Enhanced first-order methods in convex and nonconvex optimization

by

Xi Bai

Presented to the Graduate and Research Committee

of Lehigh University

in Candidacy for the Degree of

Doctor of Philosophy

in

Industrial Engineering

Lehigh University

May 2015

Approved and recommended for acceptance as a dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Date

Dissertation Advisor

Committee Members:

Katya Scheinberg, Ph.D., Committee Chair

Frank E. Curtis, Ph.D.

Reha Tütüncü, Ph.D.

Luis F. Zuluaga, Ph.D.

Acknowledgments

I am fortunate enough to have my Ph.D. life shared with so many great people.

First of all, I would like to thank my advisor, Katya Scheinberg, for her endless support. Her advice has always been invaluable, but perhaps what is more important is the freedom and an enjoying environment she creates for her students. Katya has the ability to make students happy and, from that perspective, I could not have hoped for anything more from an advisor. It is fair to say that, without Katya's priceless guidance and consistent encouragement, I would never have finished my dissertation.

I am grateful to my dissertation committee members, Frank E. Curtis, Reha Tütüncü and Luis F. Zuluaga, for their guidance in my research and insightful comments to my dissertation. I thank Frank for his absolutely high-quality courses which provide me with the knowledge in nonlinear optimization. I thank Reha for introducing me to the exciting area of financial optimization and his guidance in my career. I am grateful to Luis for many helpful discussions and comments on financial optimization, risk measures and global optimization techniques.

I am also grateful to many people around me who have played an utterly important part of my life. I would like to thank Tengjiao Xiao and all my friends at Lehigh. Their caring and support have made my Ph.D. life at Lehigh an unforgettable experience. I want to thank the 2015 Ph.D. class at ISE of Lehigh, for many heated discussions in both courses and research. Finally, I am indebted to my dad, my mom, and my cousins for their unconditional love throughout my life.

Contents

List of Figures

List of Tables

Abstract

First-order methods for convex and nonconvex optimization have been an important research topic in the past few years. In many applications in compressed sensing and machine learning, higher-order methods could be computationally prohibitive due to the large-scale input data which is often dense. As a contrast, the low per-iteration complexity of first-order methods has made them a wise alternative.

This dissertation studies and develops efficient algorithms of first-order type, to solve a variety of problems. Chapter 2 and 3 focus on the widely used gradient-based methods in composite convex optimization problems. Problems of the composite form arise extensively in compressed sensing, machine learning, etc. In particular, Chapter 2 presents some preliminaries on first-order methods. As is shown, techniques and concepts such as alternating minimization and proximal update are grounded in other more complex first-order algorithms, and also serve as foundations to many of our optimization methods in later chapters.

Chapter 3, from a practical perspective, studies an accelerated first-order scheme for composite convex problems. It is shown in Chapter 2 that, while enjoying the "optimal" convergence rate for the gradient methods, the complexity of the accelerated first-order methods introduced in Chapter 2 relies on the worst case value of the prox parameter. Our focus in Chapter 3 thus is: how the accelerated first-order and alternating linearization methods can be improved by allowing for the complexity

estimates that depend on the "average" prox parameter value. The computational results show the benefit of the new algorithm.

Chapter 4 discusses the risk parity portfolio selection problem, which aims to find such portfolios for which the contributions of risk from all assets are equally weighted. In this chapter, we discuss the problem of finding portfolios that satisfy risk parity over either individual assets or groups of assets. We describe the set of all risk parity solutions by using convex optimization techniques over orthants and we show that this set may contain an exponential number of solutions. We then apply the nonconvex least-squares technique as an alternative approach, which can be solved efficiently by the alternating direction techniques described in Chapter 5. We also propose a modified formulation which aims at selecting the most desirable risk parity solution according to a given criterion.

In Chapter 5, we present alternating direction type of methods solving structured nonlinear nonconvex problems. The problem we are interested in has special structure which allows convenient 2-block variable splitting (one example is the nonconvex risk parity model introduced in Chapter 4). Our methods rely on solving convex subproblem and the limit point obtained can be guaranteed to satisfy KKT optimality conditions. Our approach includes the alternating directions method of multipliers and the alternating linearization method and we provide convergence rate results for both classes of methods. Moreover, global optimization techniques from polynomial optimization literature are applied to complement our local methods and to provide lower bounds.

Chapter 6 deals with another application, for which methods described in Chapter 5 are applicable. The application is called the optimal power flow problem (OPF). OPF arises as one of the most important optimization problems in the power system. Since the difficulty of OPF lies in the quadratic equality where the rank-one constraint is maintained, we relax the constraint through augmented Lagrangian and propose alternating direction methods to minimize the resulted second-order least-squares. Computational results from preliminary numerical experiment show the expected efficiency of our approach.

Chapter 1

Introduction

In mid-1980s, Karmarkar published a seminal paper which brought on a new era for convex optimization. Since then, polynomial time interior-point methods (IPMs) have been a main research focus for convex optimization since it could easily reach a high accuracy solution within tens of iterations. However, for polynomial time methods it is usually hard to cope with the size of the data: the computational cost per iteration for interior-point methods grows superlinearly with the number of decision variables. This makes it impractical to perform one iteration, particularly in many machine learning and compressed sensing problems that arose recently, when the constraint matrices are often dense. For instance, in a typical machine learning problem with more than 10,000 variables, even storing the Hessian could be impractical.

In contrast, gradient-based methods which only incorporate first-order information have recently attracted intensive research interests, due to its cheap computational cost per iteration. Moreover, an increasing demand for large-scale optimization problems, which do not have to be solved accurately, makes it even more promising to apply first-order methods which have long been criticized for their poor convergence rate.

Instead of general linear and nonlinear optimization algorithms, this dissertation focuses on developing efficient optimization methods for structured convex and nonconvex problems. These problems arise in a number of areas, including image processing and compressed sensing, machine learning, portfolio selection, and energy systems. In this dissertation, the strategies such as variable splitting, alternating minimization, proximal update, etc. are heavily used together with the traditional gradient schemes, to take the most advantage of the problem structure. As is shown in this dissertation, gradient based methods sometimes can be quite powerful when solving these nontrivial problems when the problem structure is fully concerned.

The remainder of the dissertation is organized as follows. In Chapter 2, we introduce the basic first-order optimization techniques, which serve as preliminary knowledge of the methods used in later chapters. In Chapter 3, we develop practical accelerated first-order methods for composite convex optimization problems. In Chapter 4, we discuss risk parity optimization as an application which drives us to developing efficient algorithms for structured nonconvex optimization. In Chapter 5, alternating direction type of methods solving structured nonconvex problems are presented. In particular, we discuss finding a stationary point for second-order least-squares problems, with applications in portfolio selection (Chapter 4) and in optimal power flow (Chapter 6). We conclude the thesis in Chapter 7.

Chapter 2

Preliminaries on first-order methods for composite convex problems

2.1 Introduction

In this section, we are interested in composite convex programming problem of the following form:

$$\min F(x) = f(x) + g(x), \tag{2.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}$ are both convex functions and only g is possibly nonsmooth. In this section, we first survey a variety of important problems involving this form, and then review some theoretical results related to applying fast first-order method to them.

Problem of form (2.1) also arises extensively in machine learning. Many statistical learning problems involve minimizing a sum of a loss function together with a regularization term. Some interesting examples are as follows.

Example 2.1.1. *The idea of compressed sensing is to recover a sparse signal from a linear system. Consider the following problem*

$$\min \quad \frac{1}{2}\|Ax-b\|_2^2 + \rho\|x\|_1 \ , \tag{2.2}$$

where A is an $m \times n$ matrix, b is the observed measurement vector of size m. Then (2.2) *can be regarded as the noisy sparse recovery problem with the l_0-norm objective replaced by the l_1-norm relaxation.*

Example 2.1.2. *As an alternative to problem (2.2), one may wish to solve the following problem in Huber setting, and obtain the following minimization problem:*

$$\min_x F(x) = H_\nu(\|Ax-b\|_2) + \rho\|x\|_1, \tag{2.3}$$

where $\rho > 0$ and

$$H_\nu(y) = \begin{cases} \dfrac{y^2}{2\nu}, & 0 \le |y| \le \nu \\[2em] |y| - \dfrac{\nu}{2}, & |y| \ge \nu \end{cases}$$

for $\nu > 0$. We can define $f(x) := H_\nu(\|Ax-b\|_2)$ and $g(x) := \rho\|x\|_1$.

Example 2.1.3. *Sparse logistic regression maximizes the log-likelihood of a sparse vector to map real vectors $x_i \in \mathbb{R}^m$ into binary y_i, and is formulated in the following form*

$$\min_w \frac{1}{n}\sum_{i=1}^{n} \log(1+\exp(-y_i \cdot w^\top x_i)) + \rho\|w\|_1,$$

where $\{(x_i, y_i)\}_{i=1}^n \in (\mathbb{R}^m \times \{-1, 1\})$ is the training set and $\rho > 0$. Note that now we have $f(w) := \frac{1}{n}\sum_{i=1}^{n} \log(1+\exp(-y_i \cdot w^\top x_i))$ and $g(w) := \rho\|w\|_1$.

Example 2.1.4. *Matrix rank minimization problem can be regarded as a matrix extension of Basis-Pursuit DeNoising (BPDN). To avoid the combinatorial nature of the matrix rank, such problem can be approximated by a convex one using nuclear norm minimization.*

$$\begin{aligned} \min \quad & \|X\|_* \\ s.t. \quad & \|A(X) - b\|_2 \leq \theta, \end{aligned}$$

where $X \in \mathbb{R}^{m\times n}$, $A : \mathbb{R}^{m\times n} \to \mathbb{R}^p$ is a linear operator, and $b \in \mathbb{R}^p$. Note that, similar as the l_1 minimization in compressed sensing, the above problem can be reformulated as an unconstrained problem with the form (2.1).

Algorithms for solving an objective of form (2.1) have been studied extensively. In this chapter, we describe some of the existing optimization methods solving (2.1). Note that, despite aiming to solve a composite convex objective, many methods discussed in this chapter inspire our approaches in later chapters when structured nonconvex problems are under consideration.

2.2 Simple gradient descent schemes

First-order methods have been applied extensively in a number of areas such as signal processing and machine learning. The advantage of first-order methods is their low computational cost per iteration, as compared with higher order methods such as interior-point methods. In particular, there have been a number of recent applications where data is large-scale, when applying second-order methods or even storing the Hessian could be computationally prohibitive.

Consider problem of the form (2.1) but $g \equiv 0$. Then problem (2.1) becomes

$$\min_x f(x). \tag{2.4}$$

Hence, we can choose a starting point x^0 and apply gradient step to each iteration

$$x^{k+1} := x^k - \mu \nabla f(x^k), \ k \geq 0, \tag{2.5}$$

for some $\mu > 0$. The direction is well known as "steepest descent", and there are several ways to choose the stepsize μ. For instance, one can apply exact line search to choose μ such that $f(x^k - \mu \nabla f(x^k))$ is minimized, but it is rarely used in practice due to extra computational cost involved in most cases. A cheaper alternative is to apply backtracking line search (i.e. to try $\mu^0, \frac{\mu^0}{2}, \frac{\mu^0}{4}, ...$) until some conditions are satisfied (for instance, a sufficient function value decrease). Given some prior knowledge regarding the function (for instance, Lipschitz constant of the gradient), one can also use fixed steplength of a sufficiently small size. In later sections, we will see that, for a class of gradient based algorithms, step sizes (or prox parameter value which could be regarded as variants of step sizes) could play an important role in both theoretical complexity analysis and practical implementations.

Another interpretation of (2.5) is to consider a quadratic approximation of f at a proximal point y:

$$Q_\mu(x, y) := f(y) + \langle \nabla f(y), x - y \rangle + \frac{1}{2\mu} \|x - y\|_2^2. \tag{2.6}$$

Then step (2.5) simply becomes solving

$$x^{k+1} := \arg\min_x Q(x, x^k), \ k \geq 0. \tag{2.7}$$

Now let us consider the case when $g \neq 0$. Then similarly we have the following quadratic approximation

$$Q_\mu(x, y) := f(y) + \langle \nabla f(y), x - y \rangle + \frac{1}{2\mu}\|x - y\|_2^2 + g(x). \tag{2.8}$$

Note that only f is linearized and combined with a proximal penalty term, simply because only f is guaranteed to be smooth. In this case, μ is often called the "prox parameter". Consider the following update

$$x^{k+1} := \arg\min_x Q_\mu(x, x^k),\ \ k \geq 0. \tag{2.9}$$

This mapping from x^k to x^{k+1} is denoted as

$$x^{k+1} := p_\mu(x^k),\ \ k \geq 0, \tag{2.10}$$

and is sometimes referred to as "proximal operator mapping". In many applications, function g is "simple" and efficient for the proximal operator computation. Particularly, for l_1 regularized problem, $F(x) \equiv f(x) + \rho\|x\|_1$. Then the update becomes

$$x^{k+1} := T_{\rho\mu}(x^k - \mu\nabla f(x^k)), \tag{2.11}$$

where the thresholding operator is defined as

$$T_\alpha(x)_i = (|x_i| - \alpha)_+ \text{sign}(x_i). \tag{2.12}$$

The thresholding update described above can be viewed as a variation of a gradient step, and results in the Iterative Shrinkage/Thresholding Algorithm (ISTA) [7]. If we have prior knowledge of a Lipschitz constant of ∇f (denoted as $L(f)$), then μ

can be computed effectively and we can formalize ISTA as follows.

```
Algorithm 1 ISTA
1. Choose $\mu := 1/L(f)$.
2. for $k = 0, 1, ...,$
     solve $x^{k+1} := \arg\min_x Q_\mu(x, x^k)$.
```

In many applications, to compute the Lipschitz constant $L(f)$ exactly is often not easy. To overcome this potential difficulty, we can make an estimation at the first iteration and then employ a simple backtracking strategy [7].

```
Algorithm 2 ISTA-BKTR
1. Set $0 < \beta < 1$ and $\mu_0 > 0$.
2. for $k = 0, 1, ...,$
     (1) Choose $0 < \bar{\mu}_k < \mu_0$.
     (2) Find the smallest $i_k \geq 0$ such that $\mu_k = \beta^{i_k}\bar{\mu}_k$ and
           $F(p_{\mu_k}(x^k)) \leq Q_{\mu_k}(p_{\mu_k}(x^k), x^k)$,
       where $p_{\mu_k}(x^k) \equiv \arg\min_x Q_{\mu_k}(x, x^k)$.
     (3) $x^{k+1} := p_{\mu_k}(x^k)$.
```

2.3 Convergence and accelerated techniques

Throughout the section, we call $x_\epsilon \in \mathbb{R}^n$ an ϵ-optimal solution to (2.1) if $F(x_\epsilon) - F(x^*) \leq \epsilon$, where x^* is an optimal solution to (2.1) and $\epsilon > 0$. Our interest in this subsection lies on iteration complexity bounds of first-order methods (i.e. worst-case iteration number for first-order methods to compute an ϵ-optimal solution). Complexity bounds for first-order methods solving general convex optimization problems have been studied in [51, 52]. It has been shown that the best iteration complexity, when only first-order information is used, is $O(\sqrt{\frac{L}{\epsilon}})$, where L is a Lipschitz constant of the gradient of the objective function.

The convergence of ISTA has been well studied (see, for instance, [7, 27]), and it can be shown that ISTA converges at a sublinear rate. The following theorem and its proof can be found in [7].

Theorem 2.3.1. *Let $\{x^k\}$ be the sequence generated by Algorithm 1. Then for every $k \geq 1$, we have*

$$F(x^k) - F(x^*) \leq \frac{L(f)\|x^0 - x^*\|^2}{2\beta k},$$

where x^* is an optimal solution, $L(f)$ is a Lipschitz constant of ∇f, and β is the backtracking scaler.

As shown in the above theorem, the worst-case iteration complexity bound of ISTA is known to be $O(\frac{L(f)}{\epsilon})$, where $L(f)$ is a Lipschitz constant for $\nabla f(x)$. Beck and Teboulle improve the complexity bound from $O(\frac{L(f)}{\epsilon})$ to $O(\sqrt{\frac{L(f)}{\epsilon}})$, by proposing a version of Nesterov's method for composite convex functions which they call FISTA (Fast Iterative Shrinkage/Thresholding Algorithm) [7], as shown below.

Algorithm 3 FISTA

```
1. Set $t_1 = 1, 0 < \beta < 1$ and $y^1 = x^0, \mu_0 > 0$;
2. for $k = 0, 1, ...,$
    (1) Choose $0 < \bar{\mu}_k < \mu_{k-1}$;
    (2) Find the smallest $i_k \geq 0$ such that $\mu_k = \beta^{i_k}\bar{\mu}_k$ and
            $F(p_{\mu_k}(y^k)) \leq Q_{\mu_k}(p_{\mu_k}(y^k), y^k).$
        where $p_{\mu_k}(x^k) \equiv \arg\min_x Q_{\mu_k}(x, x^k)$.
    (3) $x^k := p_{\mu_k}(y^k) \equiv \arg\min_x Q_{\mu_k}(x, y^k)$.
        $t_{k+1} := \left(1 + \sqrt{1 + 4t_k^2}\right)/2$.
        $y^{k+1} := x^k + \frac{t_k - 1}{t_{k+1}}\left[x^k - x^{k-1}\right]$.
```

The main difference between ISTA and FISTA is that the latter employs the information of the previous two points by taking a linear combination. Compared with ISTA, FISTA also requires only one gradient computation per iteration and thus does not have significantly more computational effort at each iteration. Further, the following convergence results have been proved in [7]. Hence, FISTA is an "optimal" first-order method since $O(\sqrt{\frac{L(f)}{\epsilon}})$ is the best complexity bound that one can achieve by using only gradient information [51, 52].

Theorem 2.3.2. *Let $\{x^k\}$ be the sequence generated by Algorithm 3. Then for every $k \geq 1$, we have*

$$F(x^k) - F(x^*) \leq \frac{2L(f)\|x^0 - x^*\|_2^2}{\beta(k+1)^2},$$

where x^* is an optimal solution, $L(f)$ is a Lipschitz constant of ∇f, and β is the backtracking scaler.

2.4 Alternating direction methods for composite convex optimization problem

Another popular class of first-order methods solving (2.1) is the class of alternating direction methods and alternating linearization methods, which often combines the idea of variable splitting with the augmented Lagrangian method.

Augmented Lagrangian approach belongs to the class of the methods of multipliers. Unlike the classic method of multipliers, augmented Lagrangian method reduces the possibility of ill conditioning by adding a quadratic penalty term. Thus, while making relatively few assumptions, this method is often robust in practice. Moreover, this method is efficient when the subproblem is not computationally expensive, which makes it very easy to implement on problems with simple constraints.

Suppose we have a general optimization problem:

$$\begin{aligned} \min_{x \in \mathcal{X}} \quad & F(x) \\ \text{s.t.} \quad & h(x) = 0, \end{aligned} \tag{2.13}$$

where $x \in \mathcal{X}$ means that x satisfies some simple constraints other than the equality ones.

The Lagrangian function with multipliers can be written as

$$\mathcal{L}(x; \lambda) = F(x) - \sum_i \lambda_i h_i(x), \tag{2.14}$$

where λ is the multiplier vector.

With μ as a positive scalar, we can define the augmented Lagrangian function as

$$\mathcal{L}_A(x; \lambda) = F(x) - \sum_i \lambda_i h_i(x) + \frac{1}{2\mu} \sum_i h_i^2(x), \tag{2.15}$$

It can be seen that the augmented Lagrangian function differs from the standard Lagrangian function by the squared terms.

The augmented Lagrangian method can be useful if minimizing (2.15) is easier than solving (2.13). At the k-th iteration, we can choose a proper penalty parameter μ and solve the subproblem

$$x^{k+1} := \arg\min_x \mathcal{L}_A(x; \lambda^k)$$

and then update the multiplier by

$$\lambda^{k+1} := \lambda^k - \frac{1}{\mu} h(x^{k+1}).$$

We summarize our discussion as the following algorithmic framework. In later chapters, we will heavily rely on the classic augmented Lagrangian method to develop more complicated approaches for problems with some decomposable structure.

Algorithm 4 Augmented Lagrangian method (AL)

```
1. Choose $\mu^0 > 0$, $\lambda^0$, and $x^0 = y^0$.
2. For $k = 0, 1, ...$
      Solve the subproblem (approximately) $x^{k+1} := \arg\min_x \mathcal{L}_A(x; \lambda^k)$.
      Update the multiplier $\lambda^{k+1} := \lambda^k - \frac{1}{\mu^k} h(x^{k+1})$.
      Choose new penalty parameter $\mu^{k+1} \in (0, \mu^k)$.
```

Now we are ready to discuss the convex composite optimization problem. Based on the augmented Lagrangian framework, a natural way to deal with composite functions is to split the variables. Consider the following problem which is equivalent to (2.1):

$$\begin{aligned} \min \quad & f(x) + g(y) \\ \text{s.t.} \quad & x - y = 0, \end{aligned} \tag{2.16}$$

where $y \in \mathbb{R}^n$ is a new variable introduced. Hence, the subproblem in Algorithm 3.3 becomes

$$(x^k, y^k) := \arg\min_{x,y} f(x) + g(y) - \lambda^\top (x - y) + \frac{1}{2\mu}\|x - y\|^2. \tag{2.17}$$

Minimizing (2.17) over x and y jointly is often equivalent to solving the original problem. However, the composite structure of the original problem provides an opportunity for alternating minimization which is often much easier than (2.17), which leads to so-called alternating direction methods of multipliers (ADMM). The difference of ADMM, compared with classic AL, is that the multiplier is updated at each loop after minimizing over each direction, instead of after minimizing overall variables jointly. A simple framework of ADMM solving (2.16) can be implemented as follows.

Algorithm 5 Alternating direction methods of multipliers (ADMM)

```
1. Choose $\mu^0$, $\lambda^0$, and $x^0 = y^0$.
2. For $k = 0, 1, ...$
      $x^{k+1} := \arg\min_x \mathcal{L}_A(x, y^k; \lambda^k)$.
      $y^{k+1} := \arg\min_y \mathcal{L}_A(x^{k+1}, y; \lambda^k)$.
      Update the multiplier$\lambda^{k+1} := \lambda^k - \frac{1}{\mu^k}(x^{k+1} - y^{k+1})$.
      Choose new penalty parameter $\mu^{k+1} \in (0, \mu^k)$.
```

ADMM can be viewed as one of the alternating direction methods (ADMs). The idea of ADM can be tracked back to the Douglas-Rachford method in the 1950s [26] and alternating methods solving variational problems associated with PDEs in the 1970s [30, 33]. In [39, 67, 68], ADMs are applied to solve variational inequality problems. Recently, with the popularity of compressed sensing and l_1 regularized statistical learning, ADM and its variants have been extensively applied to problems arising in these areas. In [36, 75], ADMM is considered in the context of l_1 regularized problems, in which case it is also known as the Split Bregman method. More recently, ADMs have also been applied to some classes of well-structured semidefinite programming problems (SDPs) [71].

Notice that in Algorithm 5, the Lagrangian multiplier λ is updated only after the minimization with respect to y is finished at each iteration. It has been proposed in different applications that a natural extension of ADMM is to deal with x and y symmetrically [34, 56]. Given below is the so-called S-ADMM method. S-ADMM is also called Peaceman-Rachford method in some literature.

Algorithm 6 Symmetric Alternating Direction Augmented Lagrangian method (S-ADMM)

```
1. Choose $\mu^0$, $\lambda^0$, and $x^0 = y^0$;
2. For $k = 0, 1, ...$
      $x^{k+1} := \arg\min_x \mathcal{L}_A(x, y^k; \lambda^k)$.
      $\lambda^{k+\frac{1}{2}} := \lambda^k - \frac{1}{\mu^k}(x^{k+1} - y^k)$.
      $y^{k+1} := \arg\min_y \mathcal{L}_A(x^{k+1}, y; \lambda^k)$.
      $\lambda^{k+1} := \lambda^k - \frac{1}{\mu^k}(x^{k+1} - y^{k+1})$.
      Choose new penalty parameter $\mu^{k+1} \in (0, \mu^k)$.
```

It is natural to ask what is the relationship between S-ADMM and ADMM. In fact both methods can be viewed as variants of the famous forward-backward splitting scheme. We refer interested readers to a detailed explanation in [22, 33].

In problems with the form (2.1), if both f and g are smooth, a natural extension of ISTA is to treat f and g "equally" and to alternatingly linearize the two functions. Such alternating linearization can be viewed as a variant of S-ADMM. When both f and g are differentiable and λ^0 is chosen to be $-\nabla g(y^0)$, it can be derived from the first-order optimality conditions of the subproblems that the following relationship holds at the kth iteration

$$\lambda^{k+\frac{1}{2}} = \nabla f(x^{k+1}) \text{ and } \lambda^{k+1} = -\nabla g(y^{k+1}).$$

Further, define the following linearization functions:

$$Q^1_{\mu_g}(x, y) := f(x) + g(y) + \langle \nabla g(y), x - y\rangle + \frac{1}{2\mu_g}\|x - y\|^2$$
$$Q^2_{\mu_f}(x, y) := f(x) + g(y) + \langle \nabla f(x), y - x\rangle + \frac{1}{2\mu_f}\|x - y\|^2.$$

where μ_f, $\mu_g > 0$. When g is nonsmooth, we can use a subgradient in the subdifferential $\partial g(y)$ to replace "$\nabla g(y)$" in $Q^1_{\mu_g}(x, y)$. Hence, we obtain the algorithm below.

Algorithm 7 Alternating Linearization method (ALM)

```
1. Choose $\mu_f$, $\mu_g > 0$, and $x^0 = y^0$.
2. for $k = 0, 1, ...$
       $x^{k+1} := \arg\min_x Q^1_{\mu_g}(x, y^k)$.
       $y^{k+1} := \arg\min_y Q^2_{\mu_f}(x^{k+1}, y)$.
```

Similar as ISTA type of algorithms, the complexity of alternating directions methods can also be improved by applying variants of Nesterov's acceleration techniques. In particular, the Fast Alternating Linearization Method (FALM) is an accelerated version of ALM for solving (2.1), when $f(x)$ and $g(x)$ are both differentiable, and is given below as Algorithm 8.

The iteration complexity of FALM is similar to FISTA: it requires $O(\sqrt{\frac{L}{\epsilon}})$ iterations to obtain an ϵ-optimal solution, where $L = \frac{L(f)L(g)}{L(f) + L(g)}$ in the case when $f(x)$ and $g(x)$ are both differentiable [34].

Algorithm 8 Fast Alternating Linearization Method (FALM)

```
1. Choose $\mu_f > 0$ and $\mu_g > 0$ and $x^0 = y^0 = z^0$, set $t_0 = 1$.
2. for$k = 0, 1, ...$
       $x^{k+1} := \arg\min_x Q^1_{\mu_g}(x, z^k)$
       $y^{k+1} := \arg\min_y Q^2_{\mu_f}(y, x^k)$
       $t_{k+1} := (1 + \sqrt{1 + 4t_k^2})/2$
       $z^{k+1} := y^{k+1} + \frac{t_k - 1}{t_{k+1}}(y^{k+1} - y^k)$
```

Similarly, other alternating direction schemes including ADMM can also be accelerated, as recently shown in [35]. Since this chapter is for preliminaries, we do not discuss these variants here. We refer interested readers to [35] for details.

Chapter 3

Practical accelerated first-order algorithms for composite convex problems

3.1 Introduction

The primary goal of this chapter is to introduce some variants of the algorithms introduced in Chapter 2 and also to provide some empirical evidence. As discussed in Chapter 2, FISTA computes an ϵ-optimal solution in $O(\sqrt{L(f)/\epsilon})$ steps, where $L(f)$ is a bound on the Lipschitz constant for $\nabla f(x)$. Hence, it is an "optimal gradient" method since this is the best complexity bound that one can obtain using only first-order information [51] [52]. However, as discussed later, maintaining a nonincreasing sequence of proximal parameters (i.e. the inverse of the estimate of the Lipschitz constant) can substantially limit the performance of FISTA when a large Lipschitz constant estimate is encountered early in the algorithm since this causes the sizes of the steps taken at that point, and in all subsequent, to be very small. Similar situations hold in FALM described in Algorithm 8.

The convergence of FISTA relies on the following lemma from [7]. In fact, this result provides a sufficient function decrease guarantee for a family of proximal-gradient type algorithms. Moreover, in later chapters, we will extend this result to the more general nonconvex case.

Lemma 3.1.1. *For any $y, x \in \mathbb{R}^n$ and $\mu > 0$, if*

$$F(p_\mu(y)) \leq Q_\mu(p_\mu(y), y), \tag{3.1}$$

where

$$Q_\mu(u, v) := f(v) + \langle u - v, \nabla f(v)\rangle + \frac{1}{2\mu}\|u - v\|^2 + g(u). \tag{3.2}$$

$$p_\mu(v) := \arg\min_u Q_\mu(u, v). \tag{3.3}$$

Then

$$2\mu(F(x) - F(p_\mu(y))) \geq \|p_\mu(y) - y\|^2 + 2\langle y - x, p_\mu(y) - y\rangle = \|p_\mu(y) - x\|^2 - \|y - x\|^2. \tag{3.4}$$

Proof. Proof can be found in [7]. □

In this chapter, we introduce an extension of FISTA and FALM (named, FISTA-BKTR and FALM-BKTR, respectively) which allows an increase of the proximal parameter (i.e. μ in Algorithm 3). The actual value of μ at each iteration can be determined via a back-tracking line search so that condition (3.1) is satisfied. A lower bound on μ can be derived from the facts that μ is reduced by a constant factor at each line search step and as soon as $\mu \leq 1/L(f)$, condition (3.1) is satisfied and the line search terminates. Further, we discuss the concept of local composite Lipschitz constant. It can be shown that, instead of relying on the "worst case" $L(f)$, the complexity of the new algorithm only depends on the average of the "local composite" Lipschitz constants. In many applications, such average is significantly less than the global Lipschitz bound.

To understand the intuition of the extended algorithm, we first introduce the idea of local composite Lipschitz constant.

We first consider any two vectors $u, v \in \mathbb{R}^n$ and let $[u, v]$ denote the set of points on a segment between u and v, in other words, $[u, v] = \{x : x = \lambda u + (1 - \lambda)v, 0 \leq \lambda \leq 1\}$. Let $L_{[u,v]}(f)$ be the Lipschitz constant of $\nabla f(x)$ restricted to $[u, v]$; i.e.,

$$\|\nabla f(x) - \nabla f(y)\| \leq L_{[u,v]}(f)\|x - y\|, \forall x, y \in [u, v].$$

From simple calculus it follows that

$$f(u) \leq f(v) + \nabla f(v)^\top (u - v) + \frac{L_{[u,v]}(f)}{2}\|u - v\|^2. \tag{3.5}$$

Note that the roles of u and v are interchangeable.

Definition 3.1.2. *$L(f, g, w)$ is a local composite Lipschitz constant for $\nabla f(v)$ at w if*

$$\|\nabla f(v_1) - \nabla f(v_2)\| \leq L(f, g, w)\|v_1 - v_2\|, \ \forall v_1, v_2 \in [p_\mu(w), w], \forall \mu \leq 1/L(f, g, w),$$

where $[p_\mu(w), w] = \{v : v = \lambda p_\mu(w) + (1 - \lambda)w, 0 < \lambda < 1\}$.

The dependence of $L(f, g, y)$ on g arises from the dependence of $p_\mu(y)$ on g, hence we use the term "composite" to emphasize this dependence. If $g(x) \equiv 0$ then

$p_\mu(y) = y - \mu\nabla f(y)$ for any μ and, hence, $L(f,g,y)$ is a Lipschitz constant of $\nabla f(x)$ restricted to an interval $[y, y - \frac{1}{L(f,g,y)}\nabla f(x)]$.

It has been shown in [60] with two key observations:

- $L(f,g,y) \le L(f)$ for all y and
- from Definition 3.1.2 it follows that for any μ and y, such that $\mu \le 1/L(f,g,y)$ (3.1) holds (by (3.5)), and hence (3.4) holds for the given y and any x.

Let us now illustrate why $1/L(f,g,y)$ may be a better estimate for the prox parameter μ than $1/L(f)$.

Example 3.1.3. *Consider a compressed sensing or Lasso setting:*

$$(P) \qquad \min\{F(x) \equiv \|Ax-b\|^2 + \rho\|x\|_1 : x \in \mathbb{R}^n\}. \tag{3.6}$$

In this case $f(x) = \|Ax-b\|^2$ *and* $L(f) = \|AA^\top\|_2$. *Now consider a sparse vector* $\bar{y}$, *without loss of generality assume that* $\bar{y} = (\bar{y}_1, \bar{y}_2)$ *with* $\bar{y}_2 = 0$. *Also consider the gradient vector* $z = A^\top(A\bar{y} - b)$ *and assume that* $\|z_2\|_\infty \le \rho$ *(*z_2 *is the subvector of* z *that corresponds to the subvector* $\bar{y}_2$*). In this case it is easy to see from the properties of the shrinkage operator that* $p_\mu(\bar{y})_2 = 0$ *for all* $\mu > 0$. *This implies that for any* $x \in [p_\mu(\bar{y}), \bar{y}]$ $x_2 = 0$ *and hence* $L(f,g,\bar{y}) = \|A_1A_1^\top\|_2$, *which is clearly smaller than* $L(f) = \|AA^\top\|_2$, *where* A_1 *is the subset of columns of* A *that correspond to the subvector* y_1.

Since it may be difficult, in general, to compute the local composite Lipschitz constant accurately we may consider estimating it via an upper bound which is still lower than $L(f)$. For instance, assume that for a given f, g and y,

$$\|\nabla f(x) - \nabla f(y)\| \leq L(f,g,y)\|x-y\|, \forall x: \|x-y\| \leq \|p_{1/L(f,g,y)}(y) - y\|, \quad (3.7)$$

in other words, $L(f,g,y)$ is a Lipschitz constant of $\nabla f(x)$ restricted on the ball around y of radius $\|p_{1/L(f,g,y)}(y) - y\|$. Then $L(f,g,y)$ is a local composite Lipschitz constant for these f, g and y, and hence for any $\mu < 1/L(f,g,y)$ it can be shown that (3.1) holds [60].

In the next section, we use the concept of local Lipschitz constant to motivate the extension of existing fast first-order methods.

3.2 Fast first-order methods with backtracking

FISTA with backtracking (i.e. FISTA-BKTR) can be regarded as an extension of FISTA but allows for variable stepsize μ_k. Let us recall the update of proximal point and denote the following computation by $(t_{k+1}, y^{k+1}) = FistaStep(x^k, x^{k-1}, t_k, \theta_k)$:

$$\begin{aligned} t_{k+1} &:= \left(1 + \sqrt{1 + 4\theta_k t_k^2}\right)/2, \\ y^{k+1} &:= x^k + \frac{t_k - 1}{t_{k+1}}\left[x^k - x^{k-1}\right]. \end{aligned} \quad (3.8)$$

In FISTA, we have $\theta_k \equiv 1$, and $\mu_{k+1} \leq \mu_k$ is satisfied for all k. Suppose we want to allow to increase μ_k by a scaler $\beta \in (0,1)$, i.e. $\mu_{k+1} \geq \mu_k/\beta$. Then it has been shown in [60] that we need to choose θ smartly in order to maintain the optimal rate of convergence as in FISTA. To achieve it, we choose $\theta_k = \mu_k/\mu_{k+1}^0$, where μ_{k+1}^0 is an estimate of μ_{k+1}. Hence, we have the following two cases. If μ is increased at the end of the kth iteration, then we set $\mu_{k+1}^0 := \mu_k/\beta$. Otherwise, we simply choose $\mu_{k+1}^0 := \mu_k$. After an initial guess μ_{k+1}^0 is made, we linearize the function f to see whether it is a good guess or not. If not, we reduce μ_{k+1} by backtracking $\mu_{k+1} := \beta\mu_{k+1}$ and recompute $\theta_k := \theta_k/\beta$, until a good estimate of μ_{k+1} is found. We now introduce FISTA-BKTR as Algorithm 9, and later in this section we will explain why such update achieves our goal.

Algorithm 9 FISTA-BKTR

```
0. Set $t_1 = 1, 0 < \beta < 1, \theta_0 = 1$ and $y^1 = x^0 = x^{-1}, \mu_1^0 > 0$.
1. for $k = 1, 2, ...$
    (1) Set $\mu_k := \mu_k^0$.
    (2) Compute $\nabla f(y^k), p_{\mu^k}(x^k)$.
        If $F(p_{\mu_k}(y^k)) > Q_{\mu_k}(p_{\mu_k}(y^k), y^k)$,
        $\mu_k := \beta\mu_k, \theta_{k-1} := \theta_{k-1}/\beta$.
        $(t_k, y^k) := FistaStep(x^{k-1}, x^{k-2}, t_{k-1}, \theta_{k-1})$.
        Return to (2).
    (3) $x^k := p_{\mu^k}(y^k)$.
        Choose $\mu_{k+1}^0 > 0$, and set $\theta_k := \mu_k/\mu_{k+1}^0$.
        $(t_{k+1}, y^{k+1}) := FistaStep(x^k, x^{k-1}, t_k, \theta_k)$.
```

Theorem 3.2.1. *Let L_k be the estimate of the "average" of local composite Lipschitz constants encountered during the first k iterations of Algorithm 9, such that $\frac{1}{\sqrt{L_k}} = \frac{1}{k}\sum_{i=1}^{k}\frac{1}{\sqrt{L(f,g,y^i)}}$. Assume that $\mu_i^0 \geq 1/L(f,g,y^i)$ for all $1 \leq i \leq k$. Then*

$$F(x^k) - F(x^*) = v_k \leq \frac{2L_k\|x^0 - x^*\|^2}{\beta k^2}, \tag{3.9}$$

where x^* is an optimal solution and β is the backtracking scaler.

Proof. Proof can be found in [60]. □

Although in this chapter we do not show a detailed proof of Theorem 3.2.1, we here aim to provide some discussion on how the complexity is achieved and pinpoint how Algorithm 9 generalizes the original FISTA.

In Algorithm 3, if μ_0 and $\bar{\mu}_k$ are chosen so that $\mu_0 > 1/L(f)$ and $\mu_k \geq \mu_{k-1}$, then $\mu_k \geq \beta/L(f)$, $\forall k$, since as already noted, $F(p_{\mu_k}(y^k)) \leq Q_{\mu_k}(p_{\mu_k}(y_k), y^k)$ holds for any y_k if $\mu_k < 1/L(f)$. The following lemma is an immediate consequence of this.

Lemma 3.2.2. *Let $\mu_0 > 1/L(f)$, then at each iteration of Algorithm 3 there are at most $\log_{\frac{1}{\beta}}(\mu_0 L(f)) + 1$ line search backtracking steps, and hence computations of $p_{\mu_k}(y_k)$.*

If for a given y_k we define a local composite local Lipschitz constant L_k ($= L(f, g, y_k)$), as discussed in Section 3.1, then $\mu_k \geq \beta/L_k$, and the number of line search steps at iteration k is at most $\log_{\frac{1}{\beta}}(\mu_0 L_k) + 1$.

The complexity of Algorithm 3 for $\theta_k \equiv 1$ is analysed in [7]. The proof in [7] relies on the fact that μ_{k+1} is chosen so that $\mu_{k+1} \leq \mu_k$ for all $k \geq 1$, i.e., the step size μ_k is monotonically nonincreasing. However, it can be shown that (see [60]) this is not a necessary condition as long as we choose θ carefully. Instead, in order to recover the $O(1/k^2)$ complexity, it is sufficient to show that our algorithm satisfies

$$2\mu_k t_k^2 \geq \eta k^2, \tag{3.10}$$

for some fixed $\eta > 0$, while maintaining

$$\mu_k t_k^2 \geq \mu_{k+1} t_{k+1}(t_{k+1} - 1). \tag{3.11}$$

Condition 3.11 provides an opportunity to increase μ_{k+1} from μ_k. This is achieved by selecting a proper θ_k.

From the update of t_k in Algorithm 3

$$t_{k+1} = (1 + \sqrt{1 + 4\theta_k t_k^2})/2 \tag{3.12}$$

it follows that

$$\theta_k t_k^2 = t_{k+1}(t_{k+1} - 1). \tag{3.13}$$

Hence as long as $\theta_k \leq \mu_k/\mu_{k+1}$, property (3.11) holds. We know that if we implement the constraint $\mu_{k+1} \leq \mu_k$ and use $\theta_k = 1$ as is done in the FISTA algorithm in [7] both conditions (3.10) and (3.11) are satisfied with $\eta = \beta/L(f)$. However, if we want to allow $\mu_{k+1} > \mu_k$, say $\mu_{k+1} \geq \mu_k/\beta$, then we need to have $\theta_k < 1$, e.g, $\theta_k \leq \beta$. On the other hand, condition (3.10) requires $t_k \geq k\sqrt{\frac{\eta}{2\mu_k}} \geq k\sqrt{\frac{\eta}{2\mu_0}}$, hence t_k needs to grow at the same rate as k. From (3.13) and the fact that $t_k > 1$ for all $k > 1$,

$$(t_{k+1} - 1)^2 < t_{k+1}(t_{k+1} - 1) = \theta_k t_k^2,$$

and hence, $t_{k+1} < \theta_k^{1/2} t_k + 1$. Assume that for all k, we allow $\theta_k \leq \beta < 1$ then if $t_k > 1/(1 - \beta^{1/2})$, we have $t_{k+1} < t_k$. This means that the sequence t_k will not grow at the required rate if we simply allow $\theta_k < 1$ for all iterations.

To maintain FISTA's rate of convergence while allowing $\theta_k < 1$ on some iterations we need to ensure that $\theta_k > 1$ on some of the other iterations. This can be accomplished on iterations on which $\mu_k/\mu_{k+1} > 1$. The immediate difficulty is that we do not know the value of μ_{k+1} on iteration k, when θ_k and t_{k+1} are computed. Simply setting $\theta_k \leq \mu_k/\mu_0$, where μ_0 is an upper bound on the step size will imply that $\theta_k < 1$ for all k. Hence it is necessary to update θ_k and t_{k+1} along with μ_{k+1}, thus expanding the backtracking part of the algorithm,

Hence, Algorithm 9 is an extension of Algorithm 3 which allows for a full backtracking and any size μ_k. It satisfies conditions (3.10)-(3.11), while setting the step size μ_k initially to some value μ_k^0 at the beginning of each iteration. Moreover, it can be shown that Algorithm 9 is designed to maintain $\theta_k = \mu_k/\mu_{k+1}$ for all $k \geq 1$.

Recall that the parameter θ_k is used to compute t_{k+1} and y^{k+1} in Algorithm 3 using the updates

$$\begin{aligned} t_{k+1} &:= (1 + \sqrt{1 + 4\theta_k t_k^2})/2 \\ y^{k+1} &:= x^k + \frac{t_k - 1}{t_{k+1}}[x^k - x^{k-1}]. \end{aligned}$$

Let us denote this computation by

$$(t_{k+1}, y^{k+1}) = FistaStep(x^k, x^{k-1}, t_k, \theta_k).$$

At the end of iteration $k-1$ of Algorithm 9, $\theta_{k-1} = \mu_{k-1}/\mu_k^0$, where μ_k^0 is an initial "guess" for the value of μ_k. Hence, t_k and y^k are computed using this "guess". Once the backtracking is called at iteration k this "guess" may turn out to be correct or it may turn out to be an overestimate of μ_k. If the "guess" is correct, then no correction to θ_{k-1} is needed. If μ_k is reduced during the backtracking, then θ_{k-1} is recomputed to account for the new value of μ_k so that $\theta_{k-1} = \mu_{k-1}/\mu_k$ is satisfied. Another call to $FistaStep(x^{k-1}, x^{k-2}, t_{k-1}, \theta_{k-1})$ is then necessary. After such a call is made, since the iterate y^k has changed, $p_{\mu_k}(y^k)$ has to be recomputed and a new backtracking step has to be performed. The backtracking starts with the value of μ_k with which the previous backtracking stopped, and hence, which was used to compute the most recent value of θ_{k-1}. If the value of μ_k is not reduced any further then θ_{k-1}, y^k and t_k have the correct value and the backtracking part of the iteration is complete. If the value of μ_k is reduced further then θ_{k-1} is recomputed again and backtracking continues.

Hence, we apply backtracking to design Algorithm 9 such that $\theta_k = \mu_k/\mu_{k+1}$ is maintained for all $k \geq 1$. It follows immediately from the update rule that (3.11) is satisfied for each k.

Now if the other part, i.e. (3.10), holds then the complexity result follows. It turns out that this can be achieved by induction from the update rule (see, Lemma 3.4 in [60]). We can simply set η in (3.10) to be $\beta/(2L_k)$, where L_k satisfies $\frac{1}{\sqrt{L_k}} = (\sum_{i=1}^{k} \frac{1}{\sqrt{L(f,g,y^i)}})/k$. Then Theorem 3.2.1 trivially follows.

Remark 3.2.3. *If we use the worst case Lipschitz constant $L(f)$ as the local one $L(f,g,y^i)$, the analysis above still holds and we recover the complexity result:*

$$F(x^k) - F(x^*) = v_k \leq \frac{2L(f)\|x^0 - x^*\|^2}{\beta k^2}, \tag{3.14}$$

at the k-th iteration of Algorithm 9.

Hence, our algorithm and analysis are similar to those in [7], but allow for increases in the step size in Algorithm 3 while preserving the algorithm's $O(\sqrt{L(f)/\epsilon})$ complexity. As we have shown, this can be accomplished by choosing appropriate values of θ_k.

3.3 A practical backtracking FISTA algorithm for compressed sensing and Lasso problems

The additional cost of each backtracking step of Algorithm 9 compared to that of Algorithm 3 lies in a call to $FistaStep$ updates and re-computation of $\nabla f(y^k)$ which is needed to construct $Q_\mu(y^k, x)$. All remaining computational cost is the same for both algorithms. The number of backtracking steps is solely defined by the choice of μ_0^k at each iteration, as discussed in the previous section. The choice of a practical approach is likely to depend on the comparisons of the cost of computation of $\nabla f(y^k)$, $p_{\mu_k}(y^k)$ and $F(p_{\mu_k}(y^k))$. Here we consider specific application of our backtracking algorithm to the problem of the form,

$$(P) \qquad \min\{F(x) \equiv \|Ax - b\|^2 + g(x) : x \in \mathbb{R}^n\}. \tag{3.15}$$

We assume here that $g(x)$ is a simple function, such as $\|x\|_1$, as in the case of CS or Lasso [15] [66] or $\sum\limits_i \|x_i\|_2$ as in the case of group Lasso [77]. In this case $\nabla f(x) = A^\top(Ax - b)$. Recall expression for y^k:

$$y^k = x^{k-1} + \frac{t_{k-1} - 1}{t_k}[x^{k-1} - x^{k-2}]$$

which implies that

$$\nabla f(y^k) = \nabla f(x^{k-1}) + \frac{t_{k-1} - 1}{t_k}[\nabla f(x^{k-1}) - \nabla f(x^{k-2})].$$

Hence, if $\nabla f(x^{k-1})$ and $\nabla f(x^{k-2})$ are available, then $\nabla f(y^k)$ can be computed in $O(n)$ operations for any value of t_k. Since the backtracking step changes only the value of t_k but not x^{k-1} or x^{k-2}, this means that the extra cost of each backtracking step of Algorithm 9 compared to that of Algorithms 3 is only $O(n)$, which is negligible.

However, as discussed earlier, using larger values of μ_0^k may result in a higher number of backtracking steps, hence we should analyze the cost of a backtracking step itself. For simple forms of $g(x)$ computing $p_{\mu_k}(y^k)$ given $\nabla f(y^k)$ takes $O(n)$ operations. Finally computing $F(p_{\mu_k}(y^k))$ requires one matrix-vector product for computing $Ax - b$. Once x^k is determined via backtracking an additional matrix-vector product is employed to compute $\nabla f(x^k)$, however, this last computation is not a part of the backtracking procedure. Assuming that matrix-vector product comprise the dominant cost of each iteration, then the total cost of an iteration without backtracking equals two matrix vector products, while the cost of an iteration with backtracking contains additional matrix-vector product per each backtracking step. For instance, if $\mu_k^0 = \mu_{k-1}/\beta$, then it can be shown that the average cost of an iteration of Algorithm 9 is 3/2 that of Algorithms 3. Such cost increase is beneficial if the number of iterations Algorithm 9 is proportionately smaller.

In the examples we consider in our computational experiments in Section 3.5 increasing μ_k at each iterations appears to be wasteful. Hence we choose to allow for the increase of the value of the prox parameter every l iteration, where l is chosen heuristically. In particular we try several different values of l during the first 100 iterations and then settle with the value of l which gives least number of failed backtracking steps. The key assumption for using such a heuristics is that the behavior of the algorithm at later iterations is similar to that during earlier iterations. While the behavior of the algorithm is problem-dependent, in our experiments this heuristic produced good results. We believe that this is due to the fact that the local Lipschitz constants do not vary dramatically for the problems in our experiments. In case of significant changes in the Lipschitz constants we believe any backtracking heuristic will produce significant improvement over pure FISTA algorithm as it will allow the prox parameter to increase sooner or later.

3.4 A practical backtracking FALM algorithm for compressed sensing and Lasso problems

The idea of FALM with backtracking is similar to the one in FISTA. The main difference between FISTA and FALM is that FISTA approximates only the first part of the composite objective function $F(x)$, while FALM applies alternating approximations to each of the two parts of $F(x)$. Hence, we consider the following two approximations of $F(x)$ and their minimizers:

Definition 3.4.1.

$$Q^f_\mu(u,v) := f(v) + \langle u - v, \nabla f(v) \rangle + \frac{1}{2\mu}\|u - v\|^2 + g(u). \tag{3.16}$$

$$p^f_\mu(v) := \arg\min_u Q^f_\mu(u,v). \tag{3.17}$$

$$Q^g_\mu(u,v) := f(v) + \langle v - u, \lambda \rangle + \frac{1}{2\mu}\|u - v\|^2 + g(u), \tag{3.18}$$

where λ is some element of the subdifferential $\partial g(u)$.

$$p^g_\mu(u) := \arg\min_v Q^g_\mu(u,v). \tag{3.19}$$

In this section, we discuss a practical FALM with backtracking for compressed sensing and Lasso problems, which is shown in Algorithm 10. Throughout the discussion, we also frequently apply a concept named *skipping step.* Regarding alternating linearization methods (and their accelerated variants) with skipping steps, interested readers can refer to [34, 60].

We now discuss the additional cost of each backtracking step of Algorithm 10 compared to that of Algorithms 8 and a general efficient implementation of the algorithm targeted at the problems of the form (3.15).

We again assume here that $g(x)$ is a simple function, hence computing $p^g_{\mu^x}(y)$ is a relatively cheap operation. Computing $p^f_{\mu^y}(x)$, however, involves solving a system of linear equations with the matrix $A^\top A + \frac{1}{2\mu^y} I$. In some compressed sensing problems A has a special structure, such that this system can be solved in $O(n \log n)$ operations - the same work as is required to multiply A or $A^\top$ by a vector, and hence to compute the gradient $\nabla f(x) = A^\top(Ax - b)$ [34]. In cases when such special structure is not present, the work which involves factorization of $A^\top A + \frac{1}{2\mu^y} I$ may be the dominant cost of the iteration, as it generally requires $O(m^3)$ operations.

If μ^y is fixed beforehand, then the matrix $A^\top A + \frac{1}{2\mu^y} I$ can be factorized once, at the initialization stage of the algorithm, and hence the per iteration cost only involves additional matrix vector products. If μ^y is updated in an arbitrary way on each iteration, then the factorization has to be repeated each time. Recall, that ideally we want μ^y to have the largest possible value which satisfies the line search conditions in Step 2 of Algorithm 10 and that keeping μ^y constant may result in very slow progress. Hence, again, there exists a tradeoff between choosing a suitable value of prox parameter and the per iteration cost. For practical efficiency we strive to achieve a reasonable balance. Assume that we choose some value $\bar{\mu}^y_1$ at the beginning of the algorithm and we choose $\bar{\mu}^y_{k+1} = \beta^i \mu^y_k$ for some $i \in Z$ at each iteration k. Note that in this case, for all k, μ^y_k can only take values $\beta^j \mu^y_1$ for $j \in Z$. Let us consider $\beta = 0.5$. We also impose the following restriction of μ^y_k: if $\mu^y_k < \mu^y_1/1000$, then the expected improvement achieved by Step 2 is too small and the step is automatically skipped; if $\mu^y_k > 1000\mu^y_1$, the prox parameter the step size is large enough and no additional increase of μ^y_k is necessary. Hence the only values allowed for μ^y_k throughout the algorithm are $\{2^{-10}\mu_1^y, 2^{-9}\mu_1^y, ..., 0, 2\mu_1^y, 4\mu_1^y, ..., 2^{10}\mu_1^y\}$, overall 21 possible values. As soon as one of these values occurs in the algorithm the corresponding matrix factorization can be computed and stored for future use.

If the skipping of Step 2 occurs for a few consecutive iterations we may choose to automatically skip this step in the further iterations and thus avoid the additional cost of computing $p^f_{\mu^y}(x)$. In this case the FALM-BKTR algorithm reduces FISTA-BKTR. We found it beneficial to attempt Step 2 from time to time even after it has been skipped consistently on prior iterations.

The management of μ_k^x parameter and the additional per iteration cost of step 3 can be executed similarly to what is described in Section 3.3 for the FISTA-BKTR implementation.

Algorithm 10 Fast Alternating Linearization Method (FALM-BKTR)

```
1. Choose $x^0 = y^0 = y^{-1} = z^1$, $\mu_0 = \bar{\mu}_1^x = \bar{\mu}_1^y > 0, 0 < \beta < 1$, set $t_1 = 1$ and $t_0 = 0$;
2. for $k = 1, 2, ...$ do
  (1) Set $\mu_k^x := \bar{\mu}_k^x, \mu_x^y := \bar{\mu}_k^y, skip := true$
  (2) Compute $p^g_{\mu_k^x}(z^k)$
      If $F(p^g_{\mu_k^x}(z^k)) \leq Q^g_{\mu_k^x}(z^k, p^g_{\mu_k^x}(z^k))$ then $x^k := p^g_{\mu_k^x}(z^k), skip := false$
        else $\mu_k^x = 0$,$\theta_{k-1} := 2\mu_{k-1}/\mu_k^y, (t_k, z^k) := FistaStep(y^{k-1}, y^{k-2}, t_{k-1}, \theta_{k-1})$, $x^k := z^k$
  (3) Compute $p^f_{\mu_k^y}(x^k)$
      If $F(p^f_{\mu_k^y}(x^k)) \leq Q^f_{\mu_k^y}(p^f_{\mu_k^f}(x^k), x^k)$ then $y^k := p^y_{\mu_k^y}(x^k)$
          else $\mu_k^y := \beta\mu_k^y$, $\mu_k := \dfrac{\mu_k^x + \mu_k^y}{2}$, $\theta_k := \mu_{k-1}/\mu_k$, $(t_k, z^k) := FistaStep(y^{k-1}, y^{k-2}, t_{k-1}, \theta_{k-1})$
          If $skip = false$, return to (2)
          else $x_k := z_k$, return to (3)
  (4) Choose $\bar{\mu}_{k+1}^y, \bar{\mu}_{k+1}^x$
      $\theta_k := \dfrac{2\mu_k}{\bar{\mu}_{k+1}^y + \bar{\mu}_{k+1}^x}$,
      $(t_{k+1}, z^{k+1}) := FistaStep(y^k, y^{k-1}, t_k, \theta_k)$
```

3.5 Computational results

We now present numerical results for several sparse optimization problems of the standard compressed sensing or Lasso form:

$$\bar{x} := \arg\min_{x} \quad \frac{1}{2}\|Ax - b\|_2^2 + \rho\|x\|_1, \tag{3.20}$$

with $f(x) := \frac{1}{2}\|Ax - b\|_2^2$ and $g(x) := \rho\|x\|_1$.

We compare the following algorithms.

- FISTA: the original FISTA [7], as described in Algorithm 3 with $\theta_k = 1$.
- FISTA-BKTR: an efficient implementation of Algorithm 9 as discussed in Section 2.3 and 3.3.
- FALM: an implementation of Algorithm 8 as discussed in Section 2.4.
- FALM-BKTR: an efficient implementation of Algorithm 10 as discussed in Section 2.4 and 3.3.
- SpaRSA: a gradient based algorithm, with the use of shrinking, described in [28].
- Yall1: a solver based on alternating directions methods described in [78].

We compared the performance of the algorithms benchmarking them against FISTA. In particular, we ran FISTA for j iterations and recorded the best objective function value $FISTA(j)$ achieved by FISTA thus far. Then for all other algorithms, we recorded the number of the iterations it took to reach a function value which is smaller than $FISTA(j)$. We report the number of iterations as well as the number of matrix-vector multiplications. Throughout our tests, the maximum number of iterations is set to be 100000, and the tolerance is set to be 10^{-3} which means that the algorithm terminates when the objective function value is within 10^{-3} from the optimal (precomputed). We report the final objective function value when each algorithm terminates.

The main goal of our experiments is to demonstrate that our full backtracking strategy provides not only theoretical but practical advantage when applied to FISTA and FALM methods. The comparison to Yall1 and SpaRSA methods is only presented here to gauge the difficulty of the problems in our experiments and to demonstrate that behavior of our methods is reasonable. Our implementations were written in MATLAB and run in MATLAB R2010b on a laptop with Intel Core Duo 1.8 GHz CPU and 2GB RAM. We used the default setting for both Yall1 and SpaRSA, which likely accounts for the bad performance of these algorithms on some of the problems.

3.5.1 The Spear Examples

This set of instances are obtained from the Sparse Exact and Approximate Recovery Project and can be downloaded from either of the following links:

- http://imo.rz.tu-bs.de/mo/spear/
- https://coral.ie.lehigh.edu/projects/SPAROPTLIB/wiki/SparseOptimizationLibrary.

Spear10 (1024×512) Dynamic range is 3.02e+4. Sparsity is 18 (i.e. 18 nonzero elements in the true solution). This problem provides a relatively easy instance when $\rho = 1$ but the difficulty increases substantially as ρ decreases.

Table 3.1: Comparison of the algorithms for solving (3.20) with $\rho = 1$ on Spear10. $FISTA(100) = 5.3839e+5$, $FISTA(500) = 1.2799e+5$, $FISTA(1000) = 1.0035e+5$. The starting μ for FISTA/FISTA-BKTR, μ_f for FALM/FALM-BKTR and μ_g for FALM-BKTR are all set to be 1. For FALM (with skipping), we tried different values for μ_g. The starting μ_g's for FALM-S1, FALM-S2, FALM-S3 are 1, 10 and 100, respectively. Moreover, for FALM/FALM-BKTR the number in parentheses is the number of matrix factorization required overall.

solver	iter	mult	iter	mult	iter	mult	final iter	mult	final Obj.
FISTA	100	206	500	1006	1000	2006	1065	2133	9.997e+4
FISTA_BKTR	69	170	283	619	627	1343	647	1404	9.997e+4
FALM-S1	30	94 (2)	117	355 (2)	376	1037 (11)	586	1457 (11)	9.997e+4
FALM-S2	10	34 (2)	39	121 (2)	206	565 (11)	273	699 (11)	9.997e+4
FALM-S3	4	16 (2)	17	57 (4)	355	812 (11)	464	1030 (11)	9.997e+4
FALM_BKTR	8	28 (8)	26	112 (10)	85	396 (21)	136	546 (21)	9.997e+4
SpaRSA	98	196	1487	2974	1689	3378	1729	3458	9.997e+4
YALL1	18	55	30	91	89	268	93	280	9.997e+4

From Table 3.1, we see that the algorithms with backtracking (FISTA-BKTR and FALM-BKTR) were generally faster than their basic counterparts (FISTA, FALM) in terms of the number of matrix-vector multiplications. For example, it only takes 627 iterations and 1343 matrix-vector multiplications for FISTA-BKTR to reduce the objective function value below $FISTA(1000) = 1.0035e+5$. Comparing FALM-S1 and FALM-BKTR, we observe that the latter is faster for the same initial choice of μ_g ($\mu_g = 1$). Initial performance of FALM-S2 and, especially, FALM-S3 is good due to larger starting values for μ_g, however this performance slows down compared to FALM-BKTR as iterations progress. This indicates that the full backtracking strategy can help accelerate the original algorithms at any stage.

In Figure 3.1 we plot how μ changes during iterations taken by FISTA and FISTA-BKTR when solving (3.20) with $\rho = 1$ on Spear10. We see that, FISTA-BKTR can achieve larger values of μ by allowing backtracking, and thus performs large steps on some of the iterations, which corresponds with the smaller number of iterations required by FISTA-BKTR.

Table 3.2 shows the results on Spear10 problem with $\rho = 0.01$. This problem

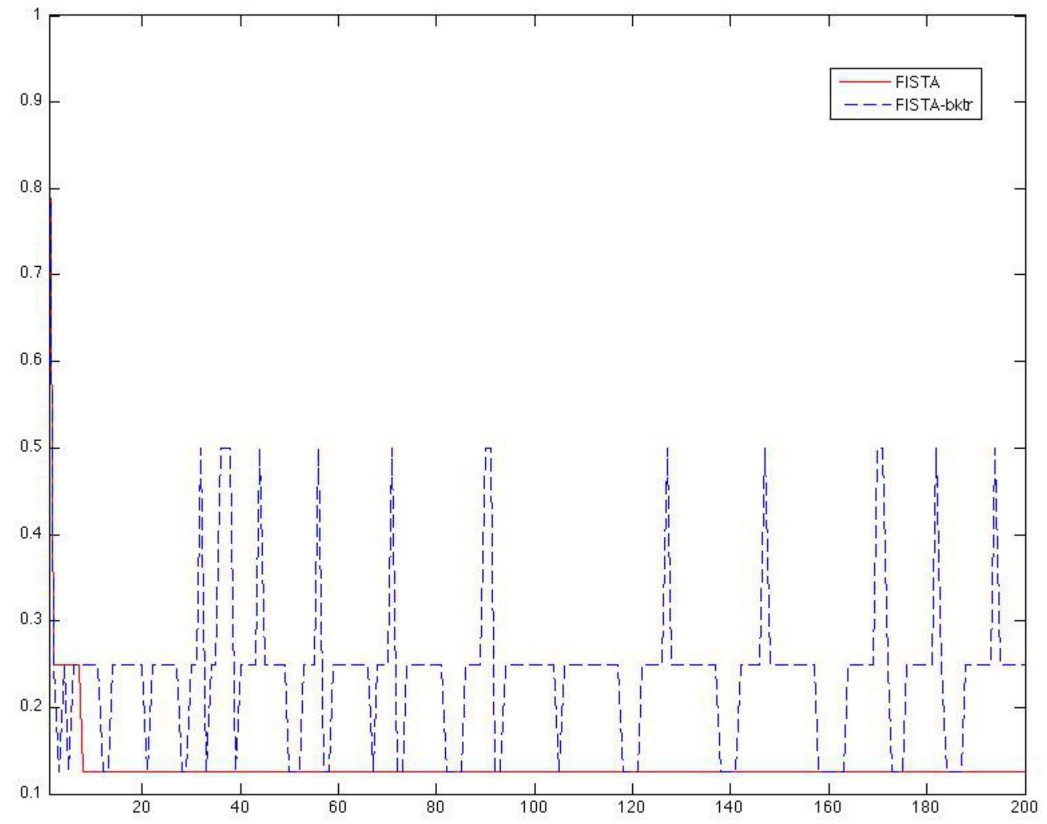


Figure 3.1: Comparison on the μ values while solving (3.20) with $\rho = 1$ on Spear10

Table 3.2: Comparison of the algorithms for solving (3.20) with $\rho = 0.01$ on Spear10. The starting μ is set to be 0.01. $FISTA(100) = 6.0980e + 3$, $FISTA(500) = 5.8943e + 3$, $FISTA(1000) = 5.4176e + 3$.

solver	iter	mult	iter	mult	iter	mult	final iter	mult	final Obj.
FISTA	100	206	500	1006	1000	2006	17846	35695	999.4
FISTA_BKTR	79	190	372	804	746	1607	12547	26527	999.4
FALM-S1	24	76 (2)	156	472 (2)	313	943 (2)	5298	15903 (11)	999.4
FALM-S2	7	25 (2)	53	163 (2)	108	328 (2)	1221	3674 (11)	999.4
FALM-S3	4	16 (2)	16	52 (2)	33	103 (2)	287	869 (11)	999.4
FALM_BKTR	7	25 (7)	12	40 (11)	18	64 (11)	252	1078 (21)	999.4
SpaRSA	30	60	687	1374	5024	10048	100000	200000	2151.1 (Failed)
YALL1	65	196	65	196	66	199	95	286	1015.3

provides a difficult instance where our backtracking methods appears to provide clear advantage. SpaRSA did not converge to the proximity of the solution, while Yall1 only achieved accuracy of 10^{-1}, but not 10^{-3}. Here we observe that FISTA-BKTR retains its advantage, while FALM-BKTR seems to slow down compared to FALM method when it gets closer to the solutions. The reasons for this behavior will be investigated in the future.

Spear3 (1024×512) with $\rho = 0.1$. Dynamic range is 2.7535e+4. Sparsity is 6. We observe that behavior of FALM-BKTR converges to that of FISTA-BKTR in later iterations due to persistent skipping of Step 2.

Table 3.3: Comparison of the algorithms for solving (3.20) with $\rho = 0.1$ on Spear3. The starting μ is set to be 1. $FISTA(100) = 1.1825e{+}4$, $FISTA(500) = 1.1793e{+}4$, $FISTA(1000) = 1.1784e + 4$.

solver	iter	mult	iter	mult	iter	mult	final iter	mult	final Obj.
FISTA	100	211	500	1011	1000	2011	28517	57045	7.33e+3
FISTA_BKTR	236	535	241	547	324	722	3149	6767	7.33e+3
FALM-S1	8	28 (2)	166	378 (11)	559	1164 (11)	27287	54620 (11)	7.33e+3
FALM-S2	6	34 (2)	112	266 (11)	547	1136 (11)	28063	56168 (11)	7.33e+3
FALM-S3	105	245 (11)	506	1047 (11)	826	1687 (11)	22430	44895 (11)	7.33e+3
FALM_BKTR	6	34 (6)	145	485 (17)	276	827 (17)	3819	10271 (17)	7.33e+3
SpaRSA	5	10	264	528	1215	2430	100000	200000	1.02e+4 (Failed)
YALL1	418	1255	418	1255	418	1255	809	2428	7.33e+3

3.5.2 Bdata problems

Bdata test set was originally created by A. Nemirovski with the aim to imitate examples with worst-case complexity for the first-order methods. This problem, however, provides a relatively easy instance probably due to presence of the ℓ_1 term in the objective. Here we present results for Bdata1 (1036×1036), with $\rho = 0.0001$; dynamic range is 5.9915 and sparsity is 16.

Table 3.4: Comparison of the algorithms for solving (3.20) with $\rho = 0.0001$ on Bdata1. The starting μ is set to be 1. $FISTA(10) = 0.0015$, $FISTA(50) = 4.6868e{-}4$, $FISTA(100) = 1.8933e{-}4$, $FISTA(200) = 1.6275e{-}4$. The toleranceϵ_b is set to be 0.001.

solver	iter	mult	iter	mult	iter	mult	final iter	mult	final Obj.
FISTA	10	22	50	102	100	202	200	402	1.63e-4
FISTA_BKTR	8	18	40	99	81	193	160	368	1.63e-4
FALM-S1	8	22 (2)	37	112 (5)	114	282 (11)	190	434 (11)	1.63e-4
FALM-S2	4	10 (2)	17	51 (5)	95	224 (11)	172	378 (11)	1.63e-4
FALM-S3	2	4 (1)	12	38 (8)	93	212 (11)	169	364 (11)	1.63e-4
FALM_BKTR	5	13 (5)	14	60 (9)	80	331 (19)	142	479 (19)	1.63e-4
SpaRSA	8	16	55	110	166	332	230	460	1.63e-4
YALL1	16	49	27	82	36	109	81	244	1.63e-4

3.5.3 Sparco problems

This category of instances are obtained from [70]. Due to the fact that the Sparco instances use function handles for matrix computation, which our FALM implementation is not equipped to utilize, we do not include FALM in this comparison. We present results for Sparco3 (2048×1024), with $\rho = 0.01$, dynamic range of 2 and sparsity of 2. We observe that, for this relatively easy instance, FISTA-BKTR has minor advantage over FISTA in terms of the number of matrix-vector multiplications. For this example FISTA outperforms the alternating direction based method Yall1, while SpaRSA seems to be the winning method for this instance.

Table 3.5: Comparison of the algorithms for solving (3.20) with $\rho = 0.01$ on Sparco3. The starting μ is set to be 1. $FISTA(10) = 13.07180$, $FISTA(50) = 8.187212$, $FISTA(100) = 2.710062$. The toleranceϵ_b is set to be 0.001.

solver	iter	mult	iter	mult	iter	mult	final iter	mult	final Obj.
FISTA	10	24	50	104	100	204	207	418	2.22278
FISTA_BKTR	6	18	38	97	78	187	173	387	2.22278
SpaRSA	7	14	11	22	72	144	99	198	2.22278
YALL1	10	20	10	20	19	38	262	534	2.22278

3.5.4 Smoothed ℓ_2 norm minimization

As an alternative to problem (3.20) one may wish to solve the following problem with exact ℓ_2 penalty term:

$$\bar{x} := \arg\min_x \quad \|Ax - b\|_2 + \rho\|x\|_1, \tag{3.21}$$

In order to apply FISTA and FALM family of methods, we can smooth the ℓ_2 term with the well-known Huber penalty function, and obtain the following minimization problem:

$$\bar{x} := \arg\min_x \quad H_\nu(\|Ax - b\|_2) + \rho\|x\|_1, \tag{3.22}$$

where

$$H_\nu(y) = \begin{cases} \dfrac{y^2}{2\nu}, & 0 \le |y| \le \nu \\ |y| - \dfrac{\nu}{2}, & |y| \ge \nu, \end{cases}$$

for $\nu > 0$. If $\nu < \epsilon$, then the solution of (3.22) is an ϵ-solution to (3.21). converges to that of (3.21). We define $f(x) := H_\nu(\|Ax - b\|_2)$ and $g(x) := \rho\|x\|_1$. It is well known that global Lipschitz constant of $\nabla f(x)$ is $O(1/\nu)$. In [60], the analysis of the local composite Lipschitz constant for the case when $g(x) \equiv 0$ is derived and it shows that away from the optimal solution the local composite Lipschitz constant is of $\nabla f(x)$ is much smaller than $O(1/\nu)$. The analysis of the case when $g(x) := \rho\|x\|_1$ is a lot more complex, but the essential expectation remains for our first-order schemes: the prox parameter μ will be relatively large away from the solutions, while it will decrease as the algorithms converges. In fact FISTA and FALM in their original form will observe the same behavior of the prox parameter, as they allow for the prox parameter to decrease, but not to increase. Hence we do not expect a significant saving using backtracking in this setting, however, we present experiments for illustration and to confirm our expectation of the prox parameter behavior.

In Tables 3.6 and 3.7 we present a comparison of the first-order methods on Spear10 data and formulation (3.22). We observe that FISTA-BKTR is much faster than FISTA and SpaRSA, in the case when the initial prox parameter value is not very large. As compared with FISTA, FISTA-BKTR allows for a huge increase in μ. If initial μ is set to 1, then FISTA performance is very slow, as is shown in Table 3.6. But if $\mu = 1000$, then FISTA performance improves, to the level of FISTA-BKTR, as is seen in 3.7. This is well explained by showing graphically the change of μ in Fig 3.2.

Table 3.6: Comparison of the algorithms for solving (3.22) with $\rho = 0.1$ on Spear10. The starting μ is set to be 1. $FISTA(100) = 4.4552e{+}4$, $FISTA(500) = 2.4248e{+}4$, $FISTA(1000) = 1.1554e + 4$

solver	iter	mult	iter	mult	iter	mult	final iter	mult	final Obj.
FISTA	100	200	500	1000	1000	2000	2242	4486	9.99e+3
FISTA_BKTR	18	36	27	54	29	62	48	144	9.99e+3
SpaRSA	1558	3116	36014	72028	57198	114396	60313	120626	9.99e+3

In Table 3.8 and Figure 3.3 we show the outcome of the experiments on the Bdata1 test set. In this case, FISTA and FISTA-BKTR perform as expected, with FISTA-BKTR retaining a small advantage. In fact, after 500 iterations, the μ value for both algorithms becomes small which indicates large Lipschitz constant for solving (3.22)

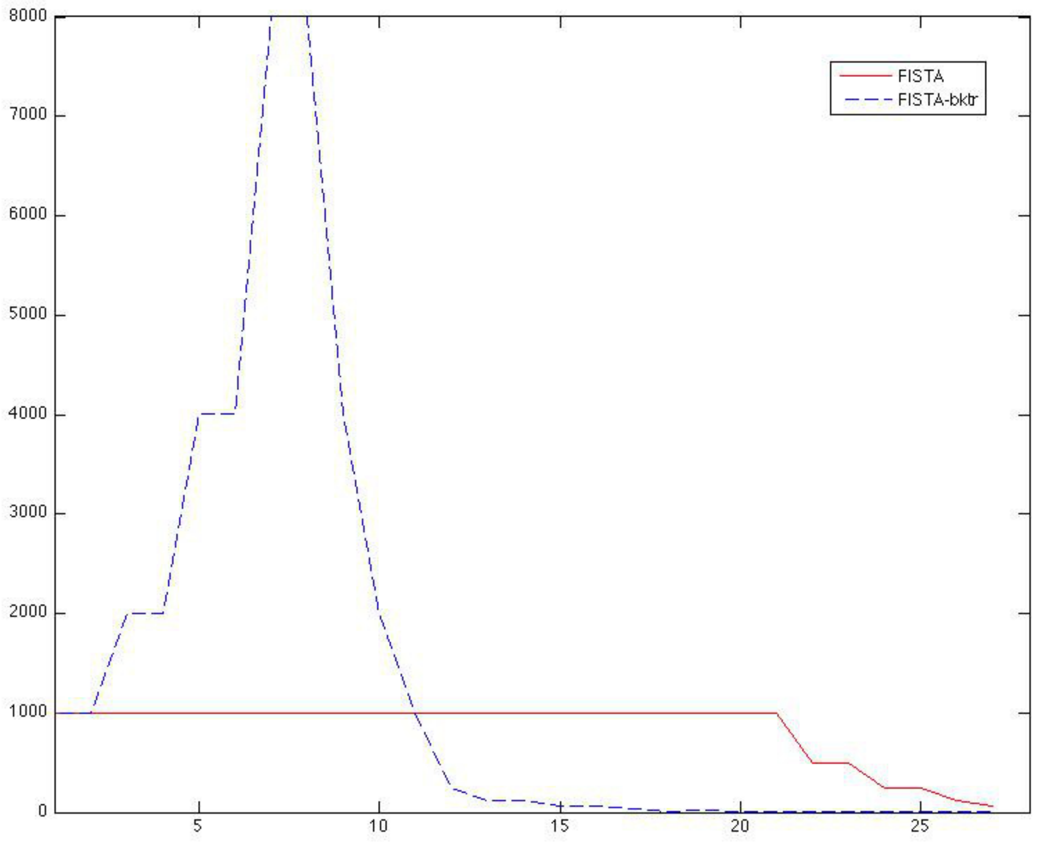


Figure 3.2: Comparison on the μ values while solving (3.22) with $\rho = 0.1$ on Spear10

Table 3.7: Comparison of the algorithms for solving (3.22) with $\rho = 0.1$ on Spear10. The starting μ is set to be 1000. $FISTA(10) = 3.4189e{+}4$, $FISTA(20) = 1.0866e+4$, $FISTA(40) = 9.9944e + 3$

solver	iter	mult	iter	mult	iter	mult	final iter	mult	final Obj.
FISTA	10	20	20	40	40	91	42	97	9.99e+3
FISTA_BKTR	7	14	11	28	25	92	28	102	9.99e+3
SpaRSA	18129	36258	57282	114564	59130	118260	59131	118262	9.99e+3

on Bdata1.

Table 3.8: Comparison of the algorithms for solving (3.22) with $\rho = 0.01$ on Bdata1. The starting μ is set to be 1. $FISTA(100) = 0.0196$, $FISTA(500) = 0.0164$, $FISTA(1000) = 0.0164$

solver	iter	mult	iter	mult	iter	mult	final iter	mult	final Obj.
FISTA	100	210	500	1010	1000	2010	2105	5249	0.0164
FISTA_BKTR	120	342	539	1290	539	1290	1624	3822	0.0164

3.5.5 ℓ_2 regularized logistic regression

Finally, we illustrate the behavior of FISTA vs. FISTA-BKTR on an example of ℓ_2 regularized logistic regression applied to an optical character recognition data set "Optdigits" from UCI repository [31]. We present this example here purely for illustration purpose to show that on settings other than compressed sensing the backtracking strategy can produce significant improvement. As we discussed, to obtain optimal performance from backtracking a careful implementation is needed that tries to take into account the problem structure. Such implementation for logistic regression and other problems is a subject of future research. Here we present results of basic approach where μ is increased by a factor of 2 on each iteration. FISTA required 5705 iterations and 11417 matrix-vector multiplications to obtain a solution with gradient norm less that 10^{-2}, while FISTA-BKTR required 1315 iterations and 3512 matrix-vector multiplications. In Figure 3.4 we plot the behavior

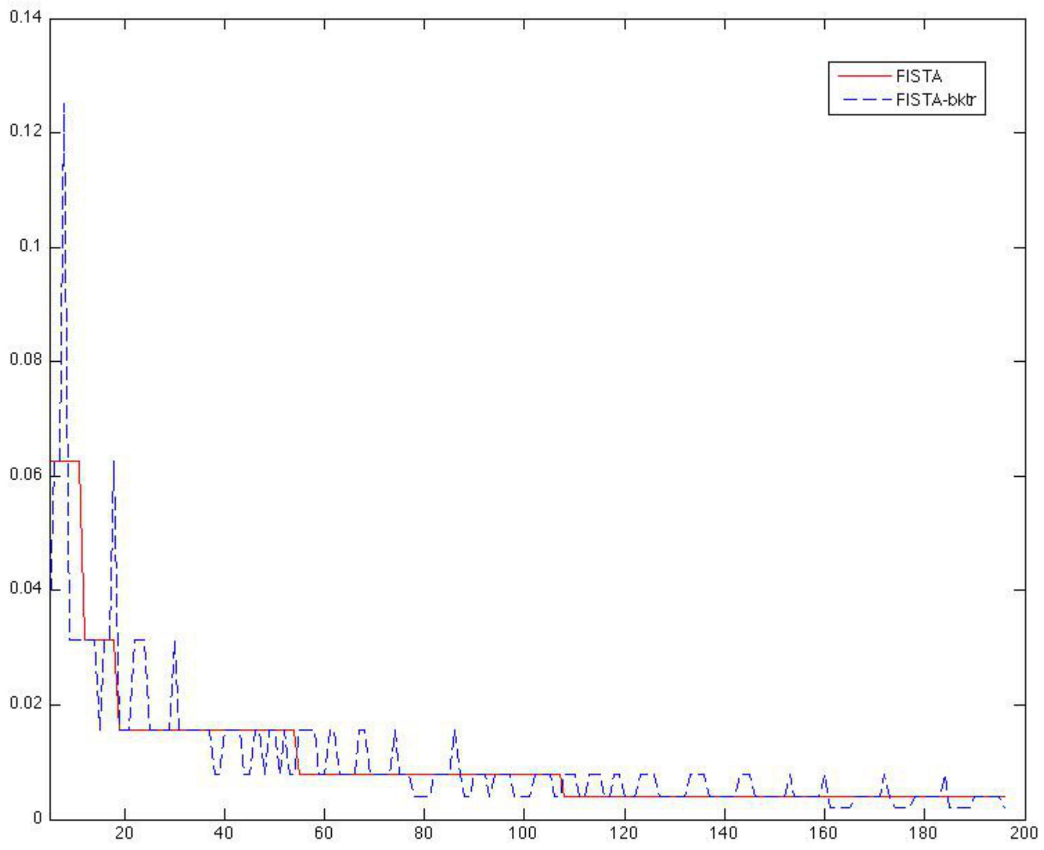


Figure 3.3: Comparison on the μ values while solving (3.22) with $\rho = 0.001$ on Bdata

of the μ parameter for both algorithms, which clearly shows that FISTA-BKTR benefits from much larger steps.

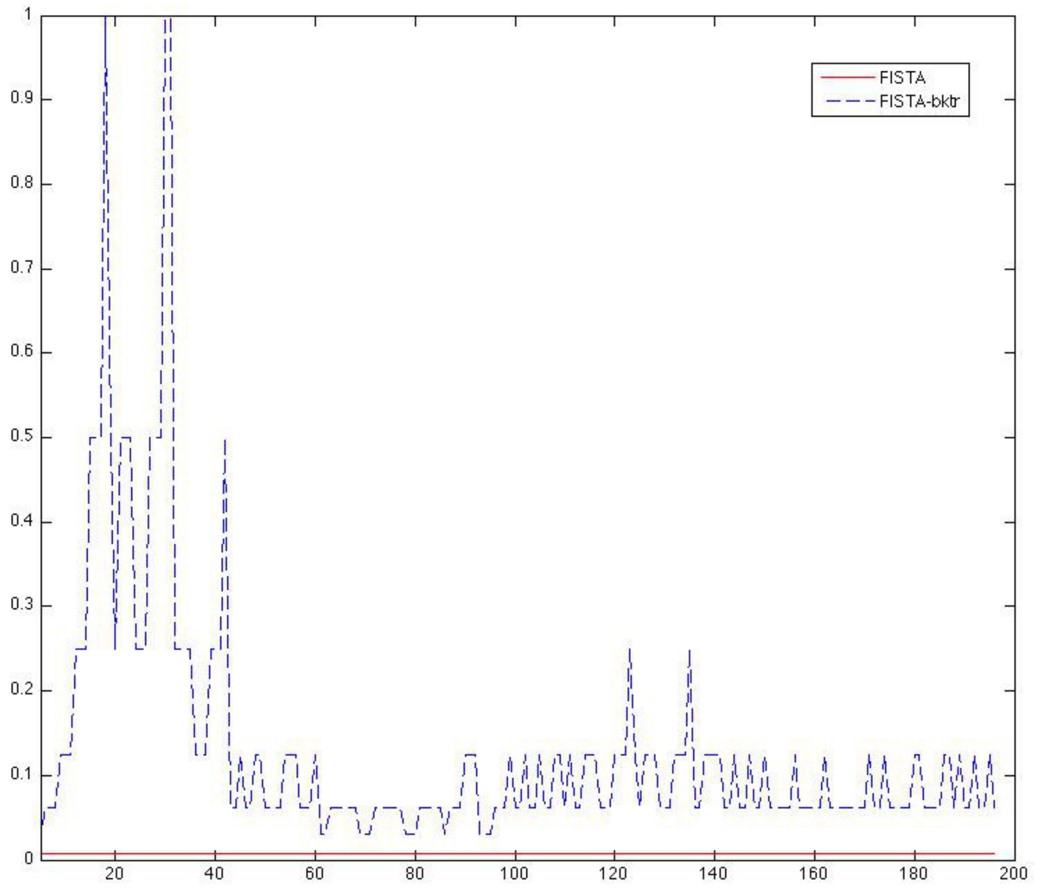


Figure 3.4: Comparison on the μ values while solving logistic regression problem on "Optdigits" dataset.

3.6 Conclusion

In this chapter we present a generalized version of accelerated first-order schemes capable of estimating the prox parameter via backtracking, thereby allowing for the value of this parameter to increase as well as decrease. We show that the value of the parameter depends on the so-called local composite Lipschitz constant of the gradient, rather than the global Lipschitz constant. Moreover, via some examples, it can be shown that the local constants can be much smaller than the global ones; hence, one could potentially obtain better convergence bounds. To produce such bounds, one would need to combine the analysis in [60] with the analysis of the iterates of a first-order algorithm, which will be the subject of a future study. Our computational experiments and the discussion of a practical implementation show that in practice our proposed backtracking schemes offer improvements in terms of accelerated first-order algorithms.

Chapter 4

Least-squares approach to risk parity in portfolio selection

4.1 Introduction

Quantitative portfolio selection has been a focus of many investors and researchers in the past decades. One central goal in quantitative portfolio selection is to manage the tradeoff between return and risk. Since 1950s, many optimization models have appeared and they continue to play an important role in making investment decisions. However, some strategies resulting from optimal portfolio selection models have been observed to be difficult to implement in practice. One of the commonly cited reasons is the reliance of many of these models on forecasts of future returns. When estimated from historical information, these forecasts can be inaccurate predictors of future behavior of the security returns, and optimal portfolios constructed using these predictors can therefore be inefficient.

One well-known example of an optimal portfolio selection strategy is the mean-variance optimization model. Proposed by Markowitz, mean-variance optimization approach has been regarded as a fundamental framework in portfolio construction. It offered the first quantitative insight into the tradeoff between returns and risk. One persistent criticism of the mean-variance model has been its sensitivity to inputs. Among others, research by [11] has shown that slight changes in input parameters, especially in the expected return estimates, can lead to dramatic changes in the optimal portfolio composition.

In practice, some investors apply much simpler asset allocation and portfolio selection strategies and might suffer from their limitations as a consequence. One of these simpler approaches is the well-known "60/40" strategy (60% equity, 40% bonds). From a risk management perspective, the relative balance between equity and fixed income instruments of this strategy is deceptive; because of the higher risk of the equity investments, often more than 90% of the risk of the portfolio comes from equities. Another simple diversification strategy is the "1/n" method, namely, portfolio construction with equal weights in all asset classes; see [23]. But, once again, the diversification is achieved only at the capital allocation level and not in terms of risk contributions, simply because the approach does not utilize any information on the assets' volatility or their correlations.

In this chapter, we investigate another portfolio selection strategy – the risk parity approach. The idea of risk parity is not new and can be regarded as a special type of diversification strategy. Using volatility as the measure of risk, the risk parity approach aims to create a portfolio with equal risk contributions from each of the assets in the portfolio. The past few years have witnessed an increasing focus on risk parity research; see, for instance, [18,19,48]. Most of the existing work considers risk parity in the context of long-only portfolios. In this case, it can be demonstrated that a risk parity portfolio can be obtained from the solution of a convex optimization problem. Under reasonable assumptions, this solution is unique. Some analysis on theoretical properties of long-only risk parity portfolios can be seen in [48, 63].

In the context of long-short portfolios, the risk parity portfolio selection is a richer and more challenging problem. For example, if shorting is allowed multiple risk-parity solutions may exist. We show that a convex model still applies if the investor identifies in advance which assets should be shorted. However, without any such restrictions we show that there could be a combinatorially large number of solutions that satisfy risk parity and these solutions can be identified through an enumeration strategy. In practice, investors may consider adding general bounds on the individual weights of the assets which may limit the total number of risk parity solutions. Moreover, if the bounds are sufficiently tight a risk parity solution satisfying such bounds may not exist at all.

We propose a generalized risk parity model which allows for short sales and applies to cases where risk parity solutions may not exist. Our model is similar to the model proposed in [48] in that we minimize a function that measures deviation from risk parity in a least-squares sense. However, our formulation has a simpler structure and allows for easier analysis and efficient algorithmic approaches. As in [48], the optimization model that we consider is not convex. When they exist, each risk parity solution is a global optimum of the least-squares model. Moreover, no local optimum solution exists on the interior of the feasible set; in other words, all local optima of our proposed formulation occur due to the presence of bounds on the weights. We develop an algorithmic framework based on alternating linearization methods (ALMs) to solve our generalized model. The framework is simple and convergent to a local optimum that is guaranteed to be a global optimum when no constraints are binding. Our formulation also easily extends to the case of the multiple objectives;

for example, to the case where one chooses the best risk parity solution according to some additional criterion, when multiple solutions exist.

The rest of the chapter is organized as follows. After a brief discussion of the minimum variance optimization framework in Section 4.2, we introduce the concept of risk parity and consider the convex log-barrier model as well as the proposed least-squares model. In Section 4.3 we discuss some properties of the local and global optima of our new model. In Section 4.4, we discuss some extensions based on the least-square model, where we aim to choose the best risk parity solution. We also propose another extension of the risk parity problem to the case where we seek parity on risk contributions of groups of assets rather than individual assets. We propose a class of algorithmic methods based on the ALM framework, to solve the least-square model in Section 4.5. Experiments and computational results are discussed in Section 4.6, followed by concluding remarks.

4.2 Risk parity problem

Numerous methods based on the famous Markowitz mean-variance framework have been proposed to overcome its drawbacks while maintaining its advantages; see, for instance, [65]. In this chapter, we focus on risk based diversification strategies. Unlike the classical mean-variance approach, risk based strategies do not incorporate expected returns into the formulation. Motivations for not using expected returns in the portfolio construction include the difficulty of estimating these quantities accurately, and the well documented sensitivity of the optimal weights to small changes in expected returns. For these reasons, [48] argue that risk based strategies are more robust than approaches using expected returns.

One prominent example of risk-based strategies in portfolio selection is the minimum variance optimization approach. This approach aims to minimize the volatility of the portfolio, or its active risk, and can be formulated as a convex quadratic optimization problem. This problem can be solved efficiently using widely available optimization software and typically has a unique solution. Here we briefly introduce the minimum variance optimization model to compare it to the risk parity approach.

Suppose we have n risky assets. Their covariance is given by a symmetric matrix Σ which is assumed to be positive definite. The following optimization problem minimizes the total variance of a fully-invested long-only portfolio:

$$\begin{aligned} \min_{x} \quad & \frac{1}{2} x^\top \Sigma x \\ \text{s.t.} \quad & x_i \geq 0 \\ & \sum_{i=1}^{n} x_i = 1, \end{aligned} \tag{4.1}$$

where $x = [x_1, x_2, \ldots, x_n]^\top$ is the vector of the weights of the n assets. The factor $1/2$ in the objective is introduced to simplify the optimality conditions and has no impact on the optimal portfolio. From the first-order optimality conditions of the above problem we see that

$$\Sigma x - \lambda - \gamma \mathrm{e} = 0, \tag{4.2}$$

where e is an n-dimensional vector of all ones, $\lambda \in \mathbb{R}^n$ and $\gamma \in \mathbb{R}$ are the Lagrange multipliers corresponding to the long-only and full investment constraints, respectively. Note that, complementary slackness conditions imply that if some x_i is strictly larger than zero, then the corresponding λ_i must be zero. Combining this observation with (4.2), we obtain

$$(\Sigma x)_i = \gamma, \forall i \text{ s.t. } x_i \neq 0. \tag{4.3}$$

From (4.3) we obtain:

$$\left(\frac{\Sigma x}{\sqrt{x^\top \Sigma x}}\right)_i = \left(\frac{\Sigma x}{\sqrt{x^\top \Sigma x}}\right)_j, \forall i, j \text{ s.t. } x_i, x_j \neq 0. \tag{4.4}$$

In this chapter, we make the common choice of using volatility $\sigma(x) = (x^\top \Sigma x)^{\frac{1}{2}}$ as our risk measure. With this definition of risk, note that $\dfrac{\partial \sigma}{\partial x} = \dfrac{\Sigma x}{\sqrt{x^\top \Sigma x}}$ is the vector of *marginal risk contributions* for the assets in the portfolio. Hence, we have

$$\frac{\partial \sigma}{\partial x_i} = \frac{\partial \sigma}{\partial x_j}, \forall i, j \text{ s.t. } x_i, x_j > 0. \tag{4.5}$$

The above condition implies that, as long as we invest in an asset, its marginal risk contribution should be the same as that of all other assets with positive weights in the portfolio. As such, minimum variance approach leads to portfolios with equal marginal risk contributions. In practice, while dominating other strategies from the perspective of low volatility, the minimum variance approach often leads to concentrated portfolios, i.e., encourages investors to concentrate on a small number of assets with lower risk profiles and to give up diversification. This behavior is often undesirable and this is exactly what risk parity optimization intends to overcome.

Risk parity portfolios can be motivated by considering Euler decomposition of a portfolio risk measure into contributions from each asset in the portfolio.

Theorem 4.2.1. *(Euler's theorem) Let a continuous and differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ be a homogeneous function of degree one; i.e. for any constant $c \in \mathbb{R}$, $f(cx) = c \cdot f(x)$. Then,*

$$f(x) \;=\; x_1 \cdot \frac{\partial f}{\partial x_1} + x_2 \cdot \frac{\partial f}{\partial x_2} + \ldots + x_n \cdot \frac{\partial f}{\partial x_n} \;.$$

Simply put, a risk parity portfolio is a portfolio where the total contribution of each asset to the portfolio risk is equal. When $\sigma(x) = (x^\top \Sigma x)^{\frac{1}{2}}$ is used as the risk measure, using Euler's theorem, we can decompose $\sigma(x)$ as $\sigma(x) = \sum_{i=1}^{n} x_i \cdot \frac{\partial \sigma}{\partial x_i}$. Here, the quantity $x_i \cdot \frac{\partial \sigma}{\partial x_i}$ represents the contribution of asset i to portfolio risk. In this context, the risk parity problem aims to find any portfolio that satisfies

$$x_i \cdot \frac{\partial \sigma}{\partial x_i} = x_j \cdot \frac{\partial \sigma}{\partial x_j}, \forall i, j. \tag{4.6}$$

We will refer to any solution satisfying (4.6) as a *risk parity solution.* If we restrict the weight vector to be normalized, we obtain the normalized risk parity problem:

$$\begin{aligned} & x_i \cdot \frac{\partial \sigma}{\partial x_i} = x_j \cdot \frac{\partial \sigma}{\partial x_j}, \forall i, j \\ & \sum_{i=1}^{n} x_i = 1. \end{aligned} \tag{4.7}$$

We will refer to any solution of (4.7) as *normalized risk parity solution.* Note that one can easily convert a solution of (4.6) into a solution of (4.7) through simple scaling, as long as the sum of the asset weights is not zero.

We will also consider situations where risk parity solutions may not exist because of the presence of additional restrictions on the portfolio weights. In such cases, we will look for portfolios that are "close to risk parity" and for this purpose it will be important to quantify the deviation from risk parity. We address this issue in Section 4.2.3.

Let us assume for a moment that the correlation between any two assets is a constant, that is $\rho_{ij} = \rho, \forall i, j$. It can be shown that a closed form solution to (4.7) can be derived under this assumption; see [48]. This solution is given by the following identity: $x_i = \frac{\sigma_i^{-1}}{\sum_{j=1}^{n} \sigma_j^{-1}}$, where σ_i^2 represent the diagonal elements of the covariance matrix Σ. Some literature refers to this case as the "naive risk parity" solution. When the correlations are not constant, a closed-form solution for x does not exist in general and numerical approaches need to be applied.

4.2.1 Long-only risk parity via convex optimization

Next, we consider the problem of finding a long-only risk parity solution, i.e., finding a vector of weights of n assets $x = [x_1, x_2, \ldots, x_n]^\top$ such that $x_i \cdot \dfrac{\partial \sigma}{\partial x_i} = x_j \cdot \dfrac{\partial \sigma}{\partial x_j}, \forall i, j$ and $x \geq 0$. In this case, it turns out that solving an artificial optimization problem that incorporates a logarithmic barrier term is equivalent to finding a risk parity solution:

$$\begin{aligned} \min_{x} \quad & \frac{1}{2} x^\top \Sigma x - c \sum_{i=1}^{n} \ln x_i \\ \text{s.t.} \quad & x_i > 0, \end{aligned} \tag{4.8}$$

where Σ is the covariance matrix and c is an arbitrary positive constant. Our use of the logarithmic barrier term in the objective function of (4.8) is motivated by a related formulation in [48] that uses a constraint incorporating the sum of the logarithms. Other authors have also utilized the logarithmic barrier function for solving the risk parity problem in independently developed studies; see [42, 63].

Since Σ is positive definite and the logarithm function is strictly concave, the objective function of (4.8) is strictly convex. Under strict convexity, we observe that this convex optimization problem has a unique solution. From the first-order optimality conditions, this solution lies at the point where the gradient of the objective function, $\Sigma x - cx^{-1}$ is zero, where $x^{-1} = [1/x_1, 1/x_2, \ldots, 1/x_n]^\top$. Hence, at optimality we have $(\Sigma x)_i = \dfrac{c}{x_i}, \forall i$, which leads to

$$x_i (\Sigma x)_i = x_j (\Sigma x)_j, \forall i, j. \tag{4.9}$$

It is now easy to see that (4.9) is equivalent to $x_i \cdot \dfrac{\partial \sigma}{\partial x_i} = x_j \cdot \dfrac{\partial \sigma}{\partial x_j}, \forall i, j$ and that risk parity is achieved at the unique optimal solution of (4.8).

There is no guarantee that the weights in the solution of (4.8) will sum to one, so the result may represent a levered or an under-invested portfolio. Fortunately, we have the following result showing the existence and uniqueness of the risk parity solution in the long-only case if we impose the additional constraint that the sum of all weights equals to one. The proof is similar to Theorem 1.1 in [63].

Lemma 4.2.2. *Let Σ be a positive definite covariance matrix. Then there exists a unique solution x which satisfies:*

$$x_i(\Sigma x)_i = x_j(\Sigma x)_j, \forall i, j \tag{4.10}$$

$$\sum_{i=1}^{n} x_i = 1, \quad x_i > 0, \quad \forall i. \tag{4.11}$$

In fact, any two long-only risk parity solutions differ by a constant factor.

Proof. The objective function of (4.8) is continuous and strongly convex and it increases to infinity at the boundary of the feasible region. Hence, the intersection of its level set and the feasible region is compact. This implies that it has a unique global minimum, say, x^c for any given $c > 0$. Consider the following optimization problem:

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top \Sigma x - \alpha c \sum_{i=1}^{n} \ln x_i \\ \text{s.t.} \quad & x_i > 0, \end{aligned} \tag{4.12}$$

where α is a positive scalar. It is easy to verify that $\sqrt{\alpha} x^c$ is the unique solution for this problem since $\sqrt{\alpha} x_i^c \left[\Sigma(\sqrt{\alpha} x^c)\right]_i = \alpha c$. Hence, as α varies from 0 to ∞, $\sum_{i=1}^{n} \sqrt{\alpha} x_i^c$ varies from 0 to ∞. Further, for $\alpha^* = \dfrac{1}{(\sum_{k=1}^{n} x_k^c)^2}$, the solution of (4.12), denoted as x^*, satisfies $\sum_{i=1}^{n} x_i^* = 1$. It is easy to see that $x_i^* = \frac{x_i}{\sum_{k=1}^{n} x_k^c}$ and x^* is unique and independent of initial choice of c.

□

4.2.2 Risk parity solutions over orthants

In this subsection, we explore the set of risk parity solutions when the long-only restriction on weights is removed. The approach discussed in Section 4.2.1 identifies only the solution in the nonnegative orthant. To find solutions in other orthants, we consider the following modified log-barrier approach:

$$\begin{aligned} \min_{x} \quad & \frac{1}{2} x^\top \Sigma x - c \sum_{i=1}^{n} \ln \beta_i x_i \\ \text{s.t.} \quad & \beta_i x_i > 0, \end{aligned} \tag{4.13}$$

where $\beta = [\beta_1, \beta_2, \ldots, \beta_n]^\top \in \{-1, 1\}^n$, defines the orthant in which the solution is sought. If $\beta_i = 1$, then $x_i \in (0, +\infty)$; otherwise, $x_i \in (-\infty, 0)$. For each choice of β, (4.13) is a convex optimization problem and thus it has a unique solution x^β. Since there are 2^n such different β vectors, there are 2^n such solutions. Let $\bar{\beta} = -\beta$ define the complementary orthant of the orthant defined by β. It is easy to see that $x^\beta = -x^{\bar{\beta}}$. We have the following lemma.

Lemma 4.2.3. *Let Σ be a positive definite covariance matrix. Then there exist at most 2^{n-1} solutions which satisfy $x_i(\Sigma x)_i = x_j(\Sigma x)_j, \forall i, j;$* and $\sum_{i=1}^{n} x_i = 1$.

Proof. For each β and x^β, $\hat{x}^\beta = \frac{x^\beta}{\sum_{i=1}^{n} x_i^\beta}$ is a normalized risk parity solution satisfying (4.7) as long as $\sum_{i=1}^{n} x_i^\beta \neq 0$. Note that $\frac{x_i}{\sum_{i=1}^{n} x_i} = \frac{-x_i}{\sum_{i=1}^{n}(-x_i)}$. Also note that for all unnormalized risk parity solutions in the same orthant either $\sum_{i=1}^{n} x_i^{\beta} = 0$ or $\sum_{i=1}^{n} x_i^{\beta} < 0$ or $\sum_{i=1}^{n} x_i^{\beta} > 0$. Hence, for every choice of β, $\bar{\beta} = -\beta$ generates the same scaled risk parity solution $\hat{x}^{\beta}$, and thus there are at most 2^{n-1} such solutions. In other words, for each orthant, the normalized risk parity solution exists if and only if $\sum_{i=1}^{n} x_i^{\beta} \neq 0$ for any of the unnormalized risk parity solutions in that orthant. If $\sum_{i=1}^{n} x_i^{\beta} < 0$, then the normalized risk parity solution lies in the complementary orthant. $\square$

Remark 4.2.4. *Note that a scaled risk parity solution does not exist in the orthant defined by* β *if* $\sum_{i=1}^{n} x_i^{\beta} = 0$ *in Lemma 4.2.3. Hence, there may be fewer than* 2^{n-1} *normalized solutions. When* $\sum_{i=1}^{n} x_i^{\beta} = 0$ *the portfolio is "market-neutral". That is,* x *represents a zero investment portfolio and its aggregate exposure to the market is zero. This case is independently interesting and can sometimes be desirable in portfolio selection.*

To illustrate Lemma 4.2.3 with an example, let us consider the simple case when the covariance matrix is diagonal, i.e.

$$\Sigma = \begin{bmatrix} \sigma_1^2 & & & \\ & \sigma_2^2 & & \\ & & \ddots & \\ & & & \sigma_n^2 \end{bmatrix}.$$

In this case risk parity is equivalent to the following system of $(n-1)$ linearly independent equations:

$$\begin{aligned} \beta_1\sigma_1 x_1 &= \beta_2\sigma_2 x_2 \\ \beta_1\sigma_1 x_1 &= \beta_3\sigma_3 x_3 \\ &\vdots \\ \beta_1\sigma_1 x_1 &= \beta_n\sigma_n x_n, \end{aligned} \tag{4.14}$$

where $\beta_i \in \{-1, 1\}, \forall i \in \{1, \ldots, n\}$. Suppose $\sigma_i^{-1} + \sum\limits_{j \neq i} \frac{\beta_j}{\beta_i}\sigma_j^{-1} \neq 0, \forall i$. Then (4.14), together with $\sum\limits_{i=1}^{n} x_i = 1$, leads to a set of closed form solutions

$$x_i = \frac{\sigma_i^{-1}}{\sigma_i^{-1} + \sum_{j \neq i} \frac{\beta_j}{\beta_i}\sigma_j^{-1}}. \tag{4.15}$$

Since there are 2^n different β, and each pair $(\beta, -\beta)$ lead to the same system of equations (4.14), there are at most 2^{n-1} different solutions. Additionally, in (4.14), if $\beta_i = 1, \forall i \in \{1, \ldots, n\}$, then (4.15) becomes $x_i = \frac{\sigma_i^{-1}}{\sum_j \sigma_j^{-1}}$, which is simply the "naive risk parity" solution discussed previously.

Given that there may be exponentially many different risk-parity solutions in the long-short case, additional preferences or restrictions on the weights can be used to narrow down these choices. For example, one may look for a risk parity solution with the least volatility or one with weights lying between given bounds. If investors know a priori which assets are desirable to short, then the log barrier approach of Section 4.2.1 can be easily extended to find the desired long-short risk-parity portfolio by selecting the appropriate β in (4.13). In the next two subsections, we discuss efficient methods for finding solutions satisfying additional preferences or restrictions without resorting to enumeration.

4.2.3 Least-squares model with general bounds

The log-barrier approach to finding risk parity solutions does not immediately extend to scenarios with additional constraints or preferences. In particular, when general bounds are added, risk parity solution may not exist, however, up to 2^{n-1} different instances of (4.13) may have to be solved before infeasibility can be established. Moreover, the log barrier formulation gives no direct guidance on how to produce a feasible solution which may be "close to risk parity". Simple approaches, such as projecting infeasible normalized risk parity solutions onto the feasible region defined by the constraints may generate portfolios that deviate substantially from risk parity. In addition, it is not clear how to extend this approach to the cases when risk parity is desirable for groups of assets rather than for individual assets (e.g., industry-based groups).

In this section we propose a least-squares formulation for solving the risk parity problem. Our formulation is similar to the following formulation proposed in [48]:

$$\begin{aligned} \min_{x} \quad & \sum_{i=1,j=1}^{n} (x_i(\Sigma x)_i - x_j(\Sigma x)_j)^2 \\ \text{s.t.} \quad & a_i \le x_i \le b_i \\ & \sum_{i=1}^{n} x_i = 1. \end{aligned} \tag{4.16}$$

Above, a_i and b_i are arbitrary constants representing the bounds on the weight of the i-th asset (a_i can be less than zero if we allow short sales). The objective function of (4.16) introduces a penalty term for each pair of risk contribution terms $x_i(\Sigma x)_i$ and $x_j(\Sigma x)_j$ that are different from each other. Alternatively, one can consider using penalty terms for deviations of risk contributions from their average value:

$$\min_x \sum_{i=1}^n (x_i(\Sigma x)_i - \frac{\sum_{j=1}^n x_j(\Sigma x)_j}{n})^2.$$

Our formulation is based on this second objective function, but replaces the average risk contribution term with a free variable θ that is also optimized:

$$\begin{aligned} \min_{x,\theta} \quad & \sum_{i=1}^n (x_i(\Sigma x)_i - \theta)^2 \\ \text{s.t.} \quad & a_i \leq x_i \leq b_i \\ & \sum_{i=1}^n x_i = 1, \end{aligned} \tag{4.17}$$

For future reference, we denote by $F(x,\theta)$ the objective function of (4.17):

$$F(x,\theta) := \sum_{i=1}^n (x_i(\Sigma x)_i - \theta)^2. \tag{4.18}$$

If the optimization problem (4.17) has an optimal value of zero, then risk parity is achieved. Otherwise, the value of the objective function $F(x,\theta)$ can be regarded as a minimum variance measure towards our goal.

The two formulations (4.16) and (4.17) are equivalent in the sense that any risk parity solution is a solution to both optimization problems. However, our formulation (4.17) offers a much simpler form of the objective function containing only n elements in the sum, while the formulation from [48], contains an order of n^2 elements. Hence, our formulation is computationally less demanding, is easier to analyze and contains fewer nonlinearities. Moreover it allows us to develop efficient optimization approaches as will be seen in Section 4.5 The auxiliary variable θ can always be set to its optimal value based on the following lemma. However, allowing θ to be a free variable significantly simplifies the formulation.

Lemma 4.2.5. *For any fixed vector x, there exists one and only one θ^* such that* (4.18) *is minimized, and $\theta^* = (\sum_{i=1}^{n} x_i(\Sigma x)_i)/n = x^\top \Sigma x/n$.*

Proof. For a fixed vector x, (4.18) is a strictly convex unconstrained function of θ. Further, the function is minimized when first-order optimality is satisfied, which implies $\theta^* = (\sum_{i=1}^{n} x_i(\Sigma x)_i)/n$. □

Example 4.2.6. *Consider three assets with volatilities $\sigma_1 = 1, \sigma_2 = 1, \sigma_3 = 2$, respectively and assume the correlation matrix is the 3×3 identity matrix.*

Thus, the covariance matrix is

$$\Sigma = \begin{bmatrix} 1.0 & & \\ & 1.0 & \\ & & 4.0 \end{bmatrix}.$$

The normalized risk-parity solution in the positive orthant is $x^0 = [0.4; 0.4; 0.2]$. Now, suppose we have a restricted feasible region: $0.5 \leq x_1 \leq \infty, x_2, x_3 \in \mathbb{R}^+$. If we project x^0 onto the feasible region, we obtain $x^p = [0.5; 0.35; 0.15]$. This solution is not "optimal" from the perspective of risk parity solution and the objective function of (4.17)*. To see that, consider the solution obtained by optimizing* (4.17)*: $x^{opt} = [0.5; 0.333; 0.167]$. We see that the objective function value at x^p equals* 0.0143*, while at x^{opt} it equals* 0.0128*. Hence, in terms of risk parity, x^{opt} is preferable to x^p.*

We can also compare these two solutions using measures of risk concentration. For this purpose, we consider two metrics: the highest risk contribution by any asset and the Herfindahl index. As its name suggests, the highest risk contribution is defined as

$$HRC(x) := \max_i \frac{x_i(\Sigma x)_i}{x^\top \Sigma x}, \tag{4.19}$$

while the Herfindahl index is defined as follows:

$$h(x) = \sum_{i=1}^{n} \left[\frac{x_i(\Sigma x)_i}{x^\top \Sigma x} \right]^2 . \tag{4.20}$$

The values for the Herfindahl index range between $\frac{1}{n}$*, for the risk-parity portfolio, and 1, for a perfectly concentrated portfolio.*

We observe that $HRC(x^p) = 0.5405$*, while* $HRC(x^{opt}) = 0.5292$*; and* $h_{x^p} = 0.4002$*, while* $h_{x^{opt}} = 0.3909$*. While the differences are small, both measures indicate that* x^{opt} *is a better solution in terms of risk concentration, suggesting that optimizing* (4.17) *leads to less concentrated portfolios.*

Note that, unlike problems (4.1) and (4.8), (4.17) is a non-convex optimization problem, hence in theory it is harder to solve and may have local solutions. However, as we will show it is a useful formulation, as simple practical and fast optimization schemes can be developed for this problem. Further, this formulation can be extended to include additional optimization criteria and different variants of risk parity.

4.3 Local and global optima issues

In this section, we investigate conditions under which first-order optimality conditions for (4.17) are sufficient for global optimality. The Lagrangian function for (4.17) can be written as

$$\mathcal{L}(x, \theta) = \sum_{i=1}^{n} (x_i(\Sigma x)_i - \theta)^2 - \lambda_a^\top (x - a) - \lambda_b^\top (b - x) + \gamma(\sum_{i=1}^{n} x_i - 1), \tag{4.21}$$

where $\lambda_a, \lambda_b \in \mathbb{R}^n_+, \gamma \in \mathbb{R}$. We now derive the first-order conditions:

$$\frac{\partial \mathcal{L}}{\partial x_i} = \frac{\partial F}{\partial x_i} - (\lambda_a)_i + (\lambda_b)_i + \gamma = 0, \ \forall i, \tag{4.22a}$$

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial F}{\partial \theta} = 0, \tag{4.22b}$$

$$\sum_{i=1}^{n} x_i - 1 = 0, \tag{4.22c}$$

$$a_i \leq x_i \leq b_i, \ \forall i, \tag{4.22d}$$

$$\lambda_a, \lambda_b \geq 0, \tag{4.22e}$$

$$(\lambda_a)_i (x_i - a_i) = 0, \ (\lambda_b)_i (b_i - x_i) = 0. \tag{4.22f}$$

The gradient of F with respect to x is

$$\nabla_x F(x, \theta) = 2 \sum_{i=1}^{n} (x_i (\Sigma x)_i - \theta)(e_i \Sigma_i + (e_i \Sigma_i)^\top) x, \tag{4.23}$$

where $e_i \in \mathbb{R}^{n \times 1}$ is the ith column of the identity and $\Sigma_i \in \mathbb{R}^{1 \times n}$ is the ith row of the covariance matrix.

When constraints are excluded from (4.17) we have the following unconstrained problem:

$$\min_{x,\theta} F(x, \theta) = \sum_{i=1}^{n} (x_i (\Sigma x)_i - \theta)^2. \tag{4.24}$$

In this case, the first order conditions are simpler:

$$\nabla_x F = 0, \frac{\partial F}{\partial \theta} = 0.$$

We have the following lemma for this unconstrained case:

Lemma 4.3.1. *A solution pair* $\{x, \theta\}$ *is a global optimum of* (4.24) *with* $F(x, \theta) = 0$

if and only if $\nabla_x F = 0, \dfrac{\partial F}{\partial \theta} = 0.$

Proof. 1) If $F = 0$, then $x_i(\Sigma x)_i - \theta = 0,\ \forall i$, from which $\nabla_x F = 2\sum_{i=1}^{n}(x_i(\Sigma x)_i - \theta)(e_i\Sigma_i + (e_i\Sigma_i)^\top)x = 0$ holds trivially.

2) If $\nabla_x F = 0$, then $x^\top \nabla_x F = 0$; hence,

$$2\sum_{i=1}^{n}(x_i(\Sigma x)_i - \theta)x^\top(e_i\Sigma_i + (e_i\Sigma_i)^\top)x = 2\sum_{i=1}^{n}(x_i(\Sigma x)_i - \theta)(x_i(\Sigma x)_i + (x_i(\Sigma x)_i)^\top) = 0. \tag{4.25}$$

Let $B_i = x_i(\Sigma x)_i$, note that $B_i = B_i^\top$. Then, ignoring the constant factor, we have

$$\sum_{i=1}^{n}(B_i - \theta)B_i = 0. \tag{4.26}$$

Applying the second condition $\dfrac{\partial F}{\partial \theta} = 0$, which implies $\theta = \dfrac{\sum_{i=1}^{n} B_i}{n}$, we have

$$n\sum_{i=1}^{n} B_i^2 = (\sum_{i=1}^{n} B_i)^2. \tag{4.27}$$

On the other hand, from Cauchy–Schwarz inequality we know that $n\sum_{i=1}^{n} B_i^2 \geq (\sum_{i=1}^{n} B_i)^2$. Furthermore, (4.27) holds only when $B_i = B_j$ for all $i, j \in \{1, \dots, n\}$. Hence, $F(x, \theta) = 0$.

$\square$

In contrast, when constraints are imposed, local optima and local stationary points can occur. The following simple example shows that the inclusion of the full investment constraint, even without additional bounds, can lead to a local stationary point that is not a global solution:

Example 4.3.2. *Consider an example of two assets with volatility* $\sigma_1 = 1, \sigma_2 = 2$*, respectively, and the covariance matrix*

$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}.$$

Suppose there are no bounds on x_1 *and* x_2 *but we have* $\sum_{i=1}^{n} x_i = 1$*. Then it is easy to solve the system of equations that satisfies KKT conditions. There are three solutions, two of which,* $x^1 = [\frac{2}{3}, \frac{1}{3}]$ *and* $x^2 = [2; -1]$ *are risk parity solutions. The third stationary point is* $x^3 = [\frac{4}{3}, -\frac{1}{3}]$*, which is not a risk parity solution. Indeed, this point corresponds to a local maximum of the function* F*.*

The next example shows that a local minimum can also occur when there are bound constraints in (4.17), even if there is a risk parity solution, and thus, a global optimum, in the same orthant.

Example 4.3.3. *Consider Example 4.3.2 but with additional bounds* $1.2 \leq x_1 \leq \infty, -\infty \leq x_2 \leq -0.2$*.* $x_1 = 2, x_2 = -1$ *is the risk parity solution that satisfies the bounds. As Figure 4.1 shows,* $x_1 = 1.2, x_2 = -0.2$ *is a local minimum but is not a global minimum.*

These last two examples suggest that care must be taken in solving (4.17) if we wish to avoid local solutions and find a global solution of the problem.

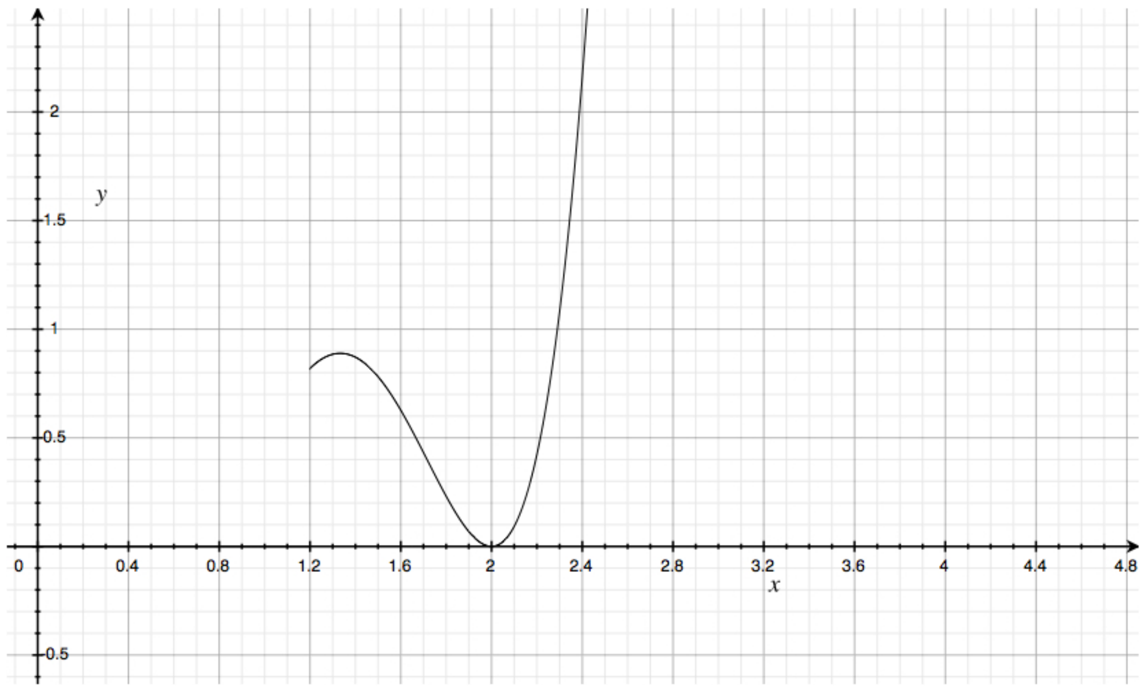


Figure 4.1: The function value in Example 4.3.3 with respect to x_1. Note that, in this 2×2 case, $x_2 = 1 - x_1$, and $\theta = \dfrac{\sigma_1^2 x_1^2 + \sigma_2^2 x_2^2}{2} = \dfrac{\sigma_1^2 x_1^2 + \sigma_2^2 (1 - x_1)^2}{2}$. Hence, the figure here shows $F(x_1, x_2, \theta) = (x_1^2 - \dfrac{x_1^2 + 4(1 - x_1)^2}{2})^2 + (4(1 - x_1)^2 - \dfrac{x_1^2 + 4(1 - x_1)^2}{2})^2$ when $x_1 \geq 1.2$. It shows that $x_1 = 1.2$ is a local optimum on the boundary.

4.4 Extended least-squares models

In this section, we discuss several useful extensions of the risk parity problems that can be easily included in our least-squares model.

4.4.1 Minimum variance with risk parity

As we discussed above, when short positions are allowed, multiple risk parity solutions may exist. In such cases, the investors have the option to define additional criteria on their portfolio preferences to narrow down the choices for risk parity portfolios, and possibly pick a "best" risk-parity portfolio according to the additional criteria. Introducing preferences based on expected returns is one possible option, but we do not consider this here as we are focused on risk-based strategies. Instead, we seek the risk-parity solution with the least variance. One possible approach to achieve this objective is to enumerate all risk parity solutions and then pick the one with the least variance. Instead of this approach that may require identification of exponentially many risk parity solutions, we consider a multi-objective formulation where the objective function is a weighted sum of total portfolio variance and least-squares risk parity term:

$$\begin{aligned} \min_{x,\theta} \quad & \sum_{i=1}^{n} (x_i(\Sigma x)_i - \theta)^2 + \rho x^\top \Sigma x \\ \text{s.t.} \quad & a_i \leq x_i \leq b_i \\ & \sum_{i=1}^{n} x_i = 1, \end{aligned} \tag{4.28}$$

where $\rho \geq 0$ is the weight parameter. Note that, in the above formulation we simply added a convex term to the objective function of (4.17).

Without the risk-parity term in the objective, (4.28) is simply the minimum variance portfolio problem and is easy to solve using quadratic programming solvers. It is also easy to show that for a large enough ρ problem (4.28) is convex in the feasible domain, if this domain is bounded. We propose an approach where we simply solve a sequence of problems (4.28) with decreasing values of ρ. This approach is described in Algorithm 11 and is intended to find a risk parity solution with the smallest variance:

Algorithm 11 Sequential min-variance risk parity algorithm

```
1. Choose $\rho^0 > 0$, $\beta \in (0, 1)$, $\varepsilon > 0$ and $x^0$;
2. for $k = 0, 1, \ldots$
      If $\rho^k \geq \varepsilon$, then find $x^{k+1}$ that solves (4.28) with $\rho = \rho^k$ using $x^k$ as a starting point.
           Set $\rho^{k+1} := \rho^k \beta$, CONTINUE.
      Else, find $x^{k+1}$ that solves (4.28) $\rho = 0$ using $x^k$ as a starting point. EXIT.
```

By setting initial ρ to a large value, we initiate Algorithm 11 with a potentially easy to solve problem and a solution that is close to the minimum variance solution but may be far from risk parity. Then, the algorithm solves a sequence of subproblems of the form (4.28) with decreasing values ρ, initializing each new subproblem with the solution of the previous subproblem. The goal is to converge to the risk parity solution that has the smallest variance among all risk parity solutions. Recall that each orthant contains at most one normalized risk parity solution. Algorithm 11 attempts to identify the correct orthant where the minimum variance risk parity solution lies. Hence, once ρ is small enough (for instance, smaller than some predefined tolerance ε, or once the convergence to the correct orthant is apparent), we can drop the minimum variance term and solve the (pure) risk parity problem in the right orthant. In this case, we have a single-orthant risk parity problem as in (4.13) and can obtain the exact risk parity solution that has small volatility.

Note that, due to the nonconvexity of the objective function, there is no theoretical guarantee that Algorithm 11 will always converge to the minimum variance risk parity solution. However, this approach is significantly more efficient than computing 2^{n-1} possible solutions. In all our experiments, Algorithm 11 reliably produced good solutions and did not get "trapped" in bad local minima.

Example 4.4.1. *Often, the long-only risk parity portfolio is conservative in terms of risk, but it may not be the least risky one among all risk parity portfolios, especially in the presence of negatively correlated assets.*

Consider three assets whose volatilities are given by $\sigma_1 = 1, \sigma_2 = 1, \sigma_3 = 2$. *Moreover, assume the correlation matrix is given by*

$$Corr = \begin{bmatrix} 1.0 & -0.9 & 0.3 \\ -0.9 & 1.0 & -0.1 \\ 0.3 & -0.1 & 1.0 \end{bmatrix}.$$

Thus, the covariance matrix is

$$\Sigma = \begin{bmatrix} 1.0 & -0.9 & 0.6 \\ -0.9 & 1.0 & -0.2 \\ 0.6 & -0.2 & 4.0 \end{bmatrix}.$$

Since this is a small instance, it is not hard to empirically check that there are a total of 4 *normalized risk parity solutions. We list these solutions in Table 4.1.*

By successively setting the parameter ρ *to be 1000, 10, 0.1, 0.001 and* 10^{-5}, *and solving the corresponding five subproblems, Algorithm 11 finds the risk parity solution* $[0.574; 0.531; -0.105]$. *As indicated in Table 4.1, this is precisely the minimum*

Table 4.1: A comparison of strategies, with the lower and upper bounds to be $a = -1, b = 2$.

Items for comparison	x_i	RC_i	Volatility
Risk parity portfolio (1)	[0.455;0.481;0.064]	[0.333;0.333;0.333]	0.289
Risk parity portfolio (2)	[-1.912;1.605;1.307]	[0.333;0.333;0.333]	3.840
Risk parity portfolio (3)	[1.784;-1.999;1.215]	[0.333;0.333;0.333]	4.805
Risk parity portfolio (4)	[0.574;0.531;-0.105]	[0.333;0.333;0.333]	0.238

variance risk parity solution.

Our results suggest that, our method is very robust with respect to the starting point. More results and larger instances are considered in numerical experiments in Section 4.6

It is clear that a similar strategy can be applied to select best risk parity portfolio based on another criterion, such as expected return or value at risk, etc.

4.4.2 Group risk parity

Another interesting extension of the risk parity problem is the case of group risk parity where we seek parity of risk contributions from groups of assets instead of individual assets. This variation is useful in the case when there are a large number of assets; for instance, in the context of equity investing. The assets can be grouped using a common risk factor such as industry membership or market capitalization and we look for equal risk contribution from each factor instead of each individual asset.

One consequence of requiring risk parity at the individual asset level is that each asset receives a positive weight. This is not always desirable when investors prefer to limit the number of positions taken, or when transaction costs are of concern. While it is possible to add cardinality constraints to the formulation to address such concerns, such an approach brings additional computational difficulty. We do not discuss cardinality constraints here in the risk parity context and refer interested readers to [13] for more details. Imposing risk parity at a group level instead of at the individual asset level can also address this issue implicitly and can produce "sparse" solutions with fewer holdings.

For group risk parity, we solve the following nonconvex problem:

$$\begin{aligned} \min_{x,\theta} \quad & \sum_{j=1}^{l} \Big(\sum_{i \in \mathcal{G}_j} x_i (\Sigma x)_i - \theta\Big)^2 \\ \text{s.t.} \quad & a_i \leq x_i \leq b_i \\ & \sum_{i=1}^{n} x_i = 1, \end{aligned} \tag{4.29}$$

where $\mathcal{G}_j$ stands for the jth group, and l is the total number of groups. Here we make two assumptions: 1) we invest in all groups (i.e., we do not aim to pursue group sparsity); 2) there is no overlap (i.e., each asset can only lie in one of the groups). Risk parity between groups is achieved if the optimal value of the objective function is zero. In Section 4.6 we show how group risk parity may produce a desirable portfolio using sector membership for grouping different assets in an equity portfolio. In the next section we introduce an efficient algorithm that can handle problem (4.17), as well as the variations (4.28) and (4.29).

4.5 Algorithms solving second order least-square problems

In this section, we briefly introduce an algorithm for solving risk parity optimization problem. Details of this algorithm and others for this class of problems can be found in Chapter 5, including convergence analysis and additional numerical results. This algorithm was initially inspired by an alternating linearization algorithm in [34]. However, here it is applied to a nonconvex problem and in a substantially different setting.

Consider the following optimization problem:

$$\min_{x \in \mathcal{X}, \theta} F(x) = \sum_i ((A_i x)^\top (B_i x) - \theta)^2, \tag{4.30}$$

where $x \in \mathbb{R}^n$, $A_i, B_i \in \mathbb{R}^{m \times n}$ and $\mathcal{X}$ is a set defined by linear constraints.

Note that the risk parity models we discussed in the previous sections can be embedded into this formulation. The objective function of risk parity problem (4.17) fits the formulation of (4.30) by setting $A_i = \Sigma_i \in \mathbb{R}^{1 \times n}$ as the ith row of the covariance matrix, and $B_i = e_i \in \mathbb{R}^{1 \times n}$ as the ith column of the identity matrix. In case of (4.29), $A_j \in \mathbb{R}^{m_j \times n}$ is defined by a submatrix of Σ which correspond to rows with indices from set $\mathcal{G}_j$. $B_j \in \mathbb{R}^{m_j \times n}$ is defined as follows: suppose the ith row of A_j corresponds to the (k_i)th row of Σ, then ith row of B_j corresponds to the (k_i)th row of the identity matrix:

$$(B_j)_{i,k} = \begin{cases} 1, & k = k_i \\ 0, & otherwise. \end{cases}$$

Note that (4.30) is equivalent to

$$\min_{x\in\mathcal{X},\theta} F(x,\theta) = \sum_{i=1}^{n} F_i(x) = \sum_{i=1}^{n}(x^\top M_i x - \theta)^2, \tag{4.31}$$

where $M_i = A_i^\top B_i \in \mathbb{R}^{n\times n}$. Clearly M_i is not generally symmetric or positive semidefinite. Hence (5.2) is a nonconvex optimization problem. We consider a variable splitting approach which replaces $F(x,\theta)$ by $F(x,y,\theta) = \sum_{i=1}^{n}(x^\top M_i y - \theta)^2$, s.t. $y = x$. For brevity, let us omit θ from the variables of F and use $F(x,y)$. Our method generates two sequences $\{x^k\}$ and $\{y^k\}$ in such a way that $x^k \to x^*$ and/or $y^k \to x^*$ where x^* is a local optimal solution of (4.30).

Given y^k

$$F(x,y^k) \equiv \sum_{i=1}^{n}(x^\top M_i y^k - \theta)^2, \tag{4.32}$$

and given x^k we have

$$F(x^k,y) \equiv \sum_{i=1}^{n}((x^k)^\top M_i y - \theta)^2, \tag{4.33}$$

Both $F(x,y^k)$ and $F(x^k,y)$ are convex functions of x and y, respectively, for any given y^k and x^k. Let $\nabla_i F$ denote the partial derivative of F with respect to either x $(i=1)$ or y $(i=2)$. In particular, using the form of (5.2), we have

$$\begin{aligned} \nabla_1 F(x,y) &= \sum_{i=1}^{n} 2(x^\top M_i y - \theta) M_i y \\ \nabla_2 F(x,y) &= \sum_{i=1}^{n} 2(x^\top M_i y - \theta) M_i^\top x. \end{aligned} \tag{4.34}$$

We now construct the following approximations of $F(x, y)$:

$$\begin{aligned} Q^1_\mu(x, y^k) &\triangleq F(x, y^k) + \left\langle \nabla_2 F(y^k, y^k), x - y^k \right\rangle + \frac{1}{2\mu}\|x - y^k\|_2^2 \\ Q^2_\mu(x^{k+1}, y) &\triangleq F(x^{k+1}, y) + \left\langle \nabla_1 F(x^{k+1}, x^{k+1}), y - x^{k+1} \right\rangle + \frac{1}{2\mu}\|x^{k+1} - y\|_2^2, \end{aligned} \tag{4.35}$$

where μ is some chosen positive scalar. In Chapter 5 the simple version of our ALM algorithm, shown in Algorithm 14.

As already discussed in previous chapters, in practice, a backtracking strategy can be applied to choose values of parameters μ at each iteration. A practical backtracking scheme is shown in Algorithm 15. Note that, in each minimization step we check whether a sufficient reduction has been obtained. If so, the minimization step is accepted and μ may be increased, otherwise it is decreased and a new candidate step is computed. Each minimization step in Algorithms 14 and 15 is a solution of a strictly convex quadratic programming problem, which can be obtained efficiently by existing methods and software. We will discuss the implemented method in the next chapter.

4.6 Numerical results

4.6.1 A comparison between strategies

We compare several asset allocation strategies on a small data set to illustrate the benefits of the risk parity strategy. Consider the following example with 5 assets and the covariance matrix of percentage returns (returns multiplied by 100) given by:

Table 4.2: A comparison of strategies, with the lower and upper bounds set to $a = 0, b = 1$.

Items for comparison	x_i	RC_i	Volatility
1/n rule	[0.200;0.200;0.200;0.200;0.200]	[0.119;0.524;0.219;-0.002;0.139]	7.07%
Minimum variance portfolio	[0.050;0.006;0.000;0.862;0.082]	[0.050;0.006;0.000;0.862;0.082]	2.16%
Risk parity portfolio	[0.125;0.047;0.083;0.613;0.132]	[0.200;0.200;0.200;0.200;0.200]	3.04%

$$\Sigma = \begin{bmatrix} 94.868 & 33.750 & 12.325 & -1.178 & 8.778 \\ 33.750 & 445.642 & 98.955 & -7.901 & 84.954 \\ 12.325 & 98.955 & 117.265 & 0.503 & 45.184 \\ -1.178 & -7.901 & 0.503 & 5.460 & 1.057 \\ 8.778 & 84.954 & 45.184 & 1.057 & 34.126 \end{bmatrix}.$$

The covariance matrix above suggests that Asset 4 is a low risk asset while Asset 2 is a high risk asset. The other assets are in the middle of the risk spectrum. First, we consider the long-only risk parity case, when the lower and upper bound of the weights are set to be 0 and 1, respectively. As we have proved in Lemma 4.2.2, in this case, there is a unique risk parity solution.

Table 4.2 shows a comparison of different strategies in terms of volatility and risk concentration. We compare three strategies: $1/n$ rule, the minimum variance portfolio, and the risk parity portfolio. The relative risk contribution of asset i, denoted by RC_i, is calculated as

$$RC_i := \frac{x_i(\Sigma x)_i}{x^\top \Sigma x}.$$

By construction, the minimum variance strategy has the least volatility among these three strategies. However, the distribution of the risk contributions for the minimum variance portfolio is very skewed — more than 86% of the total risk comes from

Table 4.3: A comparison of strategies, with the lower and upper bounds set to $a = 0.05, b = 0.35$.

Items for comparison	x_i	RC_i	Volatility
1/n rule	[0.200;0.200;0.200;0.200;0.200]	[0.119;0.524;0.219;-0.002;0.139]	7.07%
Minimum variance portfolio	[0.200;0.050;0.050;0.350;0.350]	[0.280;0.178;0.086;0.034;0.421]	4.13%
Optimal parity portfolio	[0.204;0.060;0.130;0.350;0.256]	[0.256;0.198;0.234;0.027;0.284]	4.44%

Asset 4. From Table 4.2 it can be observed that risk parity strategy also puts a high weight on Asset 4, but has equal risk contribution from each asset by construction. Further, the volatility of the risk parity portfolio is between that of the minimum variance portfolio and the equally weighted portfolio, which indicates that risk parity could be viewed as a compromise between these two strategies.

If there are binding constraints on asset weights a risk parity portfolio may not exist, as discussed in previous sections. For instance, if we change the bounds to be $a_i = 0.05, b_i = 0.35,\ i = 1, \ldots, 5$, there is no risk parity solution. Due to its much lower risk profile, Asset 4 requires a much higher weight than other assets to match their risk contributions. Since the 0.35 upper bound prevents this, risk parity cannot be achieved. Instead, we seek approximate risk parity by solving (4.17).

Let us call the optimal solution to (4.17) the "optimal parity" solution. From Table 4.3, we observe that the risk contribution of Asset 4 in the optimal parity portfolio is lower than other assets and the resulting excess is shared roughly evenly among three of the four remaining assets. No asset has risk contribution more than 30% in the optimal parity portfolio.

Next, let us consider the long-short case by removing the bounds on asset weights. In Lemma 4.2.3, we showed that for the long-short risk parity problem with n assets, there could be as many as 2^{n-1} solutions. In our example, by solving (4.13) for all orthants, we obtain $2^4 = 16$ solutions. We enumerate these solutions in Table 4.4.

We also list the orthant that each solution lies in.

Finally, we compare the long-short minimum variance portfolio to the minimum variance risk parity portfolio. We include the equally weighted portfolio in the comparison, shown in Table 4.5, for completeness. As discussed in Section 4.1, the minimum variance risk parity portfolio can be found by applying Algorithm 11. The solution, denoted as Risk parity portfolio* in Table 4.5, provides a favorable tradeoff between lowering portfolio risk and balancing risk contributions from all assets compared to the other strategies. Further, by examining Table 4.4, we verify that Risk parity portfolio* is indeed the minimum variance risk parity portfolio.

4.6.2 Strategic asset allocation

In this subsection, we compare four different static investment strategies using real-world data.

We consider the following common market indices to represent different asset classes: S&P 500, MSCI World (Net), Russell 2500, Russell 2000 Growth, Russell 2000 Value, HFRI Equity Hedged Index, MSCI Emerging Markets (Net), HFRI Emerging MKTS Total, HFRI FoF (Conservative Index), BC Treasury 5-10 Yr, BC US Corporate High Yield Index, JPMorgan GBI-EM Index, JPMorgan EMBI+ Index, S&P Global Natural Resources - Energy Index. Here, we use a monthly sampling frequency, and the sampling period is from Nov. 2002 to Aug. 2012. We use the returns of each one of these asset classes during the sampling period to estimate their annualized expected returns and volatilities. For excess return and Sharpe ratio calculations we estimate the risk-free rate from the 3-month T-bill rate which averages approximately 1.8% annually during the sampling period.

Table 4.4: A comparison of long-short risk parity solutions by enumeration.

Items for comparison	x_i	Volatility	Orthant	Complementary orthant
Risk parity portfolio (1)	[0.125;0.047;0.083;0.613;0.132]	3.04%	$[+;+;+;+;+]$	$[-;-;-;-;-]$
Risk parity portfolio (2)	[-0.223;0.074;0.125;0.820;0.204]	4.26%	$[-;+;+;+;+]$	$[+;-;-;-;-]$
Risk parity portfolio (3)	[0.157;-0.128;0.124;0.538;0.309]	3.34%	$[+;-;+;+;+]$	$[-;+;-;-;-]$
Risk parity portfolio (4)	[0.154;0.073;-0.285;0.717;0.340]	3.48%	$[+;+;-;+;+]$	$[-;-;+;-;-]$
Risk parity portfolio (5)	[0.165;-0.118;-0.255;0.537;0.671]	3.38%	$[+;-;-;+;+]$	$[-;+;+;-;-]$
Risk parity portfolio (6)	[-0.700;-0.233;-0.503;3.298;-0.863]	17.94%	$[-;-;-;+;-]$	$[+;+;+;-;+]$
Risk parity portfolio (7)	[0.272;0.247;0.450;1.495;-1.464]	5.84%	$[+;+;+;+;-]$	$[-;-;-;-;+]$
Risk parity portfolio (8)	[-0.216;-0.162;0.182;0.726;0.471]	4.70%	$[-;-;+;+;+]$	$[+;+;-;-;-]$
Risk parity portfolio (9)	[-0.245;0.117;-0.393;0.985;0.536]	5.01%	$[-;+;-;+;+]$	$[+;-;+;-;-]$
Risk parity portfolio (10)	[0.366;-0.097;-0.201;1.289;-0.356]	6.67%	$[+;-;-;+;-]$	$[-;+;+;-;+]$
Risk parity portfolio (11)	[-0.553;0.541;0.934;2.902;-2.824]	11.87%	$[-;+;+;+;-]$	$[+;-;-;-;+]$
Risk parity portfolio (12)	[-0.513;0.538;-0.426;2.707;-1.307]	11.36%	$[-;+;-;+;-]$	$[+;-;+;-;+]$
Risk parity portfolio (13)	[0.287;-0.103;0.441;1.072;-0.696]	5.58%	$[+;-;+;+;-]$	$[-;+;-;-;+]$
Risk parity portfolio (14)	[0.272;0.245;-0.220;1.403;-0.700]	5.65%	$[+;+;-;+;-]$	$[-;-;+;-;+]$
Risk parity portfolio (15)	[-0.457;-0.183;0.895;1.979;-1.234]	10.87%	$[-;-;+;+;-]$	$[+;+;-;-;+]$
Risk parity portfolio (16)	[-0.208;-0.150;-0.346;0.711;0.992]	4.69%	$[-;-;-;+;+]$	$[+;+;+;-;-]$

Table 4.5: A comparison of strategies, without any bounds on asset weights

Items for comparison	x_i	RC_i	Volatility
1/n rule	[0.200;0.200;0.200;0.200;0.200]	[0.119;0.524;0.219;-0.002;0.139]	7.07%
Minimum variance portfolio	[0.050;0.006;-0.012;0.856;0.100]	[0.050;0.006;-0.012;0.856;0.100]	2.16%
Risk parity portfolio*	[0.125;0.047;0.083;0.613;0.132]	[0.200;0.200;0.200;0.200;0.200]	3.04%

We compare the following four strategies: 60/40 rule (allocated between S&P 500 and BC Treasury 5-10 Yr indices, representing equity and fixed income investments), $1/n$ rule, the minimum variance portfolio, and the risk parity portfolio. For each strategy, the weights and risk contributions of different asset classes are shown in Figure 4.2 and 4.3, respectively. For each one of the resulting portfolios, we compare the expected excess return, volatility, Sharpe ratio, 5% nonparametric Value-at-Risk of these strategies. In addition, we compare two risk concentration metrics, namely the HRC and the Herfindahl index that are defined by (4.19) and (4.20), respectively.

It can be observed that for both of 60/40 rule and the equally weighted portfolio, the volatility and 5% VaR are high compared to the minimum variance and risk parity portfolios. The risk parity portfolio achieves the highest Sharpe ratio among the four strategies. In terms of risk concentration, the traditional 60/40 rule is dominated by equity risk (more than 95%), as expected. As in the example of the previous section, the minimum variance portfolio, also has a high risk concentration: almost two thirds of the risk is contributed by the least risky asset, namely the HFRI FoF Conservative Index. This can be clearly observed from Figure 4.3, showing risk contribution of different assets. Overall, the risk parity portfolio provides a good compromise between balancing risk contributions and achieving reasonable amount of expected returns.

Table 4.6: A comparison of strategies on asset allocation example, with the lower and upper bounds set to $a = 0, b = 1$.

Items for comparison	Expected excess return	Volatility	Sharpe Ratio	VaR 5%	Highest RC	Herfindahl index
60/40 rule	4.33%	9.36%	0.462	4.43%	95.49%	0.8783
1/n rule	8.59%	12.39%	0.693	5.80%	13.12%	0.0900
Minimum variance portfolio	2.59%	3.55%	0.730	1.17%	61.79%	0.5278
Risk parity portfolio	6.57%	7.72%	0.851	3.13%	7.14%	0.0714

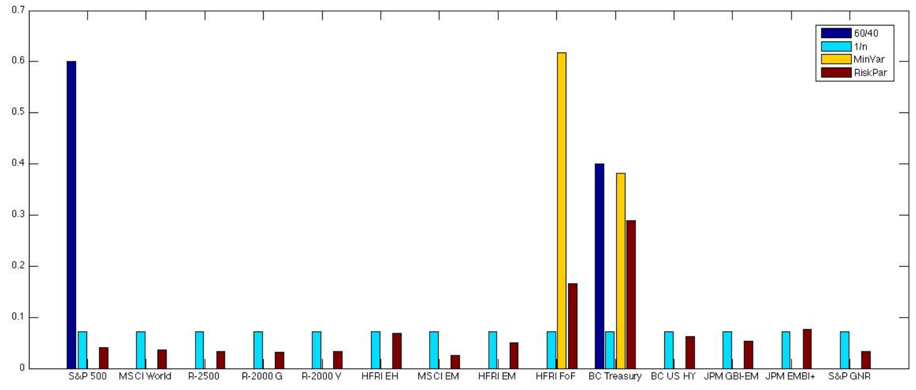


Figure 4.2: Weights of different asset classes in the asset allocation example

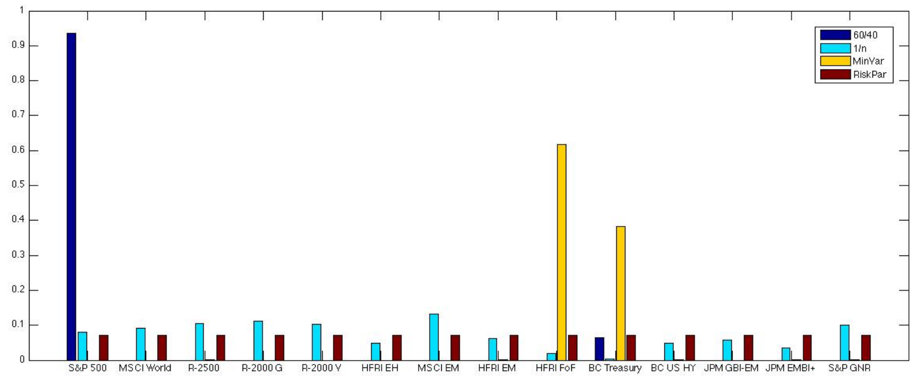


Figure 4.3: Risk contribution of different asset classes in the asset allocation example

4.6.3 US equity sector allocation

In this section, we consider a sector allocation problem for US equities. We perform a simulation over a 40-year period with annual rebalances and compare the performance of three different strategies. As in Section 4.6.1, we consider equal weighting, minimum variance and risk parity strategies. Similar experiments or simulated examples can be found in, for instance, [19, 48].

We consider 17 sectors representing the US equity universe (food, mines, oil, clths, durbl, chems cnsum, cnstr, steel, fabpr, machn, cars, trans, utils, rtail finan, other) as detailed in Kenneth French's data library.[1] We also use return data provided in this library.

We run simulations using monthly sector returns from Oct. 1972 to Sep. 2012. Since the sampling period is relatively long, we do not assume that input parameters to portfolio construction strategies are constant over time and we take into consideration the time dependency of the return and risk parameters. For each year, we estimate the forward-looking expected returns and covariance matrix based on the monthly returns of the previous j years, with $j = 3, 5$ and 10 (we do not generate results for the first j years). The risk-free rate is assumed to be the 3-month T-bill rate. Meanwhile, considering the rebalancing costs, we apply annual rebalancing. The average annualized excess return, volatility, Sharpe ratio and risk contribution are reported. We also plot the excess return and cumulative excess return over time in Figures 4.4 - 4.6.

It can be observed from Tables 4.7, 4.8 and 4.9 that minimum variance portfolio has less volatility (and sometimes enjoys a slightly higher Sharpe ratio) than the other two strategies, while its risk concentration is much higher as well. As a comparison,

[1]Details are available at http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/index.html

Table 4.7: A comparison of strategies for US sector allocation (3 yr rolling windows)

Items for comparison	Excess return	Volatility	Sharpe Ratio	Highest RC	Herfindahl index
1/n rule	8.04%	15.30%	0.525	9.76%	0.0674
Minimum variance portfolio	6.39%	11.67%	0.547	51.79%	0.4089
Risk parity portfolio	8.59%	13.77%	0.624	14.72%	0.2034

Table 4.8: A comparison of strategies for US sector allocation (5 yr rolling windows)

Items for comparison	Excess return	Volatility	Sharpe Ratio	Highest RC	Herfindahl index
1/n rule	7.97%	15.46%	0.516	9.83%	0.0676
Minimum variance portfolio	6.65%	11.66%	0.570	49.67%	0.3738
Risk parity portfolio	7.08%	14.11%	0.507	14.11%	0.1329

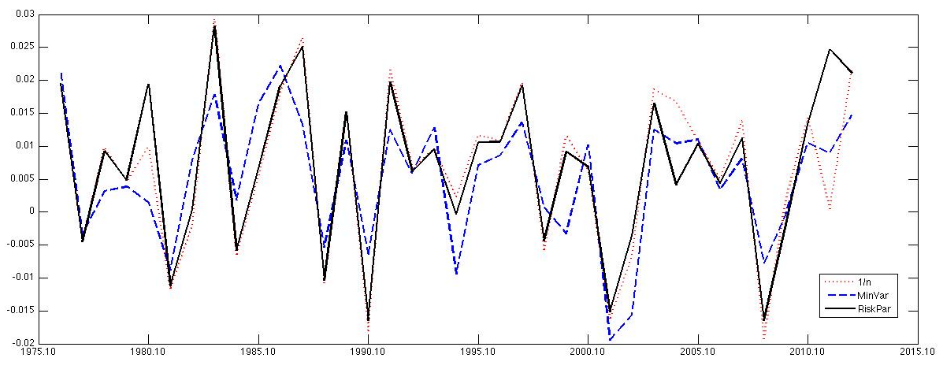
very often the volatility of risk parity portfolio lies between the other two, which once again shows that risk parity is a compromise between equal weighting and minimum variance approaches. Moreover, the risk parity strategy realizes higher returns than the minimum variance strategy under all three moving window lengths considered, as illustrated in Figures 4.4, 4.5, and 4.6.

4.6.4 Group risk parity portfolios based on the US large cap universe

In this section, we study a stock selection strategy on the US large cap equities as defined by the components of the S&P 500 index. We build our return and covariance estimates using daily returns of S&P 500 constituents, from Aug. 21, 2009 to Aug.

Table 4.9: A comparison of strategies for US sector allocation (10 yr rolling windows)

Items for comparison	Excess return	Volatility	Sharpe Ratio	Highest RC	Herfindahl index
1/n rule	8.90%	15.15%	0.588	9.98%	0.0684
Minimum variance portfolio	7.59%	11.39%	0.667	54.90%	0.4072
Risk parity portfolio	9.57%	14.54%	0.658	16.11%	0.1542



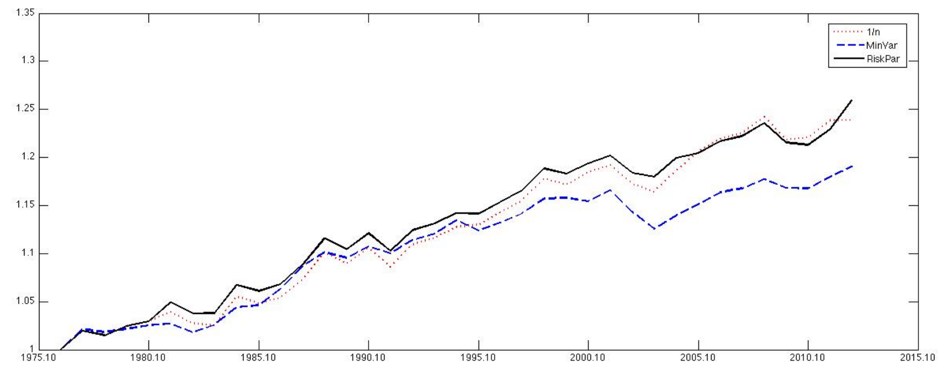


Figure 4.4: Annual and cumulative returns for US sector allocation strategies (3 yr rolling windows)

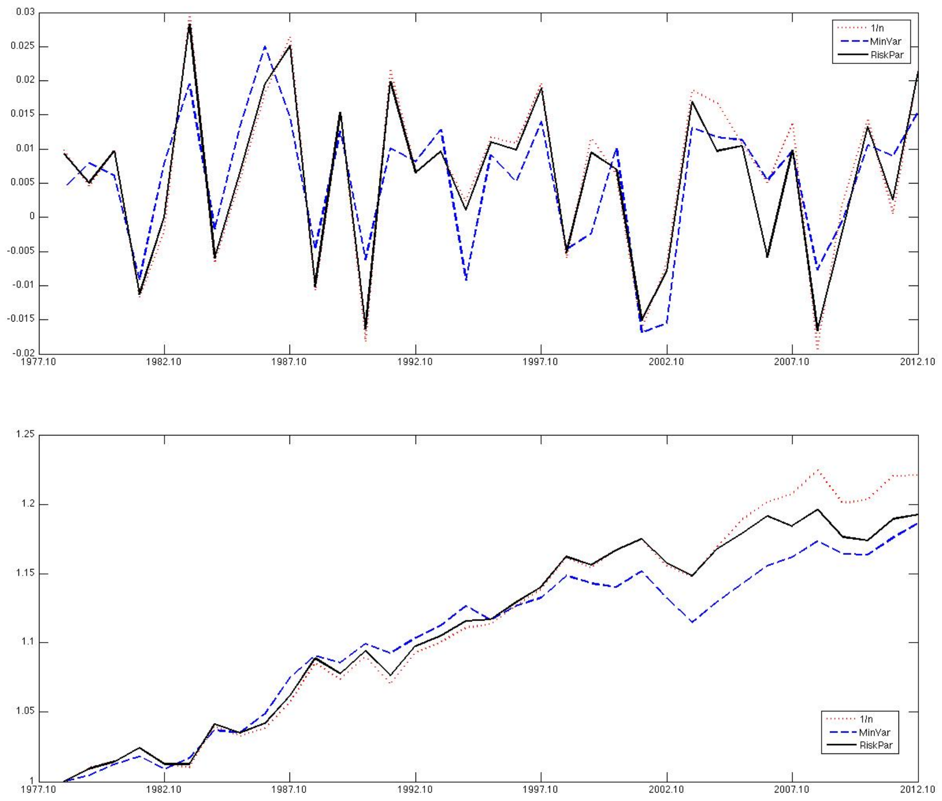


Figure 4.5: Annual and cumulative returns for US sector allocation strategies (5 yr rolling windows)

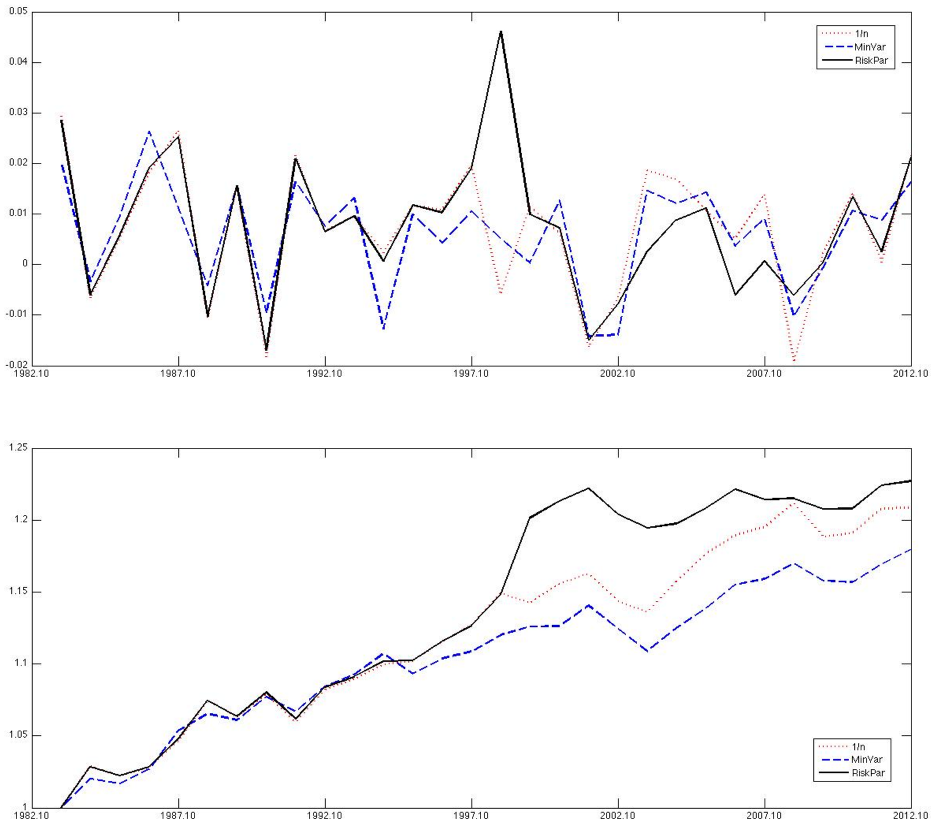


Figure 4.6: Annual and cumulative returns for US sector allocation strategies (10 yr rolling windows)

20, 2010 (see: http://pages.swcp.com/stocks/). We ignore those stocks with less than 200 trading days of data. In total, there are 482 stocks and 245 trading days considered. We now compare the simple "1/n" and minimum variance strategy with group risk parity over this universe.

The groups are determined by Global Industry Classification Standard (GICS Sector).[2] Using the GICS classification, we partition the S&P 500 stocks into ten sectors: Consumer Discretionary, Consumer Staples, Energy, Financials, Health Care, Industrials, Information Technology, Materials, Telecommunication Services and Utilities. Note that there is no overlap between groups, which means that each stock belongs to one and only one group.

Since the data is sampled from the time period of only one year, we ignore the time dependency of the parameters when testing the performance of different strategies. Here we have 244 daily returns in the sample data. We divide this data into two groups: 163 days for training, and the remaining 81 days for testing. We estimate expected excess returns and covariances using the training data, build a portfolio using various strategies using these estimates, and monitor the performance of the resulting portfolios, out-of-sample, during the testing period.

The results are shown in Table 4.10. The highest group risk contribution is defined as $HGRC := \max_j \frac{\sum_{i \in \mathcal{G}_j} x_i(\Sigma x)_i}{x^\top \Sigma x}$. Also, the group Herfindahl index is defined as

$$h_{\mathcal{G}} = \sum_{j=1}^{p} \left[\frac{\sum_{i \in \mathcal{G}_j} x_i(\Sigma x)_i}{x^\top \Sigma x} \right]^2 .$$

Since our tests use the validation data, which is different from our training data, we do not expect to achieve perfect risk parity during the testing period. But, as can be

[2]The standard can be accessed at: http://en.wikipedia.org/wiki/List_of_S%26P_500_companies.

Table 4.10: A comparison of strategies on the US large cap universe, with the lower and upper bounds to be $a = 0, b = 1$.

Items	Excess return	Volatility	Highest RC	Highest Group RC	Group Herfindahl index
1/n rule	-27.60%	28.93%	3.94%	23.81%	0.142
Min. var.	-26.03%	18.54%	40.40%	65.10%	0.449
Group risk par.	-19.96%	23.46%	2.07%	11.13%	0.101

seen from Table 4.10, the group Herfindahl index for the group risk parity strategy is not far from the perfect one (which is $1/10 = 0.1$ since we have 10 groups). As a comparison, the portfolio using 1/n rule has worse excess return (note that in this particular testing period, we have a down market) and higher volatility, and thus is dominated. Moreover, minimum variance portfolio, while having the least volatility, suffers from risk concentration both on stock level and group level. Its highest group risk contribution is 65.10%: more than half of total risk lies in the group "Consumer Staples" (this can be clearly seen in Figure 4.7). Furthermore, at the individual stock level, the maximum risk contribution is more than 40 percent (General Mills - 40.40%). This is a very high concentration of risk given that the investment universe includes hundreds of securities.

4.6.5 Long-only vs. long-short portfolios

One of the main contributions of this chapter is its introduction of strategies for determining long-short risk parity portfolios. In this section, we provide an example to compare long-only portfolios with long-short ones and illustrate that allowing short positions in a risk parity portfolio provides significant opportunities for reducing volatility and Value-at-Risk while preserving the equal risk contribution property.

Depending on investors' view on the future market trends, three types of long-short portfolios can be considered: they may have a long-side bias, as in the case of

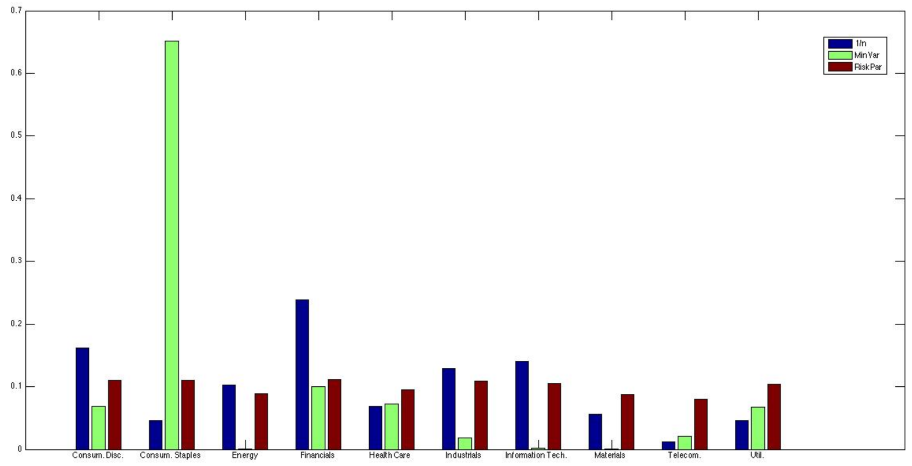


Figure 4.7: Risk contribution of different groups based on testing data

popular 130/30 strategies; they may have a short-side bias; or, they may be market-neutral. In our discussion in Section 2, we focused on long-short portfolios with a long-side bias. We proved that risk parity solutions exist in this case and furthermore there could exist as many as 2^{n-1} solutions if we enforce a full-investment constraint.

Risk parity portfolios with a short-side bias can be regarded as complementary counterparts to those with long-side biases and can be found by solving the following problem:

$$\begin{aligned} & x_i \cdot \frac{\partial \sigma}{\partial x_i} = x_j \cdot \frac{\partial \sigma}{\partial x_j}, \forall i, j \\ & \sum_{i=1}^{n} x_i = -1. \end{aligned} \tag{4.36}$$

Clearly, a solution of problem (4.36) can be generated by solving (4.7) and flipping the sign of each resulting weight.

In contrast, the case of risk parity under market neutrality is more complicated, and it is not clear whether there exists a risk parity solution beyond the trivial one (and, if yes, how many there are). In this chapter, we only consider an illustrative

Table 4.11: A comparison of long-only and long-short strategies on asset allocation example

Items for comparison	Bounds	Volatility	VaR 5%	Highest RC	Herfindahl index
Long-only min var portfolio	[0,1]	3.55%	1.17%	61.79%	0.5278
Long-short min var portfolio	[-1,2]	2.16%	1.00%	82.93%	1.6200
Long-only risk par portfolio	[0,1]	7.72%	3.13%	7.14%	0.0714
Long-short risk par portfolio	[-1,2]	2.57%	1.06%	7.14%	0.0714

asset allocation example with a long-side bias. Market neutral risk parity portfolios will be considered as a subject for future research.

In Table 4.11 we show a comparison between long-only and long-short strategies on the asset allocation example introduced in Section 4.6.2. We omit the results for 60/40 rule and 1/n rule since we reported on them earlier. Here, the long-short risk parity portfolio is computed by applying Algorithm 11. The long-short risk parity portfolio has much smaller volatility and VaR when compared to the long-only risk parity portfolio. For comparison, the weights and risk contributions of each asset are shown in Figures 4.8 and 4.9, respectively.

It can be observed that, for the long-short minimum variance portfolio, the volatility is less than its long-only counterpart, as expected from the expansion of the feasible region. However, its highest risk contribution and Herfindahl index are also larger, which indicates higher risk concentration. On the other hand, the long-short risk parity portfolio greatly reduces the volatility as compared with the long-only one, without causing any increase in either of the concentration metrics.

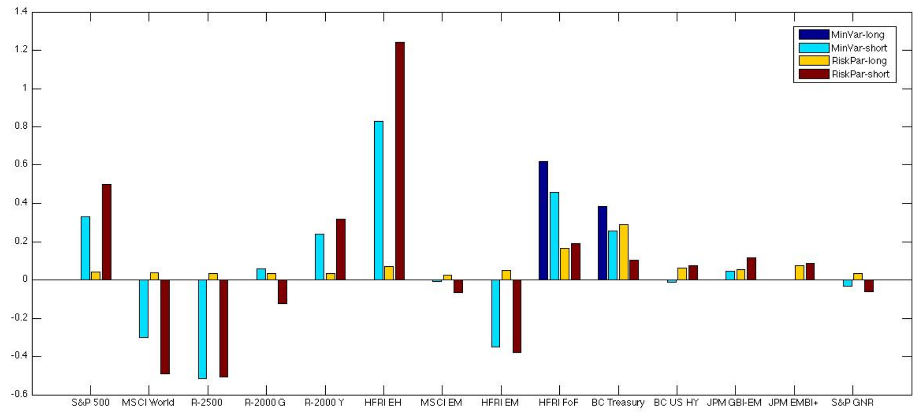


Figure 4.8: Long-only vs. long-short portfolios: weights of asset classes in asset allocation example

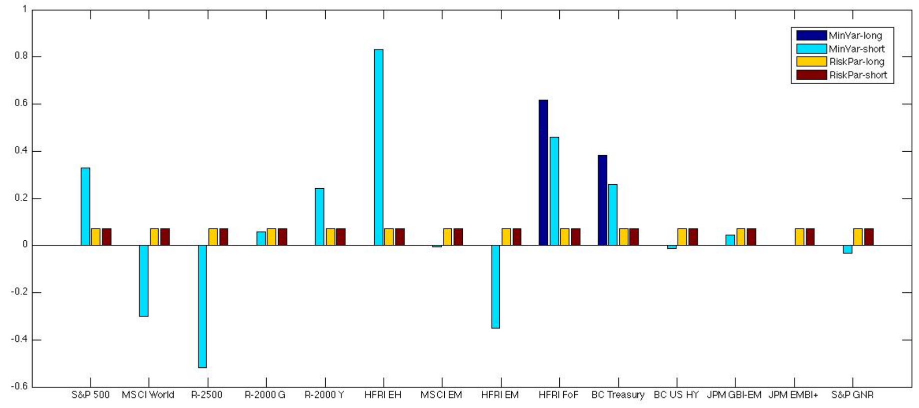


Figure 4.9: Long-only vs. long-short portfolios: risk contributions of asset classes in asset allocation example

4.6.6 Efficiency of algorithms

While this chapter is not aimed at algorithmic details, here we briefly discuss the efficiency of the algorithms shown in Section 4.5, in application to the instances discussed so far in this section. Further discussion on algorithms and corresponding numerical experiments are provided in [4].

Our implementation is written in MATLAB and experiments were performed using MATLAB 2013a on a laptop with Intel Core i5 1.8 GHz CPU and 4GB RAM. We apply Mosek 7.0 to solve the QP subproblems in Algorithm 15, when optimizing Q_1 and Q_2.

In Table 4.12, we compare the efficiency of the proposed algorithm on different data sets. A random symmetric positive definite matrix can be generated from $\Sigma = AA^\top$, where A_{ij} is uniformly distributed within the interval $[0, 1]$. Recall that the main computational cost of each algorithm lies in the number of quadratic models it solves as the subproblem. Hence, here we report the number of iterations and the total number of QPs solved. We compare the performance of the algorithms by recording the number of iterations they took until the largest KKT violation (in absolute value) fell below 10^{-4} and 10^{-6}. It can be seen that, after the largest KKT violation is less than 10^{-6}, in most cases the function value is less than 10^{-9}, which indicates that an approximate risk-parity solution has been obtained. In the scenario with tighter bounds, the objective function does not fall below 10^{-7} because there is no risk parity solution.

As an alternative to ALM, optimization methods from MATLAB optimization toolbox can be used to solve (4.17). However, our results indicate that these methods are unreliable. For instance, for the 14-asset strategic asset allocation example from

Table 4.12: A comparison of ALM with backtracking on different instances. The starting point is chosen to be the equally weighted portfolio, i.e., $x_i^0 = 1/n$. Due to the scaling of the data, we chose a large initial μ. The number of iterations (k), the number of QPs solved, and the objective function value (F-value) are compared. ε is the threshold for the largest KKT violation.

Instance (size)	Starting μ	Bounds	k ($\varepsilon = 10^{-4}$)	QP	F-value
Random I (20)	0.01	$a = 0; b = 1$	6	14	1.01×10^{-11}
Random II (200)	0.0001	$a = 0; b = 1$	5	12	1.59×10^{-12}
5 assets example (5)	1000	$a = 0; b = 1$	1	6	1.21×10^{-6}
5 assets example (5)	1000	$a = 0.05; b = 0.35$	1	6	1.21×10^{-6}
5 assets example (5)	1000	$a = -1; b = 2$	1	6	1.21×10^{-6}
Asset allocation (14)	10000	$a = 0; b = 1$	1	2	2.68×10^{-8}
US equity (482)	1000	$a = 0; b = 1$	1	2	1.11×10^{-11}
Instance (size)	Starting μ	Bounds	k ($\varepsilon = 10^{-6}$)	QP	F-value
Random I (20)	0.01	$a = 0; b = 1$	7	17	8.86×10^{-15}
Random II (200)	0.0001	$a = 0; b = 1$	7	18	6.58×10^{-18}
5 assets example (5)	1000	$a = 0; b = 1$	11	16	1.01×10^{-9}
5 assets example (5)	1000	$a = 0.05; b = 0.35$	6	20	1.63×10^{-7}
5 assets example (5)	1000	$a = -1; b = 2$	11	16	1.01×10^{-9}
Asset allocation (14)	10000	$a = 0; b = 1$	6	12	3.89×10^{-9}
US equity (482)	1000	$a = 0; b = 1$	2	4	1.10×10^{-11}

Section 4.6.2, we can solve the generalized risk parity problem (4.17) using MATLAB function *fmincon* using two algorithms - SQP and Interior Point. We set the function tolerance (which controls both the size of the latest change in the objective function value and the first-order optimality measure) as 10^{-8} and the maximum function evaluation number as 10000. Table 5.9 compares the results generated by our ALM with the results obtained by MATLAB's *fmincon*. We observe that, in this example, *fmincon*-SQP completely fails, as it does not progress from the starting point. *fmincon*-interior point performs better but still not nearly as well as our algorithm — it takes longer to find a far worse solution than ALM. Our ALM algorithm finds a solution which is close to optimum by taking only 12 iterations and by solving 26 QPs. This result shows the efficiency of the algorithm we proposed.

We also compare ALM with other methods to solve the risk parity optimization

Table 4.13: A comparison of algorithms on the 14-asset strategic asset allocation instance in Section 4.6.2. The starting point is chosen to be the equally weighted portfolio, i.e., $x_i^0 = 1/n$.

Algorithm	ALM	$fmincon$ - SQP	$fmincon$ - interior point
$x_1 \sim x_7$	[0.043 0.038 0.035 0.033 0.035 0.069 0.027]	[0.071 0.071 0.071 0.071 0.071 0.071 0.071]	[0.057 0.052 0.047 0.045 0.047 0.090 0.038]
$x_8 \sim x_{14}$	[0.052 0.170 0.259 0.066 0.056 0.083 0.035]	[0.071 0.071 0.071 0.071 0.071 0.071 0.071]	[0.069 0.122 0.116 0.087 0.074 0.109 0.047]
$RC_1 \sim RC_7$	[0.072 0.073 0.073 0.073 0.073 0.070 0.075]	[0.081 0.092 0.104 0.111 0.104 0.048 0.131]	[0.078 0.081 0.082 0.084 0.082 0.074 0.086]
$RC_8 \sim RC_{14}$	[0.072 0.071 0.057 0.072 0.073 0.074 0.073]	[0.063 0.019 0.004 0.049 0.057 0.036 0.101]	[0.075 0.041 0.013 0.076 0.076 0.072 0.081]
Success/Failure	Success	Failure	Partial success
CPU time (s)	0.047	-	0.457

Table 4.14: A CPU time comparison (in seconds) of ALM and Spinu's Newton method for solving instances of various sizes. The starting point is chosen to be the equally weighted portfolio, i.e., $x_i^0 = 1/n$.

Thresholds	10^{-3}		10^{-6}		10^{-9}	
Instance size	ALM	Newton	ALM	Newton	ALM	Newton
5×5	0.0146	0.0005	0.0188	0.0005	0.0210	0.0005
10×10	0.0188	0.0006	0.0188	0.0006	0.0213	0.0006
20×20	0.0185	0.0008	0.0185	0.0008	0.0214	0.0008
200×200	0.2597	0.0275	0.2597	0.0275	0.2597	0.0275
1000×1000	6.4051	2.2817	8.1324	2.2817	8.1324	2.2817

problem. In particular, [63] applies a damped Newton method to solve the long-only risk parity problem (4.8) — the log-barrier risk parity model. [63] shows that the log-barrier function is self-concordant and (4.8) can be efficiently solved using convex optimization. In contrast, our generalized method solves (4.17) — a nonconex quartic which is potentially more difficult. Here we test both methods on randomly generated covariance matrices with different sizes. Table 4.14 compares the efficiency in terms of CPU time required to reduce the chosen deviation measure below a certain threshold value. Here, we choose the deviation measure to be $\sum_{i=1}^{n}(\frac{x_i(\Sigma x)_i}{x^\top \Sigma x} - \frac{1}{n})^2$, and thresholds to be $10^{-3}, 10^{-6}$ and 10^{-9}. Note that, for both methods, the corresponding optimal solution has a deviation measure zero when no bounds are enforced. As shown in Table 5.9, Newton method is faster than the ALM approach in all the instances we tested. This is not surprising since, seeking to solve a more general problem, our model is computationally more complex. Further, it does not utilize the full second-order information, as shown in [4], because we want to ensure that subproblems are convex. As seen in Section 2.3, the log-barrier approach does not directly extend to the cases when general bounds are added and thus we here only compare the long-only full investment case. These results indicate that, when no customized bounds are needed, the convex model is more efficient but its application is limited compared to our general framework, which is reliable and comparatively efficient as well.

4.7 Conclusion

In this chapter, we discuss the problem of finding portfolios that satisfy risk parity of either individual assets or groups of assets as closely as possible. We analyze the limitations of the convex optimization approach, which was proposed in prior literature. We then propose an alternative nonconvex least-squares model whose set of optimal solutions includes all risk parity solutions. We also propose a modified formulation which aims at selecting the most desirable risk parity solution according to some predetermined criteria. Our model has many advantages, especially when general bounds are considered or when other constraints are added. Furthermore, we propose an alternating linearization framework to solve this nonconvex model. Numerical experiments indicate the effectiveness of our technique. Extensions of our models and methods to broader domains remain a topic of future research.

Chapter 5

Alternating direction schemes for minimizing a nonconvex objective that is *not* necessarily composite and its application in second-order least-squares

5.1 Introduction

Minimizing a sum of squares is a classical approach to finding an approximate solution to (possibly) overdetermined systems of equations. In this chapter we are interested in the case when the equations are quadratic. Specifically, consider solving the following system of quadratic equations:

$$x^\top M_i x = c_i, \forall i \in \{1, ..., m\}, \quad x \in \mathcal{X} \tag{5.1}$$

where $\mathcal{X}$ is some convex set and $M_i \in \mathbb{R}^{n\times n}$ are matrices, not necessarily symmetric. In this chapter, we consider solving this problem by minimizing a sum of squares:

$$\min_{x\in\mathcal{X}} F = \sum_{i=1}^{m} F_i(x) = \sum_{i=1}^{m} (x^\top M_i x - c_i)^2, \tag{5.2}$$

which we refer to, in this chapter, as second-order least-squares problem. Thus, we aim to minimize a forth-order (quartic) polynomial, which is nonconvex in general, over some convex set. Solving a system of quadratic equations or minimizing a quartic polynomial is NP-hard in general (see, for instance, [20, 47]).

There are several simple extensions of (5.1) and (5.2) to which methods in this chapter can be readily applied. For instance, (5.1) can be extended by adding a non-homogeneous term on the left hand side, i.e.

$$x^\top M_i x + p_i^\top x = c_i, \forall i \in \{1, ..., m\}, \tag{5.3}$$

where $m_i \in \mathbb{R}^n$ is a given vector.

Another natural extension is the introduction of a regularization term. System (5.3) may be under-determined or it only needs to be satisfied approximately, i.e. $x^\top M_i x + p_i^\top x \approx c_i, \forall i \in \{1, ..., m\}$. In these situations, we may consider a problem in this form:

$$\min_{x\in\mathcal{X}} F = \sum_{i} (x^\top M_i x + p_i^\top x - c_i)^2 + q(x), \tag{5.4}$$

where $q(x)$ is a regularization function that has some desired properties.

Below are the practical examples that provided initial motivation for the methods in this chapter.

Example 5.1.1. *Least-squares risk parity problem.*

Risk parity arises in portfolio selection when the objective is to develop portfolios for which the contributions of risk from all assets are equally weighted [48]. If volatility of the returns is chosen as the risk measure, then risk parity problem can be represented as

$$\begin{aligned} & x_i(\Sigma x)_i = x_j(\Sigma x)_j, \ \forall i, j, \\ & a_i \leq x_i \leq b_i \\ & \sum_{i=1}^{n} x_i = 1, \end{aligned} \tag{5.5}$$

where x is the weight vector of individual assets, a_i and b_i are lower and upper bounds on the weight of the i-th asset, and Σ is the covariance matrix of the assets. Depending on the constraints on x, the existence of the exact solution of (5.5) *may not be simple to establish. As an alternative, in [5], the following least-square model is proposed for risk parity portfolios:*

$$\begin{aligned} \min_{x,\theta} \quad & \sum_{i=1}^{n} (x_i(\Sigma x)_i - \theta)^2 + q(x) \\ s.t. \quad & a_i \leq x_i \leq b_i \\ & \sum_{i=1}^{n} x_i = 1, \end{aligned} \tag{5.6}$$

where $q(x)$ is a customized measure function. In the standard risk parity portfolio selection problem, $q(x) = 0$. It was shown in [5] that risk parity solution may not be unique when shorting of assets is allowed ($a_i < 0$ for some i). In that case one may aim at finding a risk parity solution with the least variance, for instance. In that case $q(x) = \rho x^\top \Sigma x$, where $\rho > 0$ is a regularization parameter.

The risk parity formulation (5.6) *clearly fits the form* (5.4)*, with* $\mathcal{X} = \{x : a \leq x \leq b\} \otimes \mathbb{R}$*,* $M_i = \Sigma_i^\top e_i$*, and* $p_i = (-e_{n+1})^\top$*, where* $\Sigma_i \in \mathbb{R}^{1\times(n+1)}$ *is the i-th row of the covariance matrix with a zero added as the* $n+1$*st element and* $e_i \in \mathbb{R}^{1\times(n+1)}$ *is the i-th row of the identity.*

Example 5.1.2. *Group risk parity problem.*

More generally, one may divide assets into groups by sectors (of industry). In this case the risk parity between these sectors rather than individual assets is desired. This leads to the following grouped formulation [5]:

$$\begin{aligned} \min_{x,\theta} \quad & \sum_{j=1}^{l}(\sum_{i\in\mathcal{G}_j} x_i(\Sigma x)_i - \theta)^2 \\ s.t. \quad & a_i \leq x_i \leq b_i \\ & \sum_{i=1}^{n} x_i = 1, \end{aligned} \tag{5.7}$$

where $\mathcal{G}_j$ *stands for the* j*th group, and* l *is the total number of sectors. Again, this problem can be written in form* (5.4)*, with* $\mathcal{X}$ *and* p_i *defined as in individual risk parity case and* $M_j = A_j^\top B_j$*, where* $A_j \in \mathbb{R}^{m_j\times(n+1)}$ *contains all rows of* Σ *indexed by* $\mathcal{G}_j$ *and appended by a zero element and* $B_j \in \mathbb{R}^{m_j\times(n+1)}$ *is defined as follows. Suppose the i-th row in* A_j *is the corresponding* (k_i)*th row in* Σ*, then*

$$(B_j)_{i,k} = \begin{cases} 1, & k = k_i \\ 0, & otherwise. \end{cases}$$

For the main part of this chapter we generalize the setting further. In particular, we consider optimizing (over a convex set $\mathcal{X}$) an objection function $F(x)$, which can be written as $F(x) = h(f(x), g(x))$, where f and g are (possibly) vector functions of x, and h is some function of the corresponding arguments. We make some standard assumptions on the smoothness and boundedness of F, but the key assumption for the purposes of this chapter is that h, f and g are such that for any fixed $\bar{x}$, $h(f(\bar{x}), g(x))$ and $h(f(x), g(\bar{x}))$ are smooth and convex functions. This assumption applies to our second-order least squares problem (5.4).

Based on the representation of function $F(x)$ we propose an algorithmic framework based on variable splitting and augmented Lagrangian technique. Our framework consists of the well-known alternating direction method of multipliers (ADMM) and alternating linearization method (ALM) which has been widely studied in recent literature, primarily for convex optimization. Our focus and main contribution is to analyze these methods in a nonconvex setting where the objective function cannot be represented as a sum of multiple functions. The problem under discussion is potentially nonconvex, with convexity being assumed in some subspace when variable splitting is applied, and our methods are convergent to a local minimum. We provide global complexity analysis for both ADMM and ALM and show that they converge at the sub-linear rate of $O(1/\sqrt{k})$. The difference between the two frameworks is that ALM requires computing partial gradient information and as a result benefits from possibility of varying the choice of proximal parameter and choosing it via backtracking. It also requires constraints $x \in \mathcal{X}$ to be enforced for each subproblem optimization. In ADMM, on the other hand, the choice of the proximal parameter has to be fixed and sufficiently small (at least in theory), while the constraints are only enforced for one of the subproblems on each iteration. Both frameworks are simple and rely on solving a sequence of convex subproblems. Our experiments show that ALM is more efficient in terms of the number of convex subproblems that need to be solved. We also show, that our alternating linearization scheme is related to the classical Levenberg-Marquard method in case of solving least-squares problems.

Since our methods find local stationary points they provide upper bounds on objective function value. While we do not focus on global optimization techniques here, we provide some lower bounds for the risk parity problems specifically, to demonstrate that our local solutions happen to be global in our numerical example. In particular, we consider the sum-of-square (SOS) techniques which is a popular approach for polynomial optimization [43, 54, 55]. SOS finds a global lower bound by checking the membership in the sum-of-square cone via semidefinite programming (SDP). There have been several variants of SOS methods and we discuss some of them in this chapter. These techniques are usually computationally expensive as the dimension of the resulting SDP problem grows fast, but they appear to produce useful results for the applications that we are interested in.

The rest of the chapter is organized as follows. After a brief discussion of the problem structure, in Section 5.2.3 we introduce the class of algorithms based on variable splitting and augmented Lagrangian function and develop and analyze the ADMM method. We introduce and analyze the ALM method in Section 5.2.4. We discuss the application of SOS technique and its variants in Section 5.4. The experiments for risk parity problem and computational results are presented in Section 5.5, followed by conclusion remarks in Section 5.6.

5.2 Alternating direction schemes for minimizing a nonconvex objective that is *not* necessarily composite

5.2.1 Notations and preliminaries

Consider the following nonlinear optimization problem:

$$\min_{x \in \mathcal{X}} F(x), \tag{5.8}$$

where $\mathcal{X} \in \mathbb{R}^n$ is a simple convex set. We seek local solutions $\bar{x}$ that satisfy first-order optimality condition

$$\langle \nabla F(\bar{x}), x - \bar{x} \rangle \geq 0, \forall x \in \mathcal{X}. \tag{5.9}$$

Suppose that the function F can be written as a function of two blocks, i.e. $F(x) = h(f(x), g(x))$, where function f is $\mathbb{R}^n \to \mathbb{R}^{m_1}$, g is $\mathbb{R}^n \to \mathbb{R}^{m_2}$ and h is $\mathbb{R}^{m_1+m_2} \to \mathbb{R}$. Note that we do not assume that the objective can be decomposed as a sum of f and g.[1]

Consider the following function of x, which is restriction of F, with the second block, g, fixed, given a fixed $\bar{x}$.

$$F_1(x, \bar{x}) = h(f(x), g(\bar{x})). \tag{5.10}$$

[1]As mentioned in previous chapters, such functions are called composite functions, in recent literate, and are assumed to have the form $F(x) = f(x) + g(x)$, or in multi-block case, $F(x) = \sum_{i=1}^{n} f_i(x)$.

Similarly, if we fix the first block, f, given $\bar{x}$, we have the following function

$$F_2(\bar{x}, x) = h(f(\bar{x}), g(x)). \tag{5.11}$$

In this section, we use the "2-block" notations so that the subscript $i = 1, 2$ indicates which block is variable. In particular the partial derivative of the objective over the ith block is denoted as $\nabla_i F$ $(i = 1, 2)$. Then $\nabla_1 F(x, \bar{x}) = \nabla_f h(f(x), g(\bar{x})) \cdot \nabla_x f(x)$ and $\nabla_2 F(\bar{x}, x) = \nabla_g h(f(\bar{x}), g(x)) \cdot \nabla_x g(x)$, are the gradients of (6.9) and (6.10), respectively, where $\nabla_f h$ and $\nabla_g h$ are the corresponding partial derivatives of h with respect to the first and second block. Note that, for any x, the relationship between the full gradient of F and our notations is simply

$$\nabla F(x) = \nabla_1 F(x, x) + \nabla_2 F(x, x). \tag{5.12}$$

In particular, this and (5.9) imply that x^* is a stationary point of (5.2) as long as

$$\langle \nabla_1 F(x^*, x^*) + \nabla_2 F(x^*, x^*), x - x^* \rangle \geq 0, \forall x \in \mathcal{X}. \tag{5.13}$$

We now list the key assumptions on the function F and the resulting functions F_1 and F_2. Two assumptions are standard - smoothness of the functions and boundedness from below. The third assumption - convexity of the functions F_1 and F_2 - is the key assumption which allows us to develop efficient framework based on alternating directions.

Assumption 5.2.1.

- **Lipschitz continuity of the gradients.** *Gradients of functions* $\nabla F(x)$,

$\nabla_1 F(x,\bar{x})$ and $\nabla_2 F(\bar{x},x)$ are Lipschitz continuous with Lipschitz constant L, for any $\bar{x} \in \mathcal{X}$, i.e. $F, F_1, F_2 \in C_L^{1,1}(\mathcal{X})$,

- **Blockwise convexity.** *$F_1(x,\bar{x}), F_2(\bar{x},x)$ are convex over x, for any $\bar{x} \in \mathcal{X}$,*
- **Boundedness from below.** *F (and hence $F_1(x,\bar{x})$ and $F_2(\bar{x},x)$) is bounded from below, for any $\bar{x} \in \mathcal{X}$, i.e. $F > \infty$.*

For each block, we define the following two functions as local approximations of F at any given $\bar{x} \in \mathbb{R}^n$.

$$\begin{aligned} Q_\mu^1(x,\bar{x}) &= F_1(x,\bar{x}) + \langle \nabla_2 F(\bar{x},\bar{x}), x-\bar{x}\rangle + \frac{1}{2\mu}\|x-\bar{x}\|_2^2 \\ Q_\mu^2(\bar{x},x) &= F_2(\bar{x},x) + \langle \nabla_1 F(\bar{x},\bar{x}), x-\bar{x}\rangle + \frac{1}{2\mu}\|x-\bar{x}\|_2^2, \end{aligned} \tag{5.14}$$

where μ is a positive scalar. These functions will be used by our alternating linearization method in Section 5.2.4.

Below are some examples of $F(x)$ and the resulting functions F_1, F_2, Q_μ^1 and Q_μ^2 . We start with the standard composite function case.

Example 5.2.2. *Assume that $h = f + g$ and thus*

$$F(x) = f(x) + g(x),$$

where $f(x)$ and $g(x)$ are convex and smooth scalar functions. Then we define $F_1(x,\bar{x}) = f(x) + g(\bar{x})$ and $F_2(\bar{x},x) = f(\bar{x}) + g(x)$, which clearly leads to

$$\begin{aligned} Q_\mu^1(x,\bar{x}) &= f(x) + g(\bar{x}) + \langle \nabla g(\bar{x}), x-\bar{x}\rangle + \frac{1}{2\mu}\|x-\bar{x}\|_2^2 \\ Q_\mu^2(\bar{x},x) &= f(\bar{x}) + g(x) + \langle \nabla f(\bar{x}), x-\bar{x}\rangle + \frac{1}{2\mu}\|x-\bar{x}\|_2^2. \end{aligned} \tag{5.15}$$

Note that our approximation functions in this case are the same as the standard block-wise proximal functions from the composite optimization literature.

We now turn to more complex function structures.

Example 5.2.3. *Assume that $h = f \cdot g$ and thus*

$$F(x) = f(x)g(x),$$

where $f(x)$ and $g(x)$ are convex, smooth and nonnegative. Then we define $F_1(x, \bar{x}) = f(x)g(\bar{x})$ and $F_2(\bar{x}, x) = f(\bar{x})g(x)$, which implies that F_1 and F_2 satisfy Assumption and we have

$$\begin{aligned} Q^1_\mu(x, \bar{x}) &= f(x)g(\bar{x}) + f(\bar{x}) \langle \nabla g(\bar{x}), x - \bar{x} \rangle + \frac{1}{2\mu}\|x - \bar{x}\|_2^2 \\ Q^2_\mu(\bar{x}, x) &= f(\bar{x})g(x) + g(\bar{x}) \langle \nabla f(\bar{x}), x - \bar{x} \rangle + \frac{1}{2\mu}\|x - \bar{x}\|_2^2. \end{aligned} \tag{5.16}$$

Finally, we present another general setting which includes the objective function of (5.2) and satisfies Assumption 5.2.3.

Example 5.2.4. *Assume that $h = \sum_i ((f^i)^\top g^i - c_i)^2$ and thus*

$$F(x) = \sum_i (f^i(x)^\top g^i(x) - c_i)^2, \tag{5.17}$$

where each $f^i(x) = [f^i_1(x), f^i_2(x), ..., f^i_{m_i}(x)]$ and $g^i(x) = [g^i_1(x), g^i_2(x), ..., g^i_{m_i}(x)]$ is an affine function of x, $\mathbb{R}^n \to \mathbb{R}^{m_i}$.

It is easy to see, that is functions f^i and g^i are affine and homogeneous (that is they do not contain a constant term), then this form is equivalent to (5.2)*, with an appropriate choice of M_i and c_i parameters, otherwise formulation* (5.4) *applies with*

$q(x) = 0$. *Clearly, when* $q(x)$ *is a smooth convex function, then* $h = \sum_i (f_i^\top g_i - c_i)^2 + q(x)$ *should be considered.*

In the risk parity case, in particular, we have

$$\begin{aligned} \min_{x,\theta} \quad & F(x) = \sum_{i=1}^{n} (x^\top M_i x - \theta)^2 \\ s.t. \quad & x \in \mathcal{X}, \end{aligned}$$

which can be written as (5.17) *with* $f^i(x, \theta) = [x, -1]$ *and* $g^i(x, \theta) = [M_i x, \theta]$, $i = 1, \ldots, n$ *are* $(n+1)$*-dimensional affine vector functions of* x *and* θ *and* $c_i = 0$, $\forall i$. *For any given* $\bar{x}$*:* $F_1(x, \bar{x}, \bar{\theta}) = \sum_{i=1}^{m} (x^\top M_i \bar{x} - \bar{\theta})^2$ *and* $F_2(\bar{x}, x) = \sum_{i=1}^{m} (\bar{x}^\top M_i x - \theta)^2$. *Both* F_1 *and* F_2 *are convex quadratic functions and Assumption 5.2.3 is satisfied.*

5.2.2 Variable splitting and augmented Lagrangian based methods

In this section, we discuss several alternating direction methods, all of which are based on the augmented Lagrangian framework with variable splitting. Augmented Lagrangian method (with variable splitting) and its variants have been increasingly popular in recent literature [2, 12, 34, 62, 64].

In particular observe that (6.8) can be equivalently written as

$$\begin{aligned} \min_{x \in \mathcal{X}, y} \quad & F(x, y) = h(f(x), g(y)) \\ \text{s.t.} \quad & x = y, \end{aligned} \tag{5.18}$$

where $x, y \in \mathbb{R}^n$. In other words, we map the dimension of decision variable from n

in (6.8) to $2n$ in (6.6).

Consider problem in the form of (6.6). Provided a penalty parameter $1/\mu$ ($\mu > 0$), we have the following augmented Lagrangian function:

$$\mathcal{L}_A(x, y; \lambda) = F(x, y) - \lambda^\top (x - y) + \frac{1}{2\mu}\|x - y\|^2, \tag{5.19}$$

and, hence, (6.6) can be solved by the augmented Lagrangian method described in Algorithm 12.

Example 5.2.5. *In the second-order least-squares case, we split the variables as follows*

$$\begin{aligned} \min_{x,y} \quad & F(x) = \sum_{i=1}^{n} (x^\top M_i y - c_i)^2 \\ s.t. \quad & x = y \\ & x \in \mathcal{X}. \end{aligned} \tag{5.20}$$

Then the augmented Lagrangian function is defined as

$$\mathcal{L}_A(x, y; \lambda) = \sum_{i=1}^{n} (x^\top M_i y - c_i)^2 - \lambda^\top (x - y) + \frac{1}{2\mu}\|x - y\|^2.$$

Algorithm 12 Augmented Lagrangian method (AL)

```
1. Choose $\mu^0, \lambda^0$, and $x^0 = y^0$;
2. for $k = 0, 1, ...,$ do
       $[x^{k+1}, y^{k+1}] := \arg\min_{x \in \mathcal{X}, y} \mathcal{L}_A(x, y; \lambda^k)$;
       update the multiplier $\lambda^{k+1} = \lambda^k - \frac{1}{\mu^k}(x^{k+1} - y^{k+1})$;
       possibly choose new penalty parameter $\mu^{k+1}$.
```

The convergence of augmented Lagrangian method (see Algorithm 12) has been well studied (see, for instance, [9, 72]). Furthermore, the minimization of the augment Lagrangian in Algorithm 12, can be performed by applying block coordinate decent method (BCD) until first-order optimality is guaranteed. Convergence of block coordinate decent method, under the assumption of uniqueness of minimizers over blocks, and for simple convex constraints is studied, for instance, in [37].

5.2.3 Alternating direction methods of multipliers

Alternating direction methods of multipliers (ADMM), or alternating direction augmented Lagrangian method (ADMM), can be regarded as a variant of augmented Lagrangian with subproblems solved inexactly by BCD. ADMM and other alternating direction methods (ADMs) can be tracked back to the Douglas-Rachford method in the 1950s [26] and ADMs for solving variational problems associated with PDEs in the 1970s [30, 33]. In ADMM the multiplier is updated after only one minimization step over each x and y blocks instead of after minimizing the augmented Lagrangian over x and y jointly. A simple framework of ADMM for solving (6.6) is given in Algorithm 13.

Algorithm 13 Alternating direction methods of multipliers (ADMM)

```
1. Choose $\mu, \lambda^0$, and $x^0 = y^0$;
2. for $k = 0, 1, ...,$ do
       $x^{k+1} := \arg\min_{x \in \mathcal{X}} \mathcal{L}_A(x, y^k; \lambda^k)$;
       $y^{k+1} := \arg\min_{y} \mathcal{L}_A(x^{k+1}, y; \lambda^k)$;
       update the multiplier $\lambda^{k+1} = \lambda^k - \frac{1}{\mu}(x^{k+1} - y^{k+1})$;
```

ADMM has been widely used and well studied, in the large-scale convex optimization setting in the case of composite structure of the objective function (see [12] for a review). In the nonconvex setting, ADMM has been applied to obtain KKT solutions often obtaining competitive results, empirically. However, its theoretical properties are not well understood in the nonconvex case. Typically, convergence is shown under the assumption that the successive differences of the iterates converge to zero (see, for instance, an ADMM for polynomial optimization described in [41]). While such assumption seems reasonable in that the iterates produced by the algorithm do not exhibit any erratic behavior, it is not clear if this condition can be verified in advance or enforced during the progress of the algorithm. Recently Hong, et al. show the convergence of a family of ADMMs without this assumption when applied to a family of n-block structured composite nonconvex problems [40]. Our results in this chapter also do not rely on this assumption, but are different from [40] in that our objective function is not assumed to be a sum of multiple functions (composite form). Note that it is not trivial to extend our convergence result to problems with more than two blocks. However, the two-block results shown in this section readily apply to the second-order least-squares problem which is the focus of our work.

Our strategy to prove the convergence of ADMM is similar to that of [40], in that it relies on obtaining a sufficient function decrease of augmented Lagrangian. In our proof, we frequently use the term "x-update" and "y-update", which refer to the update rule of each block of variables x and y, respectively. First, we have the following result to bound the augmented Lagrangian function value, which is a relaxed version of Lemma 2.3 in [40].

Lemma 5.2.6. *Let Assumption 5.2.3 hold with the Lipschitz constant L and let* $\mu \le \dfrac{1}{4L}$*. Then for the sequence of iterates* $\{x^k, y^k, \lambda^k\}$ *defined by Algorithm 13, the augmented Lagrangian function converges to some limit* $\mathcal{L}^*$*:*

$$\lim_{k\to\infty} \mathcal{L}_A(x^k, y^k; \lambda^k) = \mathcal{L}^*$$

Proof. By the first-order optimality condition of y-update in Algorithm 13, we have

$$\nabla_2 F(x^{k+1}, y^{k+1}) + \lambda^k - \frac{1}{\mu}(x^{k+1} - y^{k+1}) = 0,$$

which leads to

$$\lambda^{k+1} = \lambda^k - \frac{1}{\mu}(x^{k+1} - y^{k+1}) = -\nabla_2 F(x^{k+1}, y^{k+1}). \tag{5.21}$$

Thus, we can bound the change of λ:

$$\begin{aligned} \|\lambda^{k+1} - \lambda^k\| &= \|\nabla_2 F(x^{k+1}, y^{k+1}) - \nabla_2 F(x^k, y^k)\| \\ &\leq L\left(\|x^{k+1} - x^k\| + \|y^{k+1} - y^k\|\right), \end{aligned} \tag{5.22}$$

where L is a Lipschitz constant of the gradient.

Now, we can bound the change of the augmented Lagrangian function value, after the update of the primal variables. Since it is assumed that at each iteration $F_1(x, y^k)$ is convex, the function of subproblem of ADMM can be regarded as a sum of convex and strongly convex function and thus is strongly convex, i.e., we have $\mathcal{L}_A(x^k, y^k; \lambda^k) \geq \mathcal{L}_A(x^{k+1}, y^k; \lambda^k) - \left\langle \nabla_1 \mathcal{L}_A(x^{k+1}, y^k; \lambda^k), x^{k+1} - x^k \right\rangle + \frac{1}{2\mu}\|x^{k+1} - x^k\|^2$. From the optimality of the subproblem we obtain that

$$\mathcal{L}_A(x^k, y^k; \lambda^k) - \mathcal{L}_A(x^{k+1}, y^k; \lambda^k) \geq \frac{1}{2\mu}\|x^{k+1} - x^k\|^2. \tag{5.23}$$

Similarly, for y-update, it holds that

$$\mathcal{L}_A(x^{k+1}, y^k; \lambda^k) - \mathcal{L}_A(x^{k+1}, y^{k+1}; \lambda^k) \geq \frac{1}{2\mu}\|y^{k+1} - y^k\|^2. \tag{5.24}$$

We also bound the change of the augmented Lagrangian function value, after the update of the multipliers:

$$\begin{aligned}
&\mathcal{L}_A(x^{k+1}, y^{k+1}; \lambda^{k+1}) - \mathcal{L}_A(x^{k+1}, y^{k+1}; \lambda^k) \\
&= F(x^{k+1}, y^{k+1}) - \left\langle \lambda^{k+1}, x^{k+1} - y^{k+1} \right\rangle + \frac{1}{2\mu}\|x^{k+1} - y^{k+1}\|^2 \\
&-F(x^{k+1}, y^{k+1}) - \left\langle \lambda^{k}, x^{k+1} - y^{k+1} \right\rangle + \frac{1}{2\mu}\|x^{k+1} - y^{k+1}\|^2 \\
&= -\left\langle \lambda^{k+1} - \lambda^k, x^{k+1} - y^{k+1} \right\rangle \\
&= \mu\|\lambda^{k+1} - \lambda^k\|^2.
\end{aligned}$$

By using (5.22), we further have

$$\begin{aligned}
&\mathcal{L}_A(x^{k+1}, y^{k+1}; \lambda^{k+1}) - \mathcal{L}_A(x^{k+1}, y^{k+1}; \lambda^k) \\
&\leq \mu L^2 \left(\|x^{k+1} - x^k\| + \|y^{k+1} - y^k\|\right)^2 \\
&\leq 2\mu L^2 \left(\|x^{k+1} - x^k\|^2 + \|y^{k+1} - y^k\|^2\right).
\end{aligned} \tag{5.25}$$

With (5.23), (5.24) and (5.25), we finally have

$$\begin{aligned}
&\mathcal{L}_A(x^{k+1}, y^{k+1}; \lambda^{k+1}) - \mathcal{L}_A(x^k, y^k; \lambda^k) \\
&\leq \left(2\mu L^2 - \frac{1}{2\mu}\right)\|x^{k+1} - x^k\|^2 + \left(2\mu L^2 - \frac{1}{2\mu}\right)\|y^{k+1} - y^k\|^2,
\end{aligned} \tag{5.26}$$

which indicates a monotonic decrease of augmented Lagrangian function value as long as $\mu < \frac{1}{2L}$. For instance, we can choose μ to be $\frac{1}{4L}$.

Moreover, we can bound the value of $\mathcal{L}_A$ by F:

$$\begin{aligned}
&\mathcal{L}_A(x^{k+1}, y^{k+1}; \lambda^{k+1}) \\
&= F(x^{k+1}, y^{k+1}) - \langle \lambda^{k+1}, x^{k+1} - y^{k+1} \rangle + \frac{1}{2\mu}\|x^{k+1} - y^{k+1}\|^2 \\
&= F(x^{k+1}, y^{k+1}) + \langle \nabla_2 F(x^{k+1}, y^{k+1}), x^{k+1} - y^{k+1} \rangle + \frac{1}{2\mu}\|x^{k+1} - y^{k+1}\|^2 \\
&\geq F(x^{k+1}),
\end{aligned}$$

for $\mu \leq \frac{1}{L}$. Since it is assumed that F is bounded from below, it follows that the sequence $\{\mathcal{L}_A(x^k, y^k; \lambda^k)\}$ is also bounded from below and thus the augmented Lagrangian function value converges.

□

Remark 5.2.7. *It is possible to prove this result without assuming convexity of F_1, F_2, by choosing μ large enough so that the augmented Lagrangian is strongly convex at each iteration.*

Now we show that, under further assumptions, any limit point of the sequence $\{x^k\}$ is a stationary point of (6.8).

Theorem 5.2.8. *Let Assumption 5.2.3 hold with the Lipschitz constant L and let $\mu < \frac{1}{4L}$. Then any limit point of sequence $\{x^k\}$ generated by Algorithm 13 is a stationary point of* (6.8).

Proof. As $\mathcal{L}_A(x^k, y^k; \lambda^k)$ converges by Lemma 5.2.6, it follows from (5.22) and (5.26) that

$$\begin{aligned}
x^{k+1} - x^k &\to 0 \\
y^{k+1} - y^k &\to 0 \\
\lambda^{k+1} - \lambda^k &\to 0,
\end{aligned}$$

which indicates that $x^k - y^k \to 0$ from (5.21), since μ is bounded from above.

Assume $\{\bar{x}, \bar{y}, \bar{\lambda}\}$ is a limit point of the sequence $\{x^k, y^k, \lambda^k\}$. Our goal is to prove the first-order condition, i.e. for any x such that $x \neq \bar{x}$ and $x \in \mathcal{X}$, it holds that $\left\langle \nabla F(\bar{x}), \frac{x - \bar{x}}{\|x - \bar{x}\|} \right\rangle \geq 0$, which is equivalent to (5.9).

In fact, for any $x \neq \bar{x}$,

$$\begin{aligned}
&\left\langle \nabla F(\bar{x}), \frac{x - \bar{x}}{\|x - \bar{x}\|} \right\rangle \\
&= \frac{1}{\|x - \bar{x}\|} \langle \nabla_1 F(\bar{x}, \bar{y}) + \nabla_2 F(\bar{x}, \bar{y}), x - \bar{x} \rangle \\
&+ \frac{1}{\|x - \bar{x}\|} \langle \nabla_1 F(\bar{x}, \bar{x}) - \nabla_1 F(\bar{x}, \bar{y}), x - \bar{x} \rangle + \frac{1}{\|x - \bar{x}\|} \langle \nabla_2 F(\bar{x}, \bar{x}) - \nabla_2 F(\bar{x}, \bar{y}), x - \bar{x} \rangle \\
&= \frac{1}{\|x - \bar{x}\|} \left\langle \nabla_1 F(\bar{x}, \bar{y}) - \bar{\lambda}, x - \bar{x} \right\rangle \\
&+ \frac{1}{\|x - \bar{x}\|} \langle \nabla_1 F(\bar{x}, \bar{x}) - \nabla_1 F(\bar{x}, \bar{y}), x - \bar{x} \rangle + \frac{1}{\|x - \bar{x}\|} \langle \nabla_2 F(\bar{x}, \bar{x}) - \nabla_2 F(\bar{x}, \bar{y}), x - \bar{x} \rangle,
\end{aligned} \tag{5.27}$$

where we have $\langle \nabla_1 F(\bar{x}, \bar{x}) - \nabla_1 F(\bar{x}, \bar{y}), x - \bar{x} \rangle \leq L\|\bar{x} - \bar{y}\|\|x - \bar{x}\|$, and moreover, $\langle \nabla_2 F(\bar{x}, \bar{x}) - \nabla_2 F(\bar{x}, \bar{y}), x - \bar{x} \rangle \leq L\|\bar{x} - \bar{y}\|\|x - \bar{x}\|$, and thus the last two terms of (5.27) vanish as $\bar{x} - \bar{y} \to 0$. From the first order optimality of the x-update, we also have $\left\langle \nabla_1 F(\bar{x}, \bar{y}) - \bar{\lambda}, x - \bar{x} \right\rangle \geq 0, \forall x \in \mathcal{X}$. Thus, $\left\langle \nabla F(\bar{x}), \frac{x - \bar{x}}{\|x - \bar{x}\|} \right\rangle \geq 0$ which indicates that $\bar{x}$ is a stationary point of F.

□

We will now prove the convergence rate result for Algorithm 13. We will show that the sequence of iterates converges to the first order optimality conditions at a sublinear rate, where we say that a two-block pair $(\bar{x}, \bar{y})$ satisfies the first-order optimality conditions for (6.6) when

$$\begin{aligned}
&\langle \nabla_1 F(\bar{x}, \bar{y}), x - \bar{x} \rangle + \langle \nabla_2 F(\bar{x}, \bar{y}), x - \bar{x} \rangle \geq 0, \ \forall x \in \mathcal{X} \\
&\bar{x} - \bar{y} = 0.
\end{aligned} \tag{5.28}$$

Clearly, (5.28) is equivalent to (5.9).

Theorem 5.2.9. *Let Assumption 5.2.3 hold with the Lipschitz constant L and let $\mu < \dfrac{1}{4L}$. Let $\{x^k, y^k, \lambda^k\}$ be the sequence of iterates defined by Algorithm 13, Denote $\hat{g}_k \equiv \min\limits_{1\le i\le k}\{\|x^i - x^{i-1}\| + \|y^i - y^{i-1}\|\}$ and $\bar{\mathcal{I}}_k \equiv \{(x^i, y^i) : \|x^i - x^{i-1}\| + \|y^i - y^{i-1}\| = \hat{g}_k,\ 1 \le i \le k\}$. Then for any fixed $x \in \mathcal{X}$, which is not a stationary point of* (6.6), *and any k, one (and only one) of the following is true:*

1. *$\hat{g}_k = 0$, and* (5.28) *is satisfied by $\bar{x} = x^k$ and $\bar{y} = y^k$.*

2. *$\hat{g}_k > 0$. Consider a subsequence $\{\hat{x}_k, \hat{y}_k\}$, where $(\hat{x}_k, \hat{y}_k) \in \bar{\mathcal{I}}_k$. Then $\nabla F(\hat{x}_k, \hat{y}_k)$ satisfies first-order optimality conditions* (5.28) *with an error, which converges to zero at a sublinear rate of $\mathcal{O}(\dfrac{1}{\sqrt{k}})$.*

Proof. Consider the sequence of two-block variables $\{x^k, y^k\}$.

If $\hat{g}_k = 0$, then it is obvious that a limit point is obtained, and it is a stationary point by Theorem 2.1.

Now suppose $\hat{g}_k > 0$, and we aim to bound each condition of (5.28) and show that each condition is satisfied with an error which converges to zero at the rate of $\mathcal{O}(\dfrac{1}{\sqrt{k}})$.

First, we bound the change of primal variables in $\bar{\mathcal{I}}_k$ by the change of augmented

Lagrangian function value. From (5.26), we have

$$\begin{aligned}
&\mathcal{L}_A(x^0, y^0; \lambda^0) - \mathcal{L}_A(x^*, y^*; \lambda^*)\\
&\geq \mathcal{L}_A(x^0, y^0; \lambda^0) - \mathcal{L}_A(x^k, y^k; \lambda^k)\\
&\geq \sum_{i=1}^{k}(\frac{1}{2\mu} - 2\mu L^2)\|x^i - x^{i-1}\|^2 + \sum_{i=1}^{k}(\frac{1}{2\mu} - 2\mu L^2)\|y^i - y^{i-1}\|^2 \qquad (5.29)\\
&\geq \frac{1}{2}k\delta \left(\|x^i - x^{i-1}\| + \|y^i - y^{i-1}\|\right)^2\\
&= \frac{1}{2}k\delta \hat{g}_k^2,
\end{aligned}$$

where $\delta = \frac{1}{2\mu} - 2\mu L^2$. Since we choose $0 < \mu \leq \frac{1}{4L}$, $\delta > 0$.

Thus, we have

$$\hat{g}_k \leq \sqrt{\frac{2(\mathcal{L}_A(x^0, y^0; \lambda^0) - \mathcal{L}_A(x^*, y^*; \lambda^*))}{k\delta}}. \qquad (5.30)$$

Now we bound each condition of (5.28) by (5.30). From the x-update, we have $\left\langle \nabla_1 \mathcal{L}(x^k, y^{k-1}; \lambda^{k-1}), x - x^k \right\rangle \geq 0,\ \forall x \in \mathcal{X}$, which is written as

$$\left\langle \nabla_1 F(x^k, y^{k-1}) + \nabla_2 F(x^k, y^k) + \frac{1}{\mu}(y^k - y^{k-1}), x - x^k \right\rangle \geq 0,\ \forall x \in \mathcal{X}. \qquad (5.31)$$

Here we use the fact that $\nabla_1 \mathcal{L}(x^k, y^{k-1}; \lambda^{k-1}) = \nabla_1 F(x^k, y^{k-1}) - \lambda^{k-1} + \frac{1}{\mu}(x^k - y^{k-1})$ and $\lambda^{k-1} = \lambda^k + \frac{1}{\mu}(x^k - y^k) = -\nabla_2 F(x^k, y^k) + \frac{1}{\mu}(x^k - y^k)$.

Similarly to (5.27), we now bound $\left\langle \nabla_1 F(x^k, y^k), x - x^k \right\rangle + \left\langle \nabla_2 F(x^k, y^k), x - x^k \right\rangle$

as following

$$\begin{aligned}
&\left\langle \nabla_1 F(x^k, y^k), x - x^k \right\rangle + \left\langle \nabla_2 F(x^k, y^k), x - x^k \right\rangle \\
&= \left\langle \nabla_1 F(x^k, y^{k-1}) + \nabla_2 F(x^k, y^k) + \frac{1}{\mu}(y^k - y^{k-1}), x - x^k \right\rangle \\
&+ \left\langle \nabla_1 F(x^k, y^k) - \nabla_1 F(x^k, y^{k-1}), x - x^k \right\rangle - \left\langle \frac{1}{\mu}(y^k - y^{k-1}), x - x^k \right\rangle \\
&\geq -(L + \frac{1}{\mu})\|x - x^k\|\|y^k - y^{k-1}\|.
\end{aligned} \tag{5.32}$$

Thus, it follows that

$$\left\langle \nabla_1 F(x^k, y^k) + \nabla_2 F(x^k, y^k), \frac{x - x^k}{\|x - x^k\|} \right\rangle \geq -(L + \frac{1}{\mu})\|y^k - y^{k-1}\|. \tag{5.33}$$

Consider a subsequence $\{(\hat{x}_k, \hat{y}_k)\}$, where $(\hat{x}_k, \hat{y}_k) \in \bar{\mathcal{I}}_k$. From (5.33), we have

$$\begin{aligned}
&\left\langle \nabla_1 F(\hat{x}_k, \hat{y}_k) + \nabla_2 F(\hat{x}_k, \hat{y}_k), \frac{x - \hat{x}_k}{\|x - \hat{x}_k\|} \right\rangle \\
&\geq -(L + \frac{1}{\mu})\hat{g}_k \\
&\geq -(L + \frac{1}{\mu})\sqrt{\frac{2(\mathcal{L}_A(x^0, y^0; \lambda^0) - \mathcal{L}_A(x^*, y^*; \lambda^*))}{k\delta}}.
\end{aligned} \tag{5.34}$$

On the other hand, for any k, we have $x^k - y^k = -\mu(\lambda^k - \lambda^{k-1})$, which combined with (5.22) implies that

$$\|x^k - y^k\| = \mu\|\lambda^k - \lambda^{k-1}\| \leq \mu L \left(\|x^k - x^{k-1}\| + \|y^k - y^{k-1}\|\right). \tag{5.35}$$

(5.35) indicates that, for $(\hat{x}_k, \hat{y}_k) \in \bar{\mathcal{I}}_k$, we have

$$\|\hat{x}_k - \hat{y}_k\| \leq \bar{\mu} L \hat{g}_k \leq \bar{\mu} L \sqrt{\frac{2(\mathcal{L}_A(x^0, y^0; \lambda^0) - \mathcal{L}_A(x^*, y^*; \lambda^*))}{k\delta}}. \tag{5.36}$$

(5.34) and (5.36) demonstrate that sequences x_k, y_k and $\nabla F(\hat{x}_k, \hat{y}_k)$ satisfy first-order optimality conditions with errors that converge to zero at a sublinear rate $\mathcal{O}(\frac{1}{\sqrt{k}})$. Thus, our proof is complete.

$\square$

Remark 5.2.10. *When $\mathcal{X} = \mathbb{R}^n$, by choosing $x = \hat{x}_k - \frac{\nabla F(\hat{x}_k, \hat{y}_k)}{\|\nabla F(\hat{x}_k, \hat{y}_k)\|}$ in (5.34) we have*

$$\|\nabla F(\hat{x}_k, \hat{y}_k)\| \leq (L + \frac{1}{\mu})\sqrt{\frac{2(\mathcal{L}_A(x^0, y^0; \lambda^0) - \mathcal{L}_A(x^*, y^*; \lambda^*))}{k\delta}}, \tag{5.37}$$

which recovers the standard complexity result in the unconstrained case, i.e. it takes $\mathcal{O}(\frac{1}{\epsilon^2})$ iterations for the norm of $\nabla F(\hat{x}_k, \hat{y}_k)$ to reach a value below ϵ.

5.2.4 Alternating linearization method

In this section we consider alternating linearization method (ALM), which is closely related to ADMM. The relationship between the two methods in the composite convex setting is derived in [34]. In this chapter, we discuss alternating linearization method in a nonconvex setting where the objective may not be decomposable.

Recall the notations in (5.14). Suppose that both sequences $\{x^k\}$ and $\{y^k\}$ are bounded, and that the function F can be bounded locally from above by the following approximation term with sufficiently small μ:

$$Q^1_\mu(x, y^k) = F_1(x, y^k) + \langle \nabla_2 F(y^k), x - y^k \rangle + \frac{1}{2\mu}\|x - y^k\|_2^2.$$

Formally put, the following condition holds for sufficiently small μ:

$$F(x^{k+1}) \leq Q^1_\mu(x^{k+1}, y^k), \tag{5.38}$$

where $x^{k+1} := \arg\min_{x \in \mathcal{X}} Q^1_{\mu_1^k}(x, y^k)$. Similarly,

$$F(y^{k+1}) \leq Q^2_\mu(x^{k+1}, y^{k+1}) \tag{5.39}$$

holds for small enough μ, where $y^{k+1} := \arg\min_{y \in \mathcal{X}} Q^2_{\mu_2^k}(x^{k+1}, y)$. In the next section, it will be shown through simple expansion that our assumption holds for the second-order least-squares case. Furthermore, such μ can be found by backtracking at each iteration. The basic ALM algorithm is presented in Algorithm 14 and the backtracking version is in Algorithm 15

Algorithm 14 Alternating linearization method (ALM)

```
1. Choose $\mu_1^0 = \mu_2^0 = \mu^0$, and $x^0 = y^0$;
2. for $k = 0, 1, ...$
    (a) $x^{k+1} := \arg\min_{x \in \mathcal{X}} Q^1_{\mu_1^k}(x, y^k)$; choose $\mu_1^{k+1}$ such that (6.12) holds;
    (b) $y^{k+1} := \arg\min_{y \in \mathcal{X}} Q^2_{\mu_2^k}(x^{k+1}, y)$; choose $\mu_2^{k+1}$ such that (5.39) holds;
```

The convergence analysis of ALM is similar as the analysis in the Iterative Shrinkage-Thresholding Algorithm (ISTA and FISTA) by Beck and Teboulle [7], Alternating Linearization Method (ALM) in the convex setting by Goldfarb *et al.* [34] and Block Coordinate Decent (BCD) method by Xu and Yin [74]. However, all of the above results apply only to the convex domain, which cannot be easily embedded into the case of problem (5.2). Moreover, unlike the ALM in [34], we do not have a composite function form where the linearly separable structure can be taken advantage of (see [34] for details to apply ALM to composite convex functions).

The descent step of Algorithm 14 can be guaranteed by the following result.

Lemma 5.2.11. *Let the objective function and its block-wise representation be defined as* (6.8), (6.10) *and* (6.9). *Define*

$$\hat{x} := \arg\min_{x\in\mathcal{X}} Q^1_\mu(x,u) \equiv \arg\min_{x\in\mathcal{X}} F_1(x,u) + \langle \nabla_2 F(u,u), x-u\rangle + \frac{1}{2\mu}\|x-u\|^2,$$

where $u \in \mathbb{R}^n$. *Suppose the following condition holds*

$$F(\hat{x}) \le Q^1_\mu(\hat{x},u). \tag{5.40}$$

Then we have

$$F(u) - F(\hat{x}) \ge \frac{1}{2\mu}(\|\hat{x}-u\|^2). \tag{5.41}$$

Similarly, define

$$\hat{y} := \arg\min_{y\in\mathcal{Y}} Q^2_\mu(u,y) \equiv \arg\min_{y\in\mathcal{Y}} F_2(u,y) + \langle \nabla_1 F(u,u), y-u\rangle + \frac{1}{2\mu}\|y-u\|^2,$$

where $u \in \mathbb{R}^n$. *Suppose the following condition holds*

$$F(\hat{y}) \le Q^2_\mu(u,\hat{y}). \tag{5.42}$$

Then we have

$$F(u) - F(\hat{y}) \ge \frac{1}{2\mu}(\|\hat{y}-u\|^2). \tag{5.43}$$

Proof. From the condition (5.40), we have

$$\begin{aligned} F(u) - F(\hat{x}) &\ge F(u) - Q^1_\mu(\hat{x},u) \\ &= F(u) - (F_1(\hat{x},u) + \langle \nabla_2 F(u,u), \hat{x}-u\rangle + \frac{1}{2\mu}\|\hat{x}-u\|^2). \end{aligned} \tag{5.44}$$

Since $F_1(x,u)$ is convex for any given u, we have

$$F(u) \geq F_1(\hat{x},u) + \langle \nabla_1 F(\hat{x},u), u - \hat{x} \rangle . \tag{5.45}$$

It follows that

$$\begin{aligned} F(u) - F(\hat{x}) &\geq F(u) - Q^1_\mu(\hat{x},u) \\ &\geq F_1(\hat{x},u) + \langle \nabla_1 F(\hat{x},u), u - \hat{x} \rangle - Q^1_\mu(\hat{x},u) \\ &= \langle \nabla_1 F(\hat{x},u) + \nabla_2 F(u,u), u - \hat{x} \rangle - \frac{1}{2\mu}\|\hat{x} - u\|^2. \end{aligned} \tag{5.46}$$

Since the subproblem is solved for optimality, it satisfies the first-order optimality condition:

$$\left\langle \nabla_1 Q^1_\mu(\hat{x},u), x - \hat{x} \right\rangle \geq 0,\ \forall x \in \mathcal{X},$$

which means that this condition holds for $x = u$:

$$\left\langle \nabla_1 Q^1_\mu(\hat{x},u), u - \hat{x} \right\rangle \geq 0,$$

i.e.

$$\left\langle \nabla_1 F(\hat{x},u) + \nabla_2 F(u,u) + \frac{1}{\mu}(\hat{x} - u), u - \hat{x} \right\rangle \geq 0. \tag{5.47}$$

From (5.46) and (5.47), we have

$$\begin{aligned} F(u) - F(\hat{x}) &\geq \langle \nabla_1 F(\hat{x},u) + \nabla_2 F(u,u), u - \hat{x} \rangle - \frac{1}{2\mu}\|\hat{x} - u\|^2 \\ &\geq -\frac{1}{\mu}(\hat{x} - u)^\top (u - \hat{x}) - \frac{1}{2\mu}\|\hat{x} - u\|^2 \\ &= \frac{1}{2\mu}\|\hat{x} - u\|^2. \end{aligned} \tag{5.48}$$

Similarly, the other part of the proof follows.

□

The following result shows the convergence of the primal function value for Algorithm 14.

Theorem 5.2.12. *Let the objective function, its block-wise representation and its approximation be defined as* (6.8), (6.9), (6.10), *and* (5.14), *respectively. Let Assumption 5.2.3 hold with the Lipschitz constant L. Suppose at iteration k, μ_1^k, μ_2^k are chosen to satisfy* (6.12) *and* (5.39), *respectively, and are bounded from zero by some constant $\tau > 0$. Then, by Algorithm 14, the sequence $\{F(x^k)\}$ converges to $F(x^*)$, where x^* is a stationary point of F. Furthermore, denote $\hat{g}_k \equiv \min_{1\le i\le k} \|x^i - y^{i-1}\|$ and $\bar{\mathcal{I}}_k \equiv \{x^i : \|x^i - y^{i-1}\| = \hat{g}_k,\ 1 \le i \le k\}$. Then one and only one of the following is true:*

1. *$\hat{g}_k = 0$, and the limit point is obtained;*

2. *$\hat{g}_k > 0$. Consider a subsequence $\{\hat{x}_k\}$, where $\hat{x}_k \in \bar{\mathcal{I}}_k$. Then $\nabla F(\hat{x}_k)$ satisfies first-order optimality condition* (5.9) *with an error that converges to zero at a sublinear rate $\mathcal{O}(\frac{1}{\sqrt{k}})$.*

Proof. From Algorithm 14 we have

$$x^k := \arg\min_{x\in\mathcal{X}} Q^1_{\mu_1^k}(x, y^{k-1})$$

for $k \ge 1$.

Let $u = y^{k-1}$ in Lemma 5.2.11. Then we have

$$2\mu_1^k(F(y^{k-1}) - F(x^k)) \ge \|x^k - y^{k-1}\|^2 \ge 0. \tag{5.49}$$

Similarly, for the subproblem updating y, we have

$$2\mu_2^k(F(x^k) - F(y^k)) \geq \|x^k - y^k\|^2 \geq 0. \tag{5.50}$$

Hence,

$$F(x^{k-1}) \geq F(x^k) \text{ and } F(y^{k-1}) \geq F(y^k), \quad \text{for all } k. \tag{5.51}$$

Now we have a non-increasing sequence $F(x^k)$ bounded from below. Thus, $\{F(x^k)\}$ converges. Since the left hand sides of (5.49) and (5.50) go to zero when $k \to \infty$, we have $x^k - y^k \to 0$, and also

$$x^k - x^{k-1} \to 0. \tag{5.52}$$

If it happens that $y^{k-1} = \arg\min_{x\in\mathcal{X}} Q^1_{\mu_1^k}(x, y^{k-1})$, then it naturally follows that $x^k = y^{k-1}$ is a stationary point. If $y^{k-1} \neq \arg\min_{x\in\mathcal{X}} Q^1_{\mu_1^k}(x, y^{k-1})$, since x^k is a minimizer of $Q^1_{\mu_1^k}(x, y^{k-1})$, it satisfies the foliowing first-order condition:

$$\left\langle \nabla_1 F(x^k, y^{k-1}) + \nabla_2 F(y^{k-1}, y^{k-1}) + \frac{1}{\mu_1^k}(x^k - y^{k-1}), x - x^k \right\rangle \geq 0, \ \forall x \in \mathcal{X}. \tag{5.53}$$

Hence, we have

$$\begin{aligned}
&\left\langle \nabla_1 F(x^k, x^k) + \nabla_2 F(x^k, x^k), x - x^k \right\rangle \\
&= \left\langle \nabla_1 F(x^k, y^{k-1}) + \nabla_2 F(y^{k-1}, y^{k-1}) + \frac{1}{\mu_1^k}(x^k - y^{k-1}), x - x^k \right\rangle \\
&+ \left\langle \nabla_1 F(x^k, x^k) - \nabla_1 F(x^k, y^{k-1}), x - x^k \right\rangle \\
&+ \left\langle \nabla_2 F(x^k, x^k) - \nabla_2 F(y^{k-1}, y^{k-1}), x - x^k \right\rangle - \left\langle \frac{1}{\mu_1^k}(x^k - y^{k-1}), x - x^k \right\rangle.
\end{aligned} \tag{5.54}$$

As $x^k - x^{k-1} \to 0$, $\nabla_1 F(x^k, x^k) - \nabla_1 F(x^k, x^{k-1}) \to 0$ due to the fact that $\|\nabla_1 F(x^k, x^k) - \nabla_1 F(x^k, y^{k-1})\|_2 \leq L\|x^k - y^{k-1}\|$. Similarly, $\nabla_2 F(x^k, x^k) - \nabla_2 F(y^{k-1}, y^{k-1}) \to 0$.

That, combined with (5.52) and (5.54), leads to

$$\left\langle \nabla_1 F(x^k, x^k) + \nabla_2 F(x^k, x^k), x - x^k \right\rangle \geq 0, \;\; k \to \infty \; \forall x \in \mathcal{X}.$$

Thus, $\{F(x^k)\}$ converges to $F(x^*)$, where x^* is a stationary point.

Furthermore, from (5.54) it leads to

$$\begin{aligned} &\left\langle \nabla_1 F(x^k, x^k) + \nabla_2 F(x^k, x^k), x - x^k \right\rangle \\ &\geq -(2L + \frac{1}{\mu_1^k})\|x - x^k\| \cdot \|x^k - y^{k-1}\|, \end{aligned} \tag{5.55}$$

where L is a Lipschitz constant for the gradients of F_1, F_2. For any $x \neq x^k$ and $x \in \mathcal{X}$, we can bound $\left\langle \nabla_1 F(x^k) + \nabla_2 F(x^k), \dfrac{x - x^k}{\|x - x^k\|} \right\rangle$ from below by $-(2L + \dfrac{1}{\mu_1^k})\|x^k - y^{k-1}\|$.

Denote $\mu^{max} = \max\limits_{1 \leq i \leq k}\{\mu_1^i\}, \mu^{min} = \min\limits_{1 \leq i \leq k}\{\mu_1^i\}$. From monotonicity of the function value (5.49) and (5.50), we have

$$F(x^{i-1}) - F(x^i) \geq F(y^{i-1}) - F(x^i) \geq \frac{1}{2\mu_1^i}\|x^i - y^{i-1}\|^2,$$

and further

$$\begin{aligned} &F(x^0) - F(x^*) \\ &\geq F(x^0) - F(x^k) \\ &\geq \sum_{i=1}^{k} \frac{1}{2\mu_1^i}\|x^i - y^{i-1}\|^2 \\ &\geq \sum_{i=1}^{k} \frac{1}{2\mu^{max}}\|x^i - y^{i-1}\|^2 \\ &\geq \frac{k}{2\mu^{max}}(\hat{g}_k)^2. \end{aligned} \tag{5.56}$$

Thus, we have

$$\hat{g}_k \leq \sqrt{\frac{2\mu^{max}(F(x^0) - F(x^*))}{k}}. \tag{5.57}$$

Consider a subsequence $\{\hat{x}_k\}$, where $\hat{x}_k \in \bar{\mathcal{I}}_k$. Combining (5.55) and (5.57), we come to the following inequality for any $x \in \mathcal{X}$:

$$\begin{aligned}
&\left\langle \nabla_1 F(\hat{x}_k, \hat{x}_k) + \nabla_2 F(\hat{x}_k, \hat{x}_k), \frac{x - \hat{x}_k}{\|x - \hat{x}_k\|} \right\rangle \\
&\geq -(2L + \frac{1}{\mu^{min}})\hat{g}_k \\
&\geq -(2L + \frac{1}{\mu^{min}})\sqrt{\frac{2\mu^{max}(F(x^0) - F(x^*))}{k}}.
\end{aligned} \tag{5.58}$$

While the right-hand-side of (5.58) goes to zero at $\mathcal{O}(\frac{1}{\sqrt{k}})$, it follows that $\nabla F(\hat{x}_k)$ converges to first-order optimality at a sublinear rate.

When $\mathcal{X}$ is $\mathbb{R}^n$, by choosing $x = \hat{x}_k - \frac{\nabla F(\hat{x}_k)}{\|\nabla F(\hat{x}_k)\|}$, we have

$$\|\nabla F(\hat{x}_k)\| \leq (2L + \frac{1}{\mu^{min}})\sqrt{\frac{2\mu^{max}(F(x^0) - F(x^*))}{k}}, \tag{5.59}$$

which recovers the standard complexity result in the unconstrained case, i.e. it takes $\mathcal{O}(\frac{1}{\sqrt{k}})$ iterations for the norm of $\nabla F(\hat{x}_k)$ to go to zero.

□

5.2.5 Practical ALM with backtracking and skipping

In the previous section we discuss the alternating linearization framework. At each iteration, the sufficient function decrease is obtained by the condition that the resulting function value is no larger than the approximation function value. In the application that we are interested in, a prox parameter μ can be conveniently found so that such condition can be satisfied.

Algorithm 15 Alternating linearization method with backtracking (ALM)

```
1. Choose $\mu_1^0 = \mu_2^0 = \mu^0$, and $x^0 = y^0$;
2. for $k = 0, 1, ...$
        (a) $x^{k+1} := \arg\min_{x \in \mathcal{X}} Q^1_{\mu^k}(x, y^k)$;
        (b) if $F(x^{k+1}) \leq Q^1_{\mu_1^k}(x^{k+1}, y^k)$ then
                $\mu_1^{k+1} := \mu_1^k$;
            else
                find the smallest $n$ s.t. $\bar{\mu} := \mu_1^k \beta^n, \bar{x} := \arg\min_{x \in \mathcal{X}} Q^1_{\bar{\mu}}(x, y^k)$ and $F(\bar{x}) \leq Q^1_{\bar{\mu}}(\bar{x}, y^k)$;
                $\mu_1^{k+1} := \mu_1^k \beta^n$, $x^{k+1} := \arg\min_{x} Q^1_{\mu_1^{k+1}}(x, y^k)$;
        (c) $y^{k+1} := \arg\min_{y \in \mathcal{X}} Q^2_{\mu_2^k}(x^{k+1}, y)$;
        (d) if $F(y^{k+1}) \leq Q^2_{\mu_2^k}(x^{k+1}, y^{k+1})$ then
                $\mu_2^{k+1} := \mu_2^k$;
            else
                find the smallest $n$ s.t. $\bar{\mu} := \mu_2^k \beta^n, \bar{y} := \arg\min_{y \in \mathcal{X}} Q^2_{\bar{\mu}}(x^{k+1}, y)$ and $F(\bar{y}) \leq Q^2_{\bar{\mu}}(x^{k+1}, \bar{y})$;
                $\mu_2^{k+1} := \mu_2^k \beta^n$, $y^{k+1} := \arg\min_{y \in \mathcal{X}} Q^2_{\mu_2^{k+1}}(x^{k+1}, y^{k+1})$.
```

However, in some cases, finding μ that satisfies function reduction conditions may be time consuming, especially when one of the two block gradients has a large Lipschitz constant. When this happens to only one block, we can simply skip the step related to that block and still obtain the convergence. This idea was first proposed in [34] for convex composite case.

When skipping step is constantly applied to one block, we have what we call a partial linearization method (PLM), presented in Algorithm 16. The term "partial" is in a sense that only one block of variables is linearized. In the next section, we will give a specific interpretation when the second-order least-squares problem is considered.

Algorithm 16 Partial linearization method (PLM)

```
1. Choose $\mu^0$, $\beta \in (0,1)$ and $x^0$;
2. for $k = 0, 1, ...$
      (a) $x^{k+1} := \arg\min_{x \in \mathcal{X}} Q^1_{\mu^k}(x, x^k)$;
      (b) if $F(x^{k+1}) \leq Q^1_{\mu^k}(x^{k+1}, x^k)$ then
            $\mu^{k+1} := \mu^k$;
         else
               find the smallest $n$ s.t. $\bar{\mu} := \mu^k \beta^n, \bar{x} := \arg\min_{x \in \mathcal{X}} Q^1_{\bar{\mu}}(x, x^k)$ and $F(\bar{x}) \leq Q^1_{\bar{\mu}}(\bar{x}, x^k)$;
            $\mu^{k+1} := \mu^k \beta^n$, $x^{k+1} := \arg\min_{x \in \mathcal{X}} Q^1_{\mu^{k+1}}(x, x^k)$; go to (b).
```

5.2.6 Connection between ALM and ADMM.

Let us establish the connection between Algorithms 13 (ADMM) and 14 (ALM). Both algorithm perform optimization of two convex functions on each step. In particular, given y^k and λ^k ADMM optimizes

$$\mathcal{L}_A(x, y^k; \lambda^k) = F_1(x, y^k) - \left\langle \lambda^k, x - y^k \right\rangle + \frac{1}{2\mu}\|x - y^k\|^2,$$

while ALM optimizes

$$Q^1_\mu(x, y^k) = F_1(x, y^k) + \left\langle \nabla_2 F(y^k), x - y^k \right\rangle + \frac{1}{2\mu}\|x - y^k\|^2.$$

Hence the two steps are identical if $\lambda^k = -\nabla_2 F(y^k) \equiv -\nabla_2 F(y^k, y^k)$ and then same value of μ is used. Consider the optimality conditions satisfied by y^k, since it is an unconstrained minimizer of

$$\mathcal{L}_A(x^k, y; \lambda^{k-1}) = F_2(x^k, y) - \left\langle \lambda^{k-1}, y - x^k \right\rangle + \frac{1}{2\mu}\|x^k - y\|^2,$$

which is

$$\nabla_y F_2(x^k, y^k) + \lambda^{k-1} + \frac{1}{\mu}(y^k - x^k) = 0,$$

Hence, from the update rule of λ^k, we have

$$\lambda^k = \lambda^{k-1} - \frac{1}{\mu}(x^k - y^k) = -\nabla_y F_2(x^k, y^k) \equiv -\nabla_2 F(x^k, y^k).$$

Similarly, if the order of subproblems is reversed in Algorithms 13, and if $\mathcal{X} = \mathbb{R}^n$, then

$$\lambda^k = -\nabla_1 F(x^k, y^k).$$

Hence, ADMM can be regarded as an inexact version of ALM, where λ is not updated as the exact partial gradient but some "mixture" of gradient information. In practice, the update of multipliers in ADMM is less costly than computing the gradient information in ALM, but as a results ADMM does not guarantee a descent step. In our analysis this difference inflicts limitation on ADMM convergence results to the case of constant, sufficiently small parameter μ, while in the case of ALM we were able to analyze the case where μ^k is a variable parameter chosen to satisfy sufficient decrease condition.

5.3 The connection between A/PLM and Levenberg-Marquardt method.

If we consider problem (5.2) simply as a nonlinear least squares problem, then we can apply classical methods such as Gauss-Newton method and Levenberg-Marquardt (LM) method [72]. By exploiting the structure of the least-squares problems, these methods compute exact gradient and partial Hessian information. In particular, they approximate the Hessian using Jacobian of the constraints (the functions inside the squares) and omit the second order information from those functions, since in many cases the contribution of this second order information is negligible compared with the Jacobian induced Hessian approximation. In this section, we relate these classic methods and our alternating linearization schemes, in the second-order least-squares setting. We show that our ALM method is a simplified version of the LM method.

We will consider the least-squares problem (5.2) instead of the more general form (5.4) for simplicity of notation. We can write (5.2) as follows:

$$\min_{x \in \mathcal{X}} F = \sum_{i=1}^{m} r_i^2(x) = \sum_{i=1}^{m} (x^\top M_i x - c_i)^2, \tag{5.60}$$

where $r(x)$ is a residual vector and $r_i(x) = x^\top M_i x - c_i$ is the ith residual. Then the Jacobian of r function is:

$$\begin{aligned} J(x) &= \begin{bmatrix} (\frac{\partial r_1(x)}{\partial x})^\top \\ \vdots \\ (\frac{\partial r_m(x)}{\partial x})^\top \end{bmatrix} \\ &= \begin{bmatrix} (M_1 x + M_1^\top x)^\top \\ \vdots \\ (M_m x + M_m^\top x)^\top \end{bmatrix}. \end{aligned} \tag{5.61}$$

Both Gauss-Newton method and Levenberg-Marquardt method store the Jacobian matrix after each iteration and compute the gradient as

$$\begin{aligned} \nabla F(x) &= J(x)^\top r(x) \\ &= \sum_{i=1}^{m} (x^\top M_i x - c_i)(M_i x + M_i^\top x). \end{aligned} \tag{5.62}$$

Furthermore, the Hessian can be written explicitly as

$$\begin{aligned}
&\nabla^2 F(x) = J(x)^\top J(x) + \sum_i r_i(x)\nabla^2 r_i(x) \\
&= \begin{bmatrix} (M_1 + M_1^\top)x & \dots & (M_n + M_n^\top)x \end{bmatrix} \begin{bmatrix} (M_1 x + M_1^\top x)^\top \\ \vdots \\ (M_m x + M_m^\top x)^\top \end{bmatrix} \\
&+ \sum_{i=1}^m (x^\top M_i x - c_i)(M_i + M_i^\top).
\end{aligned} \tag{5.63}$$

In the case of both, Gauss-Newton and Levenberg-Marquardt, methods the second term in (5.63) is ignored. This gives an efficient and relatively accurate approximation of the Hessian, in the cases when the magnitude of the second term in (5.63) is significantly smaller than that of the first. In particular, this is the case, when the residual $x^\top M_i x - \theta$ is approximately zero for all i. For instance, the risk parity problems (when risk parity exists) belong to this case.

Levenberg-Marquardt method is considered more robust than Gauss-Newton because it adds a positive definite matrix to $J(x)^\top J(x)$, (usually an identity matrix multiplied by a scalar) and solves the modified subproblem. Specifically, at each iteration, it solves

$$x^{k+1} = \arg\min_x F(x^k) + \nabla F(x^k)^\top (x - x^k) + \frac{1}{2}(x - x^k)^\top [J(x)^\top J(x)](x^k) + \frac{1}{\mu} I)(x - x^k). \tag{5.64}$$

The relationship between Levenberg-Marquardt method and our alternating linearization method can be analyzed as follows. Recall our partial linearization method

(Algorithm 16). At each iteration, the following function is minimized:

$$Q_{\mu}^{1}(x) = F(x, x^k) + \nabla_2 F(x^k)^{\top}(x - x^k) + \frac{1}{2\mu}\|x - x^k\|^2, \tag{5.65}$$

and

$$F(x, x^k) = F(x^k) - \sum_i [(x^k)^{\top} M_i^{\top}(x - x^k)]^2 + 2\sum_i (x^{\top} M_i x^k - c_i)(x^k)^{\top} M_i^{\top}(x - x^k).$$

Note that

$$\begin{aligned}
&2\sum_i (x^{\top} M_i x^k - c_i)(M_i x^k) \\
&= 2\sum_i ((x^k)^{\top} M_i x^k - c_i)(M_i x^k) + 2\sum_i ((x - x^k)^{\top} M_i x^k - c_i)(M_i x^k) \\
&= \nabla_1 F(x^k) + 2\sum_i ((x - x^k)^{\top} M_i x^k - c_i)(M_i x^k)
\end{aligned}$$

and recall that $\nabla_2 F(x^k) + \nabla_1 F(x^k) = \nabla F(x^k)$. Thus, function $Q_{\mu}^{1}(x)$ in (5.65) can be written as

$$Q_{\mu}^{1}(x) \;\; = \;\; F(x^k) + \nabla F(x^k)^{\top}(x - x^k) + G(x),$$

where the first two terms are exactly the same as (5.64), and $G(x)$ contains all the remaining terms of $Q_{\mu}^{1}(x)$, and, hence,

$$G(x) = \sum_i (x - x^k)^{\top}[M_i x^k (x^k)^{\top} M_i^{\top} + \frac{1}{2\mu} I](x - x^k).$$

Recall that, in Levenberg-Marquardt method,

$$J(x^k)^{\top} J(x^k) = \sum_i (M_i x^k + M_i^{\top} x^k)(M_i x^k + M_i^{\top} x^k)^{\top}$$

Hence, our linearization approach can be regarded as a cheap variant of Levenberg-Marquardt method when only part of the second-order information is considered.

For alternating linearization method, the analysis is similar, in which case partial Hessian approximations $M_i x^k (x^k)^\top M_i^\top$ and $M_i^\top x^k (x^k)^\top M_i$ are used in an alternating manner. Specifically, for each subproblem,

$$Q^1_{\mu_1}(x) = F(y^k) + \nabla F(y^k)^\top (x - y^k) + \sum_{i=1}^{m} (x - y^k)^\top [M_i y^k (y^k)^\top M_i^\top + \frac{1}{2\mu_1} I](x - y^k)$$

$$Q^2_{\mu_2}(y) = F(x^{k+1}) + \nabla F(x^{k+1})^\top (y - x^{k+1})$$
$$+ \sum_{i=1}^{m} (y - x^{k+1})^\top [M_i^\top x^{k+1} (x^{k+1})^\top M_i + \frac{1}{2\mu_2} I](y - x^{k+1}),$$

where μ_1, μ_2 are positive scalars.

5.4 Relaxations and lower bounds

We have presented several algorithms based on alternating direction and variable splitting techniques. All these algorithms are local algorithms, hence they provide no guarantee that a global minimum will be found. Solutions obtained by these methods provide an upper bound of the global minimum. In this section, we apply relaxation schemes in an attempt to find a (possibly) good lower bound of the global optimum. In particular, in the case of risk parity problems, when risk parity exists, the global optimal solution of (5.1) is zero, otherwise it is positive. Assume that our ALM or ADMM methods find a local minimum with a positive value of the objective function. If a lower bound can be computed which is positive, this will provide a certification that a risk parity solution does not exists. Clearly, if the lower bound is equal to the local optimal value, then this certifies that a global optimum has actually been obtained.

5.4.1 SOS relaxations

The objective function of (5.1) is a quartic polynomial. Hence recent advances in polynomial optimization are generally applicable for our problem (see, for instance, [49] for a review). It is well known that polynomial optimization problem can be reduced to the problem of determining whether a polynomial is nonnegative (we expand on this below). It is also known that a polynomial is nonnegative if it has a sum-of-squares (SOS) decomposition (while the reverse in not true, examples can be seen in [59] for instance). Existence of an SOS for a given polynomial, can, in turn, be determined by solving a semidefinite programming (SDP) feasibility problem, and thus is considered as tractable for small to medium scaled problem [55]. Hence, SOS can be used to compute a global lower bound for a nonconvex optimization problem.

Consider a multivariate polynomial $p(x_1, ..., x_n) \triangleq p(x)$ of degree $2d$. Then $p(x)$ is representable as an SOS if there exists a positive semidefinite matrix Q, such that

$$p(x) = z^\top Q z, \tag{5.66}$$

where $z = [1, x_1, x_2, ..., x_n, x_1x_2, ..., x_n^d]^\top$. The size of z is $\begin{pmatrix} n+d \\ d \end{pmatrix}$. It can be shown that (5.66) is equivalent to $p(x) = \sum_i f_i^2(x)$, where $f_i(x)$ are polynomials [55]. Note that the matrix Q may not be unique.

Example 5.4.1. *Recall the objective function in our least-squares formulation in risk parity problem:* $p(x, \theta) = \sum_{i=1}^{n} (x^\top M_i x - \theta)^2$. *In order to have the form* (5.66)*, we can* *first homogenize the degree of the polynomial by substituting* $\theta = \frac{1}{n}\sum_{j=1}^{n} x^\top M_j x$ *(more on validity of that choice below). Also, note that, for any* $M \in \mathbb{R}^{n\times n}$*, there always exists a vector* $p \in \mathbb{R}^n$ *such that* $x^\top M x = p^\top \bar{x}$*, where* $\bar{x} = [x_1^2, x_1x_2, ..., x_1x_n, x_2^2, ..., x_n^2]^\top$ *is a vector of all the monomials of degree 2. The size of* $\bar{x}$ *is* $\begin{pmatrix} n+1 \\ 2 \end{pmatrix}$*. Then we have:*

$$\begin{aligned}
& \sum_{i=1}^{n}(x^\top M_i x - \frac{1}{n}\sum_{j=1}^{n} x^\top M_j x)^2 \\
= \ & \sum_{i=1}^{n}(x^\top M_i x)^2 - \frac{1}{n}(\sum_{j=1}^{n} x^\top M_j x)^2 \\
= \ & \sum_{i=1}^{n} \bar{x}^\top p_i p_i^\top \bar{x} - \frac{1}{n}\bar{x}^\top(\sum_{j=1}^{n} p_i)(\sum_{j=1}^{n} p_i)^\top \bar{x} \\
= \ & \bar{x}^\top Q \bar{x},
\end{aligned}$$

where $Q = \sum_{i=1}^{n} p_i p_i^\top - \frac{1}{n}(\sum_{j=1}^{n} p_i)(\sum_{j=1}^{n} p_i)^\top$*, and the positive definiteness of Q follows naturally from Cauchy-Schwarz.*

All of the problems of form of (5.2)) considered in this chapter can be reformulated into (5.66). The existence of an SOS decomposition of a polynomial in n variables of degree $2d$ can be decided efficiently by solving a semidefinite programming feasibility problem [54].

Example 5.4.2. *Consider the following* 2×2 *risk parity problem. The covariance matrix is given by*

$$\Sigma = \begin{bmatrix} 1 & \\ & 4 \end{bmatrix}.$$

Thus, the objective function is given by $F(x) = \sum_{i=1}^{2}(x_i(\Sigma x)_i - \frac{1}{2}x^\top \Sigma x)^2 = \frac{1}{2}(x_1^2 -$

$4x_2^2)^2$. *Thus, we can write*

$$\begin{aligned}
F(x) &= \frac{1}{2}(x_1^2 - 4x_2^2)^2 \\
&= [x_1^2 \; x_1x_2 \; x_2^2] \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{12} & q_{22} & q_{23} \\ q_{13} & q_{23} & q_{33} \end{bmatrix} \begin{bmatrix} x_1^2 \\ x_1x_2 \\ x_2^2 \end{bmatrix} \\
&= q_{11}x_1^4 + 2q_{12}x_1^3x_2 + (2q_{13} + q_{22})x_1^2x_2^2 + 2q_{23}x_1x_2^3 + q_{33}x_2^4.
\end{aligned}$$

Then we can determine an SOS decomposition by equating the corresponding coefficients and thus obtaining the following SDP feasibility problem:

$$\textit{Find } Q \succeq 0, \quad \textit{s.t. } q_{11} = 1, \; 2q_{13} + q_{22} = -8, \; q_{33} = 16.$$

Global bounds for polynomial functions

SOS technique can be used to compute the global lower bounds for polynomial functions. Consider the following problem:

$$\begin{aligned} \min \quad & F(x) \\ \text{s.t.} \quad & x \in \mathcal{X}. \end{aligned} \tag{5.67}$$

Problem (5.67) is equivalent to

$$\begin{aligned} \max \quad & \gamma \\ \text{s.t.} \quad & F(x) - \gamma \geq 0, \quad \forall x \in \mathcal{X}. \end{aligned} \tag{5.68}$$

Further, (5.68) can be approximated by

$$\begin{aligned} \max \quad & \gamma \\ \text{s.t.} \quad & F(x) - \gamma \text{ is SOS}, \quad \forall x \in \mathcal{X}. \end{aligned} \tag{5.69}$$

Obviously, the optimal value of (5.69) is a lower bound for the global minimum of the original problem (5.67).

If a polynomial is SOS, then it is nonnegative for any x. To take the constraints $x \in \mathcal{X}$ into account and hence to strengthen the lower bound on $F(x)$ the concept of Schmdgen Positivstellensatz can be applied.

Theorem 5.4.3. *(Schmdgen Theorem, '1991, [61]) Let $K = \{x \in \mathbb{R}^n : g_1(x) \geq 0, ..., g_m(x) \geq 0\}$ be a compact set. If a polynomial $p(x)$ is positive on K, then $p(x) \in P(g_1, ..., g_m)$, where*

$$P(g_1, ..., g_m) = \left\{ \sum_{\nu \in \{0,1\}^m} \sigma_\nu(x) g_1(x)^{\nu_1} ... g_m(x)^{\nu_m} : \textit{ each } \sigma_\nu \textit{ is SOS} \right\}.$$

Suppose we have the following constrained optimization problem:

$$\begin{aligned} \min \quad & F(x) \\ \text{s.t.} \quad & g_i(x) \geq 0, \quad i = 1, ..., m_1 \\ & h_j(x) = 0, \quad j = 1, ..., m_2. \end{aligned} \tag{5.70}$$

Then a lower bound of $F(x)$ can be computed by:

$$\begin{aligned}\max \quad & \gamma \\ \text{s.t.} \quad F(x)-\gamma &= \sigma_0(x)+\sigma_1(x)g_1(x)+...+\sigma_{m_1}(x)g_{m_1}(x) \\ & +\sigma_{12}(x)g_1(x)g_2(x)+...+\sigma_{12...m}(x)g_1(x)...g_{m_1}(x)+\sum_{j}^{m_2}\lambda_j(x)h_j(x),\end{aligned} \tag{5.71}$$

where $\sigma_i(x)$'s are a set of SOSs and $\lambda_j(x)$'s are a set of polynomials. Similarly to the unconstrained case, we can find such decomposition by solving an SDP. In fact, the SOSTOOLS that we apply for an implementation for risk parity optimization in the next section solves (5.71) to find a global bounds. We refer interested readers to [57] for more details.

As an alternative of Schmdgen's Positivstellensatz, we can apply Putinar's Positivstellensatz which needs a stronger assumption but also has a stronger conclusion. For a detailed review on Positivstellensatz and its applications, interested readers can refer to [49].

Theorem 5.4.4. *(Putinar Theorem, '1993, [58]) Let $K = \{x \in \mathbb{R}^n : g_1(x) \geq 0, ..., g_m(x) \geq 0\}$ be a compact set, and the quadratic module be defined as*

$$M(g_1,...,g_m) = \left\{ \sum_{i=1}^{m} \sigma_i(x)g_i(x) : \text{ each } \sigma_i \text{ is SOS} \right\}.$$

Suppose there exists N such that $N - \|x\|_2^2 \in M(g_1,...,g_m)$. If $p(x)$ is positive on K, then $p(x) \in M(g_1,...,g_m)$.

The condition $N - \|x\|_2^2 \in M(g_1,...,g_m)$ is called archimedean condition. Note that the risk parity optimization problem satisfies the archimedean condition. The reason is that in portfolio optimization the leveraging level is always bounded, which means the weights always satisfy $\sum_{i=1}^{n} x_i^2 \leq R$ for some large enough R and the archimedean condition holds.

Theorem 5.4.4 states that, under certain assumptions, a polynomial positive on K can be represented in Putinar's way as long as the degree of each σ_i is high enough. There exist upper bounds on the degree, for instance see [53], however, these bounds usually do not present a practical approach, as high degree SOS problems lead to very large SDPs. In practice, the maximum degree of the SOS is increased sequentially and for each degree an SOS approach is applied and an SDP is solved. This is referred to as increasing the hierarchy of SOS relaxations.

5.4.2 DSOS and SDSOS optimization: alternatives to SOS optimization

While computing SOS decomposition is generally tractable, the size of resulting SDP problem grows very quickly with the original dimension and the degree of the polynomial. Hence further relaxations, via diagonally dominant sum-of-squares (DSOS) and scaled-diagonally-dominant-sum-of-squares (SDSOS) decompositions, which lead to linear programs and second order cone programs, respectively, have recently been proposed [3]. The cones of polynomials that admit DSOS and SDSOS decompositions are subsets of the cone of SOS polynomials, but they lead to more tractable optimization problems.

Definition 5.4.5. *A symmetric matrix A is diagonally dominant (dd) if $a_{ii} \geq \sum_{j \neq i} |a_{ij}|$ for all i. A is scaled diagonally dominant (sdd) if there exists an elementwise positive vector y such that $a_{ii} y_i \geq \sum_{j \neq i} |a_{ij}| y_j$ for all i.*

Then two subsets of SOS cone, DSOS and SDSOS, are defined as follows.

Definition 5.4.6. (Ahmadi and Majumdar, '2013)

A polynomial p is diagonally-dominant-sum-of-squares (DSOS) if it can be written as

$$p = \sum_i \alpha_i m_i^2 + \sum_{i,j} \beta_{i,j}(m_i \pm m_j)^2,$$

for some monomials m_i, m_j and some nonnegative scalars $\alpha_i, \beta_{i,j}$.

A polynomial p is scaled-diagonally-dominant-sum-of-squares (SDSOS) if it can be written as

$$p = \sum_i \alpha_i m_i^2 + \sum_{i,j} (\beta_i m_i \pm \gamma_j m_j)^2,$$

for some monomials m_i, m_j and some constants $\alpha_i \geq 0, \beta_i, \gamma_i$.

Let $DSOS_{n,2d}$, $SDSOS_{n,2d}$, $SOS_{n,2d}$ and $PSD_{n,2d}$, denote cones of, respectively, DSOS, SDSOS, SOS and nonnegative polynomials of degree $2d$ in dimension n. It is clear that $DSOS_{n,2d} \subseteq SDSOS_{n,2d} \subseteq SOS_{n,2d} \subseteq PSD_{n,2d}$. It is shown in [3] that a polynomial p of degree $2d$ is DSOS (or SDSOS) if and only if it has a representation $p(x) = z^\top(x)Qz(x)$, where $z(x)$ is the standard monomial vector of degree d, and Q is a DD (or SDD) matrix. It has also been proven that the set of $DSOS_{n,2d}$ (or $SDSOS_{n,2d}$) has a polyhedral (second order cone) representation and thus the search over $DSOS_{n,2d}$ ($SDSOS_{n,2d}$) for a fixed d reduces to a linear programming (second-order cone programming) problem.

In the next section we will show that applying the SOS, DSOS and SDSOS relaxation to the risk parity problem produces useful lower bounds and in the case of SDSOS good lower bounds can be obtained efficiently.

5.5 Numerical results on risk parity portfolio selection problem

In this section, we use risk parity portfolio selection problem (5.6) (with $q(x) = 0$) as the specific application to perform numerical experiments. It was shown in [5] that if all $a_i = 0$, $b_i = 1$ for all i, then a unique risk parity solution exists. It was further shown in [5] that if $a_i < 0$ for some i then there may be multiple risk parity solutions and in the case when bounds are not tight at the solution, all local minima are global. Tighter box constraints, on the other hand, may result in no risk parity solution and multiple local optima may exist. We will demonstrate on some simple examples, that our alternating direction approaches find the global optimum in each case. In [5] these methods were used in case of multiple risk parity solutions and regularization term $q(x) = \frac{1}{2}x^\top \Sigma x$ and also produced global solutions.

5.5.1 A comparison of local alternating direction algorithms.

We compare algorithms described in Sections 5.2.3 and 5.2.4 on randomly generated data and real data sets.

In risk parity problem (5.6) θ is a free variable of dimension one. It is possible to derive simplified versions of ALM and PLM in such a way, that optimization over θ is performed as a separate step after optimization over x. In particular, θ can be updated by the exact minimization at the end of each iteration. We observe that,

$$\frac{\partial F}{\partial \theta} = -2\sum_{i=1}^{n}(x^\top M_i x - \theta),$$

which leads to the following simple update $\theta^k = \frac{1}{n}\sum_{i=1}^{n}(x^k)^\top M_i x^k$. Note that if such an update is used instead of simultaneous optimization over x and θ, then our ALM scheme performs three optimization steps over three blocks of variables instead of two and our theory may no longer apply. In fact, as our computational results show, updating θ separately leads to inferior results. Optimizing over x and θ simultaneously reduced overall number of iterations, while it does not result in substantially more difficult subproblems. Hence, this modification of PLM and ALM is undesirable both in theory and in practice. We simply include it here for comparison.

For both PLM and ALM, we have shown that the convergence holds if μ^k is chosen so that the function value at the new iterate is not larger than the value of the approximation function Q. Choosing small value of μ^k a priori will guarantee this, but will result in slow progress of the algorithm. Hence, we apply backtracking procedure to find an acceptable value of details about applying backtracking alternating linearization method for convex composite optimization can be found in [60].

In the case of augmented Lagrangian method and alternating direction augmented Lagrangian method (ADMM) μ should be selected sufficiently small and constant, according to our theory. In order to avoid computing the Lipschitz constant of the gradient we allow some parameter tuning to improve the results. In augmented Lagrangian method, we also allow inexact minimization for the subproblem at beginning iterations, to achieve fast convergence (see, e.g. [10]).

In what follows we compare the following algorithms.

1. ALM-θ - Algorithm 14 with separate θ update.

2. ALM - Algorithm 14 with simultaneous optimization over x and θ;

3. PLM-θ - Algorithm 16 with separate θ update;

4. PLM - Algorithm 16 with simultaneous optimization over x and θ;

5. AL-BCD - Augmented Lagrangian method with block coordinate descent method to solve the subproblem;

6. ADMM: Algorithm 13 with properly chosen μ;

Our implementations and experiments were performed in MATLAB R2013a on a laptop with Intel Core Duo 1.8 GHz CPU and 2GB RAM. Mosek 7.0 was applied to solve the QP subproblem. In [5], the basic versions of ALM were shown to be superior to MATLAB `fmincon`, hence we do not include these comparisons here. In Table 5.1 and Table 5.2, we compare the algorithms on random data. An arbitrary symmetric positive semidefinite matrix can be generated as $\Sigma = AA^\top$, where A_{ij} is uniformly distributed within the interval $[0, 1]$. Recall that the main computational cost of each algorithm lies in the number of quadratic models it solves as the subproblem. Hence, in each table we report the number of iterations and the total number of QP solved. We set the termination criterion to when the largest KKT violation falls below ϵ, with ϵ chosen to be 10^{-3}, 10^{-5}, etc. We recorded the number of the iterations and the number of subproblem (QP) solves it took each algorithm to reach this threshhold. The maximum iteration number is 10000.

Table 5.1: A comparison of algorithms on a randomly generated instance (20×20). The starting point is chosen to be equally weighted portfolio, i.e., $x_i^0 = 1/n$. The lower and upper bounds are chosen to be $a = 0, b = 1$. The starting μ is chosen to be 0.01.

Algorithm	iter. (10^{-3})	QP	F-value	iter. (10^{-5})	QP	F-value	iter. (10^{-7})	QP	F-value
PLM-θ	18	29	1.06×10^{-8}	32	50	1.28×10^{-12}	45	70	1.44×10^{-16}
PLM	7	7	2.67×10^{-9}	9	9	2.94×10^{-13}	11	11	2.53×10^{-17}
ALM-θ	12	40	2.20×10^{-8}	21	66	1.03×10^{-12}	28	89	1.93×10^{-16}
ALM	5	12	9.29×10^{-10}	6	14	1.11×10^{-12}	8	21	1.18×10^{-19}
AL-BCD	3	52	1.11×10^{-8}	5	94	3.56×10^{-14}	6	116	4.03×10^{-17}
ADMM	19	38	9.43×10^{-9}	33	66	1.27×10^{-12}	48	96	1.02×10^{-16}

Table 5.2: A comparison of algorithms with fixed steplengths on a randomly generated instance (200×200). The starting point is chosen to be equally weighted portfolio, i.e., $x_i^0 = 1/n$. The lower and upper bounds are chosen to be $a = -1, b = 2$. The starting μ is chosen to be 0.01.

Algorithm	iter. (10^{-3})	QP	F-value	iter. (10^{-5})	QP	F-value	iter. (10^{-7})	QP	F-value
PLM-θ	15	30	7.28×10^{-10}	27	50	5.65×10^{-14}	44	77	1.54×10^{-17}
PLM	4	4	1.51×10^{-10}	5	5	2.63×10^{-13}	7	7	1.16×10^{-18}
ALM-θ	14	55	7.33×10^{-10}	23	82	2.57×10^{-14}	28	100	2.16×10^{-18}
ALM	3	13	1.66×10^{-12}	4	16	3.53×10^{-17}	5	20	7.36×10^{-18}
AL-BCD	12	790	1.88×10^{-8}	17	1496	3.04×10^{-12}	22	2110	5.55×10^{-17}
ADMM	45	90	6.15×10^{-6}	92	184	9.84×10^{-12}	103	206	1.96×10^{-15}

Tables 5.1 and 5.2 contain results for randomly generated data of two different sizes. We observe that all algorithms find the global minimum. The AL-BCD method requires the least number of iterations, but the largest number of QP solves, since each iteration requires multiple such solves. Both PLM and ALM tend to require fewer QP solves, compared to ADMM and AL-BCD, which is likely the result of using the gradient information backtracking approach for selecting μ^k.

We also compare the algorithms on one data set created to simmulate difficult risk parity cases with 5 risky assets [5]. The covariance matrix of the percentage annual return is given by:

$$\Sigma = \begin{bmatrix} 94.868 & 33.750 & 12.325 & -1.178 & 8.778 \\ 33.750 & 445.642 & 98.955 & -7.901 & 84.954 \\ 12.325 & 98.955 & 117.265 & 0.503 & 45.184 \\ -1.178 & -7.901 & 0.503 & 5.460 & 1.057 \\ 8.778 & 84.954 & 45.184 & 1.057 & 34.126 \end{bmatrix}.$$

In Table 5.3 we present results for the case of lower and upper bounds set to 0 and 1, respectively. In this case risk parity solution exists and is unique [5]. In Table 5.4 we show the results for the case of tight upper and lower bounds, where a risk parity solution, satisfying these bounds, does not exists.

Finally we test the algorithms on real instances of covariance matrices of different sizes using risk parity model (5.6) and group risk parity model (5.7). These results are listed in Tables 5.5-5.7 and also include the case where risk parity does not exist.

All our results show that ALM and PLM are comparable to each other in terms of the number of QP solves and they both typically outperform other methods, such as ALM-θ and PLM-θ, as well as ADMM. The only exception in which ALM-θ outperforms ALM in terms of speed is in Table 5.7. However, ALM achieves a smaller objective function value (1.22×10^{-13}) than ALM-θ (5.41×10^{-13}).

Table 5.3: A comparison of algorithms on 5×5 instance. The starting point is chosen to be equally weighted portfolio, i.e., $x_i^0 = 1/n$. The lower and upper bounds are chosen to be $a = 0, b = 1$. The starting μ is chosen to be 0.1.

Algorithm	iter. (10^{-3})	QP	F-value	iter. (10^{-5})	QP	F-value	Final solution
PLM-θ	55	95	4.87×10^{-9}	77	131	1.34×10^{-12}	[0.125;0.047;0.083;0.613;0.132]
PLM	41	84	1.59×10^{-9}	50	101	5.41×10^{-14}	[0.125;0.047;0.083;0.613;0.132]
ALM-θ	47	174	3.08×10^{-9}	64	230	1.26×10^{-13}	[0.125;0.047;0.083;0.613;0.132]
ALM	20	82	1.65×10^{-9}	26	102	4.52×10^{-14}	[0.125;0.047;0.083;0.613;0.132]
AL-BCD	18	2058	7.04×10^{-13}	23	2744	5.67×10^{-16}	[0.125;0.047;0.083;0.613;0.132]
ADMM	91	182	3.67×10^{-9}	106	212	1.18×10^{-13}	[0.125;0.047;0.083;0.613;0.132]

Table 5.4: A comparison of algorithms on 5×5 instance. The starting point is chosen to be equally weighted portfolio, i.e., $x_i^0 = 1/n$. The lower and upper bounds are chosen to be $a = 0.05, b = 0.35$. The starting μ is chosen to be 0.1.

Algorithm	iter. (10^{-3})	QP	F-value	iter. (10^{-5})	QP	F-value	Final solution
PLM-θ	28	52	16.0344	61	87	16.0344	[0.204;0.060;0.130;0.350;0.256]
PLM	23	46	16.0344	29	55	16.0344	[0.204;0.060;0.130;0.350;0.256]
ALM-θ	19	79	16.0344	26	98	16.0344	[0.204;0.060;0.130;0.350;0.256]
ALM	14	62	16.0344	21	82	16.0344	[0.204;0.060;0.130;0.350;0.256]
AL-BCD	507	15786	16.0344	522	15846	16.0344	[0.204;0.060;0.130;0.350;0.256]
ADMM	309	618	16.0344	325	650	16.0344	[0.204;0.060;0.130;0.350;0.256]

5.5.2 Implementation of SOS optimization on risk parity optimization problem

In this section, we discuss the application of SOS relaxation and its variants discussed in Section 5.4 to risk parity. As mentioned above, it has been shown in [5], that all

Table 5.5: A comparison of algorithms on asset allocation instance (14×14). The starting point is chosen to be equally weighted portfolio, i.e., $x_i^0 = 1/n$. The lower and upper bounds are chosen to be $a = 0, b = 1$. The starting μ is chosen to be 1.

Algorithm	it. (10^{-3})	QP	F-value	it. (10^{-5})	QP	F-value	Final solution	
PLM-θ	127	212	8.61×10^{-8}	210	343	7.78×10^{-12}		
PLM	131	225	7.15×10^{-8}	215	363	8.62×10^{-12}	$x_1 \sim x_3:$	$0.0416, 0.0360, 0.0328$
ALM-θ	31	111	2.30×10^{-8}	45	159	2.95×10^{-12}	$x_4 \sim x_6:$	$0.0313, 0.0334, 0.0687$
ALM	10	58	3.40×10^{-8}	14	69	5.82×10^{-12}	$x_7 \sim x_9:$	$0.0249, 0.0506, 0.1661$
AL-BCD	42	1276	1.19×10^{-10}	44	1676	3.11×10^{-14}	$x_{10} \sim x_{12}:$	$0.2892, 0.0630, 0.0532$
ADMM	85	170	2.88×10^{-8}	106	212	6.26×10^{-13}	$x_{13} \sim x_{14}:$	$0.0760, 0.0332$

Table 5.6: A comparison of algorithms on asset allocation instance with tight bounds (14×14). The starting point is chosen to be equally weighted portfolio, i.e., $x_i^0 = 1/n$. The lower and upper bounds are chosen to be $a = 0.06, b = 0.08$. The starting μ is chosen to be 1.

Algorithm	it. (10^{-3})	QP	F-value	it. (10^{-5})	QP	F-value	Final solution	
PLM-θ	11	24	0.06645	17	32	0.06645		
PLM	11	24	0.06645	14	29	0.06645	$x_1 \sim x_3:$	$0.0594, 0.0500, 0.0500$
ALM-θ	7	34	0.06645	8	37	0.06645	$x_4 \sim x_6:$	$0.0500, 0.0500, 0.0984$
ALM	4	25	0.06645	5	27	0.06645	$x_7 \sim x_9:$	$0.0500, 0.0715, 0.1000$
AL-BCD	772	14432	0.06645	1211	16188	0.06645	$x_{10} \sim x_{12}:$	$0.1000, 0.0957, 0.0796$
ADMM	980	1960	0.06645	1508	3016	0.06645	$x_{13} \sim x_{14}:$	$0.1000, 0.0500$

unconstrained local optima of (5.6) are actually risk parity solutions. However, if a local solution has an active bound constraint, then this may be a local solution, as seen in some examples in Section 5.5.1, and it is unclear if it also is a global one. Here, we can apply SOS techniques to find a global lower bounds for such cases.

We use SOSTOOLS MATLAB toolbox for constructing and solving SOS relaxation [57]. SOSTOOLS reformulate SOSs as semidefinite programs (SDPs), which is then solved by a standard SDP solver such as SeDuMi or SDPT3.

Example 5.5.1. *Consider the 5×5 example introduced in Section 5.5.1. As shown in Table 5.4, by imposing sufficiently tight bound constraints, a risk parity solution is not reached. We now would like to verify that risk parity solution does not indeed exist, and, if possible, verify the global optimality of our local solution by constructing a lower bound. We used function* `findbound` *in SOSTOOLS. In this case, after reformulation, SDPT3 solved an SDP feasibility problem and gave a lower bound of*

Table 5.7: A comparison of algorithms on equity market instance (482×482) with group risk parity enforced. The starting point is chosen to be equally weighted portfolio, i.e., $x_i^0 = 1/n$. The lower and upper bounds are chosen to be $a = 0, b = 1$. The starting μ is chosen to be 0.1.

Algorithm	iter. (10^{-3})	QP	F-value	iter. (10^{-5})	QP	F-value
PLM-θ	38	66	4.62×10^{-9}	568	610	1.39×10^{-13}
PLM	47	88	8.45×10^{-9}	338	400	1.11×10^{-13}
ALM-θ	11	41	2.18×10^{-9}	101	228	5.41×10^{-13}
ALM	10	43	4.38×10^{-10}	261	556	1.22×10^{-13}
AL-BCD	19	184	1.75×10^{-10}	30	890	1.67×10^{-15}
ADMM	48	96	6.51×10^{-9}	77	154	7.64×10^{-13}

16.0344 which is equal to the objective value obtained by local algorithms, as is shown in Table 5.4.

While constructing an SOS representation of a nonnegative polynomial reduces to solving an SDP feasibility problem and can be done in polynomial time, the size of the resulting SDP grows very rapidly with the dimension and the degree of the polynomial. Moreover, in the constrained case, such as ours, we need to apply Positivstellensatz (5.71) which increases the size of the SDP even further. To improve efficiency of the SOS relaxations, we propose a simple adaptive strategy, where we add some of the box constraints to the formulation is a sequential manner.

Our simple sequential SOS method is stated as Algorithm 17 and the results of a comparison between applying SOSTOOLS to the original formulation and our adaptive strategy can be seen in Table 5.8. Instead of imposing all constraints at once, we first use our local algorithm to obtain a stationary point and thus have an initial “guess” which bound should be activated. We add the corresponding bounds to the formulation and solve the corresponding subproblem with SOSTOOLS and SeDuMi. Clearly each SOS subproblem provides a lower bound for our main problem, since it is a relaxation applied to a relaxed feasible set. SOSTOOLS also provides another solution x, which is a new stationary point. We then proceed by adding new constraints if they are violated by the new solutions x. Hence we obtain a sequence of tighter relaxations and nondecreasing lower bounds. In our experiments this simple strategy substantially reduced the size of the SDPs and the overall complexity. For instance, for the 5×5 case with bounds [0.05, 0.35] the original constrained problem results in a 126×2808 SDP (using SOSTOOLS). If we apply Algorithm 17, then we solve two subproblems whose sizes are 126×534 and 126×863, respectively. As can be observed in Table 5.8, this can significantly accelerate the algorithm.

We have tested the SOS approach on two 5×5 instances, in each case using two settings - tight bounds and the full $[0, 1]$ interval for each variable. Clearly, when the bounds are not tight, then risk parity solution exists and the lower bound is known to be 0 and hence there is no need for applying SOS tools. However, we use this setting for testing efficiency and accuracy of the SOS approach and Algorithm 17. In fact in some cases Algorithm 17 produced the accurate bound after solving only one SDP problem.

To demonstrate the challenge of using SOS we tested the 14×14 asset allocation instance with bounds to be $[0, 1]$. As we can see from Table 5.8, when we test the our approach find an inaccurate (positive) bound after more than 1 hour, while directly applying SOSTOOLS fails to provide an answer within 24 hours.

It is important to note that in order to apply Schmdgen Positivstellensatz in (5.71), we need the assumption that the solution lies in a compact set. Here, although we do not add all constraints at once, we can still assume, without loss of generality, that the solution lies in a large enough bounded set.

Algorithm 17 Sequential sos method for risk parity optimization

```
1. Apply local algorithms (ADMM or ALM) to solve (5.6) and obtain a stationary point $x^0$ and the corresponding objective function value $f(x^0)$;
2. Initialize the constraint set $\mathcal{X}^0$, where $\mathcal{X}^0$ contains only the bounds which are activated by $x^0$ and all the equality constraints.
3. for $k = 0, 1, ...$
        (a) Solve (5.71) by SOSTOOLS, and obtain new local solution $x^{k+1} \in \mathcal{X}^k$ and a global lower bound $\gamma^{k+1}$;
        (b) if $x^{k+1} \in \mathcal{X}^{true}$ then
            break, and $\gamma^{k+1}$ is a global lower bound for $f$;
          else
            add bounds that are violated by $x^{k+1}$ and obtain new feasible set $\mathcal{X}^{k+1}$.
Go to (a).
```

Table 5.8: A comparison of algorithms on instances with different bounds. Original SOSTOOLS application (denoted as Org. in the table) is compared with relaxed sequential algorithm (Rel.). The default tolerance on duality gap is set to 10^{-8}).

Instance		CPU time (s)		Global lower bounds		
Name (size)	Bounds	Org.	Rel.	Org.	Rel.	Opt.
5 assets-scaled (5)	[0.05, 0.35]	160.13	16.94	16.03	16.03	16.03
5 assets-unscaled (5)	[0, 1]	115.83	24.34	2.48×10^{-10}	1.54×10^{-11}	0
Rand I (5)	[0.15, 0.25]	152.97	8.46	2.01×10^{-3}	2.01×10^{-3}	2.01×10^{-3}
Rand I (5)	[0, 1]	115.98	5.65	2.70×10^{-10}	9.28×10^{-10}	0
Asset Allocation (14)	[0,1]	-	4855.32	-	2.12×10^{-7}	0

DSOS and SDSOS optimization

As discussed in Section 5.4, we can apply DSOS and SDSOS relaxations as potential alternatives of SOS. The hope is that these techniques could provide us with a quality bound at a much smaller computational cost.

Specifically, *SDSOS* approach reduces to the following problem when seeking a

Table 5.9: A comparison of DSOS and SDSOS approach, on solving a 5×5 example. We compare the final lower bound found with a increase of the degree.

Method	Bound ($d = 4$)	CPU Time	Bound ($d = 6$)	CPU Time	OPT
DSOS	9.983	2.198	14.958	12.858	16.0344
SDSOS	16.021	1.986	16.027	11.142	16.0344

lower bound for risk parity problem:

$$\begin{aligned} \max \quad & \lambda \\ \text{s.t.} \quad & \sum_{i=1}^{n}(x_i(\Sigma x)_i - \frac{1}{n}x^\top \Sigma x)^2 - \lambda - \sum_{i=1}^{n} L(x)g(x) - r(x)(\sum_{i=1}^{n} x_i - 1) \in SDSOS_{n,d} \\ & L(x) \in SDSOS_{n,d}, \end{aligned} \tag{5.72}$$

where n is the dimension (the number of assets), d is the degree of $SDSOS$.

The result using $DSOS$ and $SDSOS$ solving the 5×5 instance is listed in Table 5.9. We used package called SPOTless (see: github.com/spot-toolbox/spotless) to model the $DSOS$ and $SDSOS$ cones and apply Mosek 7.0 to solve the resulting convex optimization problem (LP and SOCP, respectively). We can see that DSOS approximation gave a poor bound (but positive) when the maximum degree was set to 4 and improved the bound to optimal when the degree was increased to 6. SDSOS relaxation obtained good results at (surprisingly) slightly lower computational cost. Further extensive testing is needed to explore the usefulness of these approaches and is a subject of future research.

5.6 Conclusion

In this chapter, we proposed a family of alternating direction methods for minimizing nonlinear nonconvex problems with special structure which allows convenient 2-block variable splitting. In particular these methods apply to minimizing sums of squares of quadratic functions. We propose an alternating directions method of multipliers and an alternating linearization method and we provide convergence rate results for both classes of methods. The experiments on risk parity optimization problem shows the efficiency of these methods and their ability to recover a global minimum for this application. Global optimization techniques from polynomial optimization literature are applied to complement our local methods and to provide lower bounds. Exploring new applications of our methods is subject of future study.

Chapter 6

Alternating direction methods for the optimal power flow problem

6.1 Introduction

Optimal power flow (OPF) problem, one of the most important and widely studied problems in the power system, aims to minimize a cost function subject to power flow and operational constraints. This is a challenging problem due to the fact that the feasible region is nonconvex and thus is hard to tackle. In fact, it has been shown in [44] that OPF is NP-hard in the worst case.

Research in recent years has seen a great amount of efforts in applying global optimization techniques in OPF problem. In [6], OPF problem can be convexified by dropping the rank-one constraint and solved using standard semidefinite programming (SDP) approach. Lavaei and Low solve the dual of the original nonconvex OPF, which is an SDP. Furthermore, they show that, for the standard IEEE benchmark library, a necessary and sufficient condition is satisfied such that the duality gap for the OPF problem is zero [44]. More recently, polynomial optimization techniques have also been considered in [32].

The limitations of SDP type of approaches lie in the fact that the relaxation is not always exact, which may lead to an infeasible solution. As an alternative to the route, efficient local methods have been proposed in several papers. In particular, You and Peng suggest a nonconvex alternating direction method of multipliers (ADMM) for OPF [76]. ADMM is a well-known method that has been widely studied by the convex optimization community. The OPF problem fits the alternating direction method because it is natural to separately deal with the convex constraints and nonconvex rank-one constraint. However, the convergence results of ADMM applied to nonconvex problems have not been studied extensively before. Hence, in spite of the effectiveness indicated by the numerical results, there is no theoretical guarantee that the method will converge.

In this chapter we propose an algorithmic framework based on classic augmented Lagrangian function. Our framework consists of two alternating direction schemes to solve the subproblem, including the well-known alternating direction method of multipliers and alternating linearization method. Our focus and main contribution is to provide an optimization method for the nonconvex OPF that can be guaranteed to converge to satisfy KKT optimality conditions. Moreover, our method is convergent to a local minimum and can always recover a rank-one solution within some predetermined tolerance.

The rest of the chapter is organized as follows. After a brief discussion of the problem structure in Section 6.2, we discuss a class of algorithms based on augmented Lagrangian function and their convergence properties in Section 6.3. In Section 4, we discuss the advantages and disadvantages of our ADMM and propose an alternative approach. We show experiments and computational results in Section 5, followed with conclusion remarks.

6.2 The optimal power flow problem

6.2.1 Notations and preliminaries

In this chapter, we use the same notation as in [44] and [32]. . Consider a power system $P = (N, E)$, where N is the set of buses and E is the set of edges connecting the buses. Note that $E \subseteq N \times N$. Also, let $G \subseteq N$ be the set of generators. For each bus $k \in N$, let $V_k = \Re[V_k] + j\Im[V_k]$ be the complex voltage. Let $S_k^d = P_k^d + jQ_k^d$ denote the net complex power load at each bus $k \in N$ and $P_k^g + jQ_k^d$ be the power generated at each bus $k \in G$, and $S_{lm} = P_{lm} + jQ_{lm}$ be the apparent power flow on the edge $(l, m) \in E$. The network admittance matrix is denoted as $y \in \mathbb{C}^{|N| \times |N|}$.

We define the parameters of the system as follows.

- $P_k^{\min}$, $P_k^{\max}$, $Q_k^{\min}$ and $Q_k^{\max}$ are the generation capacity limits (either active or reactive) at each bus $k \in G$.
- $V_k^{\min}$ and $V_k^{\max}$ are the limits of the absolute value of the voltage at each bus $k \in N$.
- $S_{lm}^{\max}$ and $P_{lm}^{\max}$ are the limit of the apparent power flow on each edge $(l, m) \in E$.
- $\triangle V_{lm}^{\max}$ is the line capacity limit on edge $(l, m) \in E$.

Thus, the classical OPF problem aims to minimize the total cost of power generation as follows.

$$
\begin{aligned}
\min \quad & \sum_{k \in G} f_k(P_k^g) & & \\
\text{s.t.} \quad & P_k^{\min} \leq P_k^g \leq P_k^{\max}, & & \forall k \in G \\
& Q_k^{\min} \leq Q_k^g \leq Q_k^{\max}, & & \forall k \in G \\
& V_k^{\min} \leq |V_k| \leq V_k^{\max}, & & \forall k \in N \\
& |S_{lm}| \leq S_{lm}^{\max}, & & \forall (l,m) \in E \\
& |P_{lm}| \leq P_{lm}^{\max}, & & \forall (l,m) \in E \\
& |V_l - V_m| \leq \triangle V_{lm}^{\max}, & & \forall (l,m) \in E.
\end{aligned}
\tag{6.1}
$$

The objective function contains a sum of the generation cost from each generator. The cost from each generator, with c_k^2, c_k^1, c_k^0 nonnegative, is often defined as

$$f_k(P_k^g) = c_k^2 (P_k^g)^2 + c_k^1 (P_k^g) + c_k^0.$$

6.2.2 Reformulation

It has been shown in [44] that problem (6.1) can be reformulated as follows

$$\begin{aligned}
\min \quad & \sum_{k\in G} f_k(x) \\
\text{s.t.} \quad & P_k^{\min} \le \mathrm{Tr}(Y_k x x^\top) + P_k^d \le P_k^{\max} \\
& Q_k^{\min} \le \mathrm{Tr}(\bar{Y}_k x x^\top) + Q_k^d \le Q_k^{\max} \\
& (V_k^{\min})^2 \le \mathrm{Tr}(M_k x x^\top) \le (V_k^{\max})^2 \\
& (\mathrm{Tr}(Y_{lm} x x^\top))^2 + (\mathrm{Tr}(\bar{Y}_{lm} x x^\top))^2 \le (S_{lm}^{\max})^2 \\
& \mathrm{Tr}(Y_{lm} x x^\top) \le P_{lm}^{\max} \\
& \mathrm{Tr}(M_{lm} x x^\top) \le (\triangle V_{lm}^{\max})^2.
\end{aligned} \tag{6.2}$$

The objective function then becomes:

$$f_k(x) = c_k^2(P_k^d + \mathrm{Tr}(Y_k x x^\top))^2 + c_k^1(P_k^d + \mathrm{Tr}(Y_k x x^\top)) + c_k^0.$$

6.3 An alternating direction approach based on rank-one relaxation

We can define $W = xx^\top$ and formulate the problem as a rank-constrained problem. Clearly, ACOPF is a nonconvex problem, due to this quadratic equality. In traditional SDP relaxation approach, the rank-one constraint for W is ignored and thus a convex relaxation problelm is solved. An alternative approach is to formulate the problem into a rank-minimization problem. However, rank-minimization approach does not always recover the rank-one feasible solution (see, for instance, [46] for a discussion).

In [4], a least-squares approach is proposed to find an approximate solution as an alternative to solving the following system of quadratic equations:

$$x^\top M_i x + m_i^\top x = \theta_i, \forall i \in \{1, ..., m\}, \tag{6.3}$$

where $M_i \in \mathbb{R}^{n\times n}$ are matrices not necessarily symmetric—let alone positive semidefinite, $m_i \in \mathbb{R}^n$ is a given vector. In the chapter, they consider solving this problem by minimizing second-order sum of squares:

$$\min_{x\in\mathcal{X}} F = \sum_{i=1}^{m} f_i(x) = \sum_{i=1}^{m} (x^\top M_i x + m_i^\top x - \theta_i)^2, \tag{6.4}$$

where $\mathcal{X}$ is some convex set.

In this chapter we propose an algorithmic framework based on augmented Lagrangian function. The goal is to relax the difficult rank-one constraint, i.e. $\text{rank}(W) = 1$, implied by $W = xx^\top$, and recover the feasibility at the limit. Our framework consists of the well-known alternating direction method of multipliers (ADMM) to solve the subproblem, and our method is convergent to a local minimum. Our framework is simple and relies on solving a sequence of convex subproblems, which makes it easy to implement on standard nonlinear optimization solvers.

6.3.1 Outer loop: augmented Lagrangian method

For the outer loop, we apply the classic augmented Lagrangian method (AL) to relax the equality constraints for a set of least-squares. A basic augmented Lagrangian method can be implemented as Algorithm 4 for minimizing a general nonlinear problem subject to an equality constraint.

Now let us describe the situation in our case. Recall that we can add a rank constraint $W = xx^\top$. Thus, we can apply Algorithm 12 and solve the following subproblem at each iteration:

$$\begin{aligned}
\min \quad & \mathcal{L}_A := \sum_{k \in G} f_k(W) - \mathrm{Tr}(\Lambda^\top (W - xx^\top)) + \frac{1}{2\mu} \|W - xx^\top\|_F^2 \\
\text{s.t.} \quad & P_k^{\min} \leq \mathrm{Tr}(Y_k W) + P_k^d \leq P_k^{\max} \\
& Q_k^{\min} \leq \mathrm{Tr}(\bar{Y}_k W) + Q_k^d \leq Q_k^{\max} \\
& (V_k^{\min})^2 \leq \mathrm{Tr}(M_k W) \leq (V_k^{\max})^2 \\
& (\mathrm{Tr}(Y_{lm} W))^2 + (\mathrm{Tr}(\bar{Y}_{lm} W))^2 \leq (S_{lm}^{\max})^2 \\
& \mathrm{Tr}(Y_{lm} W) \leq P_{lm}^{\max} \\
& \mathrm{Tr}(M_{lm} W) \leq (\triangle V_{lm}^{\max})^2 \\
& W \succeq 0.
\end{aligned} \tag{6.5}$$

For simplicity, for future reference we will denote the feasible region of (6.5) as $W \in \mathcal{S}_{\mathcal{W}}$, since x now is a free variable.

The convergence of augmented Lagrangian method has been well studied in many literatures [9, 72].

6.3.2 Alternating direction method of multipliers (ADMM)

For the subproblem, here we apply variable splitting and solve the following problem which is equivalent to (6.5)

$$
\begin{aligned}
\min \quad & \mathcal{L}_A(W, x, z) := \sum_{k \in G} f_k(W) - \operatorname{Tr}(\Lambda^\top (W - xz^\top)) + \frac{1}{2\mu}\|W - xz^\top\|_F^2 \\
\text{s.t.} \quad & W \in \mathcal{S}_{\mathcal{W}} \\
& x = z,
\end{aligned} \tag{6.6}
$$

where $x, z \in \mathbb{R}^n$. In other words, we map the dimension of decision variable from $(\frac{n(n+1)}{2} + n)$ in (6.2) to $(\frac{n(n+1)}{2} + 2n)$ in (6.6).

Consider problem in the form of (6.6). Given a penalty parameter $1/\nu$ ($\nu > 0$), we have the following augmented Lagrangian function:

$$
\begin{aligned}
\mathcal{L}_A^{admm}(W, x, z; \delta) \;:=\; & \sum_{k \in G} f_k(W) - \operatorname{Tr}(\Lambda^\top (W - xz^\top)) \\
+ \; & \frac{1}{2\mu}\|W - xz^\top\|_F^2 - \delta^\top (x - z) + \frac{1}{2\nu}\|x - z\|^2,
\end{aligned} \tag{6.7}
$$

where δ is a multiplier. Note that throughout the iterations for each subproblem, Λ is maintained as a constant.

As described in previous chapters, alternating direction methods of multipliers (ADMM), or so-called alternating direction augmented Lagrangian method (ADAL), can be regarded as AL-BCD with each subproblem solved only inexactly. A simple framework of ADMM can be implemented as Algorithm 18.

Algorithm 18 Alternating direction methods of multipliers (ADMM) for ACOPF subproblem

```
1. Choose $\mu^0, \lambda^0$, and $W^0, x^0 = z^0$;
2. for $k = 0, 1, ...,$ do
    $[W^{k+1}, x^{k+1}] := \arg\min_{W \in \mathcal{S}_{\mathcal{W}}, x} \mathcal{L}_A^{admm}(W, x, z^k; \delta^k)$;
    $z^{k+1} := \arg\min_z \mathcal{L}_A^{admm}(W^{k+1}, x^{k+1}, z; \delta^k)$;
    update the multiplier $\delta^{k+1} = \delta^k - \frac{1}{\nu^k}(x^{k+1} - z^{k+1})$;
    choose new penalty parameter $\nu^{k+1}$.
```

Convergence properties of ADMM

The convergence properties of ADMM solving nonconvex problems with certain structures have been shown in Chapter 5. Here we argue that our case satisfies the function format that guarantees the convergence results in Chapter 5.

Recall that in Chapter 5, we consider the objective function in the following form:

$$\min_{x \in \mathcal{X}} F(x), \tag{6.8}$$

where $x \in \mathbb{R}^n$. Further, we assume that the variables can be split into two blocks, i.e. $F(x) = h(f(x), g(x))$ but do not assume that the objective can be decomposed as a linear sum of functions consisting of these blocks. Denote the following as a new function of x at some given $\bar{x}$

$$F_1(x, \bar{x}) = h(f(x), g(\bar{x})), \tag{6.9}$$

which is indeed the function of x with the second block, g, fixed at a given $\bar{x}$. Similarly, if we fix the first block at some x_0 and vary the other, we have another function

$$F_2(\bar{x}, x) = h(f(\bar{x}), g(x)). \tag{6.10}$$

Then under the assumption that F_1, F_2 are convex, as well as some mild conditions, we can show that augmented Lagrangian function value converges by the iteration in Algorithm 13, and further, any limit point of sequence $\{x^k\}$ generated by Algorithm 13 is a stationary point of F.

In the ADMM-OPF case, we have W and x as variables. However, we only split x rather than W and x together. Now we will show that such partial splitting fits into the above case.

Since we need to solve the augmented Lagrangian form with ADMM, we can rewrite the objective function in (6.5) as

$$\mathcal{L}_A(W, x) = F_1(W) + F_2(W, x),$$

where $F_1(W) = \sum_{k\in G} f_k(W)$ and $F_2(W, x) = -\text{Tr}(\Lambda^\top(W - xx^\top)) + \frac{1}{2\mu}\|W - xx^\top\|_F^2$. Furthermore, it is obvious that $F_2(W, x) = h(f^\top g)$, where $h(M) = -\text{Tr}(\Lambda^\top M) + \frac{1}{2\mu}\|M\|_F^2$, $f(W, x) = \begin{bmatrix} W \\ x^\top \end{bmatrix}$ and $g(W, x) = \begin{bmatrix} I \\ -x^\top \end{bmatrix}$ are affine functions. Thus, we can apply the splitting:

$$\begin{aligned} \min \quad & \mathcal{L}_A(W, V, x, z) := F_1(W) + F_2(W, V, x, z) = F_1(W) + h(f(W, x)^\top g(V, z)) \\ \text{s.t.} \quad & W \in \mathcal{S}_{\mathcal{W}},\ x = z,\ W = V. \end{aligned} \tag{6.11}$$

Moreover, we can write down the augmented Lagrangian function to relax the splitting constraint.

$$\begin{aligned} & \mathcal{L}_A^{admm}(W, V, x, z; \Lambda, \delta) \\ & := F_1(W) + h(f(W, x)^\top g(V, z)) - \text{Tr}(\Lambda^\top(W - V)) \\ & + \frac{1}{2\mu}\|W - V\|_F^2 - \delta^\top(x - z) + \frac{1}{2\nu}\|x - z\|^2, \end{aligned}$$

where Δ and δ are multipliers; μ and ν are positive scalars.

Note that, since $g(V, z) = \begin{bmatrix} I \\ -z^\top \end{bmatrix}$, V does not appear in g. Thus, for small enough μ, V should always be equal to W to minimize $\mathcal{L}_A^{admm}$, in which case we have

(6.7) and further Algorithm 18. Thus, we have shown that we may apply partial splitting (i.e. splitting between x's and z's) and include W in the minimization of one block and maintained in the other, and our convergence result holds.

Remark 6.3.1. *Note that, given the augmented Lagrangian formulation* (6.5)*, it is natural to consider ADMM to solve the outer loop directly. The reason is simple. Although the rank-one set is nonconvex, a symmetric positive semidefinite matrix being mapped to the rank-one set resulted in an analytic solution. For instance, a simple ADMM heuristic scheme solving OPF is considered in Algorithm 19 (see, for instance, [76]). However, there is no guarantee that Algorithm 19 would converge.*

Algorithm 19 An ADMM heuristic scheme for ACOPF

```
1. Choose $\mu^0, \lambda^0$, and $W^0, Z^0$;
2. for $k = 0, 1, ...,$ do
       $W^{k+1} := \arg\min_{W \in \mathcal{S}_\mathcal{W}} \mathcal{L}_A(W, x^k; \Lambda^k)$;
       $x^{k+1} := \sqrt{\sigma_1^{k+1}} v_1^{k+1}$,
       where $\sigma_1, v_1$ are the top singular value/vector of $W^{k+1}$.
       update the multiplier $\Lambda^{k+1} = \Lambda^k - \frac{1}{\mu^k}(W^{k+1} - x^{k+1}(x^{k+1})^\top)$;
       choose new penalty parameter $\mu^{k+1}$.
```

6.3.3 Practical issues

The efficiency of the algorithm lies in the number of (sparse) QPs that our method tends to solve. In terms of practical implementation, there are several issues worth being pointed out.

Warm-start

It is known that both AL and ADMM can be slow, especially when the starting point is far from the optimal solution and high accuracy is desired. In the OPF case, we can warm start our algorithm by the solution obtained by the SDP relaxation. To be specific, we solve an SDP without the rank-one constraint. We then map the solution W^0, which is obtained by solving (6.5), to the rank-one set, and obtain the starting points x^0 and z^0 through a partial singular value decomposition (SVD). Simply put,

$$x^0 = z^0 = \sqrt{\sigma_1} v_1,$$

where σ_1, v_1 are the top singular value/vector of W^0.

Subproblem 1: QP/SDP

The main computational cost of ADMM lies in minimizing (6.7), which has a quadratic objective. This subproblem can be reformulated into SDP with a linear objective by adding linear matrix inequalities (LMI). Let $\xi = \text{vec}(W - xz^\top) \in \mathbb{R}^{n^2}, s \in \mathbb{R}$. Then $\|W - xz^\top\|_F^2 \leq s$ is equivalent to

$$\begin{bmatrix} s & \xi \\ \xi^\top & I \end{bmatrix} \succeq 0.$$

The reformulation of $\sum_{k \in G} f_k(W)$ and the constraint $(\text{Tr}(Y_{lm}W))^2 + (\text{Tr}(\bar{Y}_{lm}W))^2 \leq (S_{lm}^{\max})^2$ can be seen (formula (6) and (5), respectively) in [44].

Subproblem 2: closed-form solution

In ADMM, very often the computational efforts for each subproblem are quite different. In Algorithm 18, note that z is not contained in either f_k or the constraint set $\mathcal{S}_{\mathcal{W}}$. Thus, we can simply solve the correponding subproblem in closed form. By settting the gradient to zero, we derive that

$$z^{k+1} = \frac{\mu\nu}{\nu(x^{k+1})^\top x^{k+1} + \mu} \left[(\frac{1}{\nu} I + \frac{1}{\mu} W^{k+1} - \Lambda^j)^\top x^{k+1} - \delta^k \right],$$

where I is an identity matrix, Λ^j is the multiplier matrix obtained from the jth outer loop.

Stopping criterion and the choice of prox parameter

Since we aim to find a KKT point, we could stop the algorithm when the largest KKT violation from (6.2) is below some $\epsilon > 0$. In practice, we can often test if the residuals are within the ϵ-band [12], i.e.

$$\begin{aligned} &\|x^k - z^k\| \le \epsilon_1 \\ &\|W^k - x^k (z^k)^\top\|_F \le \epsilon_2, \end{aligned}$$

where ϵ_1 controls the feasibility of the AL subproblem, ϵ_2 together with ϵ_1 controls the feasibility of the original problem. Or we can use, equivalently, $\|W^k - x^k(x^k)^\top\|_F \le \epsilon_{primal}$, to measure the feasibility of the primal problem. For the dual problem, we may measure the dual feasiblity by

$$\frac{1}{\mu}\|x^{k+1}(x^{k+1})^\top - x^k(x^k)^\top\|_F^2 \le \epsilon_{dual}.$$

As for the choice of the prox parameter, μ is often decreased (to enforce feasibility) after a certain number of iterations, in AL. For instance, we can apply the augmented Lagrangian algorithm shown in Algorithm 4, and choose $\mu^k \in (0, \beta\mu^{k-1}]$ where $0 <$

$\beta < 1$.

For ADMM, the convergence results are shown to be valid when ν is maintained as a constant (see Chapter 5 for details). However, in practice we rarely use a small prox parameter at the very beginning and maintain its value throughout the iterations. Instead, we can apply some backtracking tricks to increase the potential stepsizes and accelerate the algorithms.

Inexactness

It is known that augmented Lagrangian method converges while the subproblem is solved inexactly. Hence, throughout implementation, we sequentially increase the accuracy in solving the subproblem. For instance, we may terminate the subproblem by setting $\epsilon_1 = \gamma\beta^k$, where γ, β are nonnegative constants and k is the outer iteration number.

6.3.4 An alternating linearization method to solve the subproblem

As an alternative of ADMM, one may consider the alternating linearization method to solve the subproblem. Instead of using AL and ADMM described previously, in this section we consider alternating linearization method (ALM). ALM can be naturally related to ADMM since the update of the multiplier λ in the dual space in some sense can be regarded as computing the gradient in the primal space. In this section, we discuss applying alternating linearization method to solve (6.5).

Define the following approximation function with smaller enough ν:

$$Q_\nu^1(W, x, z^k) = F_{AL}(W, x, z^k) + \left\langle \nabla_2 F_{AL}(z^k), x - z^k \right\rangle + \frac{1}{2\nu}\|x - z^k\|^2,$$

where $F_{AL}(W, x, z) = \sum_{k \in G} f_k(W) - \mathrm{Tr}(\Lambda^\top (W - xz^\top)) + \frac{1}{2\mu}\|W - xz^\top\|_F^2$ and further $\nabla_2 F_{AL}(z^k) = \Lambda^\top z^k - \frac{1}{\mu}\left(W^k - z^k (z^k)^\top\right)^\top z^k$. Suppose the following condition holds for small enough μ:

$$F(W^{k+1}, x^{k+1}, x^{k+1}) \leq Q_\nu^1(W^{k+1}, x^{k+1}, z^k), \tag{6.12}$$

where x^{k+1} is in the neighborhood centered at z^k, i.e. $x^{k+1} \in \mathcal{B}(z^k; r)$. Such μ can be found by backtracking at each iteration.

Similarly, we can define

$$Q_\nu^2(W^{k+1}, x^{k+1}, z) = F_{AL}(W^{k+1}, x^{k+1}, z) + \left\langle \nabla_1 F_{AL}(x^k), z - x^{k+1} \right\rangle + \frac{1}{2\nu}\|x^{k+1} - z\|^2,$$

where $\nabla_1 F_{AL}(x^{k+1}) = \Lambda x^{k+1} - \frac{1}{\mu}\left(W^{k+1} - x^{k+1}(x^{k+1})^\top\right) x^{k+1}$.

Thus, we obtain the following alternating linearization method (Algorithm 20). Note that many situations are similar as ADMM. For instance, we only update W in the block with x to be minimized, and keep W fixed when minimizing z. Also, since minimizing z is a unconstrained problem, at each iteration we can carry a closed-form solution to obtain z^{k+1}.

Algorithm 20 Alternating linearization method for OPF (ALM-OPF)

```
1. Choose $\nu_1^0 = \nu_2^0 = \nu^0$, and $x^0 = z^0$;
2. for $k = 0, 1, ...$
      (a) $[W^{k+1}, x^{k+1}] := \arg\min_{W \in \mathcal{S}, x} Q^1_{\nu_1^k}(W, x, z^k)$; choose $\nu_1^{k+1}$;
      (b) $z^{k+1} := \arg\min_{y \in \mathcal{X}} Q^2_{\nu_2^k}(W^{k+1}, x^{k+1}, z)$; choose $\nu_2^{k+1}$;
```

Again, we refer to Chapter 5 for the convergence properties of ALM-OPF.

6.4 Numerical results

In this section, we provide some preliminary results for numerical experiments. Our implementations were written in MATLAB and run in MATLAB R2013a on a laptop with Intel Core Duo 1.8 GHz CPU and 2GB RAM. We use CVX as the problem parser and apply Sedumi as the subproblem solver.

We first apply our augmented Lagrangian framework in a standard testing set: case9 (see: http://www.pserc.cornell.edu//matpower/docs/ref/matpower5.0/ for a full description). As we discussed earlier, the computational cost of our methods (both AL-ADMM and AL-ALM) relies on the number of subproblems solved. Thus, we compare the number of QPs/SDPs solved when the rank-one violation below a certain tolerance. We also list the number of outer iteration (i.e. augmented Lagrangian iteration) in the parentheses.

Through all instances, the optimal bound is checked by using Lavaei and Low bound, which is tight in the instances tested.

We can observe that, although the number of outer iteration is comparable (as expected) for the two algorithms, the number of subproblems solved by ALM are clearly less than ADMM. In fact, in previous chapters when we test ALM and ADMM on other second-order least-square type of problems (e.g. risk parity problems), we observe similar behavior. We emphasize that this may be the result of the fact that ALM takes advantage of the gradient information as well as its backtracking strategy to obtain a good estimate of the prox parameter.

Table 6.1: A comparison of algorithms solving case9. The starting prox parameter is set to be 1.

Algorithms	k(10^{-2})	k(10^{-3})	k(10^{-4})	Final value	Matpower	Optimal
AL-ADMM	16(3)	58(7)	131(10)	5.297×10^3	5.297×10^3	5.297×10^3
AL-ALM	16(3)	28(7)	40(10)	5.297×10^3		

Table 6.2: A comparison of algorithms solving case14. The starting prox parameter is set to be 0.1.

Algorithms	k(10^{-2})	k(10^{-4})	k(10^{-5})	Final value	Matpower	Optimal
AL-ADMM	17(2)	41(7)	120(12)	8.082×10^3	8.082×10^3	8.082×10^3
AL-ALM	13(2)	21(3)	30(5)	8.082×10^3		

We also report the final function value (i.e. optimal bound) with the one solved by Matpower. As we can see, they are fully comparable with the optimal one. Similar behavior can be observed for case14.

We now test a more difficult instance: case2w. The standard OPF package Matpower cannot find a good solution when trying to solve this instance. Our methods clearly find a better KKT point in this case. Note that, by default, Matpower applies Newton method to solve OPF problem. The solutions generated by Matpower can be with high accuracy but may be far from the optimal [14, 32].

Table 6.3: A comparison of algorithms solving case2w. The starting prox parameter is set to be 1.

Algorithms	k(10^{-2})	k(10^{-3})	k(10^{-4})	Final value	Matpower	Optimal
AL-ADMM	26(3)	55(5)	143(6)	8.778×10^2	9.078×10^2	8.778×10^2
AL-ALM	15(3)	23(5)	25(6)	8.778×10^2		

6.5 Conclusion

In this chapter, we propose a family of alternating direction methods minimizing second-order least-squares problems, and further to solve the optimal power flow problem (OPF). Our methods rely on solving convex subproblem and can the limit point obtained can be guaranteed to satisfy KKT optimality conditions. Despite being slow when a high accuracy is desired, our gradient-based methods can be effective when medium-accuracy is required. Preliminary experiment fulfills our expectation. How to further enhance our algorithms in terms of both speed and the quality of solutions remain a topic of further research.

Chapter 7

Conclusion

This dissertation sets out new algorithmic approaches for structured convex and nonconvex problems. All of our methods are motivated by gradient-based methods, which have been researched heavily in the last few years. We first enhance existing accelerated first-order methods by backtracking line search and provide improved complexity estimate. Motivated by our methods in composite convex optimization, we also establish new alternating direction approaches to tackle some nonconvex problems which are difficult in general but have certain structure. Numerical results in both risk parity optimization and optimal power flow indicate the efficiency of our methods.

In Chapter 3 we study an accelerated first-order scheme for composite convex problems, which can be used as an extension of FISTA and FALM. It has been shown that, the combination between FISTA and variable stepsizes (prox parameters) yield an efficient method in solving problems where a large Lipschitz estimate is encountered. The computational results show that FISTA and FALM can better off by allowing for the complexity estimates that depend on the "average" prox parameter value.

From Chapter 4 to Chapter 6, we focus on the optimization of some structured nonlinear nonconvex problems. In Chapter 4 we discuss the risk parity portfolio selection problem. Risk parity optimization aims to find such portfolios for which the contributions of risk from all assets are equally weighted. Due to the problem structure, the standard convex approach has a number of limitations, especially in terms of versatility to extend the model when general bounds or other constraints are added. Due to these limitations, we propose a nonconvex generalized risk parity model and apply the alternating direction framework to solve this model. Numerical experiments indicate the effectiveness of our technique.

Chapter 5 deals with the convergence and complexity theory of alternating direction type of methods solving structured nonlinear nonconvex problems, and in particular, second-order least-squares problem. Moreover, we apply some global optimization techniques from polynomial optimization literature to complement our local methods and to provide lower bounds.

Chapter 6 applies our theory in second-order least-squares in the optimal power flow problem (OPF). The OPF problem is NP-hard in general due to the rank-one recovery constraint, and our method aims to find a stationary point. Compared with SDP type of approach, our method is convergent to a local minimum and can always recover a rank-one solution within some predetermined tolerance. Preliminary numerical results show the efficiency of our approach.

Bibliography

[1] M. Afonso, J. Bioucas-Dias, and M. Figueiredo. Fast image recovery using variable splitting and constrained optimization. *preprint available at http://arxiv.org/abs/0910.4887*, 2009.

[2] M. Afonso, J. Bioucas-Dias, and M. Figueiredo. An augmented Lagrangian approach to the constrained optimization formulation of imaging inverse problems. *Image Processing, IEEE Transactions on*, 20(3):681–695, 2011.

[3] A. Ahmadi and A. Majumdar. DSOS and SDSOS optimization: LP and SOCP-based alternatives to sum of squares optimization. In *Information Sciences and Systems (CISS), 2014 48th Annual Conference on*, pages 1–5. IEEE, 2014.

[4] X. Bai and K. Scheinberg. Alternating direction methods for non convex optimization with applications to second-order least-squares and risk parity portfolio selection. *Available at optimization-online*, 2015.

[5] X. Bai, K. Scheinberg, and R. Tutuncu. Least-squares approach to risk parity in portfolio selection. *Available at SSRN*, 2013.

[6] X. Bai, H. Wei, K. Fujisawa, and Y. Wang. Semidefinite programming for optimal power flow problems. *International Journal of Electrical Power & Energy Systems*, 30(6):383–392, 2008.

[7] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[8] S. Becker, J. Bobin, and E. J. Candès. NESTA: A Fast and Accurate First-Order Method for Sparse Recovery. *SIAM Journal on Imaging Sciences*, 4(1):1–39, Jan. 2011.

[9] D. Bertsekas. *Constrained optimization and Lagrange multiplier methods*, volume 1. Academic Press, 1982.

[10] D. Bertsekas. *Nonlinear programming.* Athena Scientific, 1999.

[11] M. Best and R. Grauer. On the sensitivity of mean-variance-efficient portfolios to changes in asset means: some analytical and computational results. *Review of Financial Studies*, 4(2):315–342, 1991.

[12] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[13] R. Brito and L. Vicente. Efficient cardinality/mean-variance portfolios. In *System Modeling and Optimization*, pages 52–73. Springer Berlin Heidelberg, 2014.

[14] W. Bukhsh, A. Grothey, K. McKinnon, and P. Trodden. Local solutions of the optimal power flow problem. *Power Systems, IEEE Transactions on*, 28(4):4780–4788, 2013.

[15] E. Candès. Compressive sampling. *Proc. International Congress of Mathematics*, 3:1433–1452, 2006.

[16] E. Candès and J. Romberg. l_1-Magic: Recovery of Sparse Signals via Convex Programming. Technical report, Caltech, 2005.

[17] E. Candès, J. Romberg, and T. Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52:489–509, 2006.

[18] D. Chaves, J. Hsu, F. Li, and O. Shakernia. Risk parity portfolio vs. other asset allocation heuristic portfolios. *Journal of Investing*, 20(1):108–118, 2011.

[19] D. Chaves, J. Hsu, F. Li, and O. Shakernia. Efficient algorithms for computing risk parity portfolio weights. *Journal of Investing*, 21(3):150–163, 2012.

[20] N. Courtois, A. K., J. Patarin, and A. Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *Advances in Cryptology-EUROCRYPT 2000*, pages 392–407. Springer, 2000.

[21] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications in Pure and Applied Mathematics*, 57:1413–1457, 2004.

[22] D. Davis and W. Yin. Convergence rate analysis of several splitting schemes. August 2014.

[23] V. DeMiguel, L. Garlappi, and R. Uppal. Optimal versus naive diversification: How inefficient is the 1/n portfolio strategy? *Review of Financial Studies*, 22(5):1915–1953, 2009.

[24] D. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52:1289–1306, 2006.

[25] D. Donoho. For most large underdetermined systems of linear equations, the minimal l_1 norm solution is also the sparsest solution. *Communications on Pure and Applied Mathematics*, 59(6):797–829, 2006.

[26] J. Douglas and H. Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Transactions of the American mathematical Society*, 82(2):421–439, 1956.

[27] M. Figueiredo and R. Nowak. An EM algorithm for wavelet-based image restoration. *IEEE Transactions on Image Processing*, 12:906–916, 2003.

[28] M. Figueiredo, R. Nowak, and S. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal on Selected Topics in Signal Processing*, 1(4), 2007.

[29] M. Figueiredo, R. Nowak, and S. Wright. Gradient Projection for Sparse Reconstruction: Application to Compressed Sensing and Other Inverse Problems. *Selected Topics in Signal Processing, IEEE Journal of*, 1(4):586–597, 2007.

[30] M. Fortin and R. Glowinski. *Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems*. Elsevier, 2000.

[31] A. Frank and A. Asuncion. UCI machine learning repository. 2010.

[32] B. Ghaddar, J. Marecek, and M. Mevissen. Optimal power flow as a polynomial optimization problem. *arXiv preprint arXiv:1404.3626*, 2014.

[33] R. Glowinski and P. Le Tallec. *Augmented Lagrangian and operator-splitting methods in nonlinear mechanics*, volume 9. SIAM, 1989.

[34] D. Goldfarb, S. Ma, and K. Scheinberg. Fast alternating linearization methods for minimizing the sum of two convex functions. *Mathematical Programming*, 141(1-2):349–382, 2013.

[35] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk. Fast alternating direction optimization methods. *SIAM Journal on Imaging Sciences*, 7(3):1588–1623, 2014.

[36] T. Goldstein and S. Osher. The split Bregman algorithm for l_1 regularized problems. *UCLA CAM Report 08-29*, 2008.

[37] L. Grippo and M. Sciandrone. On the convergence of the block nonlinear gauss–seidel method under convex constraints. *Operations Research Letters*, 26(3):127–136, 2000.

[38] E. Hale, W. Yin, and Y. Zhang. A fixed-point continuation method for l_1-regularized minimization with applications to compressed sensing. Technical report, CAAM TR07-07, 2007.

[39] B. He, H. Yang, and S. Wang. Alternating Direction method with self-adaptive penalty parameters for monotone variational inequalities. *Journal of optimization theory and applications*, 106(2):337–356, 2000.

[40] M. Hong, Z. Luo, and M. Razaviyayn. Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. *Available at ArXiv*, 2014.

[41] B. Jiang, S. Ma, and S. Zhang. Alternating direction method of multipliers for real and complex polynomial optimization models. *Optimization*, 63(6):883–898, 2014.

[42] H. Kaya and W. Lee. Demystifying risk parity. *Neuberger Berman,*, 2012.

[43] J. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.

[44] J. Lavaei and S. Low. Zero duality gap in optimal power flow problem. *Power Systems, IEEE Transactions on*, 27(1):92–107, 2012.

[45] D. Lorenz. Constructing test instances for Basis Pursuit Denoising. pages 1–5.

[46] R. Louca, P. Seiler, and E. Bitar. A rank minimization algorithm to enhance semidefinite relaxations of optimal power flow. In *Allerton*, pages 1010–1020, 2013.

[47] Z. Luo and S. Zhang. A semidefinite relaxation scheme for multivariate quartic polynomial optimization with quadratic constraints. *SIAM Journal on Optimization*, 20(4):1716–1736, 2010.

[48] S. Maillard, T. Roncalli, and J. Teïletche. The properties of equally weighted risk contribution portfolios. *The Journal of Portfolio Management*, 36(4):60–70, 2010.

[49] M. Mevissen. Introduction to concepts and advances in polynomial optimization. Technical report, Institute for Operations Research, ETH Zurich, 2007.

[50] B. Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24:227–234, 1995.

[51] A. Nemirovski and D. Yudin. *Informational complexity and efficient methods for solution of convex extremal problems.* J. Wiley & Sons, New York, 1983.

[52] Y. Nesterov. Introductory lectures on convex optimization. 87:xviii+236, 2004.

[53] J. Nie and M. Schweighofer. On the complexity of Putinar's positivstellensatz. *Journal of Complexity*, 23(1):135–150, 2007.

[54] P. Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization.* PhD thesis, Citeseer, 2000.

[55] P. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical programming*, 96(2):293–320, 2003.

[56] D. Peaceman and H. Rachford. The numerical solution of parabolic elliptic differential equations. *SIAM Journal on Applied Mathematics*, 3:28–41, 1955.

[57] S. Prajna, A. Papachristodoulou, and P. Parrilo. Introducing SOSTOOLS: A general purpose sum of squares programming solver. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, volume 1, pages 741–746. IEEE, 2002.

[58] M. Putinar. Positive polynomials on compact semi-algebraic sets. *Indiana University Mathematics Journal*, 42(3):969–984, 1993.

[59] B. Reznick. Some concrete aspects of Hilbert's 17th problem. *Contemporary Mathematics*, 253:251–272, 2000.

[60] K. Scheinberg, D. Goldfarb, and X. Bai. Fast first-order methods for composite convex optimization with backtracking. *Foundations of Computational Mathematics*, 14(3):389–417, 2014.

[61] K. Schmüdgen. The k-moment problem for compact semi-algebraic sets. *Mathematische Annalen*, 289(1):203–206, 1991.

[62] Y. Shen, Z. Wen, and Y. Zhang. Augmented Lagrangian alternating direction method for matrix separation based on low-rank factorization. *Optimization Methods and Software*, 29(2):239–263, 2014.

[63] F. Spinu. An algorithm for computing risk parity weights. *Available online at: http://ssrn.com/abstract=2297383 (accessed 18 October 2013).*, 2013.

[64] S. Sra, S. Nowozin, and S. Wright. *Optimization for machine learning.* MIT Press, 2012.

[65] M. Steinbach. Markowitz revisited: Mean-variance models in financial portfolio analysis. *SIAM review*, 43(1):31–85, 2001.

[66] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal Royal Statistical Society B*, 58:267–288, 1996.

[67] P. Tseng. Further applications of a splitting algorithm to decomposition in variational inequalities and convex programming. *Mathematical Programming*, 48:249–263, 1990.

[68] P. Tseng. Applications of a splitting algorithm to decomposition in convex programming and variational inequalities. *SIAM J. Control and Optimization*, 29(1):119–138, 1991.

[69] E. van den Berg and M. Friedlander. SPGL1: A solver for large-scale sparse reconstruction, June 2007.

[70] E. van den Berg, M. Friedlander, G. Hennenfent, F. Herrmann, R. Saab, and O. Yilmaz. Sparco: A testing framework for sparse reconstruction. Technical Report TR-2007-20, Dept. Computer Science, University of British Columbia, Vancouver, Oct. 2007.

[71] Z. Wen, D. Goldfarb, and W. Yin. Alternating direction augmented Lagrangian methods for semidefinite programming. *Mathematical Programming Computation*, 2(3-4):203–230, Sept. 2010.

[72] S. Wright and J. Nocedal. *Numerical optimization*, volume 2. Springer New York, 1999.

[73] S. Wright, R. Nowak, and M. Figueiredo. Sparse reconstruction by separable approximation. *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 3373–3376, 2008.

[74] Y. Xu and W. Yin. A block coordinate descent method for multi-convex optimization with applications to nonnegative tensor factorization and completion. Technical report, DTIC Document, 2012.

[75] J. Yang and Y. Zhang. Alternating direction algorithms for l_1 problems in compressive sensing. *preprint*, 2009.

[76] S. You and Q. Peng. A non-convex alternating direction method of multipliers heuristic for optimal power flow. In *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*, pages 788–793. IEEE, 2014.

[77] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[78] Y. Zhang. YALL1: Your algorithms for L1. *http://www.caam.rice.edu/˜optimization/L1/YALL1/*, 2009.

Appendix A

Sample statistics in Chapter 6

In this section, we provide the sample statistics used in Section 4.6.2. We consider the following common market indices to represent different asset classes: S&P 500, MSCI World (Net), Russell 2500, Russell 2000 Growth, Russell 2000 Value, HFRI Equity Hedged Index, MSCI Emerging Markets (Net), HFRI Emerging MKTS Total, HFRI FoF (Conservative Index), BC Treasury 5-10 Yr, BC US Corporate High Yield Index, JPMorgan GBI-EM Index, JPMorgan EMBI+ Index, S&P Global Natural Resources - Energy Index, from Nov. 2002 to Aug. 2012.

Here we use the historical return and covariance matrix.

Table A.1: Asset classes and sample statistics.

Portfolio Indices	Excess return	Volatility
S&P 500	3.8528%	14.971%
MSCI World (Net)	6.4733%	16.374%
Russell 2500	10.0418%	19.1458%
Russell 2000 Growth	9.7307%	20.8973%
Russell 2000 Value	9.2627%	20.1313%
HFRI Equity Hedged	4.1882%	8.8111%
MSCI Emerging MKTS	17.4651%	24.2452%
HFRI Emerging MKTS	9.835%	12.1583%
HFRI FoF	1.0761%	4.4455%
BC Treasury 5-10 Yr	5.0464%	5.6112%
BC US Corporate High Yield	9.4121%	10.633%
JPMorgan GBI-EM	9.7298%	12.6448%
JPMorgan EMBI+	10.8364%	9.0293%
S&P GNR - Energy	13.2877%	21.7562%

Table A.2: The correlation matrix (14 × 14) of asset classes.

Portfolio Indices	Correlation matrix													
S&P 500	1	0.972	0.945	0.908	0.917	0.850	0.816	0.755	0.617	0.031	0.712	0.660	0.556	0.696
MSCI World (Net)		1	0.922	0.887	0.877	0.913	0.892	0.850	0.690	0.076	0.741	0.737	0.611	0.776
Russell 2500			1	0.983	0.974	0.867	0.806	0.740	0.608	-0.007	0.719	0.638	0.538	0.655
Russell 2000 G				1	0.948	0.863	0.797	0.726	0.596	-0.062	0.681	0.607	0.490	0.637
Russell 2000 V					1	0.779	0.731	0.648	0.500	-0.017	0.660	0.620	0.497	0.589
HFRI Equity Hedged						1	0.933	0.943	0.841	0.001	0.724	0.673	0.544	0.814
MSCI Em. MKTS							1	0.947	0.717	0.105	0.705	0.758	0.637	0.795
HFRI Em. MKTS								1	0.842	0.121	0.704	0.714	0.623	0.780
HFRI FoF									1	0.031	0.644	0.505	0.457	0.671
BC Tr. 5-10 Yr										1	0.222	0.326	0.645	0.001
US Corp. High Yield											1	0.591	0.728	0.520
JPMorgan GBI-EM												1	0.723	0.620
JPMorgan EMBI+													1	0.435
S&P GNR - Energy														1

Appendix B

SparOptLib: a library for sparse solution recovery problem

Optimization problems with sparse or low-rank optimal solutions arise in a number of practical applications. Most notably, compressed sensing and signal processing rely on recovering sparse solutions to linear underdetermined systems [8, 16, 25, 28], and thus belong to the domain of Sparse Solution Recovery problem (SSR). The central idea for SSR is to obtain sparse approximate solutions rather than exact solutions because the latter are usually computationally more prohibitive and practically less robust. Although convex relaxations of these problems can be formulated so that standard methods of linear programming (LP) and semi-definite programming (SDP) can be applied, these are often not applicable in practice due to large-scale and dense data.

We now discuss compressed sensing as an important example of this set of optimization problems. The idea of compressed sensing is to recover a sparse signal from a linear system. Consider the following system

$$Ax = b,$$

where A is an $m \times n$ matrix, b is the observed measurement vector of size m, and x is the true solution. In many applications, $m \ll n$ which means the system is under-determined. Suppose, among the infinitely many solutions that satisfy the system, we would like to find the sparsest one. Then, we have the following optimization problem:

$$\begin{aligned} \min \quad & \|x\|_0 \\ \text{s.t.} \quad & Ax = b, \end{aligned} \tag{B.1}$$

where $\|\cdot\|_0$ denotes the zero-norm which is defined as the number of nonzero components of a vector. It can be shown that this problem is NP-hard [50]. In practice, instead of solving this combinatorial problem, a l_1 minimization problem is used instead as a convex relaxation

$$\begin{aligned} \min \quad & \|x\|_1 \\ \text{s.t.} \quad & Ax = b. \end{aligned} \tag{B.2}$$

It can be shown that, under some assumptions, the optimal solution of the combinatorial problem (B.1) is given by the optimal solution of its convex relaxation (B.2) with high probability [24] [17].

On the other hand, there is no need to find the exact x but an approximate one, i.e. $Ax \approx b$, or simply

$$Ax + r = b,$$

where r is the noise vector. Hence, we aim to find such an optimization model that: (i) the solution should be sparse and approximately satisfying the system of equations; (ii) the problem should be relatively easy to solve. These days there are three kinds of convex optimization problems that are used in compressed sensing literature.

1. Basis-Pursuit DeNoising (BPDN)

$$\begin{array}{ll} \min & \|x\|_1 \\ \text{s.t.} & \|Ax - b\|_2 \leq \epsilon, \end{array} \tag{B.3}$$

2. Lagrangian relaxation of the BPDN formulation (LAG, also called BPDN with penalty)

$$\min \quad \frac{1}{2}\|Ax - b\|_2^2 + \rho\|x\|_1 \ , \tag{B.4}$$

and

3. Least Absolute Shrinkage and Selection Operator (LASSO)

$$\begin{array}{ll} \min & \|Ax - b\|_2^2 \\ \text{s.t.} & \|x\|_1 \leq \sigma, \end{array} \tag{B.5}$$

where ϵ, ρ, σ are positive scalars.

With the sparse solution recovery problems becoming popular, there is an increasing need for a universal testing environment for researchers working in this area to test and compare their solvers. For the past decades, collections of test problems exist in various areas of optimization, including NETLIB for linear programming problems, CUTEr for nonlinear optimization, SDPLIB for semi-definite programming, and MIPLIB for mixed-integer linear programming problems. These collections have become standard for testing algorithms, benchmarking and calibrating parameters to improve algorithm robustness and convergence speed and for providing a wide spectrum of problems in their respective areas. In turn, this has led to the development of improved algorithms and has provided insight into problem structures that can be leveraged for better algorithmic convergence.

Through SparOptLib, we aim to provide researchers with a similar collection of problems and testing framework in the area of SSR. Currently, the library contains over 300 instances drawn from a variety of applications and sources. The problems reflect a wide range of difficulty and size and we hope they provide a complex enough environment for testing the robustness of different solvers and solution approaches. This chapter provides a background on the SSR problem, a short inventory of the solver packages available to solve SSR, a detailed description of the library and a user's manual that we hope will enable researchers to benefit from SparOptLib.

Recently, there has been an explosion of interest in the special classes of optimization problems mentioned in the previous chapter, mainly because of an increased interest in compressed sensing, machine learning and the development of efficient algorithms for these areas. These circumstances led to a very active research interest in developing efficient algorithms for recovering sparse solutions, and a number of software packages exist today.

Instead of a detailed review of the packages and the optimization methods they applied, here we provide a short list with reference. Interested readers can refer to their paper or website for details.

- IST (Iterative Shrinkage-Thresholding), by Daubechies, Defrise and De Mol [21]
- FPC (Fixed-Point Continuation) and FPS_AS, by Hale, Yin and Zhang [38]
- FISTA (Fast Iterative Shrinkage/Thresholding algorithm), by Beck and Teboulle [7]
- NESTA (Nesterov's Algorithm), by Bobin, Becker, and Candès [8]
- GPSR (Gradient Projection for Sparse Reconstruction), by Figueiredo, Nowak, Wright [29]
- SpaRSA (Sparse Reconstruction by Separable Approximation), by Wright, Nowak, and Figueiredo [73]
- FALM (Fast Alternating Linearization Methods), by Goldfarb, Ma and Scheinberg [34]
- YALL1 (Your Algorithm for L1), by Zhang, Deng, Yang, and Yin. [78]
- SALSA and C-SALSA (Constrained/Split Augmented Lagrangian Shrinkage Algorithm), by Afonso, Bioucas-Dias and Figueiredo [1]
- SPGL1 (Spectral Projected-gradient Algorithm), by Friedlander and van den Berg [69]

B.1 Instance source and categories

The wide array of solvers available, the various formulations and relaxations that they tackle, and the lack of a standardized reference problem set create issues in assessing the relative difficulty of a problem and the solver performance, as well as in improving the robustness of the algorithms. We created SparOptLib to provide a standardized format for sparse solution recovery instances, which we found to be compatible with most of the solvers we came across.

The instances in SparOptLib are available online and can be downloaded from COR@L.

1

SparOptLib currently contains over 300 instances of varying size and difficulty. These problems can be grouped according to their origin, into three categories.

The first category (30+ instances) is generated using the Sparco Toolbox (2007). Sparco represents a collection of sparse signal recovery problems and an environment to create new problems using the suite of linear operators provided. For each instance, we created three sizes: a "small" instance, which was the original problem provided by the toolbox; a scaled "medium"-sized version, which made each of the dimensions of the problem five times bigger than in the "small" version; a "large" version, which similarly increased each of the dimensions tenfold as opposed to its "small" counterpart (that is, one hundred times larger for the data size). This procedure allowed us to introduce size variability, which has been documented to significantly impact solver performance. The naming convention kept the original Sparco ID of the problem and appended the relative size as a one-letter suffix (for instance, "spaco1s", "sparco1m" and "sparco1l" represent three instances of different sizes of the same problem, which has the Sparco ID 1). For readers interested in learning more about Sparco and how an instance is generated, we refer to [70] for further details.

The second category of problems is obtained from the Sparse Exact and Approx-

[1]https://coral.ie.lehigh.edu/projects/SPAROPTLIB/wiki/SparseOptimizationLibrary.

imate Recovery (SPEAR) suite [45]. This project is a collaboration between the Institute for Mathematical Optimization and the Institute for Analysis and Algebra from Technische Universitt Braunschweig and it aims to develop a better understanding of the conditions under which sparse solution recovery is possible. We re-cast 273 problems adapted from the L1-Test Set developed as part of the SPEAR project in the standard format proposed and included in SparOptLib, contributing a very large proportion of the library. For the readers interested in reading more about the SPEAR project, please refer to the project website and the co-responding technical report.

The third category comes from the contribution of A. Nemirovski. Originally, this data set is designed for the following problem

$$\begin{aligned} \min_{x} \quad & \|Ax - b\|_2 \\ \text{s.t.} \quad & \|x\|_1 \le \gamma, \end{aligned} \tag{B.6}$$

where A, b, x are defined similarly as above and has the following properties:

- A has the spectral norm not exceeding some L (in the data set, L is chosen to be 1).
- R is chosen to be 1 when creating the data set.
- For the true sparse solution x^*, which is provided with the data so that $Ax^* = b$.

The primary purpose of this data set is to create problems on which CG converges slowly, creating possible difficulties for other first order methods.

B.2 Problem format

All instances can be downloaded from the link given previously, in MATLAB data format. Each instance is stored in a structure p with the following sub-fields:

- A: an $m \times n$ matrix, with m and n provided
- rhs: right hand side vector of observed measurements
- sol: true solution (usually provided by the authors of the problems)
- noise: noise level
- info: other information

Users can easily access the information in "p.sub-field-name".

A quick note is made here about the problems generated using the Sparco Toolbox. Sparco represents an environment for creating sparse signal reconstruction problems using a suite of linear operators provided. In the current version, all problems that contain "sparco" in the file name have been created using the toolbox. These problems store the information contained in 'A' as a function handle, rather than a matrix. In order to recover the information in A and comply with solver input setups, the user needs to download and install the Sparco Toolbox or the Spotbox (a lightweight version of Sparco that consists only of the linear operators needed to recover the matrix from the handle). The users can download the operator package online from the Sparco website: http://www.cs.ubc.ca/labs/scl/sparco/.

B.3 Reference for solution accuracy

Through SparOptLib, we aim to provide both a test set to be used as a reference, and a method to assess solution quality and solver performance. The difficulty in the latter comes mainly from the wide array of approaches taken to solve the problem. Not only are there multiple possible relaxations to the original sparse solution recovery problem, but there are also many solvers, each with a different approach to solving varying relaxations of the problem. Thus, it becomes difficult to evaluate whether one problem is more difficult than another, or to characterize circumstances under which a particular solution approach is better than another one. We propose a framework through which such evaluations can be more easily made.

Suppose we have the optimal solution for either (B.3), (B.4) or (B.5) denoted as x^*. With a particular solver, x is the solution obtained. Moreover, we define two positive scalars ϵ_x and ϵ_b to capture the accuracy of the solutions in the following way:

$$\|x\|_1 \leq (1+\epsilon_x)\|x^*\|_1 \tag{B.7}$$

$$\|Ax-b\|_2 \leq \|Ax^*-b\|_2 + \epsilon_b\|b\|_2 \tag{B.8}$$

Then, a good solution means either ϵ_x or ϵ_b, or both, is reasonably small, depending on which objective function and what scalar we use. Intuitively, that means, we want to obtain an approximate sparse solution and the infeasibility when solving the original system of equations should be bounded by a small error (noise).

The procedure of our proposed method to compare algorithms can be described as following. First, we choose a reference solver. In this library, we choose the

SPGL1 (the Spectral Projected Gradient Algorithm for L-1 minimization problem), developed by M. Friedlander and E. van den Berg. The reason to choose SPGL1 is two-fold. On the one hand, numerous computational experiments have shown its competitiveness. On the other hand, SPGL1 is designed mainly to solve BPDN and can be easily embedded into our proposed method.

We use the reference solver to solve BPDN. As discussed previously, there is no need to obtain an exact solution but an approximate one. Hence, we solve BPDN given (B.8) to be the problem constraint, with ϵ_b chosen to be $10^{-8}, 10^{-4}, 10^{-2}$, respectively. For each ϵ_b, the reference solver finds the corresponding x_s, with which we can compare the true solution x^* (precomputed) and calculate a corresponding ϵ_x based on (B.7). We allow some reasonably small relaxation. If the ϵ_x is less than 10^{-8}, we assume that x_s is close enough to x^* and simply let $\epsilon_x = 10^{-8}$. Thus, we see that, by the reference solver, the x_s obtained is within an ϵ_x-accuracy of the true solution, given a reasonable bound of feasibility. Since such x_s is obtainable by a competitive solver, it can be used as a benchmark for other solvers.

Biography

Xi Bai received B.Eng. from Huazhong University of Science and Technology in 2010. In the same year he joined Lehigh University for the Ph.D. program. Presently, Xi's research is focused on developing efficient algorithms, in a variety of areas including compressed sensing, machine learning, portfolio optimization, etc. In Jun. 2015, Xi is joining Goldman Sachs Asset Management on the full time basis.