolerance $\epsilon > 0$ and parameters $\beta, \theta, \eta \in (0,1)$
  0. Set $p_1 \in \operatorname{Arg\,min}_{x \in X} h(p_0, x)$, $lb_1 = h(p_0, p_1)$, $ub = f(p_1)$ and $s = 1$.
  1. If $ub_s - lb_s \le \epsilon$ terminate the algorithm with output $p_s$.
  2. Set $(p_{s+1}, lb_{s+1}) = \mathcal{G}_{SAPL}(p_s, lb_s, \beta, \theta)$ and $ub_{s+1} = f(p_{s+1})$.
  3. $s \leftarrow s + 1$ and go to 1.
gap reduction procedure $(p^+, lb^+) = \mathcal{G}_{APL}(p, lb, \beta, \theta)$
  0. Set
     $$x_0^u = p, \quad \bar{f}_0 = f(x_0^u), \quad \underline{f}_0 = lb, \quad \ell = \beta \underline{f}_0 + (1-\beta)\bar{f}_0 \qquad (3.3)$$
     $$x_0 = p, \quad X_0' = X, \quad k = 1, \quad d_\omega(x) = \frac{1}{2}\|x - x_{k-1}\|_2^2$$
  1. lower bound update:
     $$x_k^\ell = (1-\alpha_k)x_{k-1}^u + \alpha_k x_{k-1},$$
     $$\underline{h}_k := \min_{x \in X_{k-1}'} h(x_k^\ell, x) \qquad (3.4)$$
     $$\underline{f}_k := \max\{\underline{f}_{k-1}, \min\{\ell, \underline{h}_k\}\}$$
     If $\underline{f}_k \ge (1-\theta)\ell + \theta \underline{f}_0$ terminate the procedure with $p^+ = x_{k-1}^u$, $lb^+ = \underline{f}_k$.
  2. prox-center update: $x_k = \mathcal{FW}(\underline{X}_{k-1}, d_\omega(\cdot), x_{k-1}, \eta_k)$
  3. upper bound update:
     $$\bar{f}_k = \min\{\bar{f}_{k-1}, f(\alpha_k x_k + (1-\alpha_k)x_{k-1}^u)\}$$
     choose $x_k^u$ so that $f(x_k^u) = \bar{f}_k$
     If $\bar{f}_k \le (1-\theta)\ell + \theta \bar{f}_0$ terminate the procedure with $p^+ = x_k^u$, $lb^+ = \underline{f}_k$
  4. update localizer: Choose $X_k'$ so that $\underline{X}_k \subseteq X_k' \subseteq \bar{X}_k$ where
     $$\underline{X}_k := \{x \in X_{k-1}' \ : \ h(x_k^\ell, x) \le \ell\}, \qquad \bar{X}_k := \{x \in X \ : \ \langle \nabla d_\omega(x_k), x_k - x\rangle \le \eta\} \qquad (3.5)$$
end $\mathcal{G}_{APL}$ procedure
$\mathcal{FW}$ procedure $u^+ = \mathcal{FW}(X, d_\omega, u, \eta)$
  1. Set $u_1 = u$ and $t = 1$.
  2. Let $v_t$ be the optimal solution for the subproblem of
     $$V_{g,u}(u_t) := \max_{x \in X} \langle u_t - u, u_t - x\rangle \qquad (3.6)$$
  3. If $V_{g,u}(u_t) \le \eta$, set $u^+ = u_t$ and terminate the procedure.
  4. Set $u_{t+1} = (1-\alpha_t)u_t + \alpha_t v_t$ where $\alpha_t = \min\left\{1, \langle u - u_t, v_t - u_t\rangle / \|v_t - u_t\|^2\right\}$
  5. Set $t \leftarrow t + 1$ and go to step 2.
end $\mathcal{FW}$ procedure
```

3.2 Theoretical Analysis

In this section we discuss the convergence results of the SAPL method and it analysis. Theorem 2 first finds a bound on $f(x_k^u) - \ell$ that is critical for termination of $\mathcal{G}_{SAPL}$ in Steps 1 and 3 of this procedure, and second it finds a conceptual bound on the number of iterations performed by $\mathcal{G}_{SAPL}$ in a typical phase of SAPL. And finally Theorem 2 find a bound on number of calls to the linear optimization oracles in the most inner iteration of SAPL and in $\mathcal{FW}$ procedure in one typical iteration $\mathcal{G}_{SAPL}$.

Theorem 2. *Let $\gamma_k \in (0,1]$, and $x_k^\ell, x_k, x_k^u \in X$ for $k = 1, 2, \cdots$ be given, where ℓ is the bound of the level set in $\mathcal{G}_{SAPL}$. Then*

a) For $k \geq 1$ we have

$$f(x_k^u) - \ell \leq (1 - \gamma_1) f(x_0^u - \ell) + \frac{LD_X^2}{2} \max_{1 \leq i \leq k} \{\gamma_i^2 / \Gamma_i\} \left(D_X^2 + \sum_{i=1}^{k-1} \eta_i \right) \tag{3.7}$$

for $k \geq 1$, where Γ_k is defined in (2.17).

b) If γ_k, $k = 1, 2, \cdots$ are chosen so that for some $c > 0$,

$$\gamma_1 = 1, \qquad \text{and} \qquad \Gamma_k \max_{1 \leq i \leq k} \{\gamma_i^2 / \Gamma_i\} \leq c k^{-2} \qquad \text{and} \qquad \sum_{i=1}^{k-1} \eta_i \leq D_X^2, \tag{3.8}$$

then the number of iterations performed by $\mathcal{G}_{SAPL}$ can be bounded by

$$K_{SAPL}(\Delta_0) := \left\lceil \sqrt{\frac{cLD_X^2}{\beta \theta \Delta_0}} \right\rceil \tag{3.9}$$

where $\Delta_0 = \bar{f}_0 - \underline{f}_0$.

c) Under assumptions in part a) and b) the number of iterations performed by $\mathcal{FW}$ procedure is bounded by

$$K_{\mathcal{FW}}(k) = \left\lceil \frac{6 D_X^2}{\eta_k} \right\rceil \tag{3.10}$$

Proof. First we show part a. Note that by definition of $\bar{X}$ in (3.5)

$$\langle x - x_0, x_k - x\rangle \le \eta_k \qquad \forall x \in X' \subseteq \bar{X}.$$

Therefore, since $x_{k+1} \in X'_k$ for $k \ge 1$ we have

$$\begin{aligned}\|x_{k+1} - x_k\|^2 &\le \langle x_k - x_0, x_{k+1} - x_k\rangle \\ &= \|x_{k+1} - x_0\|^2 - \|x_k - x_0\|^2 - \langle x_{k+1} - x_0, x_{k+1} - x_k\rangle \\ &\le \|x_{k+1} - x_0\|^2 - \|x_k - x_0\|^2 + \eta_k.\end{aligned}$$

Summing up the above inequalities we obtain

$$\sum_{i=1}^{k} \|x_i - x_{i-1}\|^2 \le \|x_k - x_0\|^2 + \sum_{i=1}^{k-1} \eta_i \tag{3.11}$$

Next, denoting $\tilde{x}_k^u = \gamma_k x_k + (1-\gamma_k)x_{k-1}^u$, then by Lemma 6 for all $k \ge 1$ and definition of x_k^u and $\tilde{x}_k^u$ we have

$$f(x_k^u) \le f(\tilde{x}_k^u) \le (1-\gamma_k)f(x_{k-1}^u) + \gamma_k \ell + \frac{L}{2}\|\gamma_k(x_k - x_{k-1})\|^2.$$

Subtracting both sides of above inequality we obtain

$$f(x_k^u) - \ell \le (1-\gamma_k)[f(x_{k-1}^u) - \ell] + \frac{L}{2}\|\gamma_k(x_k - x_{k-1})\|^2 \qquad \forall k \ge 1.$$

Using Lemma 2 for above inequality we obtain

$$\begin{aligned}f(x_k^u) - \ell &\le (1-\gamma_1)f(x_0^u - \ell) + \frac{L}{2}\sum_{i=1}^{k}\frac{\gamma_k^2}{\Gamma_k}\|x_k - x_{k-1}\|^2 \\ &\le (1-\gamma_1)f(x_0^u - \ell) + \frac{L}{2}\max_{1\le i\le k}\{\gamma_i^2/\Gamma_i\}\sum_{i=1}^{k}\|x_k - x_{k-1}\|^2 \\ &\le (1-\gamma_1)f(x_0^u - \ell) + \frac{L}{2}\max_{1\le i\le k}\{\gamma_i^2/\Gamma_i\}\left(D_X^2 + \sum_{i=1}^{k-1}\eta_i\right) \qquad \forall k \ge 1\end{aligned}$$

where the last inequality holds by (3.11).

To show part b, let $K \equiv K_{APL}(\epsilon)$ be the total number of iterations performed by $\mathcal{G}_{APL}$ and suppose that conditions (3.8) hold. Then by (3.7) and (3.8) we have

$$f(x_K^u) - \ell \leq \frac{cLD_X^2}{K^2} \leq \theta\beta\Delta_0 = \theta(\bar{f}_0 - \ell),$$

where the last equality follows the fact that $\ell = \beta \underline{f}_0 + (1-\beta)\bar{f}_0 = \bar{f}_0 - \beta\Delta_0$. Therefore, procedure $\mathcal{G}_{APL}$ must terminate in step 3 of the K-th iteration.

□

Note that the bound $K_{\mathcal{FW}}(k)$ is conceptual as parameter $\{\eta_k\}$ is still needed to be specified. In Theorem 3 and in part a) we inf a bound on total number of phases performed by the SAPL method, in part b) we find a bound on the total number of iterations performed by the SAPL method, and finally in part c) after selecting the parameter η we fins a bound on the total number of calls to the first order linear optimization oracle.

Theorem 3. *Let $\gamma_k \in (0,1]$, $k = 1, 2, \cdots$ in $\mathcal{G}_{SAPL}$ are chosen so that conditions* (3.8) *hold for some $c > 0$. Then*

a) The number of phases performed by SAPL method is bounded by

$$\bar{S}(\epsilon) = \left\lceil \max\left\{0, \log_{\frac{1}{q}} \frac{LD_X^2}{2\epsilon}\right\}\right\rceil$$

b) The total number of iterations performed by the SAPL method can be bounded by

$$\bar{S}(\epsilon) + \frac{1}{1-\sqrt{q}}\left(\frac{cLD_X^2}{\beta\theta\epsilon}\right)^{\frac{1}{2}}$$

c) Let

$$\eta \equiv \frac{D_X^2}{K_{SAPL}(\Delta_0)}$$

where $K_{SAPL}(\Delta_0)$ is defined in (3.9) *and $\Delta_0 = \bar{f}_0 - \underline{f}_0$. Then η satisfies conditions* (3.8) *and the number of iterations performed by $\mathcal{FW}$ procedure is bounded by*

$$\mathcal{O}\left(\frac{LD_X^2}{\epsilon}\right).$$

Proof. Let us denote $\delta_s \equiv ub_s - lb_s,\ s \geq 1$. Also, let us suppose that $\delta_1 > \epsilon$ because otherwise the statement is clearly true. First, note that if $\mathcal{G}_{APL}$ terminated in step 1 of its k-th iteration we must have $\underline{f}_k \geq \ell - \theta(\ell - \underline{f}_0)$. Since $f(p^+) \leq \bar{f}_0$ and by (1.41) we have

$$\begin{aligned} f(p^+) - lb^+ = f(p^+) - \underline{f}_k &\leq \bar{f}_0 - [\ell - \theta(\ell - \underline{f}_0)] \\ &= [1 - (1-\beta)(1-\theta)](\bar{f}_0 - \underline{f}_0). \end{aligned} \tag{3.12}$$

And if $\mathcal{G}_{APL}$ terminates in step 3 of its k-th iteration we must have $\bar{f}_k \leq \ell + \theta(\bar{f}_0 - \ell)$. Since $lb^+ \geq \underline{f}_0$ and also by definition of ℓ in (1.41) we have

$$f(p^+) - lb^+ = \bar{f}_k - lb^+ \leq \ell + \theta(\bar{f}_0 - \ell) - f_0 = [1 - (1-\theta)\beta](\bar{f}_0 - \underline{f}_0). \tag{3.13}$$

equations (3.12) and (3.13) together imply that at termination of $\mathcal{G}_{APL}$ we have

$$f(p^+) - lb^+ \leq q[f(p) - lb], \tag{3.14}$$

where

$$q \equiv q(\beta, \theta) := 1 - (1-\theta)\min\{\beta, 1-\beta\}. \tag{3.15}$$

Therefore, from (3.14) and (3.15) and origin of ub_s and lb_s we have

$$\delta_{s+1} \leq q\delta_s, \quad s \geq 1. \tag{3.16}$$

Also note that, from initial phase of SAPL and Lipschitz continuity of of f' we have

$$\delta_1 = f(p_1) - h(p_0, p_1) = f(p_1) - (f(p_0) + \langle f'(p_0), p_1 - p_0\rangle) \leq \frac{L}{2}\|p_1 - p_0\|^2 \leq \frac{LD_X^2}{2}. \tag{3.17}$$

From (3.16) and (3.17) we obtain

$$\delta_{s+1} \leq q^{s-1}\delta_1 \leq q^s \frac{LD_X^2}{2}.$$

To obtain an ϵ-gap we must have $q^s LD_X^2 < 2\epsilon$ which implies the desired result in part a.

Next, we prove part b. Let $\bar{s}$ be the total number of calls for $\mathcal{G}_{APL}$ for some $1 \le \bar{s} \le \bar{S}(\epsilon)$. Therefore, from (3.16) and the fact that $\delta_{\bar{s}} > \epsilon$ we have $\delta_s > \epsilon q^{s-\bar{s}},\ s = 1, \cdots, \bar{s}$. Using this observation we obtain

$$\sum_{s=1}^{\bar{s}} \delta_s^{-\frac{1}{2}} < \sum_{s=1}^{\bar{s}} \frac{q^{\frac{1}{2}(\bar{s}-s)}}{\sqrt{\epsilon}} = \sum_{t=0}^{\bar{s}-1} \frac{q^{\frac{t}{2}}}{\sqrt{\epsilon}} \le \frac{1}{(1-q^{\frac{1}{2}})\sqrt{\epsilon}}. \tag{3.18}$$

Using this observation and by Theorem 2, the number of iterations performed by SAPL method is bounded by

$$\begin{aligned} \sum_{s=1}^{\bar{s}} K_{SAPL}(\delta_s) &\le \bar{s} + \sum_{s=1}^{\bar{s}} \left(\frac{cLD_X^2}{\beta\theta\delta_s}\right)^{\frac{1}{2}} \\ &\le \bar{s} + \frac{1}{(1-q^{\frac{1}{2}})} \cdot \left(\frac{cLD_X^2}{\beta\theta\epsilon}\right)^{\frac{1}{2}}. \end{aligned}$$

To show part c, note that η satisfies the condition (3.8) because if $K \equiv K_{SAPL}(\epsilon)$ then

$$\sum_{i=1}^{K-1} \eta_i \le \sum_{i=1}^{K} \frac{D_X^2}{K} \le D_X^2.$$

Next, from (3.10) we have

$$K_{\mathcal{FW}}(k) = \left\lceil \frac{6D_X^2}{\eta_k} \right\rceil = \lceil 6K_{SALP}(\Delta_0) \rceil \le 7K_{SAPL}(\Delta_0).$$

This along with (3.18) imply that the total number of iterations performed by $\mathcal{FW}$ is bounded by

$$\sum_{s=1}^{\bar{s}} 7K_{SAPL}^2(\delta_s) \le 7\sum_{s=1}^{\bar{s}} \frac{cLD_X^2}{\theta\beta\delta_s} = \frac{cLD_X^2}{\theta\beta} \sum_{s=1}^{\bar{s}} \frac{1}{\delta_s} \le \frac{7cLD_X^2}{(1-q)\theta\beta\epsilon}.$$

To prove part (c) let us denote $\phi \equiv \phi_k := 1/2\,\|x - x_{k-1}\|^2$. Then the $\mathcal{FW}$ procedure can be view as specialized version of FW applied to $\min_{x\in X} \phi(x)$. We can see that $V_{g,u,\beta}(u_t)$ in (3.6) is equivalent to $\max_{x\in X} \langle \phi'(u_t), u_t - x\rangle$ which is the Wolfe-gap. Indeed, x_k obtained in step 2. of $\mathcal{G}_{SAPL}$ procedure is an approximate solution of for the projection subproblem $\min_{x\in X} \phi(x)$ such that

$$\langle \phi'(x_k), x_k - x\rangle = \langle x_k - x_{k-1}, x_k - x\rangle \le \eta_k, \quad \forall x \in X$$

for some $\eta_k \geq 0$. Now let $\phi^* \equiv \min_{x \in X} \phi(x)$. Also let us denote

$$\lambda_t := \frac{2}{t} \quad \text{and} \quad \Lambda_t = \frac{2}{t(t-1)}, \tag{3.19}$$

which implies that

$$\Lambda_{t+1} = \Lambda_t(1 - \lambda_{t+1}) \quad \forall t \geq 2.$$

Let us define $\bar{u}_{t+1} := (1 - \lambda_{t+1})u_t + \lambda_{t+1}v_t$. Clearly we have $\bar{u}_{t+1} - u_t + \lambda_{t+1}(v_t - u_t)$. Observe that $u_{t+1} = (1 - \alpha_t)u_t + \alpha_t v_t$ and α_t is an optimal solution to $\arg\min_{\alpha \in [0,1]} \phi((1-\alpha)u_t + \alpha v_t)$ and hence $\phi(u_{t+1}) \leq \phi(\bar{u}_{t+1})$. Using this observation, (A.5), and the fact that ϕ has Lipschitz continuous gradients, we have

$$\begin{aligned}
\phi(u_{t+1}) &\leq \phi(\bar{u}_{t+1}) \\
&\leq \phi(u_t) + \langle \phi'(u_t), \bar{u}_{t+1} - u_t \rangle + \frac{1}{2}\left\| \bar{u}_{t+1} - u_t \right\|^2 \\
&= \phi(u_t) + \lambda_{t+1} \langle \phi'(u_t), v_t - u_t \rangle + \frac{\lambda_{t+1}^2}{2} \left\| v_t - u_t \right\|^2 \\
&= \phi(u_t) - \lambda_{t+1}\phi(u_t) + \lambda_{t+1} \left(\phi(u_t) + \langle \phi'(u_t), v_t - u_t \rangle \right) + \frac{\lambda_{t+1}^2}{2} \left\| v_t - u_t \right\|^2 \\
&\leq (1 - \lambda_{t+1})\phi(u_t) + \lambda_{t+1} \left(\phi(u_t) + \langle \phi'(u_t), x - u_t \rangle \right) + \frac{\lambda_{t+1}^2}{2} \left\| v_t - u_t \right\|^2 \\
&\leq (1 - \lambda_{t+1})\phi(u_t) + \lambda_{t+1}\phi(x) + \frac{\lambda_{t+1}^2}{2} \left\| v_t - u_t \right\|^2 .
\end{aligned} \tag{3.20}$$

Subtracting $\phi(x)$ from both sides implies that

$$\phi(u_{t+1}) - \phi(x) \leq (1 - \lambda_{t+1})(\phi(u_t) - \phi(x)) + \frac{\lambda_{t+1}^2}{2} \left\| v_t - u_t \right\|^2 \quad \forall x \in \mathcal{X}.$$

By Lemma 3, for any $x \in \mathcal{X}$ and $t \geq 1$

$$\begin{aligned}
\phi(u_{t+1}) - \phi(x) &\leq \Lambda_{t+1} \left(\frac{1 - \lambda_2}{\Lambda_1} (\phi(1) - \phi(x)) \right) + \sum_{i=2}^{t+1} \frac{\lambda_i^2}{2\Lambda_i} \left\| v_{i-1} - u_{i-1} \right\|^2 \\
&= \Lambda_{t+1} \sum_{i=1}^{t} \frac{i}{i+1} \left\| v_i - u_i \right\|^2 \\
&\leq \frac{2D_{\mathcal{X}}^2}{t+1}
\end{aligned} \tag{3.21}$$

Now, let the gap function $V_{g,u}$ be defined in (3.6). Also let us denote $\Delta_j = \phi(u_j) - \phi^*$. It then follow from (3.6), and (3.20) that for any $j = 1, \cdots, t$,

$$\phi(u_{j+1}) \leq \phi(u_j) + \lambda_{j+1} \langle \phi'(u_j), v_j - u_j \rangle + \frac{\lambda_{j+1}^2}{2} \|v_j - u_j\|^2 .$$

Hence,

$$\lambda_{j+1} \langle \phi'(u_j), u_j - v_j \rangle \leq \phi(u_j) - \phi(u_{j+1}) + \frac{\lambda_{j+1}^2}{2} \|v_j - u_j\|^2 ,$$

which implies that

$$\begin{aligned} \lambda_{j+1} V_{g,u}(u_j) &\leq \phi(u_j) - \phi(u_{j+1}) + \frac{\lambda_{j+1}^2}{2} \|v_j - u_j\|^2 \\ &= \Delta_j - \Delta_{j+1} + \frac{\lambda_{j+1}^2}{2} \|v_j - u_j\|^2 . \end{aligned}$$

Dividing both sides of above inequality by Λ_{j+1} and summing up the resulting inequalities, we obtain

$$\begin{aligned} \sum_{j=1}^t \frac{\lambda_{j+1}}{\Lambda_{j+1}} V_{g,u}(u_j) &\leq \sum_{j=1}^t \frac{\Delta_j - \Delta_{j+1}}{\Lambda_{j+1}} + \sum_{j=1}^t \frac{\lambda_{j+1}^2}{2\Lambda_{j+1}} \|v_j - u_j\|^2 \\ &= -\frac{1}{\Lambda_{t+1}} \Delta_{t+1} + \sum_{j=2}^t \left(\frac{1}{\Lambda_{j+1}} - \frac{1}{\Lambda_j} \right) \Delta_j + \Delta_1 + \sum_{j=1}^t \frac{\lambda_{j+1}^2}{2\Lambda_{j+1}} \|v_j - u_j\|^2 \\ &\leq \sum_{j=2}^t \left(\frac{1}{\Lambda_{j+1}} - \frac{1}{\Lambda_j} \right) \Delta_j + \Delta_1 + \sum_{j=1}^t \frac{\lambda_{j+1}^2}{2\Lambda_{j+1}} \|v_j - u_j\|^2 \\ &\leq \sum_{j=1}^t j \Delta_j + \sum_{j=1}^t \frac{j}{j+1} D_{\mathcal{X}}^2 \\ &\leq \sum_{j=1}^t j \Delta_j + t D_{\mathcal{X}}^2, \end{aligned}$$

where the last inequality follow from the definition of λ_t and Λ_t in (3.19). Using the above inequality and the bound on Δ_j given in (3.21), we conclude that

$$\min_{j=1,\ldots,t} V_{g,u}(u_j) \sum_{j=1}^t \frac{\lambda_{j+1}}{\Lambda_{j+1}} \leq \sum_{j=1}^t \frac{\lambda_{j+1}}{\Lambda_{j+1}} V_{g,u}(u_j) \leq \sum_{j=1}^t j \frac{2D_{\mathcal{X}}^2}{j} + t D_{\mathcal{X}}^2 = 3t D_{\mathcal{X}}^2.$$

Since $\sum_{j=1}^{t} \lambda_{j+1}/\Lambda_{j+1} = t(t+1)/2$, then

$$\min_{j=1,\ldots,t} V_{g,u}(u_j) \left(\frac{t(t+1)}{2} \right) \leq 3tD_{\mathcal{X}}^2,$$

Therefore,

$$\min_{j=1,\ldots,t} V_{g,u}(u_j) \leq \frac{6D_{\mathcal{X}}^2}{t+1},$$

which implies part (c). □

As we can observe from Theorem 3 the SAPL method achieves the optimal bound $\mathcal{O}(1/\sqrt{\epsilon})$ for smooth programming problem and the optimal bound $\mathcal{O}(1/\epsilon)$ for number of linear optimization calls. Note that despite that fact that APL method requires solving a projection subproblem in each phase of its iterations but it is a parameter free algorithm. On the other hand, while SAPL addresses the issue of the projection in APL, it requires the knowledge of diameter D_X of the feasible set X and also the Lipschitz constant L of the gradient ∇f.

Chapter 4

CGS Variants for Video Co-Localization Problem

In this chapter, we use the numerical methods introduced in the previous chapters to solve video co-localization problems. Problems on recognizing and localizing a particular objects in images and videos receive many attention in recent years, as the internet photo and video sharing becomes more and more popular. Co-localization is the problem of localizing with bounding boxes in a set of images, or videos as a sequence of images (frames). We will first have a quick review on the formulations proposed in [23] for image and video co-localization problems. We will then propose algorithms for solving such problems and demonstrate the efficiently of proposed algorithms through numerical experiments.

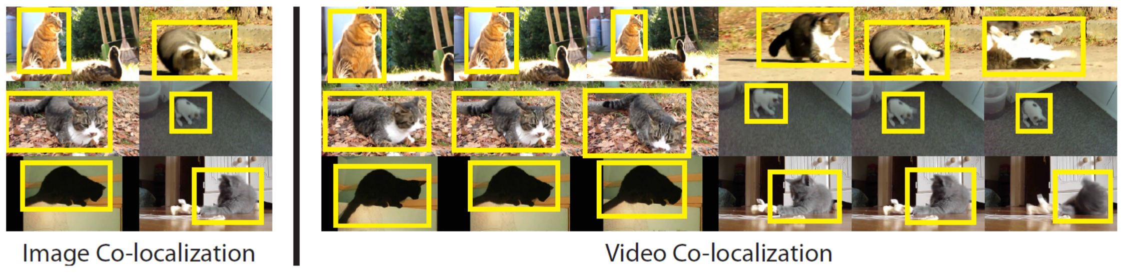

Figure 4.1: Localizing the common objects of the same class simultaneously in a set of images or videos using co-localization techniques (image from [23]).

4.1 Model Setup for Images

Our ultimate goal is to localize the common object in a set of images or in a series of frames of a video. Here we first have a brief review of image and video models based on formulation in [23]. To this end we review the required back grounds in each step as much as the features and variables in the mathematical programming model become understandable. Note that this formulation is based on formulation introduced in [45] for image co-localization. Quadratic formulation that we review in this section localizes any set of images and videos, simultaneously. In [7, 11, 12] also, we can find similar discrete optimization approaches in various computer vision applications.

4.1.1 Objectness for Images

Suppose that we have a set $\mathcal{I} = \{I_1, I_2, \dots, I_n\}$ of n given images, and our goal is to localize the common object in each image. One approach is to find candidate boxes in each image that potentially contain an object using *objectness* [1].

While object detectors for images are usually specialized for one object class such as cars, airplanes, cats, or dogs, objectness quantifies how likely it is for an image window to cover an object of any class. In an image, objects have a well-defined boundary and center, cats, dogs, and chairs, as opposed to indefinite background, such as walls, sky, grass, and road. Figure 4.2 illustrates the desired behavior of an objectness measure. Green windows must score highest windows fitting an object tight, blue windows should score lower windows covering partly an object and partly the background, and red windows should score lowest windows containing only partial background. This approach and the way we score the windows is designed in [1] and explicitly trained to distinguish windows containing an object from background windows.

Using objectness, we generate m candidate boxes (e.g. green boxes in Figure 4.2) for each image that could potentially contain an object. In other words, if $j \in \{1, 2, \dots, n\}$ we define $\mathcal{B}_j$ to be the set of all boxed in image $I_j \in \mathcal{I}$. Then the goal is to select the box that contains the object, from each image, jointly. Also. for simplicity let $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2 \cup \cdots \cup \mathcal{B}_n$ and $n_b = nm$ the total number of boxes in all images.

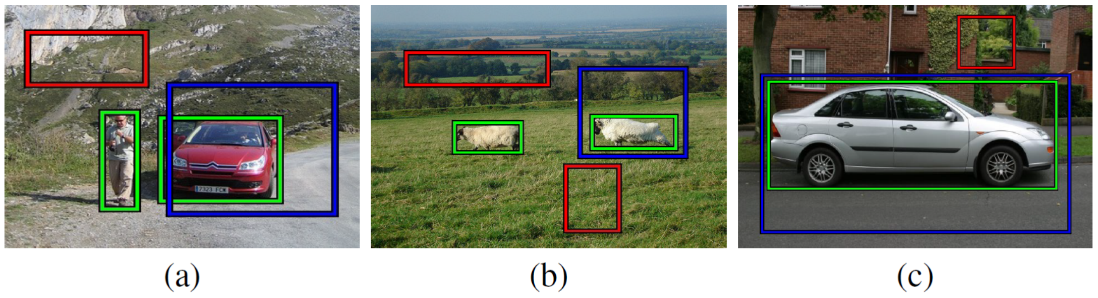

Figure 4.2: The objectness measure should score the blue windows, partially covering the objects, lower than the ground truth windows in green, and score even lower the red windows containing only the background. Image is from [1].

4.1.2 Feature representation

Assume that we have determined m candidate boxes in each of two the different images I_i and I_j for any $i, j \in \{1, 2, \ldots, m\}$. A common object in I_i and I_j might be in different shape, scale, color, brightness, angle and many other features. Therefore, it is critical to extract distinctive invariant features from images that can be used to perform reliable matching between different views of an object. David G. Lowe in [32] introduces a method that finds features that are invariant to image scaling and rotation, and partially invariant to change in illumination and 3D camera view point. Using his method, large number of features can be extracted from typical images with efficient algorithms, as well as the cost of extracting these features is minimized. The major stages of computation used to generate the set of image features are as follows.

1. **Scale-space extrema detection:** The first stage of computation searches over all scales and image locations. It is implemented efficiently by using a difference-of-Gaussian function to identify potential interest points that are invariant to scale and orientation.

2. **Keypoint localization:** At each candidate location, a detailed model is fit to determine location and scale. Keypoints are selected based on measures of their stability.

3. **Orientation assignment:** One or more orientations are assigned to each keypoint location based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing invariance to these transformations.

4. **Keypoint descriptor:** The local image gradients are measured at the selected scale in the region

around each keypoint. These are transformed into a representation that allows for significant levels of local shape distortion and change in illumination.

This process is called Scale Invariant Feature Transform (SIFT). SIFT transforms image data into scale-invariant coordinates relative to local features. Using SIFT we can generate large numbers of features that densely cover the image over full range of scales and locations.

Let b_k be a box in $\mathcal{B}$. Then we denote the SIFT feature representation of b_k as $x_k \in \mathbb{R}^d$ where $d = 10,000$ is the dimensional feature descriptor for each box in $\mathcal{B}$. Finally, we stack the feature vectors to form a feature matrix $X \in \mathbb{R}^{n_b \times d}$.

4.1.3 Prior, Similarity, and Discriminability of boxes

Let us denote the boxes that contain an instance of the common object as *positive* boxes, and the ones that don't as *negative* boxes. Then a prior is introduced for each box that represents a score that the box is positive. This happens using a saliency map [41] for each box and the prior is in fact the average saliency within the box, weighted by the size of the box. Finally we stack these values into the n_b dimensional vector $\mathbf{m}$ as the prior vector.

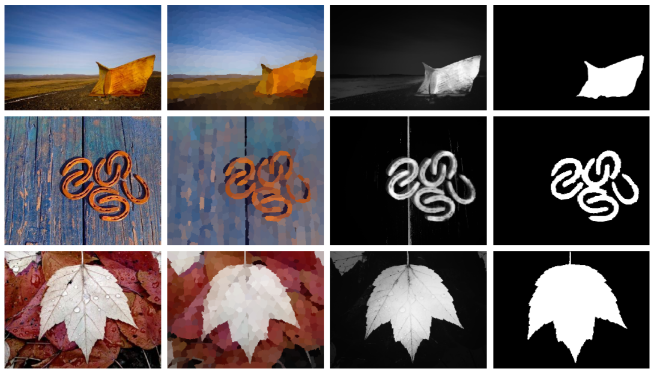

Figure 4.3: An example of saliency mappings for images from left to right. Image is from [41].

In addition, boxes that have the similar appearance should be labeled the same. This happens through a matrix called similarity matrix denoted by S. Similarity matrix of boxes in $\mathcal{B}$ is based on the box feature matrix X described above. Let b_i and b_j be any two boxes in $\mathcal{B}$ where $i, j \in \{1, 2, \dots, n_b\}$. Then similarity matrix $S \in \mathbb{R}^{n_b \times n_b}$ is computed based on the χ^2-distance as

$$S_{ij} = \exp\left\{-\gamma \sum_{k=1}^{d} \frac{(x_{ik} - x_{jk})^2}{x_{ik} + x_{jk}}\right\},$$

where $\gamma = (10d)^{-1/2}$. For i and j where boxes b_i and b_j belong to the same image we set $S_{ij} = 0$. Then the normalized Laplacian matrix [44] is computed as

$$\mathcal{L} = I_{n_b} - D^{-1/2} S D^{-1/2}, \tag{4.1}$$

where D is the diagonal matrix composed of row sums of S.

4.1.4 Model Formulation

Associated with each box $b_{j,k} \in \mathcal{B}_j$ we define a binary variable $z_{j,k}$ where $z_{j,k} = 1$ when $b_{j,k}$ is a positive box (contains an instance of the common object) and 0 otherwise. Then we define the integer vector variable

$$\mathbf{z} = (z_{1,1}, \ldots, z_{1,m}, \ldots, z_{n,1}, \ldots, z_{n,m})^T \in \{0,1\}^{n_b}.$$

Making the assumption that in each image there exist at most 1 positive box, our set of constraints are define by

$$\sum_{k=1}^{m} z_{j,k} = 1, \qquad \forall j \in \{1, \ldots, n\}. \tag{4.2}$$

As we introduced a prior for each box and defined the n_b dimensional vector of average saliency within the boxes, we obtain a linear term that penalizes less salient boxes as part of the objective function:

$$f_p(\mathbf{z}) := -\mathbf{z}^T \log(\mathbf{m}). \tag{4.3}$$

Similarly, our choice of normalized Laplacian matrix $\mathcal{L}$ defined in (4.1) results in a quadratic term that handles the selection of similar boxes:

$$f_L(\mathbf{z}) := \mathbf{z}^T \mathcal{L} \mathbf{z}. \tag{4.4}$$

This is motivated by the work of Shi and Malik [44] in which they have taken advantage of eigenvalues of the Laplacian for clustering $\mathbf{z}$ by the similarity matrix. In fact, they have shown that with the eigenvector corresponding to the second smallest eigenvalue of a normalized Laplacian matrix we can cluster $\mathbf{z}$ along the graph defined by the similarity matrix, leading to normalized cuts when used for image segmentation. Also, Belkin and Niyogi [4] showed that this problem is equivalent to minimizing (4.4) under linear constraints. In fact, the similarity term works as a generative term which selects boxes that cluster well together [45].

Although discriminative learning techniques such as support vector machines and ridge regression has been widely used on many supervised problems in which there are know labels, they can be used in this unsupervised case where the labels of boxes are unknown [3, 49]. Motivated by [22], we consider the ridge regression objective function for boxes:

$$\min_{w\in\mathbb{R}^d,\ c\in\mathbb{R}} \quad \frac{1}{n_b}\sum_{j=1}^{n}\sum_{k=1}^{m}\|z_{j,k}-wx_{j,k}-c\|_2^2-\frac{\kappa}{d}\|w\|_2^2\,,$$

where w is the d dimensional weight vector of the classifier, and c is the bias. This cost function is being used among discriminative cost functions because the ridge regression problem has a explicit (closed form) solution for weights w and bias c which implies the quadratic function in the box labels [3]:

$$f_D(\mathbf{z}) := \mathbf{z}^T\mathcal{A}\mathbf{z}, \tag{4.5}$$

where

$$\mathcal{A} = \frac{1}{n_b}\Pi_{n_b}\left(I_{n_b}-X(X^T\Pi_{n_b}X+n_b\kappa I_{n_b})^{-1}X^T\right)\Pi_{n_b}, \tag{4.6}$$

is the discriminative clustering term and $\Pi_{n_b}=I_{nb}-\frac{1}{n_b}\mathbf{1}_{n_b}\mathbf{1}_{n_b}^T$ in (4.6) is the centering projection matrix. Note that this quadratic term allows us to utilize a discriminative objective function to penalize the selection of boxes whose features are not easily linearly separable from other boxes.

Summing up our results in (4.2), (4.3), (4.4), and (4.5), the optimization problem to select the best box in each image is given by

$$\begin{aligned}
\min_{\mathbf{z}} \quad & \mathbf{z}^T(\mathcal{L}+\mu\mathcal{A})\mathbf{z} - \lambda\ \mathbf{z}^T\log(\mathbf{m}) \\
\text{s.t} \quad & \sum_{k=1}^{m} z_{j,k} = 1, \qquad j = 1,\dots,n \\
& \mathbf{z} = (z_{1,1},\dots,z_{1,m},\dots,z_{n,1},\dots,z_{n,m})^T \in \{0,1\}^{n_b},
\end{aligned} \tag{4.7}$$

where parameter μ regularizes the trade-off between the quadratic terms (4.4) and (4.5), and parameter λ handles the trade-off between the linear term (4.3) and the quadratic terms (4.4) and (4.5). Recall that the linear constraints ensures that one box from each image is selected in the optimal solution. Note that Hastie, Tibshirani, and Friedman in [20] showed that $\mathcal{A}$ is a positive semi-definite matrix. Also, since matrix $\mathcal{L}$ is positive semi-definite as well, the objective function of (4.7) is convex.

4.2 Model Setup for Videos

Co-localization in a video is very similar to the image case, as a video is a sequence of images that are called frames. While an object might not have an extreme change in size, shape, color, etc in two frames in row, co-localization in a video could be a simpler task at some point. In this section we describe the localization of a common object in a set of videos. In fact, if $\mathcal{V} = \{V_1, V_2, \dots, V_n\}$ is a set of n given videos, we explore an approach to localize a common object in each frame of each video. More precisely, we consider $\mathcal{I}_i = \{I_{i1}, I_{i2}, \dots, I_{il_i}\}$ to be the temporally ordered set of frames of video V_i. Here I_{ij} is the i-th frame of the j-th video and l_i is the total number of frames, or the length of V_i for $i = 1,\dots,n$ and $j = 1,\dots,l_i$. Similar to what we did in image case, we set $\mathcal{B}_{i,j}$ to be the set of m generated candidate boxes, using objectness [1], for j-th of i-th video. Then, considering l_i frames in video i and m boxes in each frame, we set $n_b^v = \sum_{i=1}^{n} l_i m$ to be the total number of boxes in $\mathcal{V}$, the set of all videos.

Note that, if we set $\mathcal{I} = \{\mathcal{I}_1, \mathcal{I}_2, \dots, \mathcal{I}_n\}$ to be the ordered set of all frames in $\mathcal{V}$, model (4.7) returns a single box in each frame (image) as an optimal solution. Although the objective function of this model capture the box prior, similarity, and discriminability within different videos, as we can define a more efficient similarity mapping withing boxes in the sequence of frames in a video.

4.2.1 Temporal Consistency In Frames of a Video

As discussed earlier in this section, objects in consecutive frames in video data are less likely to change drastically in appearance, position, and size. This is a motivation to use a separate prior for frames or images in video case. Temporal consistency [2, 5, 19, 40, 42, 46, 50] is a powerful prior that is often leveraged in video tasks such as tracking [23]. In this approach, in consecutive frames, boxes with great difference in size and position should be unlikely to be selected together. To this end, a simple temporal similarity measure is defined between two boxes b_i and b_j from consecutive frames with:

$$s_{\text{temporal}}(b_i, b_j) := \exp\left\{ -\left\| b_i^{\text{center}} - b_j^{\text{center}} \right\|_2 - \left\| \frac{\left| b_i^{\text{area}} - b_j^{\text{area}} \right|}{\max(b_i^{\text{area}}, b_j^{\text{area}})} \right\|_2 \right\}. \tag{4.8}$$

A few comments comes in place about the prior defines in (4.8). First, b_i^{area} is the vector of the pixel area of box b_i and b_i^{center} are the vectors of the center coordinates of box b_i, normalized by the width and height of the frame. Second, the metric defined in (4.8) is a similarity metric that is defined between all pairs of boxes in adjacent frames. From this metric we can define a weighted graph $\mathcal{G}_i$ for video $\mathcal{V}_i$ for $i = 1, 2, \ldots, n$ with nodes being the boxes in each frame and edges connecting boxes in consecutive frames and weights of edges defined as temporal similarity in (4.8). Figure 4.4 is a graphical representation of graph $\mathcal{G}_i$. For small values of similarity measure with some threshold we disconnect the nodes and remove the edge. Finally, as long as we can create a weighted graph with boxes, any similarity measure other than the temporal consistency in (4.8) can be used to weight the edges between two boxes, which makes the temporal framework pretty flexible.

Let us define

$$S_t(i, j) = \begin{cases} s_{\text{temporal}}(b_i, b_j) & \text{if frames } i \text{ and } j \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases}$$

to be the similarity matrix define by the temporal similarity measure, where b_i and b_j are any two boxes in the set of all boxes in $\mathcal{V}$. Similar to our approach to obtain (4.1), with S_t we can compute the normalized Laplacian

$$U = I_{n_b^v} - D^{-1/2} S_t D^{-1/2}, \tag{4.9}$$

where D is the diagonal matrix composed of the row sums of S_t. This matrix encourages us to select boxes that are similar based on the temporal similarity metric (4.8).

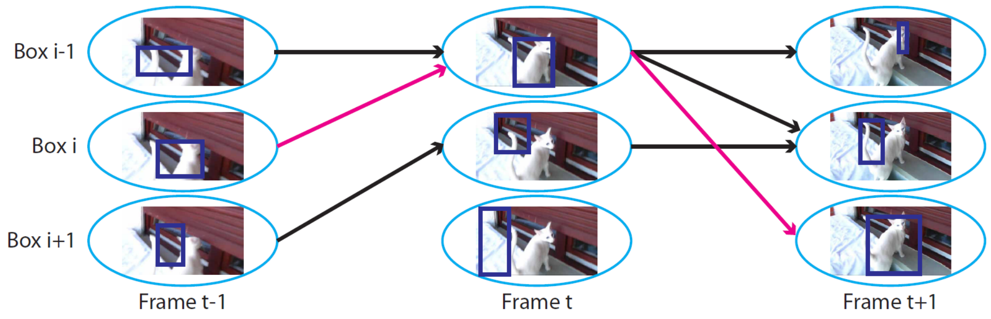

Figure 4.4: Nodes (blue circles) represent candidate boxes in each frame, and the directed edges between nodes are weighted by a temporal similarity metric (e.g. 4.8) that measures the similarity in size and position of boxes. To reduce the dimension of the graph, edges with low similarity are removed to limit the number of possible paths through the graph from the first to the last frame. The magneta edges represent the optimal path in this example. Image is from [23].

4.2.2 Video Model Formulation

As we discussed above, temporal similarity suggests a weighted graph $\mathcal{G}_i$ for video $\mathcal{V}_i$ for $i = 1, 2, \ldots, n$. In fact, a valid path in $\mathcal{G}_i$ from the the first to the last frame in $\mathcal{V}_i$ corresponds to feasible boxes chosen in each frame of $\mathcal{V}_i$. This motivates us to define a binary variable to be on when there is an edge between any two nodes in $\mathcal{G}_i$ and off otherwise. In better words, we define the binary variable $y_{i,j,k}$ for video i and boxes b_j and b_k in $\mathcal{V}_i$ as

$$y_{i,j,k} = \begin{cases} 1 & \text{if boxes } b_j \text{ and } b_k \text{ contain the common object} \\ 0 & \text{otherwise.} \end{cases}$$

In fact, variable $y_{i,j,k}$ corresponds to the existence of edge between boxes b_j and b_k in $\mathcal{V}_i$. Also, we define the binary variable $z_{i,j,k}$ to be 1 if the box b_k in frame j of video i contains the common object, and 0 otherwise. A type of constraint that we need to consider here is the fact that there might exist an edge between boxes b_j and b_k only if they are boxes in two consecutive frames. Then, for a typical box b_k in frame j of video $\mathcal{V}_i$, we define index sets $p(k_j)$ and $c(k_j)$ to be the set of indices of parents and children boxes in frames $j+1$ and $j-1$, respectively, that are connected to b_k in frame j in the graph $\mathcal{G}_i$. Therefore, a required set of constraints for localization in video case are defines by:

$$z_{i,j,k} = \sum_{l \in p(k_j)} y_{i,l,k_j} = \sum_{l \in c(k_j)} y_{i,k_j,l}, \qquad i = 1, \ldots, n,\ j = 1, \ldots, l_i,\ k = 1, \ldots, m. \tag{4.10}$$

The other set of constraints, which are quite similar to the image co-localization case, are the set of constraints restricting each frame of each video to has only one box that contains the common object. These constraints are defined by:

$$\sum_{k=1}^{m} z_{i,j,k} = 1, \qquad i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, l_i. \tag{4.11}$$

Finally, we define the vectors of variables $\mathbf{z} = (z_{1,1,1}, z_{1,1,2}, \ldots, z_{i,j,k}, \ldots, z_{n,l_n,m})^T \in \{0,1\}^{n_b^v}$ where $n_b^v = m \sum_{i=1}^{n} l_i$. Then if we combine the temporal terms defined by (4.9) with the terms in the objective function of the original image model (4.7), then with constraint defines in (4.10) and (4.11), we obtain the following optimization formulation to select the box containing the common object in each frame of video:

$$\begin{aligned} \min_{\mathbf{z},\ y} \quad & \mathbf{z}^T (L + \mu A + \mu_t U)\mathbf{z} - \lambda\, \mathbf{z}^T \log(\mathbf{m}) \\ \text{s.t.} \quad & \sum_{k=1}^{m} z_{i,j,k} = 1, \qquad i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, l_i, \\ & z_{i,j,k} = \sum_{l \in p(k_j)} y_{i,l,k_j} = \sum_{l \in c(k_j)} y_{i,k_j,l} \\ & \qquad\qquad i = 1, \ldots, n,\ j = 1, \ldots, l_i,\ k_j = 1, \ldots, m, \\ & y_{i,s,t} \in \{0,1\}, \quad i = 1, \ldots, n,\ s,t = 1, \ldots, m \\ & \mathbf{z} = (z_{1,1,1}, z_{1,1,2}, \ldots, z_{i,j,k}, \ldots, z_{n,l_n,m})^T \in \{0,1\}^{n_b^v}, \end{aligned} \tag{4.12}$$

where μ_t is the trade-off weight for the temporal Laplacian matrix. Note that with the new objective function in problem (4.12) the extra constraint (4.10) in video case is necessary and without that the temporal Laplacian matrix would lead the solution to an invalid path. This formulation allows us to incorporate temporal consistency into the image model.

4.3 Optimization

The formulation (4.7) obtained to find the best box in each image of the set of the given images is a standard binary constrained quadratic problem. The only issue that makes this problem a non-convex problem are the binary constraints. Relaxing these constraints to the continuous linear constraints lead the problem to the convex optimization problem and can be solved efficiently using standard methods. In fact, first order methods such as like Frank-Wolfe method that we discussed in previous chapters can handle the relaxed problem efficiently as they linearize the quadratic objective function and use a linear optimization oracle in each iteration.

Denoting the feasible region of the problem (4.12) by $\mathcal{P}$, we can follow a similar approach for this problem as we did for (4.7). We can relax the discrete non-convex set $\mathcal{P}$ into the convex hull, or the integer hull for this specific case, $\text{conv}(\mathcal{P})$. Although standard algorithms such as interior point methods can be applied to solve this problem, but as the number of videos increases to hundreds and the dimension of the problem increases exponentially, such problems with complexity of $\mathcal{O}(N^3)$ with number of boxes, would perform very weakly. Similarly, for the relaxation of the video problem we will show in our implementations section that suggested first order methods perform efficiently. We will also propose a first order method later in this chapter and will show that it performs better than other first order methods that have been applied to this problem.

Note that, the constraints defining the set $\mathcal{P}$ are separable in each video. In fact, for each video, these constraints are equivalent to the constraints of the shortest-path problem. This implies that the linear optimization step appears in each iteration of the first order methods are actually shortest-path problems that can be solved efficiently using dynamic programming.

Recall that Frank-Wolfe algorithm is a first order method that in each of its iteration updates the new point toward a direction by calling a linear optimization oracle. This objective function of this linear optimization is in fact a linear approximation of the objective function of (4.7), and (4.12). Frank-Wolfe algorithm specifically results in a simple linearizations with integer solution for the image and video co-localization optimization problems. For the image model, the linearlized cost function is separable for each image, and we can efficiently find the best integer solution with some threshold for this problem. For the video model also, the cost function and the constraints are separable for each video and optimizing the linearized function over the feasible region results in the shortest-path problem for each video.

In the following section we will propose an algorithm that can be applied on image and video co-localization optimization problems efficiently and we finally compare the performance of the proposed algorithm to the algorithms that are applied to these problems.

4.4 Proposed Algorithms

Conditional Gradient Sliding (CGS) algorithm [30], as we discussed in chapter 1, is a first order projection free method for solving convex optimization problems in which the feasible region is a convex and compact set. The major advantage of the CGS algorithm is that it skips gradient evaluation from time to time and uses the same information within some inner iterations. This property of the CGS algorithm becomes helpful when the dimension of the problem as size of the variable is relatively large and computations become more and more expensive.

As showed in previous chapters, CGS algorithm and its proposed variant, Conditional Gradient Sliding with Linesearch (CGS-ls) perform very well in many practical instances. Although the CGS and CGS-ls algorithms out-perform the Frank-Wolfe (FW) algorithm many cases, the variants of FW, such as Away-steps FW 1 or Pairwise FW 1 converge faster to the optimal value than CGS for the image and video co-localization problem as we will show this in numerical experiments later in this chapter.

Motivated from the CGS algorithm and also Away-steps and pairwise FW methods, we propose an algorithms called Away-Steps Conditional Gradient Sliding (ACGS) and Pairwise Conditional Gradient Sliding (PCGS) that perform very well for image and video co-localization problems. ACGS and PCGS methods have iterations of the CGS method but the direction to update the new point in each iteration is motivated from the away steps and pairwise steps in the Away-steps and Pairwise FW. We will also show that the ACGS and PCGS out-perform all of the variants of the FW applied to the image and Video co-localization problem.

4.4.1 Away-Steps and Pairwise Conditional Gradient Sliding

The basic scheme of the ACGS and PCGS methods is obtained by performing a new search direction in CGS method, if the new direction leads the algorithm to smaller Wolfe gap. Also, similar to the CGS algorithm, the classical FW method (as $\mathcal{FW}$ procedure) is incorporated in this algorithm to solve the projection subproblems in the accelerated gradient (AG) with some approximations. The

ACGS and PCGS algorithms are described as in 12 and 13.

Algorithm 12 The Away-Steps Conditional Gradient Sliding Algorithm

```
Initial point $x_0 \in \mathcal{A}$ and iteration limit $N$.
Let $\beta_k \in \mathbb{R}^n_{++},\ \gamma_1 = 1, \mathcal{S}^{(0)} := \{x_0\}$, and $\eta_k \in \mathbb{R}_+,\ k = 1, 3\cdots$, be given and set $y_0 = x_0$.
for $k = 1, \ldots, N$ do
    $z_k = y_{k-1} + \gamma_k(x_{k-1} - y_{k-1})$
    $x_k = \mathcal{FW}(f'(z_k), x_{k-1}, \beta_k, \eta_k)$,
    $d_k^{\text{CGS}} = x_k - y_{k-1}$,
    $v_k = \arg\max_{v \in \mathcal{S}^{(t)}} \langle f'(y_{k-1}), v \rangle$,                    (4.13)
    $d_k^{\text{away}} = y_{k-1} - v_k$                                                           (4.14)
    if $\langle -f'(y_{k-1}), d_k^{\text{CGS}} \rangle \le \epsilon$ then return $y_{k-1}$      (4.15)
    if $\langle -f'(y_{k-1}), d_k^{\text{CGS}} \rangle \ge \langle -f'(y_{k-1}), d_k^{\text{away}} \rangle$, then   (4.16)
        $d_k := d_k^{\text{CGS}}$
        $\gamma_{\max} := 1$                                                                       (4.17)
    else
        $d_k := d_k^{\text{away}}$
        $\gamma_{\max} = \alpha_{v_k}/(1 - \alpha_{v_k})$                                          (4.18)
    end if                                                                                         (4.19)
    $\gamma_{k+1} \in \arg\min_{\gamma \in [0, \gamma_{\max}]} f(x_k + \gamma d_k)$                (4.20)
    $y_k = y_{k-1} + \gamma_{k+1} d_k$
    $\mathcal{S}^{(k)} := \{v \in \mathcal{A};\ \alpha_v^{(k)} > 0\}$
end for
procedure $u^+ = \mathcal{FW}(g, u, \beta, \eta)$
    1. Set $u_1 = u$ and $t = 1$.
    2. Let $w_t$ be the optimal solution for the subproblem of
           $V_{g,u,\beta}(u_t) := \max_{x \in \mathcal{X}} \langle g + \beta(u_t - u), u_t - x \rangle$
    3. If $V_{g,u,\beta}(u_t) \le \eta$, set $u^+ = u_t$ and terminate the procedure.
    4. Set $u_{t+1} = (1 - \tilde{\alpha}_t)u_t + \tilde{\alpha}_t w_t$, with
           $\tilde{\alpha}_t = \min\left\{1, \frac{\langle \beta(u - u_t) - g, w_t - u_t \rangle}{\beta \|w_t - u_t\|^2}\right\}$
    5. Set $t \leftarrow t + 1$ and go to step 2.
end procedure
```

Note that the purpose of the proposed algorithm is to be applied to the image and video co-localization problems (4.7) and (4.12). The objective function in both problems, as discussed before, are convex functions, and the feasible region is a set of finite binary vectors called *atoms* in $\mathbb{R}^d$ for some d. We denote this set by $\mathcal{A}$ and its convex hull conv$(\mathcal{A})$ by $\mathcal{M}$. As $\mathcal{A}$ is finite, $\mathcal{M}$ is a polytope.

The first difference between the AGCS(PCGS) and the CGS method is that we incorporate the set $\mathcal{S}^{(k)}$ of active atoms in the ACGS(PCGS) algorithm. This set keeps record of atoms (integer points) in $\mathcal{A}$ that are being used for the *away* direction d_K^{away} at each iteration such that the point y_k at current iteration is the sum of corners in $\mathcal{S}^{(k)}$ reweighted by $\alpha^{(k)}$. This direction that is given in (4.14), is defined by finding the atom v_k in $\mathcal{S}^{(k)}$ that maximized the potential of descent given by $\langle -f'(y_{k-1}), y_{k-1} - v_k\rangle$. Note that obtaining v_k in (4.13) is fundamentally easier as the linear optimization is over the $\mathcal{S}^{(k)}$, the active set of possibly small finite set of points.

The second difference is in the way we update the step-size to update the new iteration point. As we observe in (4.20) we incorporate a line-search method to obtain a step-size with maximum reduction in the objective toward a prespecified direction from the point at current iteration. With $\gamma_{\max}$ defined in (4.17) and (4.18) as the maximum step-size for the line-search step the algorithm guarantees that the new iterates $y_k = y_{k-1} + \gamma_{\max} d_k^{\text{away}}$ stays feasible in each iteration. Note that the parameter γ_k in CGS algorithm is required to be set up in appropriate way to maintain the feasibility in each iteration. Such set ups are represented in Theorem 8 and Corollary 5 as $\gamma_k = 3/(k+2)$ and $\gamma_k = 2/(k+1)$ and in fact, we can us these set ups for CGS steps in step (4.17) as the upper bound for γ_k instead of 1 in line-search step (4.20). Also, it is easy to check that for the special case of the image and video co-localization problem in which the objective is a convex quadratic function γ_k in step (4.20) has the closed form

$$\gamma_k = -\frac{d^T \nabla f(x)}{d^T Q d},$$

if $Q \succeq 0$ is the quadratic term in the objective. This value is projected to 0 or $\gamma_{\max}$ if is outside of the range $[0, \gamma_{\max}]$ for (4.20) case.

Finally, we incorporate the Wolfe gap as an stopping criterion in the ACGS and PCGS algorithms. In fact, at steps (4.15) and (4.21), the algorithms checks if they have reached the given threshold to stop before the preset max number of iterations N. As in classical FW, the Wolfe gap

Algorithm 13 The Pairwise Conditional Gradient Sliding Algorithm

```
Initial point $x_0 \in \mathcal{A}$ and iteration limit $N$.
Let $\beta_k \in \mathbb{R}^n_{++}$, $\gamma_1 = 1, \mathcal{S}^{(0)} := \{x_0\}$, and $\eta_k \in \mathbb{R}_+$, $k = 1, 3 \cdots$, be given and set $y_0 = x_0$.
for $k = 1, \ldots, N$ do
            $z_k = y_{k-1} + \gamma_k(x_{k-1} - y_{k-1})$
            $x_k = \mathcal{FW}(f'(z_k), x_{k-1}, \beta_k, \eta_k)$,
            $d_k^{\mathrm{CGS}} = x_k - y_{k-1}$,
            $v_k = \arg\max_{v \in \mathcal{S}^{(t)}} \langle f'(y_{k-1}), v \rangle$,

            if $\left\langle -f'(y_{k-1}), d_k^{\mathrm{CGS}} \right\rangle \le \epsilon$ then return $y_{k-1}$        (4.21)

            $d_k^{\mathrm{PCGS}} = x_k - v_k$,        (4.22)
            $\gamma_{\max} = \alpha_{v_k}$,        (4.23)
            $\gamma_{k+1} \in \arg\min_{\gamma \in [0, \gamma_{\max}]} f(x_k + \gamma d_k^{\mathrm{PCGS}})$
            $y_k = y_{k-1} + \gamma_{k+1} d_k$
            $\mathcal{S}^{(k)} := \{v \in \mathcal{A};\ \alpha_v^{(k)} > 0\}$
end for
procedure $u^+ = \mathcal{FW}(g, u, \beta, \eta)$
  1. Set $u_1 = u$ and $t = 1$.
  2. Let $w_t$ be the optimal solution for the subproblem of
         $V_{g,u,\beta}(u_t) := \max_{x \in \mathcal{X}} \langle g + \beta(u_t - u), u_t - x \rangle$
  3. If $V_{g,u,\beta}(u_t) \le \eta$, set $u^+ = u_t$ and terminate the procedure.
  4. Set $u_{t+1} = (1 - \tilde{\alpha}_t) u_t + \tilde{\alpha}_t w_t$, with
         $\tilde{\alpha}_t = \min\left\{1, \frac{\langle \beta(u - u_t) - g, w_t - u_t \rangle}{\beta \|w_t - u_t\|^2}\right\}$
  5. Set $t \leftarrow t + 1$ and go to step 2.
end procedure
```

is an upper bound on the unknown suboptimality and from the convexity of the objective f we have

$$f(x_k) - f(x^\star) \le \langle -f'(x_k), x^\star - y_{k-1} \rangle \le \langle -f'(x_k), x_k - y_{k-1} \rangle \le \epsilon.$$

Note that for the image and video co-localization problem with binary decision variables in a CGS step we have

$$\mathcal{S}^{(k+1)} = \begin{cases} \{x_k\} & \text{if } \gamma_k = 1 \\ \mathcal{S}^{(k)} \cup \{x_k\} & \text{otherwise.} \end{cases}$$

Also, for $v \in \mathcal{S}^{(k)} \setminus \{s_k\}$ we have

$$\alpha_{s_t}^{(k+1)} := (1-\gamma_k)\alpha_{s_t}^{(k)} + \gamma_k \quad \text{and} \quad \alpha_v^{(k+1)} := (1-\gamma_k)\alpha_v^{(k)}.$$

On the other hand, for an away step we have

$$\mathcal{S}^{(k+1)} = \begin{cases} \mathcal{S}^{(k)} \setminus \{v_k\} & \text{if } \gamma_k = \gamma_{\max} \\ \mathcal{S}^{(k)} & \text{otherwise.} \end{cases}$$

This step is called a *drop step.* Also, for $v \in \mathcal{S}^{(k)} \setminus \{v_k\}$ we have

$$\alpha_{v_t}^{(k+1)} := (1+\gamma_k)\alpha_{v_t}^{(k)} + \gamma_k \quad \text{and} \quad \alpha_v^{(k+1)} := (1+\gamma_k)\alpha_v^{(k)}.$$

ACGS and PCGS algorithms are slightly different in the direction that they use to update the new point at each iteration. More precisely, steps (4.16) to (4.19) in Algorithm 12 are replaced with steps (4.22) and (4.23) in Algorithm 13. Similar to the Paiwise FW, the idea here is to only move weight from the away atom v_k to the CGS atom x_k and keep all other α weight unchanged. In other words

$$\alpha_{v_t}^{(k+1)} := \alpha_{v_t}^{(k)} - \gamma \quad \text{and} \quad \alpha_{x_k}^{(k+1)} := \alpha_{s_k}^{(k)} + \gamma,$$

for some $\gamma \leq \gamma_{\max} := \alpha_{v_t}^{(k)}$.

An important property of the formulation (4.7) and (4.12) is that their constraints are separable for each image and video. This helps computation to be more efficient if we use parallel computing. This, however, is a property of any first-order method and practically it is very memory efficient. In addition, as a solution to the convex relaxation is not necessarily an integer solution optimal or feasible to the original problem, we need to come up with a solution as close as possible to the obtained relaxation optimum. In image and video co-localization case, the most natural way of finding such a solution is to solve

$$\min_{p \in \mathcal{P}} \quad \|p - y\|_2^2 \,, \tag{4.24}$$

where $\mathcal{P}$ is the feasible region of the original problem and y is the solution to the relaxed problem. It is easy to check that the projection problem (4.24) is equivalent to

$$\max_{p \in \mathcal{P}} \quad \langle p, y \rangle \,,$$

which for the video model is just a shortest path problem that can be solved efficiently using dynamic programming.

4.5 Experimental Results

In this section we experiment the proposed Algorithm 12 to the problems introduced in (4.7) and (4.12) for image and video co-localization task. Recall that these problems are quadratic problems over the convex hull of paths in a network, the linear minimization oracle in first order methods is equivalent to find a shortest path in the network. We compare the performance of the proposed algorithm with the works in [23] and [24] on FW algorithm and its variants for the similar problem. For this comparison we reuse the codes available and shared for [23,24,45] and the included dataset of airplanes consist of 660 variables.

We begin this section by reviewing the performance of Away steps Frank-Wolfe (AFW) and its comparison to the solvers such as Gurobi and Mosek. These results are derived and shown in [23] and the goal in this section is to show how AFW outperforms other methods for our problem of interest. In [24], however, Joulin A., Tang K., and Fei-Fei L. showed that their proposed Pairwise Frank-Wolfe (PairFW) algorithm outperforms any other variants of FW in solving this problem. We will end this section by showing that our proposed ACGS algorithm performs better any first order methods that have been utilized to solve the video co-localization problem.

4.5.1 FW v.s. Mosek and Gurobi

Algorithm 14 is a variant of FW algorithm proposed in [23] in which the authors examined it on two datasets, the PASCAL VOC 2007 dataset [14] and the Youtube-Objects dataset [43]. This algorithm is in fact the AWF Algorithm 4 introduced in Chapter 1 with some slight changes and some extra rounding steps. Also, the set $\mathcal{D}$ in this algorithm is conv($\mathcal{P}$) the convex hull of the feasible region of problems (4.7) or (4.12). Their implementation of Algorithm 14 was coded in MATLAB and they compare it to two standard Quadratic Programming (QP) solvers, Mosek and Gurobi on a single-core 2.66GHz Intel CPU with 6GB of RAM. In addition, they set $\mu = 0.4$ for the image model and $\mu = 0.6$ for the video model and $\mu_t = 1.8$ and $\lambda = 0.1$, for both image and video models. They extracted 20 objectness boxes from each image and sample each video every 10 frames as there is little change frames in short amount time.

Algorithm 14 Frank-Wolfe Algorithm with Away Steps and Rounding [23]

```
Initialization $y_0 \in \mathcal{D}, \epsilon > 0,\ k = 0,\ z = y_0, \mathcal{S}_0 = \{y_0\},\ \alpha_0 = \{1\}$.
while duality-gap($z$) $\geq \epsilon$ do
    $k \leftarrow k + 1$;
    $y_k \leftarrow \arg\min_{y \in \mathcal{D}} \langle y, \nabla f(z) \rangle$ (FW direction);
    $x_k \leftarrow \arg\max_{y \in \mathcal{S}_{k-1}} \langle y, \nabla f(z) \rangle$ (away direction);
    if $\langle y_k - z, \nabla f(z) \rangle \leq \langle z - x_k, \nabla f(z) \rangle$ then
        $d_k = y_k - z$;
        $\gamma_{\max} = 1$;
    else
        $d_k = z - x_k$;
        $\gamma_{\max} = \alpha_k(x_k)$;
    end
    $\gamma_k = \min_{\gamma \in [0, \gamma_{\max}]} f(z + \gamma d_k)$;
    $\mathcal{S}_k, \alpha_k \leftarrow \textit{update\_active\_set}(d_k, \gamma_k)$;
    Update $z \leftarrow z + \gamma_k d_k$;
    if $f(y_k) < f(y^\star)$ then
        $y^\star \leftarrow y_k$ (rounding 1);
    end
end while
    $y_r \leftarrow \arg\max_{y \in \mathcal{D}} \langle y, z \rangle$;
    if $f(y_r) < f(y^\star)$ then
        $y^\star \leftarrow y_r$ (combining rounding);
    end
```

The stopping criterion of Algorithm 14 is based on the relative duality gap. This criterion, that is given in function duality-gap(z) in the algorithm, is defined as $d = (f - g)/g$, where f is the objective function and g is its dual. In the implementation of this algorithm, authors consider two values $1e$ - 2 and $1e$ - 3 for the stopping threshold ϵ.

Figures 4.5 presents some comparisons of the Algorithm 14 as a variant of FW algorithm with QP solvers Mosek and Gurobi in logarithmic scale. Indeed, this comparison is based on the CPU time performance of the algorithms depending on the number of images and videos, or in better words, the dimension of the decision variables. This time is the time that takes that algorithms reach a duality gap less than the threshold ϵ. As we can observe from these plots, the variant of FW algorithm with away steps outperforms the standard QP solvers Mosek and Gurobi.

The reason that we review and represent these comparisons directly from [23]local is that in our implementations in next section we will only compare our proposed algorithms to some other first order methods. These first order methods include the AWF algorithm that we already know from this section that it outperforms standard QP solvers.

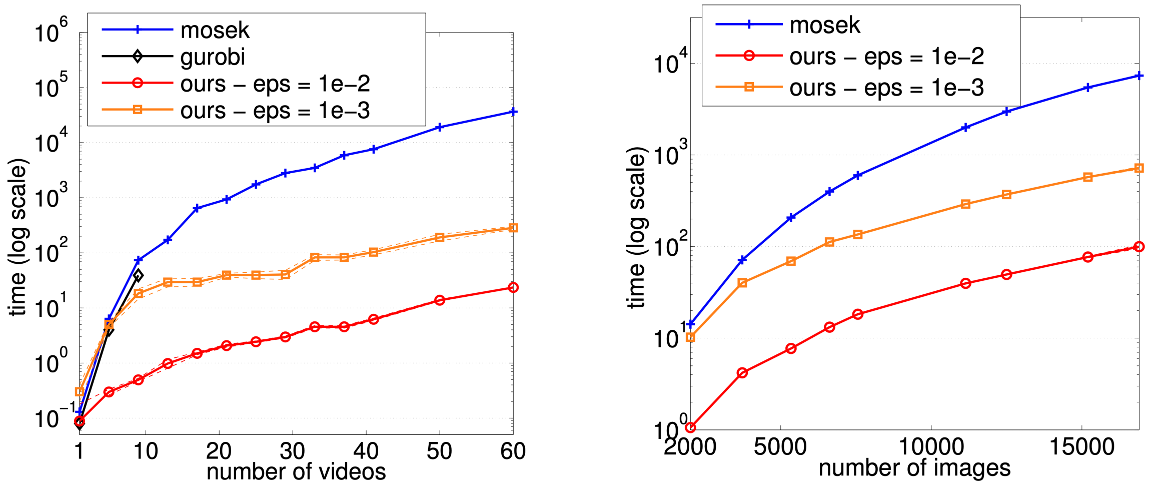

Figure 4.5: "Ours" in the legend of these plots refers to the Algorithm 14. Note that for $\epsilon = 1e - 3$ Algorithm 14 performs 100 times faster than standard solvers for more than 20 videos. Plots are from [23].

A visualization of image co-localization by solving the problem (4.7) for images on PASCAL Visual Object Classes 2007 dataset [14] is shown in Figure 4.6. This dataset provides standardized image data of 20 objects for object classes recognition along with annotations for images and bounding box and object class label for each object. Challenges and competitions have been used to recognize objects from a number of visual object classes in realistic scenes.

The YouTube-Objects dataset [43] consists of YouTube videos collected for 10 classes from PASCAL [14]: "aeroplane", "bird", "boat", "car", "cat", "cow", "dog", "horse", "motorbike", and

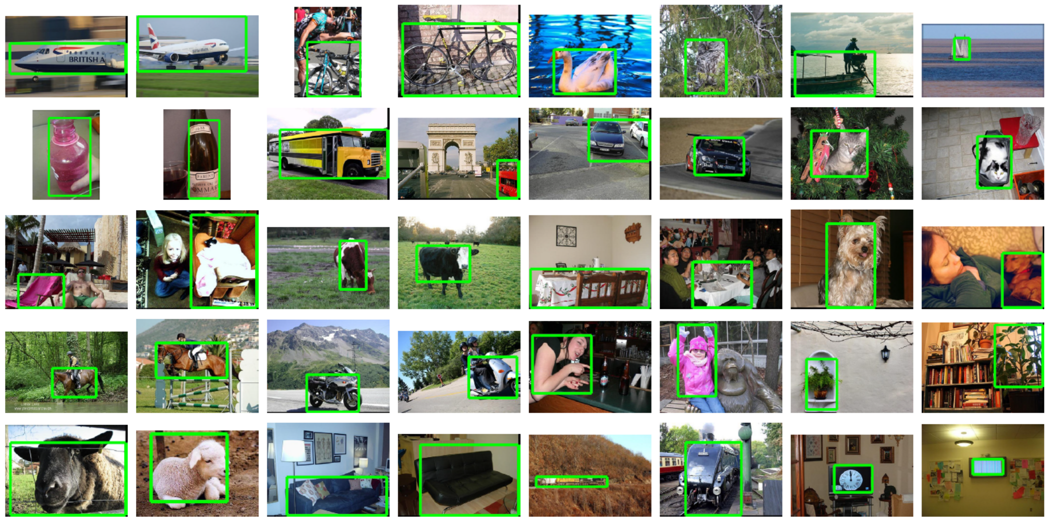

Figure 4.6: Example co-localization results on PASCAL07. From left to right, every two images belong to the same classes. Image is from [23].

”train”. A visualization of video co-localization by solving the problem (4.12) for videos on this dataset is also shown in Figure 4.7. Although authors in [23] did the study on multiple objects of this dataset, in our implementations our focus will be on the ”aeroplane” object class.

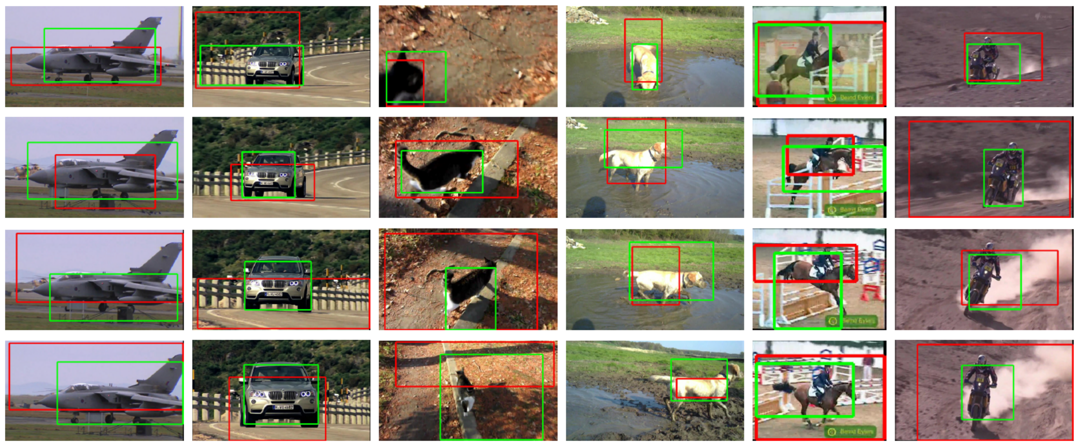

Figure 4.7: Example co-localization results on YouTube-Objects for the video model 4.12 with optimal green boxes and the image model 4.7 with optimal red boxes. Each column corresponds to a different class, and consists of frame samples from a single video. Image is from [23].

4.5.2 Implementations

Knowing that AFW Algorithm 14 outperforms the standard QP solvers Mosek and Gurobi from the works in [23], in this section we compare our proposed variants of the CGS algorithm, the ACGS Algorithm 12 and the PCGS Algorithm 13 to some other first order methods, including the AFW method. More precisely, we will compare the performance of our algorithms to all of the variants of the FW namely, the FW Algorithm 3 itself, the FW Algorithm 14 with away steps (AFW), and the pairwise FW Algorithm 5. We also compare our algorithms to the original CGS Algorithm 6. These comparisons include the duality gap, CPU time, and objective function value versus the iterations.

The implementations are over the YouTube Objects dataset [14] explained in previous section, and specifically its "aeroplane" class. We obtain the dataset for this class and also the codes for AFW and Pairwise FW algorithms available in the repositories for [23, 24, 45]. We only consider the task of video co-localization with the problem formulation defined in (4.12) for this implementation. All algorithms are coded in MATLAB and run on a computer with Interl Core i5-6500 CPU 3.2 GHz processor with 16 GB of RAM.

In our implementations, we set all algorithms to stop either after the maximum number of iterations or after reaching the Wolfe duality gap threshold. We set the threshold to $\epsilon = 1e-5$ and the max number of iterations to 2000 iterations. All of the parameters exist in (4.12) are set the same as in [24] for consistency in the comparison.

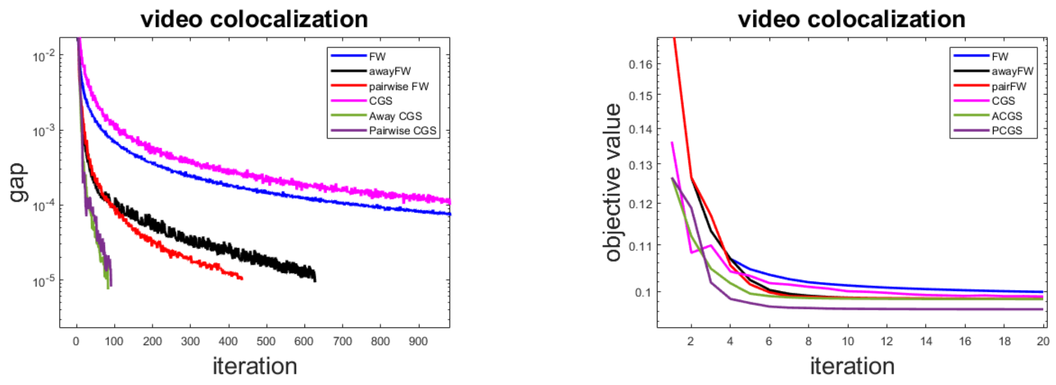

Figure 4.8: On the left we observe the difference in gap reduction and on the right the objective value improvement, both with iteration increments. Here $\epsilon = 1e-5$ and max number of iteration is 2000.

Note that both original versions of FW and CGS algorithms do not reach the desired duality gap before the preset 2000 max number of iterations. Also, the AFW algorithm takes 628 iterations, the Pairwise FW takes 436 iterations, the ACGS takes 84 iterations, and PCGS takes 82 iterations to reach the threshold for the duality gap.

As we observe in Figure 4.8 both proposed variants of CGS algorithm, the ACGS and PCGS algorithms outperform the FW algorithms and its variants as well as the original CGS algorithm. The performance of the algorithms in terms of the CPU time versus iterations increments also is represented in Figure 4.9. As we observe in this figure the CPU time per iteration of AFW and ACGS and PCGS are quite similar, although the ACGS and PCGS algorithms reach the gap much earlier than the AFW algorithm.

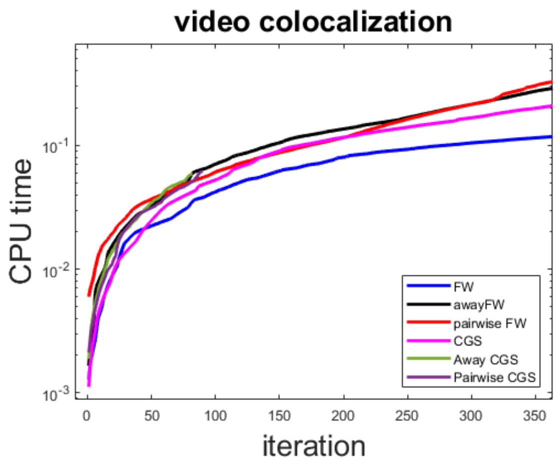

Figure 4.9: The difference in CPU time of the algorithms versus the iteration increments with $\epsilon = 1e - 5$ and max number of iteration is 2000.

In addition, while FW algorithm requires one linear optimization oracle per iteration, its CPU time per iteration is not significantly better than the other algorithms. Also, note that out of 84 iteration of the ACGS algorithm, it chooses the away direction in 34 iteration which improves the performance of CGS (with more than 2000 iterations) for this problem significantly.

Finally, authors in [24] proved, for the first time, the global linear convergence of the variants of FW algorithms, AFW and Pairwise FW, under strong convexity of the objective. One potential research work related to the current chapter is figure out the convergence of the proposed algorithms 12 and 13.

Appendices

Appendix A

Supplementary Material

In this chapter we state the convergence analysis of algorithms introduced in introduction chapter. For each algorithm we will state the mail convergence theorem and their parameter setting, unless the algorithm is parameter free.

Definition 1. *A function $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function if for any $x, y \in \mathbb{R}^n$ and any $\lambda \in [0, 1]$, we have*

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)y. \tag{A.1}$$

Theorem 4. *A smooth function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if and only if*

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle, \quad \forall x, y \in \mathbb{R}^n. \tag{A.2}$$

Proof. Let us fix any $x, y \in \mathbb{R}^n$, and denote $x_\lambda := \lambda x + (1 - \lambda)y$. Suppose that (A.2) holds. Then for any $\lambda \in [0, 1]$ we have

$$f(x) \geq f(x_\lambda) + \langle \nabla f(x_\lambda), x - x_\lambda \rangle \text{ and } f(y) \geq f(x_\lambda) + \langle \nabla f(x_\lambda), y - x_\lambda \rangle.$$

Noting that $x - x_\lambda = (1-\lambda)(x-y)$ and $y - x_\lambda = \lambda(y-x)$, using the above relations, we have

$$\begin{aligned}\lambda f(x) + (1-\lambda) f(y) &\geq \lambda \left[f(x_\lambda) + (1-\lambda)\langle \nabla f(x_\lambda), x - y\rangle\right] + (1-\lambda)\left[f(x_\lambda) + \lambda\langle \nabla f(x_\lambda), y - x\rangle\right] \\ &= f(x_\lambda), \ \forall \lambda \in [0,1].\end{aligned}$$

Therefore (A.1) holds and f is convex. Let us consider the other direction and suppose that (A.1) holds. Then for any $\lambda \in [0,1)$ we have

$$f(x_\lambda) \leq \lambda f(x) + (1-\lambda) f(y),$$

or

$$f(y) \geq \frac{f(x_\lambda) - \lambda f(x)}{1-\lambda} = f(x) + \frac{f(x_\lambda) - f(x)}{1-\lambda}. \tag{A.3}$$

Letting $\lambda \to 1$, we have

$$\lim_{\lambda\to 1} \frac{f(x_\lambda) - f(x)}{1-\lambda} = -\lim_{\lambda \to 1} \frac{f(y + \lambda(x-y)) - f(x)}{\lambda - 1} = -\frac{d}{d\lambda}\Big|_{\lambda=1} f(y+\lambda(x-y)) = -\langle f(x), x - y\rangle. \tag{A.4}$$

Combining (A.4) and (A.3) we obtain (A.2) and conclude the theorem.

□

Corollary 2. *Let f be a convex smooth function. We have,*

$$0 \leq f(y) - f(x) - \langle \nabla f(x), y - x\rangle \leq \frac{L}{2}\|y-x\|^2, \ \forall x, y \in \mathbb{R}^n. \tag{A.5}$$

A.1 Projected Gradient Descent Method

In this section we present the analysis of the convergence of PGD Algorithm 1.

Theorem 5. *Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is a smooth convex function. If the parameters η_k in Algorithm 1 satisfy*

$$\eta_k \equiv \eta \geq L, \tag{A.6}$$

then we have

$$f(\bar{x}_N) - f^* \leq \frac{\eta}{N}\frac{1}{2}\left\|x_0 - x^*\right\|_2^2,$$

where

$$\bar{x}_N := \frac{1}{N}\sum_{k=1}^{N} x_k.$$

Proof. Since f is a smooth convex function, from the Corollary 2 and Theorem 4 we have

$$\begin{aligned}
f(x_k) \leq & f(x_{k-1}) + \langle \nabla f(x_{k-1}), x_k - x_{k-1}\rangle + \frac{L}{2}\|x_k - x_{k-1}\|_2^2 \\
= & f(x_{k-1}) + \langle \nabla f(x_{k-1}), x - x_{k-1}\rangle + \langle \nabla f(x_{k-1}), x_k - x\rangle + \frac{L}{2}\left\|x_k - x_{k-1}\right\|^2 \quad \text{(A.7)} \\
\leq & f(x) + \langle \nabla f(x_{k-1}), x_k - x\rangle + \frac{\eta}{2}\left\|x_k - x_{k-1}\right\|^2 \quad \text{(A.8)} \\
\leq & f(x) + \frac{\eta}{2}\left(\left\|x_{k-1} - x\right\|^2 - \left\|x_{k-1} - x_k\right\|^2 - \left\|x_k - x\right\|^2\right) + \frac{\eta}{2}\left\|x_k - x_{k-1}\right\|^2 \\
= & f(x) + \frac{\eta}{2}\left(\left\|x_{k-1} - x\right\|^2 - \left\|x_k - x\right\|^2\right),
\end{aligned}$$

where the equality (A.7) is from (A.6), inequality (A.8) is from convexity of f. Hence,

$$f(x_k) \leq f(x) + \frac{\eta}{2}\left(\left\|x_{k-1} - x\right\|^2 - \left\|x_k -, x\right\|^2\right).$$

Summing the above inequality up from $k = 1$ to N, we obtain

$$\begin{aligned}
\sum_{k=1}^{N} f(x_k) &\leq Nf(x) + \frac{\eta}{2}\left(\left\|x_{k-1}, x\right\|^2 - \left\|x_k - x\right\|^2\right) \\
&\leq Nf(x) + \frac{\eta}{2}\left\|x_0 - x\right\|^2.
\end{aligned}$$

Setting $x = x^*$ in above relation and using the convexity of f, we have

$$f(\bar{x}_N) \leq \frac{1}{N}\sum_{k=1}^{N} f(x_k) \leq f(x^*) + \frac{\eta}{2N}\left\|x_0 - x^*\right\|^2.$$

Therefore,

$$f(\bar{x}_N) - f^* \leq \frac{\eta}{2N} \|x_0 - x^*\|^2 .$$

□

From the above theorem, we observe that in order to compute an approximate solution $\bar{x}_N$ such that $f(\bar{x}_N) - f^* \leq \varepsilon$, the number of iterations that are required is bounded by $\mathcal{O}\left(L\|x_0-x^*\|^2/\epsilon\right)$.

A.2 Nesterov's Accelerated Gradient Descent Method

In this section we present the analysis of the convergence of NAGD Algorithm 2. Theorem 6 gives a general form of an upper bound on $f(y_k) - f^*$ where y_k is the the output of the Algorithm 2 at iteration k and f^* is the optimal value.

Theorem 6. *Suppose that y_k and z_k in Algorithm 2 satisfy*

$$f(y_k) \leq f(z_k) + \langle \nabla f(z_k), y_k - z_k \rangle + \frac{L_k}{2} \|y_k - z_k\|^2 \tag{A.9}$$

for some $L_k > 0$, and that the parameters in Algorithm 2 satisfy

$$\gamma_1 = 1,\ \gamma_k \in [0, 1),\ \text{and } \eta_k \geq L_k \gamma_k, \quad \forall k \geq 1. \tag{A.10}$$

Letting Γ_k be a parameter that satisfies $\Gamma_1 > 0$ and

$$\Gamma_k = (1 - \gamma_k)\Gamma_{k-1},\ \forall k > 1, \tag{A.11}$$

then we have

$$f(y_k) - f^* \leq \Gamma_k \sum_{i=1}^{k} \frac{\gamma_i \eta_i}{2\Gamma_i} \left[\|x_{i-1} - x^*\|^2 - \|x_i - x^*\|^2 \right],$$

where x^* is a solution to (1.1).

Proof. Noting (1.6) and (1.8) we have $y_k - z_k = \gamma_k(x_k - x_{k-1})$, and also

$$\begin{aligned}
y_k - z_k &= y_k - y_{k-1} + y_{k-1} - z_k \\
&\overset{(1.8)}{=} \gamma_k(x_k - y_{k-1}) + y_{k-1} - z_k \\
&= \gamma_k \left[(x_k - x) + (x - z_k) + (z_k - y_{k-1})\right] + y_{k-1} - z_k \\
&= (1-\gamma_k)(y_{k-1} - z_k) + \gamma_k \left((x - z_k) + (x_k - x)\right) \\
&= (1-\gamma_k)(y_{k-1} - z_k) + \gamma_k \left(x_k - z_k\right).
\end{aligned}$$

Using above the inequality, (A.9) becomes

$$\begin{aligned}
f(y_k) &\le f(z_k) + (1-\gamma_k)\langle \nabla f(z_k), y_{k-1} - z_k\rangle + \gamma_k \langle \nabla f(z_k), x_k - z_k\rangle + \frac{L_k \gamma_k^2}{2}\|x_k - x_{k-1}\|^2 \\
&= (1-\gamma_k)[f(z_k) + \langle \nabla f(z_k), y_{k-1} - z_k\rangle] + \gamma_k[f(z_k) + \langle \nabla f(z_k), x - z_k\rangle + \langle \nabla f(z_k), x_k - x\rangle] \\
&\quad + \frac{L_k \gamma_k^2}{2}\|x_k - x_{k-1}\|^2, \quad \forall x \in \mathcal{X}.
\end{aligned}$$

Let us make three observations. First, by (A.5), we have

$$f(z_k) + \langle \nabla f(z_k), y_{k-1} - z_k\rangle \le f(y_{k-1}),$$

and

$$f(z_k) + \langle \nabla f(z_k), x - z_k\rangle \le f(x).$$

Second, we have

$$\langle \nabla f(z_k), x_k - x\rangle \le \frac{\eta_k}{2}\left[\|x_{k-1} - x\|^2 - \|x_{k-1} - x_k\|^2 - \|x_k - x\|^2\right], \ \forall x \in \mathcal{X}.$$

Summarizing the two observations, we have

$$\begin{aligned}
f(y_k) &\leq (1-\gamma_k)f(y_{k-1}) + \gamma_k f(x) + \frac{\gamma_k\eta_k}{2}\left[\|x_{k-1}-x\|^2 - \|x_{k-1}-x_k\|^2 - \|x_k - x\|^2\right] \\
&\quad + \frac{L_k\gamma_k^2}{2}\|x_{k-1}-x_k\|^2 \\
&= (1-\gamma_k)f(y_{k-1}) + \gamma_k f(x) + \frac{\gamma_k\eta_k}{2}\left[\|x_{k-1}-x\|^2 - \|x_k - x\|^2\right] \\
&\quad - \frac{1}{2}\gamma_k(\eta_k - L_k\gamma_k)\|x_{k-1}-x_k\|^2 \\
&\leq (1-\gamma_k)f(y_{k-1}) + \gamma_k f(x) + \frac{1}{2}\gamma_k\eta_k\left[\|x_{k-1}-x\|^2 - \|x_k - x\|^2\right].
\end{aligned}$$

Here the last inequality is from (A.10). Summing up the above two inequalities, we have

$$f(y_k) \leq (1-\gamma_k)f(y_{k-1}) + \gamma_k f(x) + \frac{1}{2}\gamma_k\eta_k\left[\|x_{k-1}-x\|^2 - \|x_k - x\|^2\right].$$

In particular, letting $x = x^*$ where x^* is a solution to (1.1), we can reformulate the above to

$$f(y_k) - f^* \leq (1-\gamma_k)(f(y_{k-1}) - f^*) + \frac{1}{2}\gamma_k\eta_k\left[\|x_{k-1}-x^*\|^2 - \|x_k - x^*\|^2\right]^2.$$

Dividing both sides by Γ_k, and using (A.11) and (A.10), we have

$$\frac{1}{\Gamma_k}(f(y_k) - f^*) \leq \frac{1}{\Gamma_{k-1}}(f(y_{k-1}) - f^*) + \frac{\gamma_k\eta_k}{2\Gamma_k}\left[\|x_{k-1}-x^*\|^2 - \|x_k - x^*\|^2\right], \quad \forall k > 1.$$

Also, when $k = 1$, noting that $\gamma_1 = 1$ by (A.10), we have

$$\frac{1}{\Gamma_1}(f(y_k) - f^*) \leq \frac{\gamma_1\eta_1}{2\Gamma_1}\left[\|x_0 - x^*\|^2 - \|x_1 - x^*\|^2\right].$$

Using induction on the two inequalities above, we conclude that

$$\frac{1}{\Gamma_k}(f(y_k) - f^*) \leq \sum_{i=1}^{k}\frac{\gamma_i\eta_i}{2\Gamma_i}\left[\|x_{i-1} - x^*\|^2 - \|x_i - x^*\|^2\right].$$

□

Having a general upper bound from Theorem 6 we can define a setting for the parameters of Algorithm 2 in Corollary 3 to get the optimal convergence results.

Corollary 3. *If we set*

$$\gamma_k = \frac{2}{k+1}, \quad \eta_k = \frac{2L}{k}$$

in Algorithm 2, then

$$f(y_k) - f^* \leq \frac{2L}{k(k+1)} \|x_0 - x^*\| .$$

Proof. Clearly (A.9) and (A.10) hold, and $\Gamma_k = {}^2/_{k(k+1)}$ satisfies (A.11) with $\Gamma_1 = 1$. Therefore, by Theorem 6, we have

$$f(y_k) - f^* \leq \frac{2}{k(k+1)} \sum_{i=1}^{k} L \left[\|x_{i-1} - x^*\|^2 - \|x_i - x^*\|^2 \right] = \frac{2L}{k(k+1)} \left[\|x_0 - x^*\|^2 - \|x_k - x^*\|^2 \right] .$$

Knowing that $V(x_k, x^*) \geq 0$ we conclude

$$f(y_k) - f^* \leq \frac{2L}{k(k+1)} \|x_0 - x^*\|^2 .$$

□

From Corollary 3 we can see that in order to compute an approximate solution such that $f(y_k) - f^* \leq \varepsilon$, we need

$$k \geq \sqrt{\frac{2L \|x_0 - x^*\|^2}{\varepsilon}}.$$

Therefore, the iteration complexity upper bound is $\mathcal{O}(\sqrt{{}^1/_\varepsilon})$. Note that the Nesterov's accelerated gradient method is optimal for solving smooth convex optimization with Lipschitz constant L [35].

A.3 Frank-Wolfe Method

In this section we present the analysis [15] of the convergence of FW Algorithm 3. We first state a simple technical result that will be used in the analysis of the algorithm.

Lemma 3. *Let $w_t \in (0,1]$, $t = 1, 2, \cdots,$ be given. Also let us denote*

$$W_t := \begin{cases} 1, & t = 1, \\ (1 - w_t)W_{t-1}, & t \geq 2. \end{cases}$$

Suppose that $W_t > 0$ for all $t \geq 2$ and that the sequence $\{\delta_t\}_{t \geq 0}$ satisfies

$$\delta_t \leq (1 - w_t)\delta_{t-1} + B_t, \quad t = 1, 2, \cdots. \tag{A.12}$$

Then for any $1 \leq l \leq k$, we have

$$\delta_k \leq W_k \left(\frac{1 - w_l}{W_l} \delta_{l-1} + \sum_{i=l}^{k} \frac{B_i}{W_i} \right).$$

Proof. Dividing both sides of (A.12) by W_t, we obtain

$$\frac{\delta_1}{W_1} \leq \frac{(1 - w_1)\delta_0}{W_1} + \frac{B_1}{W_1}$$

and

$$\frac{\delta_i}{W_i} \leq \frac{(1 - w_i)\delta_{i-1}}{W_i} + \frac{B_i}{W_i} = \frac{\delta_{i-1}}{W_{i-1}} + \frac{B_i}{W_i} \quad \forall i \geq 2.$$

The result then immediately follows by summing up the above inequalities for $i = 1 \cdots, k$ and rearranging the terms. □

In the following theorem and corollary the notation below will be used:

$$\Gamma_k = \begin{cases} 1 & \text{if } k = 1 \\ (1 - \gamma_k)\Gamma_{k-1} & \text{if } k \geq 2. \end{cases} \tag{A.13}$$

Theorem 7. *For parameters $\gamma_k \in (0,1)$ in Algorithm we have*

$$f(y_k) - f^* \leq \frac{LD^2\Gamma_k}{2} \sum_{i=1}^{k} \frac{\gamma_i^2}{\Gamma_i}.$$

Proof. Fist, we can write (1.11) as $y_k = y_{k-1} + \gamma_k(x_k - y_{k-1})$, so

$$y_k - y_{k-1} = \gamma_k(x_k - y_{k-1}). \tag{A.14}$$

Also, from (1.9) and (1.9) we observe that

$$\begin{aligned} y_k - z_k &= y_k - y_{k-1} + y_{k-1} - z_k \\ &\overset{(A.14)}{=} \gamma_k(x_k - y_{k-1}) + y_{k-1} - z_k \\ &= \gamma_k\left[(x_k - x) + (x - z_k) + (z_k - y_{k-1})\right] + y_{k-1} - z_k \\ &= \gamma_k\left((x - z_k) + (x_k - x)\right) + (1-\gamma_k)(y_{k-1} - z_k). \end{aligned} \tag{A.15}$$

Since f is a smooth convex function from Corollary (2) and also from (1.11) we have

$$f(y_k) \le f(z_k) + \langle \nabla f(z_k), y_k - z_k\rangle + \frac{L}{2}\left\|y_k - z_k\right\|^2.$$

Now from (A.15), for any $x \in X$ we have

$$\begin{aligned} f(y_k) \le{}& (1-\gamma_k)\left[f(z_k) + \langle \nabla f(z_k), y_{k-1} - z_k\rangle\right] \\ &+ \gamma_k\left[f(z_k) + \langle \nabla f(z_k), x - z_k\rangle + \langle \nabla f(z_k), x_k - x\rangle\right] \\ &+ \frac{L}{2}\left\|y_k - z_k\right\|^2 \\ ={}& (1-\gamma_k)(f(z_k) + \langle \nabla f(z_k), y_{k-1} - z_k\rangle) \\ &+ \gamma_k(f(z_k) + \langle \nabla f(z_k), x - z_k\rangle) \\ &+ \gamma_k\langle \nabla f(z_k), x_k - x\rangle + \frac{L}{2}\left\|y_k - z_k\right\|^2. \end{aligned}$$

Here we note that since x_k is an optimal solution to the subproblem (1.10), then by optimality condition we have

$$\langle \nabla f(z_k), x_k - x\rangle \le 0,$$

and also since f is a convex function we have

$$f(z_k) + \langle \nabla f(z_k), y_{k-1} - z_k \rangle \le f(y_{k-1})$$

and

$$f(z_k) + \langle \nabla f(z_k), x - z_k \rangle \le f(x).$$

In addition, from (1.9) and (1.11) we have $y_k - z_k = \gamma_k(x_k - x_{k-1})$. Summarizing all these and using (1.3) we obtain

$$f(y_k) \le (1 - \gamma_k) f(y_{k-1}) + \gamma_k f(x) + \frac{L\gamma_k^2}{2} D^2,$$

or equivalently,

$$f(y_k) - f(x) \le (1 - \gamma_k)(f(y_{k-1}) - f(x)) + \frac{L\gamma_k^2}{2} D^2.$$

Dividing both sides of the above inequality by Γ_k for $k \ge 2$ we obtain

$$\frac{f(y_k) - f(x)}{\Gamma_k} \le \frac{(1 - \gamma_k)}{\Gamma_k}(f(y_{k-1}) - f(x)) + \frac{L\gamma_k^2}{2\Gamma_k} D^2.$$

Also, for $k = 1$ we have

$$f(y_1) - f(x) \le \frac{LD^2\gamma_1^2}{2}.$$

Summing up from 1 to k we conclude that

$$\frac{1}{\Gamma_k}(f(y_k) - f(x)) \le \frac{LD^2}{2} \sum_{i=1}^{k} \frac{\gamma_i^2}{\Gamma_i}.$$

□

Note that the sequence of parameters γ_k in FW is conceptual and clearly there might be many choices to set these parameters properly to get the best convergence result for this algorithm. In Corollary 4 below we provide a setting for this parameter and prove the convergence rate corresponding to our setting.

Corollary 4. *In Algorithm 3, if we set the parameter* $\gamma_k = {}^2/_{(k+1)}$*, then*

$$f(y_k) - f^* \leq \frac{2LD^2}{k+1}. \tag{A.16}$$

Proof. With γ_k defined in assumption we have

$$\Gamma_k = \frac{2}{k(k+1)}. \tag{A.17}$$

Using (A.17) we obtain

$$\sum_{i=1}^{k} \frac{\gamma_i^2}{\Gamma_i} = \sum_{i=1}^{k} \frac{4}{(i+1)^2} \times \frac{i(i+1)}{2} = \sum_{i=1}^{k} \frac{2i}{i+1} = 2(\sum_{i=1}^{k} 1 - \sum_{i=1}^{k} \frac{1}{i+1}) \leq 2k.$$

Therefore,

$$\begin{aligned} f(y_k) - f^* &\leq \frac{LD^2}{2} \times \frac{2}{k(k+1)} \times 2k \\ &= \frac{2LD^2}{k+1}. \end{aligned}$$

□

Corollary 4 shows that the FW method computes an ϵ-solution to the problem (1.1) in $\mathcal{O}({}^{LD^2}/_{\epsilon})$ iterations. This means that in order to compute an ϵ solution, FW requires more evaluations of ∇f than NAGD method, which only requires $\mathcal{O}(\sqrt{{}^1/_{\epsilon}})$ evaluations. This drawback is resolved in *conditional gradient sliding* method [30]. In particular, conditional gradient sliding method requires $\mathcal{O}(\sqrt{{}^1/_{\epsilon}})$ evaluations of $\nabla f(\cdot)$ and $\mathcal{O}({}^1/_{\epsilon})$ evaluations for linear optimization problems of form (1.10). we will discuss the conditional gradient sliding method in later sections.

According to [25, 36] the number of evaluations of linear optimization problems of form (1.10) can not be improved from the lower complexity bound $\mathcal{O}({}^1/_{\epsilon})$. Also, it should be noted that the FW does not require knowledge on the Lipschitz constant L, the norm $\|.\|$, and diameter D. In particular, if there exists a norm $\|.\|$ that yields the smallest possible value of LD^2, then the convergence result (A.16) will follow such smallest value. In other words, the FW is a first-order method that would automatically adapt to the best possible geometric properties of the problem.

A.4 Conditional Gradient Sliding Method

In this section we present the analysis [30] of the convergence of CGS Algorithm 6. We first state a simple technical result that will be used in the analysis of the algorithm.

Lemma 4. *Let $\{\lambda_i\}$ and $\{a_i\}$ be sequences of nonnegative real numbers. Then for a fixed k,*

1- If the sequence $\{\lambda_i\}$ is a decreasing sequence, then

$$\sum_{i=1}^{k} \lambda_i (a_{i-1} - a_i) \leq \lambda_0 a_0.$$

2- If the sequence $\{\lambda_i\}$ is an increasing sequence, then

$$\sum_{i=1}^{k} \lambda_i (a_{i-1} - a_i) \leq \lambda_k \max_{0 \leq t \leq k} a_t.$$

Proof. In order to prove part 1 we have

$$\begin{aligned}
\sum_{i=1}^{k} \lambda_i (a_{i-1} - a_i) &= -\sum_{i=1}^{k} \lambda_i (a_i - a_{i-1}) \\
&= -\sum_{i=1}^{k} \lambda_i a_i + \sum_{i=1}^{k} (\lambda_i - \lambda_{i-1}) a_{i-1} + \sum_{i=1}^{k} \lambda_{i-1} a_{i-1} \\
&= -\lambda_k a_k - \sum_{i=1}^{k-1} \lambda_i a_i + \sum_{i=1}^{k} (\lambda_i - \lambda_{i-1}) a_{i-1} + \lambda_0 a_0 + \sum_{i=1}^{k-1} \lambda_i a_i \\
&= \lambda_0 a_0 - \lambda_k a_k - \sum_{i=1}^{k} (\lambda_{i-1} - \lambda_i) a_{i-1} \\
&\leq \lambda_0 a_0.
\end{aligned}$$

Where the last inequality holds because $\{\lambda_i\}$ is decreasing.

To prove the second part we have

$$\begin{aligned}\sum_{i=1}^{k} \lambda_i(a_{i-1} - a_i) &= -\lambda_k a_l + \lambda_0 a_0 + \sum_{i=1}^{k}(\lambda_i - \lambda_{i-1}) a_{i-1} \\ &\leq \lambda_0 a_0 + \sum_{i=1}^{k}(\lambda_i - \lambda_{i-1}) \max_{0\leq t\leq k} a_t \\ &= \lambda_0 a_0 + (\lambda_k - \lambda_0) \max_{0\leq t\leq k} a_t \\ &\leq \lambda_k \max_{0\leq t\leq k} a_t.\end{aligned}$$

□

Theorem 8 describes the main convergence properties of the above CGS method.

Theorem 8. *Let* Γ_k *be defined in* (A.13). *Suppose that* $\{\beta_k\}$ *and* $\{\gamma_k\}$ *in the CGS algorithm satisfy*

$$\gamma_1 = 1 \quad \text{and} \quad L\gamma_k \leq \beta_k, \quad k \geq 1. \tag{A.18}$$

(a) If

$$\frac{\beta_k \gamma_k}{\Gamma_k} \geq \frac{\beta_{k-1}\gamma_{k-1}}{\Gamma_{k-1}}, \quad k \geq 2, \tag{A.19}$$

then for any $x \in \mathcal{X}$ *and* $k \geq 1$,

$$f(y_k) - f(x^*) \leq \frac{\beta_k \gamma_k}{2} D_{\mathcal{X}}^2 + \Gamma_k \sum_{i=1}^{k} \frac{\eta_i \gamma_i}{\Gamma_i},$$

where x^* *is an arbitrary optimal solution of* (1.1) *and* $D_{\mathcal{X}}$ *is defined in* (1.3).

(b) If

$$\frac{\beta_k \gamma_k}{\Gamma_k} \leq \frac{\beta_{k-1}\gamma_{k-1}}{\Gamma_{k-1}}, \quad k \geq 2,$$

then for any $x \in \mathcal{X}$ *and* $k \geq 1$,

$$f(y_k) - f(x^*) \leq \frac{\beta_1 \Gamma_k}{2} \|x_0 - x^*\|^2 + \Gamma_k \sum_{i=1}^{k} \frac{\eta_i \gamma_i}{\Gamma_i},$$

(c) *Under the assumptions either in part (a) or (b), the number of inner iterations performed at the kth outer iteration can be bounded by*

$$T := \left\lceil \frac{6\beta_k D_{\mathcal{X}}^2}{\eta_k} \right\rceil \quad \forall k \geq 1. \tag{A.20}$$

Proof. To prove part (a) note that by (1.25) and (1.27) we have $y_k - z_k = \gamma_k(x_k - x_{k-1})$. Also, from (1.27) we have

$$\begin{aligned} y_k - z_k &= (1-\gamma_k)y_{k-1} + \gamma_k x_k - z_k \\ &= (y_{k-1} - z_k) + \gamma_k(x_k - y_{k-1}) \\ &= (1-\gamma_k)(y_{k-1} - z_k) + \gamma_k(x_k - z_k) \end{aligned}$$

Using this and also (A.5) we have

$$\begin{aligned} f(y_k) &\leq f(z_k) + \langle \nabla f(z_k), y_k - z_k \rangle + \frac{L}{2}\left\| y_k - z_k \right\|^2 \\ &= (1-\gamma_k)(f(z_k) + \langle \nabla f(z_k), y_{k-1} - z_k \rangle) \\ &\quad + \gamma_k(f(z_k) + \langle \nabla f(z_k), x_k - z_k \rangle) + \frac{L\gamma_k^2}{2}\left\| x_k - x_{k-1} \right\|^2 \\ &\leq (1-\gamma_k)f(y_{k-1}) + \gamma_k(f(z_k) + \langle \nabla f(z_k), x_k - z_k \rangle) + \frac{\beta_k \gamma_k}{2}\left\| x_k - x_{k-1} \right\|^2 \end{aligned} \tag{A.21}$$

where the last inequality follows from convexity of f and (A.18). Also note that from the optimality condition we have

$$\langle \nabla f(z_k) + \beta_k(x_k - x_{k-1}), x_k - x \rangle \leq \eta_k \qquad \forall x \in \mathcal{X},$$

and so

$$\langle x_k - x_{k-1}, x_k - x \rangle \leq \frac{\eta_k}{\beta_k} + \frac{1}{\beta_k}\langle f'(z_k), x - x_k \rangle \qquad \forall x \in \mathcal{X}. \tag{A.22}$$

Also, note that

$$\begin{aligned} \frac{1}{2}\left\| x_{k-1} - x \right\|^2 &= \frac{1}{2}\left\| (x_{k-1} - x_k) + (x_k - x) \right\|^2 \\ &= \frac{1}{2}\left\| x_k - x_{k-1} \right\|^2 + \langle x_{k-1} - x_k, x_k - x \rangle + \frac{1}{2}\left\| x_k - x \right\|^2, \end{aligned}$$

which implies that

$$\begin{aligned}\frac{1}{2}\left\|x_k - x_{k-1}\right\|^2 &= \frac{1}{2}\left\|x_{k-1} - x\right\|^2 - \langle x_{k-1} - x, x_k - x\rangle - \frac{1}{2}\left\|x_k - x\right\|^2 \\ &\leq \frac{1}{2}\left\|x_{k-1} - x\right\|^2 + \frac{1}{\beta_k}\left\langle f'(z_k), x - x_k\right\rangle - \frac{1}{2}\left\|x_k - x\right\|^2 + \frac{\eta_k}{\beta_k}.\end{aligned}$$

Hence,

$$\frac{\beta_k\gamma_k}{2}\left\|x_k - x_{k-1}\right\|^2 \leq \frac{\beta_k\gamma_k}{2}\left(\left\|x_{k-1} - x\right\|^2 - \left\|x_k - x\right\|^2\right) + \gamma_k\left\langle \nabla f(z_k), x - x_k\right\rangle + \eta_k\gamma_k. \tag{A.23}$$

Combining (A.21), (A.23) and (A.22) we obtain

$$\begin{aligned}f(y_k) &\leq (1-\gamma_k)f(y_{k-1}) + \gamma_k(f(z_k) + \langle \nabla f(z_k), x_k - z_k\rangle + \langle \nabla f(z_k), x - x_k\rangle) \\ &\quad + \frac{\beta_k\gamma_k}{2}\left(\left\|x_{k-1} - x\right\|^2 - \left\|x_k - x\right\|^2\right) + \eta_k\gamma_k \\ &= (1-\gamma_k)f(y_{k-1}) + \gamma_k(f(z_k) + \langle \nabla f(z_k), x - z_k\rangle) \\ &\quad + \frac{\beta_k\gamma_k}{2}\left(\left\|x_{k-1} - x\right\|^2 - \left\|x_k - x\right\|^2\right) + \eta_k\gamma_k \\ &\leq (1-\gamma_k)f(y_{k-1}) + \gamma_k f(x) + \frac{\beta_k\gamma_k}{2}\left(\left\|x_{k-1} - x\right\|^2 - \left\|x_k - x\right\|^2\right) + \eta_k\gamma_k,\end{aligned}$$

where the last inequality is from convexity of f. Subtracting $f(x)$ from both sides of above inequality gives

$$f(y_k) - f(x) \leq (1-\gamma_k)(f(y_{k-1}) - f(x)) + \frac{\beta_k\gamma_k}{2}\left(\left\|x_{k-1} - x\right\|^2 - \left\|x_k - x\right\|^2\right) + \eta_k\gamma_k \quad \forall x \in \mathcal{X}.$$

Now using Lemma 3,

$$\begin{aligned}f(y_k) - f(x) &\leq \frac{\Gamma_k(1-\gamma_1)}{\Gamma_1}\left[f(y_0) - f(x)\right] \\ &\quad + \Gamma_k\sum_{i=1}^{k}\frac{\beta_i\gamma_i}{2\Gamma_i}\left(\left\|x_{i-1} - x\right\|^2 - \left\|x_i - x\right\|^2\right) + \Gamma_k\sum_{i=1}^{k}\frac{\eta_i\gamma_i}{\Gamma_i}.\end{aligned} \tag{A.24}$$

Note that $\gamma_1 = 1$ and since from the assumption $\{\beta_k\gamma_k/\Gamma_k\}$ is increasing; then from Lemma 4

$$\sum_{i=1}^{k}\frac{\beta_i\gamma_i}{2\Gamma_i}\left(\left\|x_{i-1} - x\right\|^2 - \left\|x_i - x\right\|^2\right) \leq \frac{\beta_k\gamma_k}{\Gamma_k}D_{\mathcal{X}}^2. \tag{A.25}$$

Hence, we have

$$f(y_k) - f(x) \leq \frac{\beta_k \gamma_k}{2} D_{\mathcal{X}}^2 + \Gamma_k \sum_{i=1}^{k} \frac{\eta_i \gamma_i}{\Gamma_i}.$$

which completes the proof of part (a).

To prove part (b), from (A.25) and Lemma 4 the assumption we have

$$\sum_{i=1}^{k} \frac{\beta_i \gamma_i}{2\Gamma_i} \left(\|x_{i-1} - x\|^2 - \|x_i - x\|^2 \right) \leq \beta_1 \|x_0 - x\|^2$$

Therefore, by (A.24) for any $x \in \mathcal{X}$ we have

$$f(y_k) - f(x) \leq \frac{\Gamma_k}{2} \beta_1 \|x_0 - x\|^2 + \Gamma_k \sum_{i=1}^{k} \frac{\eta_i \gamma_i}{\Gamma_i},$$

Which is true for $x = x^*$, and this completes the proof of part (b).

To prove part (c) let us denote $\phi \equiv \phi_k := \langle f'(z_k), x \rangle + \beta_k/2 \|x - x_{k-1}\|^2$ and $\phi^* \equiv \min_{x \in \mathcal{X}} \phi(x)$. Also let us denote

$$\lambda_t := \frac{2}{t} \quad \text{and} \quad \Lambda_t = \frac{2}{t(t-1)}, \tag{A.26}$$

which implies that

$$\Lambda_{t+1} = \Lambda_t (1 - \lambda_{t+1}) \quad \forall t \geq 2.$$

Let us define $\bar{u}_{t+1} := (1 - \lambda_{t+1}) u_t + \lambda_{t+1} v_t$. Clearly we have $\bar{u}_{t+1} - u_t + \lambda_{t+1}(v_t - u_t)$. Observe that $u_{t+1} = (1 - \alpha_t) u_t + \alpha_t v_t$ and α_t is an optimal solution to $\arg\min_{\alpha \in [0,1]} f((1-\alpha) u_t + \alpha v_t)$ and hence $\phi(u_{t+1}) \leq \phi(\bar{u}_{t+1})$. Using this observation, (A.5), and the fact that ϕ has Lipschitz continuous gradients, we have

$$\begin{aligned}\phi(u_{t+1}) &\leq \phi(\bar{u}_{t+1}) \\ &\leq \phi(u_t) + \langle \phi'(u_t), \bar{u}_{t+1} - u_t \rangle + \frac{\beta}{2} \left\| \bar{u}_{t+1} - u_t \right\|^2 \\ &= \phi(u_t) + \lambda_{t+1} \langle \phi'(u_t), v_t - u_t \rangle + \frac{\beta}{2} \lambda_{t+1}^2 \left\| v_t - u_t \right\|^2 \\ &= \phi(u_t) - \lambda_{t+1}\phi(u_t) + \lambda_{t+1} \left(\phi(u_t) + \langle \phi'(u_t), v_t - u_t \rangle \right) + \frac{\beta}{2} \lambda_{t+1}^2 \left\| v_t - u_t \right\|^2 \\ &\leq (1 - \lambda_{t+1})\phi(u_t) + \lambda_{t+1} \left(\phi(u_t) + \langle \phi'(u_t), x - u_t \rangle \right) + \frac{\beta}{2} \lambda_{t+1}^2 \left\| v_t - u_t \right\|^2 \\ &\leq (1 - \lambda_{t+1})\phi(u_t) + \lambda_{t+1}\phi(x) + \frac{\beta}{2} \lambda_{t+1}^2 \left\| v_t - u_t \right\|^2 . \end{aligned} \tag{A.27}$$

Subtracting $\phi(x)$ from both sides implies that

$$\phi(u_{t+1}) - \phi(x) \leq (1 - \lambda_{t+1})(\phi(u_t) - \phi(x)) + \frac{\beta}{2} \lambda_{t+1}^2 \left\| v_t - u_t \right\|^2 \quad \forall x \in \mathcal{X}.$$

By Lemma 3, for any $x \in \mathcal{X}$ and $t \geq 1$

$$\begin{aligned}\phi(u_{t+1}) - \phi(x) &\leq \Lambda_{t+1} \left(\frac{1 - \lambda_2}{\Lambda_1} (\phi(1) - \phi(x)) \right) + \sum_{i=2}^{t+1} \frac{\beta \lambda_i^2}{2\Lambda_i} \left\| v_{i-1} - u_{i-1} \right\|^2 \\ &= \Lambda_{t+1} \beta \sum_{i=1}^{t} \frac{i}{i+1} \left\| v_i - u_i \right\|^2 \\ &\leq \frac{2\beta D_{\mathcal{X}}^2}{t+1} \end{aligned} \tag{A.28}$$

Now, let the gap function $V_{g,u,\beta}$ be defined in (1.28). Also let us denote $\Delta_j = \phi(u_j) - \phi^*$. It then follow from (1.28), and (A.27) that for any $j = 1, \cdots, t$,

$$\phi(u_{j+1}) \leq \phi(u_j) + \lambda_{j+1} \langle \phi'(u_j), v_j - u_j \rangle + \frac{\beta \lambda_{j+1}^2}{2} \left\| v_j - u_j \right\|^2 .$$

Hence,

$$\lambda_{j+1} \langle \phi'(u_j), u_j - v_j \rangle \leq \phi(u_j) - \phi(u_{j+1}) + \frac{\beta \lambda_{j+1}^2}{2} \left\| v_j - u_j \right\|^2 ,$$

which implies that

$$\begin{aligned}\lambda_{j+1}V_{g,u,\beta}(u_j) &\le \phi(u_j) - \phi(u_{j+1}) + \frac{\beta\lambda_{j+1}^2}{2}\|v_j - u_j\|^2 \\ &= \Delta_j - \Delta_{j+1} + \frac{\beta\lambda_{j+1}^2}{2}\|v_j - u_j\|^2.\end{aligned}$$

Dividing both sides of above inequality by Λ_{j+1} and summing up the resulting inequalities, we obtain

$$\begin{aligned}\sum_{j=1}^{t}\frac{\lambda_{j+1}}{\Lambda_{j+1}}V_{g,u,\beta}(u_j) &\le \sum_{j=1}^{t}\frac{\Delta_j - \Delta_{j+1}}{\Lambda_{j+1}} + \sum_{j=1}^{t}\frac{\beta\lambda_{j+1}^2}{2\Lambda_{j+1}}\|v_j - u_j\|^2 \\ &= -\frac{1}{\Lambda_{t+1}}\Delta_{t+1} + \sum_{j=2}^{t}\left(\frac{1}{\Lambda_{j+1}} - \frac{1}{\Lambda_j}\right)\Delta_j + \Delta_1 + \sum_{j=1}^{t}\frac{\beta\lambda_{j+1}^2}{2\Lambda_{j+1}}\|v_j - u_j\|^2 \\ &\le \sum_{j=2}^{t}\left(\frac{1}{\Lambda_{j+1}} - \frac{1}{\Lambda_j}\right)\Delta_j + \Delta_1 + \sum_{j=1}^{t}\frac{\beta\lambda_{j+1}^2}{2\Lambda_{j+1}}\|v_j - u_j\|^2 \\ &\le \sum_{j=1}^{t} j\Delta_j + \beta\sum_{j=1}^{t}\frac{j}{j+1}D_{\mathcal{X}}^2 \\ &\le \sum_{j=1}^{t} j\Delta_j + t\beta D_{\mathcal{X}}^2,\end{aligned}$$

where the last inequality follow from the definition of λ_t and Λ_t in (A.26). Using the above inequality and the bound on Δ_j given in (A.28), we conclude that

$$\min_{j=1,\ldots,t} V_{g,u,\beta}(u_j)\sum_{j=1}^{t}\frac{\lambda_{j+1}}{\Lambda_{j+1}} \le \sum_{j=1}^{t}\frac{\lambda_{j+1}}{\Lambda_{j+1}}V_{g,u,\beta}(u_j) \le \sum_{j=1}^{t} j\frac{2\beta D_{\mathcal{X}}^2}{j} + t\beta D_{\mathcal{X}}^2 = 3t\beta D_{\mathcal{X}}^2.$$

Since $\sum_{j=1}^{t}\lambda_{j+1}/\Lambda_{j+1} = t(t+1)/2$, then

$$\min_{j=1,\ldots,t} V_{g,u,\beta}(u_j)\left(\frac{t(t+1)}{2}\right) \le 3t\beta D_{\mathcal{X}}^2,$$

Therefore,

$$\min_{j=1,\ldots,t} V_{g,u,\beta}(u_j) \le \frac{6\beta D_{\mathcal{X}}^2}{t+1},$$

which implies part (c). □

Clearly, there exist various options to specify the parameters $\{\beta_k\}$, $\{\gamma_k\}$, and $\{\eta_k\}$ so as to guarantee the convergence of the CGS method. In the following corollaries, we provide two different parameter settings for $\{\beta_k\}$, $\{\gamma_k\}$, and $\{\eta_k\}$, which lead to optimal complexity bounds on the total number of calls to the first-order and linear optimization oracles for smooth convex optimization.

Corollary 5. *If $\{\beta_k\}$, $\{\gamma_k\}$, and $\{\eta_k\}$ in the CGS method are set to*

$$\beta_k = \frac{3L}{k+1}, \quad \gamma_k = \frac{3}{k+2} \quad \text{and} \quad \eta_k = \frac{LD_{\mathcal{X}}^2}{k(k+1)}, \qquad \forall k \geq 1, \tag{A.29}$$

then for any $k \geq 1$,

$$f(y_k) - f(x^*) \leq \frac{15LD_{\mathcal{X}}^2}{(k+1)(k+2)}.$$

As a consequence, the total number of calls to the first-order and linear optimization oracles performed by the CGS method for finding an ϵ-solution of (1.1) *can be bounded by $\mathcal{O}(\sqrt{LD_{\mathcal{X}}^2/\epsilon})$ and $\mathcal{O}(LD_{\mathcal{X}}^2/\epsilon)$, respectively.*

Proof. It can be easily seen from (A.29) and (A.18) holds. Also note that by (A.29), we have

$$\Gamma_k = \frac{6}{k(k+1)(k+2)} \tag{A.30}$$

and

$$\frac{\beta\gamma_k}{\Gamma_k} = \frac{9L}{(k+1)(k+2)} \cdot \frac{k(k+1)(k+2)}{6} = \frac{3Lk}{2},$$

which implies that (A.19) is satisfied. It then follows from Theorem 8(a), (A.29), and (A.30) that

$$f(y_k) - f(x^*) \leq \frac{9LD_{\mathcal{X}}^2}{2(k+1)(k+2)} + \frac{6}{k(k+1)(k+2)} \sum_{i=1}^{k} \frac{\eta_i\gamma_i}{\Gamma_i} = \frac{15LD_{\mathcal{X}}^2}{2(k+1)(k+2)},$$

which implies that the total number of outer iterations performed by the CGS method for finding an ϵ-solution can be bounded by $N = \sqrt{15LD_{\mathcal{X}}^2/2\epsilon}$. Moreover, it follows from the bound in (A.20) and (A.29) that the total number of inner iterations can be bounded by

$$\sum_{k=1}^{N} T_k \leq \sum_{k=1}^{N} \left(\frac{6\beta D_{\mathcal{X}}^2}{\eta_k} + 1 \right) = 18 \sum_{k=1}^{N} k + N = 9N^2 + 10N,$$

which implies that the total number of inner iterations is bounded by $\mathcal{O}({}^{LD_{\mathcal{X}}^2}/_{\epsilon})$.

□

Observe that in the above result, the number of calls to the linear optimization oracle is not improvable in terms of their dependence on ϵ, L and $D_{\mathcal{X}}$ for linear optimization-based convex programming methods [25]. Similarly, the number of calls to the FO oracle is also optimal in terms of its dependence on ϵ and L [35].

A.5 Accelerated Bundle-Level method

In this section we state the main convergence results of ABL. Specifically, Theorem 9 presents a bound on the maximum number of gap reductions performed by ABL Algorithm 8 in a typical phase. In Theorem 10 we can find the total number of phases and also total number of iterations performed by ABL algorithm to get an ϵ solution $\bar{x} \in X$ such that $f(x) - f(\bar{x}) < \epsilon$. While the proof of the results follow a similar logic and trend as the results for APL method we leave the proofs and the way that parameters are set for this algorithm to the reader that can be found in [26].

Theorem 9. *Let $\lambda \in (0,1)$ and $\alpha_k \in (0,1], k = 1,2,\dots,$ be given. Also, let $\Delta_k = \bar{f}_k - \underline{f}_k$ denote the optimality gap obtained at k-th iteration of procedure $\mathcal{G}_{ABL}$ before it terminates. Then for any $k = 1,2,\dots$ we have*

$$\Delta_k \leq \gamma_k(\lambda)\left[(1-\lambda\alpha_1)\Delta_0 + \frac{MD_X^{1+\nu}}{1+\nu}\left\|\Gamma_k(\lambda,\nu)\right\|_{\frac{2}{1-\nu}}\right],$$

where $D_X := \max_{x,y\in X}\|x-y\|$ and

$$\gamma_k(\lambda) := \begin{cases} 1 & k = 1 \\ (1-\lambda\alpha_k)\gamma_{k-1}(\lambda) & k \geq 2 \end{cases}$$

$$\Gamma_k(\lambda,\nu) := \left\{\gamma_1(\lambda)^{-1}\alpha_1^{1+\nu}, \gamma_2(\lambda)^{-1}\alpha_2^{1+\nu}, \dots, \gamma_k(\lambda)^{-1}\alpha_k^{1+\nu}\right\}.$$

In particular, if λ and $\alpha_k \in (0,1]$, $k = 1,2,\dots,$ are chosen such that for some $c_1, c_2 > 0$

$$\gamma_k(\lambda) \leq c_1 k^{-2} \quad \text{and} \quad \gamma_k \left\|\Gamma_k(\lambda,\nu)\right\|_{\frac{2}{1-\nu}} \leq c_2 k^{-\frac{1+3\nu}{2}}, \tag{A.31}$$

then the number of iterations performed by procedure $\mathcal{G}_{ABL}$ can be bounded by

$$K_{ABL}(\Delta_0) := \left\lceil \sqrt{\frac{2c_1(1-\lambda\alpha_1)}{\lambda}} + \left(\frac{2c_2 M D_X^{1+\nu}}{(1+\nu)\Delta_0}\right)^{\frac{2}{1+3\nu}} \right\rceil .$$

Theorem 10. *Suppose that $\lambda \in (0,1)$ and $\alpha_k \in (0,1]$, $k = 1, 2, \ldots,$ in procedure $\mathcal{G}_{ABL}$ are chosen such that* (A.31) *holds for some $c_1, c_2 > 0$. Then*

a) The number of phases performed by ABL method does not exceed

$$S(\epsilon) = \left\lceil \max\left\{0, \log_{\frac{1}{q}} \frac{MD_X^{1+\nu}}{(a+\nu)\epsilon}\right\} \right\rceil .$$

b) The total number of iteration performed by ABL method can be bounded by

$$\left(1 + \sqrt{\frac{2c_1}{\lambda}}\right) S(\epsilon) + \frac{1}{1-\lambda^{\frac{2}{1+3\nu}}} \left(\frac{2c_2 M D_X^{1+\nu}}{(1+\nu)\epsilon}\right)^{\frac{2}{1+3\nu}} .$$

Note that ABL method achieves the optimal complexity bounds for smooth, nonsmooth and weakly smooth convex optimization. For example, when $\nu = 0$ and function f is smooth the total number of iterations performed by ABL is bounded by $\mathcal{O}(1/\sqrt{\epsilon})$. Also, this algorithm is independent of the parameter M defined in (1.33).

A.6 Accelerated Prox-Level Method

In this section we perform the convergence analysis of the APL method. We begin with some technical results that will be used in the analysis.

Lemma 5. *Let X' be a localizer of the level set X_ℓ for some $\ell \in \mathbb{R}$ and $h(z,x)$ be defined in* (1.34). *Denoting*

$$h' := \min\{h(z,x) \ : \ x \in X'\} \tag{A.32}$$

we have

$$\min\{\ell, h'\} \le f(x), \quad \forall x \in X. \tag{A.33}$$

Proof. If $\ell \le f(x)$ for all $x \in X$, then $X_\ell = X' = \emptyset$ and the subprblem (A.32) is infeasible and so $h' = \infty$. In this case (A.33) clearly holds. Otherwise, let $h_\ell := \min\{h(z,x) \ : \ x \in X_\ell\}$. Then from (1.40) we clearly have $h' \le h_\ell \le f(x) \le \ell$ for all $x \in X_\ell$. In this case (A.33) then follows from the convexity of f. $\square$

Lemma 6. *Let $x_{k-1}, \ x^u_{k-1} \in X$ be given from iterations of $\mathcal{G}_{APL}$ for $k \ge 1$ and $h(z,\cdot)$ be defined in 1.34 and $x_k, \tilde{x}^u_k \in X$ are so that*

$$\begin{aligned} &h(x^\ell_k, x_k) \le \ell \qquad \textit{for some } \ell \\ &\tilde{x}^u_k = \gamma_k x_k + (1-\gamma_k) x^u_{k-1} \qquad \gamma \in (0,1) \end{aligned} \tag{A.34}$$

Then,

$$f(\tilde{x}^u_k) \le (1-\gamma_k) f(x^u_{k-1}) + \gamma_k \ell + \frac{L}{2} \left\| \gamma_k (x_k - x_{k-1}) \right\|^2$$

Proof. Note that from (1.42) and (A.34) we have

$$\tilde{x}^u_k - x^\ell_k = \gamma_k (x_k - x_{k-1}).$$

Also, from Lipschitz continuity of f' and convexity of f we have

$$\begin{aligned} f(\tilde{x}^u_k) &\le f(x^\ell_k) + \langle f'(x^\ell_k), \tilde{x}^u_k - x^\ell_k \rangle + \frac{L}{2} \left\| \tilde{x}^u_k - x^\ell_k \right\|^2 \\ &= (1-\gamma_k) h(x^\ell_k, x^u_{k-1}) + \gamma_k h(x^\ell_k, x_k) + \frac{L}{2} \left\| \tilde{x}^u_k - x^\ell_k \right\|^2 \\ &\le (1-\gamma_k) f(x^u_{k-1}) + \gamma_k \ell + \frac{L}{2} \left\| \gamma_k (x_k - x_{k-1}) \right\|^2 \end{aligned}$$

$\square$

Lemma 7 presents some observations regarding the the execution of the gap reduction procedure $\mathcal{G}_{APL}$.

Lemma 7. *The following statements hold for $\mathcal{G}_{APL}$ in Algorithm 9.*

a) $\{X'_k\}_{k \ge 0}$ is a sequence of localizers of the level set X_ℓ;

b) $\underline{f}_0 \le \underline{f}_1 \le \cdots \le \underline{f}_k \le f^*$ and $\bar{f}_0 \ge \bar{f}_1 \ge \cdots \ge \bar{f}_k \ge f^*$ for any $k \ge 1$;

c) Problem (1.44) *is feasible for all iterations of procedure;*

d) $\emptyset \neq \underline{X}_k \subseteq \bar{X}_k$ *for any* $k \geq 1$ *and hence Step 4 is feasible for all iterations of procedure.*

Proof. We show part a) by induction. For base case note that $X_\ell \subseteq X'_0 = X$. Now suppose that X'_{k-1} is a localizer of the level set X_ℓ. So if $x \in X_\ell$ then $x \in X'_{k-1}$. Also, by definition of h for any $x \in X_\ell$ we have $h(x^\ell_k, x) \leq f(x) \leq \ell$. Using these observations and the definition of $\underline{X}$ we conclude that $X_\ell \subseteq \underline{X}_k \subseteq X'_k$ which means that X'_k is a localizer.

To show part b), note that the first relation follows from Lemma 5, (1.42), (1.43) and the fact that $X'_k,\ k \geq 0$, are localizers of X_ℓ by part a). The second relation follows from the definition of $\bar{f}_k,\ k \geq 0$.

We show part c) by contradiction. Suppose that problem (1.44) is infeasible. Then by (1.43) we have $\underline{h}_k > \ell$ and so $\underline{f}_k \geq \ell$ which implies that $\mathcal{G}_{APL}$ sould have terminated in Step 1 at iteration k.

To show part d), note that by part c), the set $\underline{X}_k$ is nonempty. Also by optimality condition of (1.44) and the definition of $\underline{X}_k$ we have $\langle x_k, x - x_k\rangle \geq 0$ for any $x \in \underline{X}_k$ which implies that $\underline{X}_k \subseteq \bar{X}_k$. □

In Theorem 11 we present the conceptual bound on the number of iterations performed by $\mathcal{G}_{APL}$ in a typical phase of APL algorithm and note that parameters $\{\gamma_k\}$ are in conceptual and possible selections of them will be given in later propositions.

Theorem 11. *Let* $\gamma_k \in (0,1]$*, and* $x^\ell_k, x_k, x^u_k \in X$ *for* $k = 1, 2, \cdots$ *be given, where* ℓ *is the bound of the level set in* $\mathcal{G}_{APL}$*. Then*

$$f(x^u_k) - \ell \leq (1-\gamma_1) f(x^u_0 - \ell) + \frac{LD_X^2}{2} \max_{1 \leq i \leq k} \{\gamma_i^2/\Gamma_i\} \tag{A.37}$$

for $k \geq 1$*, where* Γ_k *is defined in* (2.17)*. In particular if* $\gamma_k \in (0,1],\ k = 1, 2, \cdots$ *are chosen so that for some* $c > 0$*,*

$$\gamma_1 = 1, \qquad \textit{and} \qquad \Gamma_k \max_{1 \leq i \leq k} \{\gamma_i^2/\Gamma_i\} \leq ck^{-2}, \tag{A.38}$$

then the number of iterations performed by $\mathcal{G}_{APL}$ can be bounded by

$$K_{APL}(\Delta_0) := \left\lceil \sqrt{\frac{cLD_X^2}{\beta\theta\Delta_0}} \right\rceil$$

where $\Delta_0 = \bar{f}_0 - \underline{f}_0$.

Proof. First, note that by definition of $\bar{X}$

$$\langle x - x_0, x_k - x \rangle \leq 0 \qquad \forall x \in X' \subseteq \bar{X}.$$

Therefore, since $x_{k+1} \in X'_k$ for $k \geq 0$ we have

$$\begin{aligned}\|x_{k+1} - x_k\|^2 &\leq \langle x_k - x_0, x_{k+1} - x_k \rangle \\ &= \|x_{k+1} - x_0\|^2 - \|x_k - x_0\|^2 - \langle x_{k+1} - x_k, x_{k+1} - x_0 \rangle \\ &\leq \|x_{k+1} - x_0\|^2 - \|x_k - x_0\|^2.\end{aligned}$$

Summing up the above inequalities we obtain

$$\sum_{i=1}^{k} \|x_i - x_{i-1}\|^2 \leq \|x_k - x_0\|^2 \tag{A.39}$$

Next, denoting $\tilde{x}_k^u = \gamma_k + (1-\gamma_k)x_{k-1}^u$, then by Lemma 6 for all $k \geq 1$ and definition of x_k^u and $\tilde{x}_k^u$ we have

$$f(x_k^u) \leq f(\tilde{x}_k^u) \leq (1-\gamma_k)f(x_{k-1}^u) + \gamma_k \ell + \frac{L}{2}\|\gamma_k(x_k - x_{k-1})\|^2.$$

Subtracting bothsides of above inequality we obtain

$$f(x_k^u) - \ell \leq (1-\gamma_k)[f(x_{k-1}^u) - \ell] + \frac{L}{2}\|\gamma_k(x_k - x_{k-1})\|^2 \qquad \forall k \geq 1.$$

Using Lemma 2 for above inequality we obtain

$$
\begin{aligned}
f(x_k^u) - \ell &\le (1-\gamma_1) f(x_0^u - \ell) + \Gamma_k \frac{L}{2} \sum_{i=1}^{k} \frac{\gamma_i^2}{\Gamma_i} \|x_i - x_{i-1}\|^2 \\
&\le (1-\gamma_1) f(x_0^u - \ell) + \frac{L}{2} \Gamma_k \max_{1\le i\le k} \{\gamma_i^2/\Gamma_i\} \sum_{i=1}^{k} \|x_i - x_{i-1}\|^2 \\
&\le (1-\gamma_1) f(x_0^u - \ell) + \frac{LD_X^2}{2} \Gamma_k \max_{1\le i\le k} \{\gamma_i^2/\Gamma_i\} \qquad \forall k \ge 1
\end{aligned}
$$

where the last inequality holds by (A.39).

Now, let $K \equiv K_{APL}(\epsilon)$ be the total number of iterations performed by $\mathcal{G}_{APL}$ and suppose that conditions (A.38) hold. Then by (A.37) and (A.38) we have

$$
f(x_K^u) - \ell \le \frac{cLD_X^2}{2K^2} \le \theta\beta\Delta_0 = \theta(\bar{f}_0 - \ell),
$$

where the last equality follows the fact that $\ell = \beta \underline{f}_0 + (1-\beta)\bar{f}_0 = \bar{f}_0 - \beta\Delta_0$. Therefore, procedure $\mathcal{G}_{APL}$ must terminate in step 3 of the K-th iteration. □

In Proposition 1 we discuss a few selections of $\{\gamma_k\}$, which satisfies conditions (A.38) and thus guarantee the termination of $\mathcal{G}_{APL}$. Note that this setting is independent of any problem parameters like L os D_X.

Proposition 1. *Let γ_k and Γ_k be defined in* (2.17)*. Then*

a) If $\gamma_k = 2/(k+1),\ k = 1, 2, \cdots$ then $\gamma_k \in (0, 1]$ and conditions (A.38) *hold with $c = 4$.*

b) If γ_k and $\Gamma_k,\ k = 1, 2, \cdots$ are recursively defined by

$$
\gamma_1 = \Gamma_1 = 1, \quad \Gamma_k = \gamma_k^2 = (1 - g_k)\Gamma_{k-1}, \quad \forall k \ge 2. \tag{A.40}
$$

Then we have $\gamma_k \in (0, 1]$ for any $k \ge 1$ and conditions (A.38) *hold with $c = 4$*

Proof. To show part a) note that if $\gamma_k = 2/(k+1)$ then $\Gamma_k = 2/k(k+1)$ and then knowing this we obtain

$$
\frac{\gamma_k^2}{\Gamma_k} = \left(\frac{2}{k+1}\right)^2 \frac{k(k+1)}{2} = \frac{2k}{k+1} \le 2.
$$

Using this and also the origin of Γ_k imply that $\Gamma_k \max_{1\le i\le k}\{\gamma_i^2/\Gamma_i\} \le 4k^{-2}$.

To show part b) first note that solving (A.40) for γ_k we obtain

$$\gamma_k = \frac{1}{2}\left(-\Gamma_{k-1} + \sqrt{\Gamma_{k-1}^2 + 4\Gamma_{k-1}}\right) \qquad k \ge 2, \tag{A.41}$$

which implies that $\gamma_k > 0,\ k \ge 2$. Now using induction we show that $\gamma_k \le 1$. So suppose that $\Gamma_{k-1} \le 1$. Then by (A.41) we have for any $k \ge 2$,

$$\gamma_k \le \frac{1}{2}\left(-\Gamma_{k-1} + \sqrt{\Gamma_{k-1}^2 + 4\Gamma_{k-1} + 4}\right) = 1.$$

Next, note that $\Gamma_k = (1-\gamma_k)\Gamma_{k-1} \le \Gamma_{k-1}$ meaning that Γ_k is decreasing in k. This also implies that for $k \ge 2$

$$\frac{1}{\sqrt{\Gamma_k}} - \frac{1}{\sqrt{\Gamma_{k-1}}} = \frac{\gamma_k/\Gamma_k}{\sqrt{1/\Gamma_k} + \sqrt{1/\Gamma_{k-1}}} \ge \frac{\gamma_k}{2\sqrt{\Gamma_k}} = \frac{1}{2}$$

which with the fact that $\Gamma_1 = 1$ we conclude

$$\Gamma_k \le \frac{4}{(k+1)^2} \le \frac{4}{k^2}.$$

Using this and definition Γ_k we obtain

$$\Gamma_k \max_{1\le i\le k}\{\gamma_i^2/\Gamma_i\} \le \frac{4}{k^2}.$$

□

Finally, in Theorem 12 we summarize the convergence properties of the APL method.

Theorem 12. *Let $\gamma_k \in (0,1],\ k = 1,2,\cdots$ in $\mathcal{G}_{APL}$ are chosen so that conditions* (A.38) *hold for some $c > 0$. Then*

a) The number of phases performed by APL method is bounded by

$$\bar{S}(\epsilon) = \left\lceil \max\left\{0, \log_{\frac{1}{q}} \frac{LD_X^2}{2\epsilon}\right\}\right\rceil$$

b) The total number of iterations performed by the APL method can be bounded by

$$\bar{S}(\epsilon) + \frac{1}{1-\sqrt{q}} \left(\frac{cLD_X^2}{2\beta\theta\epsilon} \right)^{\frac{1}{2}}$$

Proof. Let us denote $\delta_s \equiv ub_s - lb_s,\ s \geq 1$. Also, let us suppose that $\delta_1 > \epsilon$ because otherwise the statement is clearly true. First, note that if $\mathcal{G}_{APL}$ terminated in step 1 of its k-th iteration we must have $\underline{f}_k \geq \ell - \theta(\ell - \underline{f}_0)$. Since $f(p^+) \leq \bar{f}_0$ and by (1.41) we have

$$\begin{aligned} f(p^+) - lb^+ = f(p^+) - \underline{f}_k &\leq \bar{f}_0 - [\ell - \theta(\ell - \underline{f}_0)] \\ &= [1 - (1-\beta)(1-\theta)](\bar{f}_0 - \underline{f}_0). \end{aligned} \tag{A.42}$$

And if $\mathcal{G}_{APL}$ terminates in step 3 of its k-th iteration we must have $\bar{f}_k \leq \ell + \theta(\bar{f}_0 - \ell)$. Since $lb^+ \geq \underline{f}_0$ and also by definition of ℓ in (1.41) we have

$$f(p^+) - lb^+ = \bar{f}_k - lb^+ \leq \ell + \theta(\bar{f}_0 - \ell) - f_0 = [1 - (1-\theta)\beta](\bar{f}_0 - \underline{f}_0). \tag{A.43}$$

equations (A.42) and (A.43) together imply that at termination of $\mathcal{G}_{APL}$ we have

$$f(p^+) - lb^+ \leq q[f(p) - lb], \tag{A.44}$$

where

$$q \equiv q(\beta, \theta) := 1 - (1-\theta)\min\{\beta, 1-\beta\}. \tag{A.45}$$

Therefore, from (A.44) and (A.45) and origin of ub_s and lb_s we have

$$\delta_{s+1} \leq q\delta_s, \quad s \geq 1. \tag{A.46}$$

Also note that, from initial phase of APL and Lipschitz continuity of of f' we have

$$\delta_1 = f(p_1) - h(p_0, p_1) = f(p_1) - (f(p_0) + \langle f'(p_0), p_1 - p_0 \rangle) \leq \frac{L}{2}\|p_1 - p_0\|^2 \leq \frac{LD_X^2}{2}. \tag{A.47}$$

From (A.46) and (A.47) we obtain

$$\delta_{s+1} \leq q^{s-1}\delta_1 \leq q^s \frac{LD_X^2}{2}.$$

To obtain an ϵ-gap we must have $q^s LD_X^2 < 2\epsilon$ which implies the desired result in part a.

Next, we prove part b. Let $\bar{s}$ be the total number of calls for $\mathcal{G}_{APL}$ for some $1 \leq \bar{s} \leq \bar{S}(\epsilon)$. Therefore, from (A.46) and the fact that $\delta_{\bar{s}} > \epsilon$ we have $\delta_s > \epsilon q^{s-\bar{s}}, \ s = 1, \cdots, \bar{s}$. Using this observation we obtain

$$\sum_{s=1}^{\bar{s}} \delta_s^{-\frac{1}{2}} < \sum_{s=1}^{\bar{s}} \frac{q^{\frac{1}{2}(\bar{s}-s)}}{\sqrt{\epsilon}} = \sum_{t=0}^{\bar{s}-1} \frac{q^{\frac{t}{2}}}{\sqrt{\epsilon}} \leq \frac{1}{(1-q^{\frac{1}{2}})\sqrt{\epsilon}}.$$

Using this observation and by Theorem 11, the number of iterations performed by APL method is bounded by

$$\begin{aligned}\sum_{s=1}^{\bar{s}} K_{APL}(\delta_s) &\leq \bar{s} + \sum_{s=1}^{\bar{s}} \left(\frac{cLD_X^2}{2\beta\theta\delta_s}\right)^{\frac{1}{2}} \\ &\leq \bar{s} + \frac{1}{(1-q^{\frac{1}{2}})} \cdot \left(\frac{cLD_X^2}{2\beta\theta\epsilon}\right)^{\frac{1}{2}}.\end{aligned}$$

□

In view of Theorem 12 we can see that the APL method achieves the optimal complexity bound $\mathcal{O}(1/\sqrt{\epsilon})$ for convex smooth programming problems. Lan [26] shows that the APL method uniformly achieves the optimal complexity for solving smooth, weakly smooth and nonsmooth CP problems.

Bibliography

[1] Alexe, B., Deselaers, T., Ferrari, V.: Measuring the objectness of image windows. IEEE transactions on pattern analysis and machine intelligence **34**(11), 2189–2202 (2012)

[2] Babenko, B., Yang, M.H., Belongie, S.: Robust object tracking with online multiple instance learning. IEEE transactions on pattern analysis and machine intelligence **33**(8), 1619–1632 (2010)

[3] Bach, F., Harchaoui, Z.: Diffrac: a discriminative and flexible framework for clustering. Advances in Neural Information Processing Systems **20** (2007)

[4] Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural computation **15**(6), 1373–1396 (2003)

[5] Berclaz, J., Fleuret, F., Turetken, E., Fua, P.: Multiple object tracking using k-shortest paths optimization. IEEE transactions on pattern analysis and machine intelligence **33**(9), 1806–1819 (2011)

[6] Bonesky, T., Bredies, K., Lorenz, D.A., Maass, P.: A generalized conditional gradient method for nonlinear operator equations with sparsity constraints. Inverse Problems **23**(5), 2041 (2007)

[7] Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Transactions on pattern analysis and machine intelligence **23**(11), 1222–1239 (2001)

[8] Bredies, K., Lorenz, D.A.: Iterated hard shrinkage for minimization problems with sparsity constraints. SIAM Journal on Scientific Computing **30**(2), 657–683 (2008)

[9] Bubeck, S., et al.: Convex optimization: Algorithms and complexity. Foundations and Trends® in Machine Learning **8**(3-4), 231–357 (2015)

[10] Canon, M.D., Cullum, C.D.: A tight upper bound on the rate of convergence of frank-wolfe algorithm. SIAM Journal on Control **6**(4), 509–516 (1968)

[11] Delong, A., Gorelick, L., Veksler, O., Boykov, Y.: Minimizing energies with hierarchical costs. International journal of computer vision **100**(1), 38–58 (2012)

[12] Delong, A., Osokin, A., Isack, H.N., Boykov, Y.: Fast approximate energy minimization with label costs. International journal of computer vision **96**(1), 1–27 (2012)

[13] Dunn, J.C.: Rates of convergence for conditional gradient algorithms near singular and nonsingular extremals. SIAM Journal on Control and Optimization **17**(2), 187–211 (1979)

[14] Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International journal of computer vision **88**(2), 303–338 (2010)

[15] Frank, M., Wolfe, P.: An algorithm for quadratic programming. Naval research logistics quarterly **3**(1-2), 95–110 (1956)

[16] Guélat, J., Marcotte, P.: Some comments on wolfe's 'away step'. Mathematical Programming **35**(1), 110–119 (1986)

[17] Gurobi Optimization, L.: Gurobi optimizer reference manual (2020). URL `http://www.gurobi.com`

[18] Harchaoui, Z., Juditsky, A., Nemirovski, A.: Conditional gradient algorithms for norm-regularized smooth convex optimization. Mathematical Programming **152**(1-2), 75–112 (2015)

[19] Hare, S., Saffari, A., Torr, P., Struck, S.: Structured output tracking with kernels. In: IEEE International Conference on Computer Vision. IEEE, pp. 263–270

[20] Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition. Springer Series in Statistics. Springer (2009)

[21] Jaggi, M.: Revisiting frank-wolfe: Projection-free sparse convex optimization. In: ICML (1), pp. 427–435 (2013)

[22] Joulin, A., Bach, F., Ponce, J.: Discriminative clustering for image co-segmentation. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 1943–1950. IEEE (2010)

[23] Joulin, A., Tang, K., Fei-Fei, L.: Efficient image and video co-localization with frank-wolfe algorithm. In: European Conference on Computer Vision, pp. 253–268. Springer (2014)

[24] Lacoste-Julien, S., Jaggi, M.: On the global linear convergence of frank-wolfe optimization variants. Advances in neural information processing systems **28** (2015)

[25] Lan, G.: The complexity of large-scale convex programming under a linear optimization oracle. arXiv preprint arXiv:1309.5550 (2013)

[26] Lan, G.: Bundle-level type methods uniformly optimal for smooth and nonsmooth convex optimization. Mathematical Programming **149**(1), 1–45 (2015)

[27] Lan, G.: Gradient sliding for composite optimization. Mathematical Programming **159**(1-2), 201–235 (2016)

[28] Lan, G.: First-order and Stochastic Optimization Methods for Machine Learning. Springer (2020)

[29] Lan, G., Pokutta, S., Zhou, Y., Zink, D.: Conditional accelerated lazy stochastic gradient descent. In: International Conference on Machine Learning, pp. 1965–1974 (2017)

[30] Lan, G., Zhou, Y.: Conditional gradient sliding for convex optimization. SIAM Journal on Optimization **26**(2), 1379–1409 (2016)

[31] Levitin, E.S., Polyak, B.T.: Constrained minimization methods. USSR Computational mathematics and mathematical physics **6**(5), 1–50 (1966)

[32] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International journal of computer vision **60**(2), 91–110 (2004)

[33] Mitchell, B., Dem'yanov, V.F., Malozemov, V.: Finding the point of a polyhedron closest to the origin. SIAM Journal on Control **12**(1), 19–26 (1974)

[34] Nemirovski, A., Onn, S., Rothblum, U.G.: Accuracy certificates for computational problems with convex structure. Mathematics of Operations Research **35**(1), 52–78 (2010)

[35] Nesterov, Y.: Introductory lectures on convex optimization: A basic course, vol. 87. Springer Science & Business Media (2013)

[36] Nesterov, Y.: Complexity bounds for primal-dual methods minimizing the model of objective function. Mathematical Programming **171**(1-2), 311–330 (2018)

[37] Nesterov, Y.E.: A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. Doklady AN SSSR **269**, 543–547 (1983). Translated as Soviet Math. Docl.

[38] Nesterov, Y.E.: Introductory Lectures on Convex Optimization: A Basic Course. Kluwer Academic Publishers, Massachusetts (2004)

[39] Nesterov, Y.E.: Universal gradient methods for convex optimization problems. Mathematical Programming **152**(1-2), 381–404 (2015)

[40] Pang, Y., Ling, H.: Finding the best from the second bests-inhibiting subjective bias in evaluation of visual tracking algorithms. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2784–2791 (2013)

[41] Perazzi, F., Krähenbühl, P., Pritch, Y., Hornung, A.: Saliency filters: Contrast based filtering for salient region detection. In: 2012 IEEE conference on computer vision and pattern recognition, pp. 733–740. IEEE (2012)

[42] Pérez, P., Hue, C., Vermaak, J., Gangnet, M.: Color-based probabilistic tracking. In: European Conference on Computer Vision, pp. 661–675. Springer (2002)

[43] Prest, A., Leistner, C., Civera, J., Schmid, C., Ferrari, V.: Learning object class detectors from weakly annotated video. In: 2012 IEEE Conference on computer vision and pattern recognition, pp. 3282–3289. IEEE (2012)

[44] Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on pattern analysis and machine intelligence **22**(8), 888–905 (2000)

[45] Tang, K., Joulin, A., Li, L.J., Fei-Fei, L.: Co-localization in real-world images. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1464–1471 (2014)

[46] Tang, K., Ramanathan, V., Fei-Fei, L., Koller, D.: Shifting weights: Adapting object detectors from image to video. Advances in Neural Information Processing Systems **25** (2012)

[47] Wibisono, A.: Accelerated gradient descent. url=http://awibisono.github.io/2016/06/20/accelerated-gradient-descent.html (2016)

[48] Wolfe, P.: Convergence theory in nonlinear programming. Integer and Nonlinear Programming pp. 1–36 (1970)

[49] Xu, L., Neufeld, J., Larson, B., Schuurmans, D.: Maximum margin clustering. Advances in neural information processing systems **17** (2004)

[50] Yilmaz, A., Javed, O., Shah, M.: Object tracking: A survey. Acm computing surveys (CSUR) **38**(4), 13–es (2006)