FIRST-ORDER METHODS FOR LARGE SCALE

CONVEX OPTIMIZATION

A Dissertation in
The Harold and Inge Marcus Department of Industrial and Manufacturing Engineering
by
Zi Wang

© 2016 Zi Wang

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

December 2016
The dissertation of Zi Wang was reviewed and approved* by the following:

Necdet Serhat Aybat  
Assistant Professor of Industrial Engineering  
Dissertation Advisor, Chair of Committee

Enrique Del Castillo  
Distinguished Professor of Industrial Engineering

Uday V. Shanbhag  
Gary And Sheila Bello Chair Professor

Constantino Lagoa  
Professor of Electrical Engineering

Janis Terpenny  
Peter And Angela Dal Pezzo Department Head/Professor  
Department Head

*Signatures are on file in the Graduate School.
Abstract

The revolution of storage technology in the past few decades made it possible to gather tremendous amount of data anywhere from demand and sales records to web user behavior, customer ratings, software logs and patient data in healthcare. Recognizing patterns and discovering knowledge from large amount of data becomes more and more important, and has attracted significant attention in operations research (OR), statistics and computer science field. Mathematical programming is an essential tool within these fields, and especially for data mining and machine learning, and it plays a significant role for data-driven predictions/decisions and pattern recognition.

The major challenge while solving those large-scale optimization problems is to process large data sets within practically tolerable run-times. This is where the advantages of first-order algorithms becomes clearly apparent. These methods only use gradient information, and are particularly good at computing medium-accuracy solutions. In contrast, interior point method computations that exploit second-order information quickly become intractable, even for moderate-size problems, since the complexity of each factorization of a $n \times n$ matrix in interior point methods is $\mathcal{O}(n^3)$. The memory required for second-order methods could also be an issue in practice for problems with dense data matrices due to limited RAM. Another benefit of using first-order methods is that one can exploit additional structural information of the problem to further improve the efficiency of these algorithms.

In this dissertation, we studied convex regression, and multi-agent consensus optimization problems; and developed new fast first-order iterative algorithms to efficiently compute $\epsilon$-optimal and $\epsilon$-feasible solutions to these large-scale optimization problems in parallel, distributed, or asynchronous computation settings while carefully managing memory usage. The proposed algorithms are able to take advantage of the structural information of the specific problems we considered in this dissertation, and have strong capability to deal with large-scale problems. Our numerical results showed the advantages of our proposed methods over other traditional methods in terms of speed, memory usage, and especially communication requirements for distributed methods.
Table of Contents

List of Figures vi
List of Tables vii

Chapter 1
Introduction 1
1.1 Parallel APG Method for Convex Regression . . . . . . . . . . . . . . . . 2
1.2 A Distributed Proximal Gradient Method for Composite Convex Consen-
sus Optimization (CCCO) . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
1.3 Distributed Linearized ADMM for CCCO . . . . . . . . . . . . . . . . . 4

Chapter 2
A Parallelizable Dual Smoothing Method for Large Scale Convex Regression Problems 6
2.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
2.2 Methodology . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
  2.2.1 Separability . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
  2.2.2 Projected Subgradient Method for Dual . . . . . . . . . . . . . . . . 12
  2.2.3 Tikhonov Regularization Approach . . . . . . . . . . . . . . . . . . 13
  2.2.4 Parallel Accelerated Proximal Gradient (P-APG) Algorithm . . . 16
  2.2.5 Continuation Method for Convex Regression . . . . . . . . . . . . . 23
  2.2.6 Computational complexity of P-APG iterations . . . . . . . . . . . 25
2.3 Competitive Methods . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
  2.3.1 Active Set Method (ASM) . . . . . . . . . . . . . . . . . . . . . . . 31
  2.3.2 Multi-block ADMM . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
2.4 Numerical Study . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36
  2.4.1 Convergence behavior of P-APG on the regularized problem . . . 37
  2.4.2 Comparison with ASM and ADMM . . . . . . . . . . . . . . . . . . 40
2.5 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 44
Chapter 3
Distributed Consensus Optimization 46
3.1 Introduction and Motivation 46
3.2 Problem Statement 49
3.3 Related work 51

Chapter 4
Distributed Proximal Gradient Method for Composite Convex Consensus Optimization 56
4.1 APG Algorithm for the Centralized Model 57
4.2 DFAL Algorithm for the Decentralized Model 58
4.2.1 DFAL iteration complexity 59
4.2.2 Overall computational complexity 65
4.2.3 Synchronous algorithm for distributed optimization 68
4.2.4 Asynchronous implementation 71
4.2.5 Improved rate for asynchronous DFAL 72
4.3 Numerical results 75
4.3.1 A distributed ADMM algorithm 76
4.3.2 Implementation details and numerical results 77

Chapter 5
Distributed Linearized ADMM for Composite Convex Consensus Optimization 82
5.1 Proximal Gradient ADMM 84
5.1.1 Preliminaries, Notations and Simple Identities 86
5.1.2 Convergence Rate of PG-ADMM and SPG-ADMM 87
5.2 Distributed Proximal Gradient Methods 95
5.2.1 DPGA-I Algorithm 96
5.2.2 Error Bounds for DPGA-I & Effect of Network Topology 98
5.2.3 DPGA-II Algorithm 102
5.2.4 Error Bounds for DPGA-II & Effect of Topology 104
5.2.5 Adaptive step-size strategy 108
5.2.6 Stochastic gradient variants of DPGA-I and DPGA-II 108
5.3 Numerical results 109
5.3.1 Distributed ADMM Algorithm 111
5.3.2 PG-EXTRA 112
5.3.3 Implementation details and numerical results 114
5.4 Conclusion 116

Bibliography 117
List of Figures

2.1 Accelerated Proximal Gradient Algorithm ........................................... 16
2.2 Parallel APG Algorithm (P-APG) ....................................................... 18
2.3 Active Set Algorithm (ASM) .............................................................. 32
2.4 Multi-block ADMM (ADMM) ............................................................... 35
2.5 Duality Gap for P-APG and Dual Gradient Ascent: (left) P-APG and Dual Gradient Ascent, (right) Zoom-in for P-APG Method ............................................ 38
2.6 Distance to Feasible Region for P-APG and Dual Gradient Ascent. ............ 38
2.7 P-APG vs ASM $(n,N) = (10,100)$: (left) Relative Suboptimality, (right) Zoom-in for P-APG method ................................................................. 41
2.8 P-APG vs ASM $(n,N) = (10,100)$: (left) Number of Active Constraints for ASM, (right) Normalized infeasibility for P-APG .............................................. 42
2.9 Wall-time Ratio between ADMM and P-APG ........................................ 42

3.1 Three Different Classifiers ..................................................................... 47

4.1 First-order Augmented Lagrangian algorithm ......................................... 59
4.2 Multi Step - Accelerated Prox. Gradient (MS-APG) algorithm .................. 66
4.3 Dist. First-order Aug. Lagrangian (DFAL) alg. ......................................... 70
4.4 Randomized Block Coordinate Descent (RBCD) alg. .............................. 72
4.5 Accelerated Randomized Proximal Block Coordinate Descent (ARBCD) algorithm ................................................................. 73
4.6 Split ADMM algorithm ........................................................................ 76

5.1 Distributed Proximal Gradient Algorithm I (DPGA-I) ............................ 98
5.2 Distributed Proximal Gradient Algorithm II (DPGA-II) .......................... 104
5.3 Stochastic Distributed Proximal Gradient Algorithm I .......................... 110
5.4 Stochastic Distributed Proximal Gradient Algorithm II ......................... 110
5.5 ADMM algorithm ............................................................................... 111
5.6 Split ADMM algorithm ........................................................................ 112
5.7 Proximal Gradient Exact First-order Algorithm (PG-EXTRA) ................. 113
List of Tables

2.1 Comparison of Memory Usage ................................................. 31
2.2 Comparison with test function $\frac{1}{2}x^TQx$ for $n = 20$ .................... 39
2.3 Comparison with test function $\frac{1}{2}x^TQx$ for $n = 80$ .................... 40
2.4 Comparison with test function $\exp(p^Tx)$ for $n = 20$ .................... 40
2.5 Comparison with test function $\exp(p^Tx)$ for $n = 80$ .................... 41
2.6 Comparison of PAPG and other methods $\frac{1}{2}x^TQx$ .................... 43
2.7 Comparison of PAPG and other methods $\exp(p^Tx)$ .................... 44
3.1 Comparison of our method with the previous work .................... 51
3.2 Comparison on data amount stored and communicated per node per communication round (iteration) ............................ 53
4.1 Comparison of DFAL, AFAL (Asynchronous DFAL), ADMM, and SADMM. (Termination time $T=1800$ sec.) .................... 80
5.1 Comparison of DPGA-I, DPGA-II, PG-EXTRA, ADMM, and SADMM (Termination time $T=1800$ sec) ............................ 114
Chapter 1  
Introduction

Big data is suddenly everywhere. The world’s technological capacity to store information has been roughly doubling in every 40 months since the beginnings of 1980s [1]. In this era, extraction of patterns and knowledge from large amount of data is changing the way we understand the world, and has attracted much more attention in academic research communities. One important approach of extracting the useful information from large quantities of data is to cleverly exploit the modern optimization techniques. For instance, manufacturers may determine their optimal resource allocation among all the stock keeping units by maximizing the revenue such that the demand of customer order will be satisfied by the end of each contract. For on-demand Internet streaming media providers, they desire to show the content that is the most potentially relevant to users’ preferences and watch history. This problem can be analyzed by using regression models to study consumers’ behavior, which requires the minimization of regression error subject to certain shape constraints on user utility functions. Another example could be online advertisement allocation problems, where online search engines may consider maximizing their total revenue using optimal ad-placement strategies, yet without violating the customers’ budget constraints.

However, an inevitable issue with these optimization problems is that they all require the capability of processing large data sets within practically tolerable run-times. It is where the so-called first-order algorithms shine. This type of algorithms only utilize gradient or subgradient information; thereby, leading to simple and low-cost iterative schemes. In contrast to first-order methods, one single iteration of interior point methods that requires second-order information, quickly becomes intractable even for moderate-size problems, as the complexity of each factorization step in interior point methods is
\(O(n^3)\) –see [2][3], where \(n\) denotes the size of the factorized system. Furthermore, the memory requirement for interior point methods on problems with dense data matrices is another issue arising in practice, and it becomes more and more significant as the problem dimension increases. Secondly, first-order methods naturally suit for solving large-scale problems; mainly because they are particularly good at computing medium-accuracy solutions – often high-accuracy solutions are vacuous as the problem data is usually somewhat corrupted, noisy or missing anyway in real-world applications. Another benefit of using first-order methods is being able to further exploit problem-specific structural information on the objective and/or constraints, which significantly helps improve the efficiency of these algorithms.

In this dissertation, we developed new fast first-order algorithms for computing large-scale convex optimization problems. Iterations in the proposed algorithms are relatively simple with low computational cost; hence, enabling one to deal with large data sets. Specifically, not only do they outperform the interior point methods in terms of run-time for computing medium-accurate solutions, but also due to efficient memory-management, they can solve practically “unsolvable” large-scale problems.

In the rest of this chapter, we discuss the motivation for each following chapter, and briefly highlight the important results therein.

1.1 Parallel APG Method for Convex Regression

In Chapter 2, we propose a parallel method to solve large-scale convex regression via computing the least squares estimator (LSE), which is defined as the solution to the following problem:

\[
\begin{align*}
\min \ & \{y_\ell, \xi_\ell\}_{\ell=1}^N \subset \mathbb{R} \times \mathbb{R}^n \\
\text{s.t.} \ & y_\ell - \bar{y}_\ell \leq 0 \\
& y_\ell_1 \geq y_\ell_2 + \xi_\ell_2 \left(x_\ell_1 - x_\ell_2\right) \quad 1 \leq \ell_1 \neq \ell_2 \leq N.
\end{align*}
\]

Our proposed method is motivated by the major drawback of LSE: as the number of observations \(N\) increases, the number of shape constraints increases quadratically in \(N\), i.e., the number of convex shape constraints is \(O(N^2)\). Consequently, the problem quickly becomes too massive, even for moderate number of observations, to be solved using interior points methods. We provide iteration complexity results for the proposed
method, P-APG, to compute an $\epsilon$-optimal solution to the LSE problem.

We design the P-APG method with the objective of reducing the impact of $\mathcal{O}N^2$ constraints on the computational cost per iteration of the algorithm. To achieve this goal, we partition the observations into $K$ subsets to introduce separability on the Lagrangian function by partially relaxing some of the constraints. We then solve the partial dual problem using an accelerated dual ascent algorithm, where each iteration requires solving $K$ quadratic subproblems of much smaller dimension corresponding to each subset. This relaxation scheme enables us to compute all $K$ subproblems in parallel. To achieve feasibility of the primal iterate sequence in the limit, we adopted the Tikhonov regularization approach. More importantly, we provide theoretical guarantees on the closeness between the solutions to the regularized and original problems, which addresses one of the most common issues of regularization schemes: how to select an appropriate regularizer to ensure the quality of regularized solutions when compared to the solutions to the original problem. Thanks to the theoretical bound we derived, the decision makers only need to specify a tolerance on the distance between two solutions, and the regularizer can be determined accordingly in practice. We also provide a continuation scheme that can solve the original LSE problem with the same iteration complexity by inexactley solving a sequence of regularized problems.

1.2 A Distributed Proximal Gradient Method for Composite Convex Consensus Optimization (CCCO)

In Chapter 4, we focus on a multi-agent problem over a connected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$. We study how to compute an optimal decision on which there is consensus among agents $\mathcal{N} = \{1, \ldots, N\}$ and that minimizes the sum of agent-specific private convex composite functions $F_i(x) := \rho_i(x) + \gamma_i(x)$ while respecting privacy requirements. Assuming only agents connected by an edge can directly communicate, we propose a distributed first-order augmented Lagrangian algorithm (DFAL) for efficiently computing a solution to the following composite convex optimization problem:

$$F^* := \min_{x \in \mathbb{R}^n} F(x) = \sum_{i=1}^{N} \rho_i(x) + \gamma_i(x),$$
where \( \rho_i : \mathbb{R}^n \to \mathbb{R} \) is a possibly non-smooth convex function, and \( \gamma_i : \mathbb{R}^n \to \mathbb{R} \) is a smooth convex function.

The motivations for considering a distributed optimization approach for the above problem are as follows: (1) solving such a problem in a centralized fashion can be very expensive both from communication and computation perspectives when the problem dimension is large; (2) consolidating the data at a central node may violate agent-specific privacy requirements in the sense that node \( i \) may not want to reveal the details of its data, which it may privately own; (3) the central node needs to have large enough memory to be able to accommodate all the problem data.

We first design an inexact augmented Lagrangian algorithm for a more general problem:

\[
\tilde{F}^* := \min_{x \in \mathbb{R}^{nN}} \tilde{F}(x) := \tilde{\rho}(x) + \tilde{\gamma}(x) \quad \text{s.t.} \quad Ax = b;
\]

we then study its distributed implementation for solving the multi-agent problem by properly choosing the matrix \( A \) and analyze the effect of communication network topology on the speed of convergence in terms of suboptimality and consensus violation, where \( \tilde{\rho}(x) := \sum_{i=1}^{N} \rho_i(x_i) \), \( \tilde{\gamma}(x) := \sum_{i=1}^{N} \gamma_i(x_i) \), and \( A \in \mathbb{R}^{m \times nN} \) such that \( \text{rank}(A) = m \), i.e., the linear map is surjective. The iterate sequence \( \{x^{(k)}\} \) is constructed by inexact solving a sequence of subproblems of the form:

\[
x^{(k)} \in \arg\min_{x \in \mathbb{R}^{nN}} P^{(k)}(x) := \lambda^{(k)} \tilde{\rho}(x) + f^{(k)}(x),
\]

for appropriately chosen sequences of penalty parameters \( \{\lambda^{(k)}\} \) and dual variables \( \{\theta^{(k)}\} \), where \( f^{(k)}(x) := \lambda^{(k)} \tilde{\gamma}(x) + \frac{1}{2} \|Ax - b - \lambda^{(k)} \theta^{(k)}\|^2_2 \). We showed that \text{DFAL} algorithm proposed in Chapter 4 can compute an \( \epsilon \)-optimal, and \( \epsilon \)-feasible solution within at most \( \mathcal{O}\left(\frac{\kappa(A)}{\sigma_{\min}(A)}\epsilon\right) \) proximal operations, where \( \kappa(A) \) and \( \sigma_{\min}(A) \) denote the condition number and the minimum singular value of the matrix \( A \), respectively. We also include a section dedicated to asynchronous/randomized implementation of \text{DFAL} and its acceleration.

### 1.3 Distributed Linearized ADMM for CCCO

In Chapter 5, we consider the same setup as in Chapter 4 and propose another distributed proximal gradient algorithm based on linearized alternating direction method of multiplier
(PG-ADMM) to minimize the sum of agent-specific private convex composite functions $\Phi_i := \xi_i + f_i$ without violating privacy requirements. In one iteration of these algorithms, each agent-$i$ computes both the prox map of $\xi_i$ and gradient of $f_i$, and this is followed by local communications with the neighboring agents that are connected to agent-$i$ with communication links. We also study its stochastic gradient variant, SPG-ADMM, which can only access to noisy estimates of $\nabla f_i$ at each agent-$i$. These computational models in Chapter 4 and 5 abstract a number of applications in distributed sensing, machine learning and statistical inference. We also show the ergodic convergence in both sub-optimality error and consensus violation for PG-ADMM and SPG-ADMM with rates $O(1/t)$ and $O(1/\sqrt{t})$, respectively. We implement PG-ADMM and SPG-ADMM on two different, but equivalent, consensus formulations and examine the effect of the underlying network topology on their convergence rates.
Chapter 2 | A Parallelizable Dual Smoothing Method for Large Scale Convex Regression Problems

2.1 Introduction

Convex regression (CR) problem deals with fitting a convex function to a given finite set of location/observation pairs, where each pair consists of a vector of independent variables and corresponding scalar dependent variable. In particular, suppose $N$ location/observation pairs are given \( \{(\bar{x}_\ell, \bar{y}_\ell)\}_{\ell=1}^N \subset \mathbb{R}^n \times \mathbb{R} \) satisfying

\[
\bar{y}_\ell = f_0(\bar{x}_\ell) + \varepsilon_\ell, \quad \ell = 1, \ldots, N, \tag{2.1}
\]

where \( f_0 : \mathbb{R}^n \to \mathbb{R} \) is a convex function, and \( \varepsilon_\ell \) is a random noise with \( E[\varepsilon_\ell] = 0 \) for all \( \ell \). The objective is to infer the convex function \( f_0 \) from the noisy observations \( \{(\bar{x}_\ell, \bar{y}_\ell)\}_{\ell=1}^N \).

CR problems arise in various applications coming from diverse fields such as statistics, operations research, economics, and electrical engineering. M. Mousavi [4] used CR to estimate the value function for Markov chains with expected infinite-horizon discounted rewards, which naturally arises in various control problems, and estimating value functions is essential for approximate dynamic programming and applied probability. In economics, CR has been adopted for approximating consumers’ concave utility functions from empirical data [5]. Moreover, in queueing network context, when the expectation of a performance measure is convex in model parameters – see [6], then using Monte Carlo methods to compute the expectation gives rise to a CR problem [7].

CR was first studied in [8] for estimating concave production functions. Later, various
solution methods were proposed in the uni-variate setting, e.g., [9]–[11]. The problem of fitting a convex function in the multi-variate setting has been considered in [12], [13] where the minimization of the least squares (LS) error subject to the first-order convexity shape constraints is studied; furthermore, [14], [15] also considered the same approach with additional second-order convexity constraints. The most well-known method for CR is to solve the LS problem,

\[
\hat{f}_N = \arg\min_{f \in \mathcal{C}} \sum_{\ell=1}^{N} \left( f(\bar{x}_\ell) - \bar{y}_\ell \right)^2,
\]

(2.2)

where \( \mathcal{C} \triangleq \{ f : \mathbb{R}^n \to \mathbb{R} \text{ such that } f \text{ is convex} \} \). This infinite dimensional problem is equivalent to a finite dimensional quadratic problem (QP) given in (2.3) – see Proposition 1 in [7],

\[
\min_{y_\ell \in \mathbb{R}, \xi_\ell \in \mathbb{R}^n} \left\{ \sum_{\ell=1}^{N} |y_\ell - \bar{y}_\ell|^2 : y_{\ell_1} - y_{\ell_2} + \xi_\ell_1^\top (\bar{x}_{\ell_1} - \bar{x}_{\ell_2}) \geq 0, \quad 1 \leq \ell_1 \neq \ell_2 \leq N \right\}.
\]

(2.3)

Indeed, let \( \{(y^*_\ell, \xi^*_\ell)\}_{\ell=1}^{N} \) be an optimal solution to (2.3), it is easy to show that when \( N \geq n + 1 \), \( \{y^*_\ell\}_{\ell=1}^{N} \) is unique, \( \hat{f}_N(\bar{x}_\ell) = y^*_\ell \) and \( \xi^*_\ell \in \partial \hat{f}_N(\bar{x}_\ell) \) for all \( \ell \), where \( \partial \) denotes the subdifferential operator. The theoretical behavior of the LS estimator has been studied thoroughly in the past 50 years. In the univariate setting, i.e., \( n = 1 \), the consistency of the LS estimator is proved in [16]; and the convergence rate of the estimator is established in [17]. Groeneboom et al. [18] extended these results and derived the asymptotic distribution of LS estimator at a fixed point of positive curvature. In the multivariate setting, the consistency is shown in [7], i.e., \( \hat{f}_N \to f_0 \) almost surely as \( N \) increases.

Besides LS estimator, there are other methods for solving CR problem in the multivariate setting. A heuristic approach is proposed in [19] to compute locally optimal fits, which has no convergence guarantee. A convex adaptive partitioning (CAP) method is proposed in [20], which creates a globally convex regression model via computing locally linear fits on adaptively selected covariate partitions. Both methods use the piecewise linear model, and minimize the least square error. In addition, more recently, Hannah and Dunson [21] proposed a new estimator based on using traditional ensemble methods to average over multiple piecewise linear estimators, and proved its consistency when CAP is the underlying estimator. However, LS estimator has some significant advantages over the methods mentioned above. First, LS estimator is a non-parametric regression...
method as discussed in [22], which does not require any tuning parameters and avoids the issue of selecting an appropriate estimation structure; however, as also pointed out in [4], the methods proposed in [20, 23] are semi-parametric, and require adjusting several parameters before fitting a convex function. Second, LS estimator can be computed by solving the QP in (2.3); therefore, at least in theory, it can be solved very efficiently using interior point methods (IPM). A major drawback of the LS estimator in practice is that the number of shape constraints in (2.3) is $O(N^2)$. Consequently, the problem quickly becomes massive even for moderate number of observations: for off-the-shelf IPMs that do not exploit any structural properties of (2.3), the complexity of each factorization step is $O(N^3(n + 1)^3)$, and the memory requirement is $O(N^2(n + 1)^2)$ assuming Cholesky factors are stored - see [2, 3] – for more detailed discussion on memory usage and computational complexity of both IPM and our proposed method (exploiting the structure), see Section 2.2.6.

In this chapter, we propose a new parallelizable method for computing the LS estimator on large-scale CR problems. The proposed method can efficiently solve large-scale instances of (2.3) by carefully managing the memory usage through parallelization, and exploiting the underlying problem structure. In particular, the proposed method, P-APG, is based on dual smoothing, i.e., regularizing the objective in (2.3) with a strongly convex function. More specifically, we adopted Tikhonov regularization, which leads to a differentiable dual function with a Lipchitz continuous gradient. Compared to the traditional dual decomposition methods, the dual smoothing based approaches can guarantee feasibility of primal iterate sequence in the limit. To briefly summarize, P-APG is an iterative method to solve the regularized QP problem in (2.7) through solving a number of small-size QPs in each iteration. In our main results, Theorem 2.12 and 2.13, we establish error bounds on the quality of inexact solutions to the regularized problem; particularly, we investigate how well the inexact solutions can approximate i) function values of the LS estimator, i.e., $\hat{f}_N(\bar{x}_\ell)$, and ii) subgradients from the subdifferential of the LS estimator, i.e., $\partial \hat{f}_N(\bar{x}_\ell)$. Next, we study the convergence behavior of P-APG to compute these function value and subgradient approximations. In Section 2.2.5, we show that using a continuation method, we can construct an iterate sequence that is asymptotically optimal to the original LS problem in (2.3) with a provable convergence rate. We adopted a primal-dual IPM to solve the small-size QP subproblems arising in P-APG iterations, and analyzed the computational complexity of an P-APG iteration by exploiting the special structure of the constraints and the objective function. In the
As alternatives to P-APG, we examined how an active set method (ASM) can be efficiently implemented to solve (2.3), and briefly discussed a recently proposed ADMM algorithm \[24\] for (2.3). Finally, we conclude with a number of numerical examples comparing P-APG, ASM, and ADMM. Our results show that P-APG is the method of choice for large \(N\).

**Notations:** Throughout, i.i.d. is short for independent and identically distributed. \(I_n\) denotes the \(n \times n\)-identity matrix. Given \(x \in \mathbb{R}^n\), \((x)_+ \triangleq \max\{x, 0\}\) and \((x)_- \triangleq \min\{x, 0\}\); hence, \(x = (x)_+ + (x)_-\). For \(x, y \in \mathbb{R}^n\), \(\langle x, y \rangle \triangleq x^\top y\) represents the standard inner product. \(1\) denotes the vector of all ones, and \(e_i \in \mathbb{R}^n\) denotes the \(i\)-th unit vector for each \(i \in \{1, \ldots, n\}\).

### 2.2 Methodology

Let \(f_0 : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}\) be the unknown proper convex function generating the observed data \(\{(\bar{x}_\ell, \bar{y}_\ell)\}_{\ell=1}^N \subset \mathbb{R}^n \times \mathbb{R}\) as in (2.1), and let \(\mathcal{N} := \{1, \ldots, N\}\) denote the set of indices corresponding to \(N\) observations. Suppose \(B_x > 0\) such that \(\|\bar{x}_\ell\|_2 \leq B_x\) for all \(\ell \in \mathcal{N}\). Define the long-vector notations for the variables: \(y = [y_\ell]_{\ell \in \mathcal{N}} \in \mathbb{R}^N\), and \(\xi = [\xi_{\ell}]_{\ell \in \mathcal{N}} \in \mathbb{R}^{Nn}\).

Consider (2.3) in the following compact form:

\[
\chi^* \triangleq \arg\min_{y \in \mathbb{R}^N, \xi \in \mathbb{R}^{Nn}} \left\{ \frac{1}{2} \|y - \bar{y}\|_2^2 : \quad A_1 y + A_2 \xi \geq 0 \right\},
\]

where \(A_1 \in \mathbb{R}^{N(N-1) \times N}\) and \(A_2 \in \mathbb{R}^{N(N-1) \times Nn}\) are the matrices corresponding to constraints in (2.3). Let \((y^*, \xi^*)\) be the least-norm optimal solution in \(\chi^*\), i.e.,

\[
(y^*, \xi^*) \triangleq \arg\min_{y, \xi} \left\{ \frac{1}{2} \|y\|_2^2 + \frac{1}{2} \|\xi\|_2^2 : \quad (y, \xi) \in \chi^* \right\}.
\]

It is easy to show that \(y^*\) is unique to (2.4), i.e., if \((y, \xi) \in \chi^*\), then \(y = y^*\) — see Proposition 1 in [7]. Hence, it follows from (2.5) that \(\xi^*\) has the least norm, i.e., for all \((y, \xi) \in \chi^*\), one has \(\|\xi\|_2 \geq \|\xi^*\|_2\). Moreover, since (2.4) is a convex QP, strong duality holds, and an optimal dual solution \(\theta^* \in \mathbb{R}^{N(N-1)}\) exists.

Note for each \((\ell_1, \ell_2) \in \mathcal{P} \triangleq \{(\ell_1, \ell_2) \in \mathcal{N} \times \mathcal{N} : \ell_1 \neq \ell_2\}\), there is a constraint in (2.3), i.e., \(y_{\ell_2} - y_{\ell_1} + \xi_{\ell_1}^\top (\bar{x}_{\ell_1} - \bar{x}_{\ell_2}) \geq 0\) corresponds to \((\ell_1, \ell_2) \in \mathcal{P}\). In order to fix \(A_1\) and \(A_2\), we sort the rows according to increasing lexicographic order on the index set
\( P \), i.e., the row for the constraint corresponding to \((\ell_1, \ell_2)\) comes before than the one corresponding to \((\ell_3, \ell_4)\) if either \(\ell_1 < \ell_3\), or \(\ell_2 < \ell_4\) in case \(\ell_1 = \ell_3\). Next, we give explicit forms for \(A_1\) and \(A_2\).

**Definition 2.1.** Let \(T_\ell \in \mathbb{R}^{N-1 \times N}\) such that 
\[
T_\ell = [e_1 \cdots e_{\ell-1} - 1 e_\ell \cdots e_{N-1}]
\]
for \(\ell \in \mathcal{N}\), where \(e_j \in \mathbb{R}^{N-1}\) is the \(j\)-th unit vector for \(j \in \{1, \ldots, N-1\}\). Moreover, let \(\bar{X} \in \mathbb{R}^{N \times n}\) such that \(\bar{X} = [\bar{x}_\ell]_{\ell \in \mathcal{N}}\), i.e., \(\{\bar{x}_\ell\}_{\ell \in \mathcal{N}}\) are the rows of \(\bar{X}\). Then \(A_1 = [T_\ell]_{\ell \in \mathcal{N}}\), obtained by vertically concatenating \(\{T_\ell\}_{\ell \in \mathcal{N}}\), and \(A_2 = \text{diag}\left(\{-T_\ell \bar{X}\}_{\ell \in \mathcal{N}}\right)\) is a block-diagonal matrix as given below.

\[
A_1 = \begin{pmatrix} T_1 \\ T_2 \\ \vdots \\ T_N \end{pmatrix}, \quad A_2 = \begin{pmatrix} X_1 & 0 & \cdots & 0 \\ 0 & X_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & X_N \end{pmatrix}, \quad X_\ell \triangleq -T_\ell \bar{X}, \quad \ell \in \mathcal{N}.
\tag{2.6}
\]

### 2.2.1 Separability

Given the regularization parameter \(\gamma \geq 0\), consider

\[
(y^*_\gamma, \xi^*_\gamma) \triangleq \arg\min_{y, \xi} \left\{ r_\gamma(y, \xi) \triangleq \frac{1}{2} \|y - \bar{y}\|_2^2 + \frac{\gamma}{2} \|\xi\|_2^2 : A_1 y + A_2 \xi \geq 0 \right\}.
\tag{2.7}
\]

Note simply setting \(\gamma = 0\) in (2.7), we obtain the original problem (2.4).

To reduce the curse of dimensionality and develop a parallelizable method that can solve problems in (2.4) and (2.7) for large \(N\), we employ dual decomposition to induce separability. To this aim, we partition the observation set into \(K\) subsets \(\{C_i\}_{i \in \mathcal{K}}\), where \(\mathcal{K} \triangleq \{1, \ldots, K\}\) denote the set of indices corresponding to \(K\) subsets of \(\mathcal{N}\). In particular, we choose \(\{C_i\}_{i \in \mathcal{K}}\) as a partition of \(\mathcal{N}\) such that \(|C_i| \geq n + 1\) for all \(i\). To simplify the notation, throughout this chapter we make the following assumption.

**Assumption 2.2.** Suppose \(N = K\bar{N}\) for some \(\bar{N} > n + 1\), and without loss of generality assume that \(C_i \triangleq \{ (i-1)\bar{N} + 1, (i-1)\bar{N} + 2, \ldots, i\bar{N} \} \) for \(i \in \mathcal{K}\).

Throughout the chapter, for each \(i \in \mathcal{K}\), let \(y_i \in \mathbb{R}^{\bar{N}}\) and \(\xi_i \in \mathbb{R}^{n\bar{N}}\) denote the sub-vectors of \(y \in \mathbb{R}^N\) and \(\xi \in \mathbb{R}^{Nn}\) corresponding to indices in \(C_i\), respectively. In particular, for all \(i \in \mathcal{K}\), \(y_i = [y_\ell]_{\ell \in C_i}\) and \(\xi_i = [\xi_\ell]_{\ell \in C_i}\). Similarly, we define the same long-vectors for the observation data: \(\bar{y}_i = [\bar{y}_\ell]_{\ell \in C_i} \in \mathbb{R}^{\bar{N}}\).
Definition 2.3. Define \( P \triangleq \{ (\ell_1, \ell_2) \in N \times N : \ell_1 \neq \ell_2 \} \) and \( G \triangleq \{ (i, j) \in K \times K : i \neq j \} \). For each \( i \in K \), let \( A_1^{ij} \in \mathbb{R}^{N(N-1) \times N} \) and \( A_2^{ij} \in \mathbb{R}^{\tilde{N}(N-1) \times N} \) be the submatrices of \( A_1 \) and \( A_2 \) such that they consist of the rows corresponding to row indices \((\ell_1, \ell_2) \in P\) for \( \ell_1, \ell_2 \in C_i \). Similarly, for each \((i, j) \in G\), let \( A_1^{ij} \in \mathbb{R}^{N_2 \times N} \) and \( A_2^{ij} \in \mathbb{R}^{\tilde{N}_2 \times N} \) be the submatrices of \( A_1^{ij} \) and \( A_2^{ij} \) such that \( A_1^{ij} \) consists of the columns of \( A_1^{ij} \) corresponding to \( y_i \); and \( A_2^{ij} \) consists of the columns of \( A_2^{ij} \) corresponding to \( \xi_i \).

Furthermore, for each \( i \in K \), let \( \bar{A}_1^{ii} \in \mathbb{R}^{N(N-1) \times \bar{N}} \) and \( \bar{A}_2^{ii} \in \mathbb{R}^{\tilde{N}(N-1) \times \bar{N}} \) be the submatrices of \( \bar{A}_1^{ii} \) and \( \bar{A}_2^{ii} \) such that \( \bar{A}_1^{ii} \) consists of the columns of \( \bar{A}_1^{ii} \) corresponding to \( \bar{y}_i \); and \( \bar{A}_2^{ii} \) consists of the columns of \( \bar{A}_2^{ii} \) corresponding to \( \bar{\xi}_i \).

Note that for every ordered pair \((\ell_1, \ell_2) \in P\), there corresponds a constraint in (2.3), which is represented by a row in matrices \( A_1 \) and \( A_2 \) of formulations (2.4) and (2.7). Consider all the constraints in (2.3) corresponding to those pairs \((\ell_1, \ell_2)\) such that they belong to different sets in the partition, i.e., \( \ell_1 \in C_i, \ell_2 \in C_j \) for some \((i, j) \in G\), let \( \theta_{ij} \in \mathbb{R}^{N_2} \) denote the associated dual variables, and \( \Theta = [\theta_{ij}]_{(i,j) \in G} \in \mathbb{R}^{N_2K(K-1)} \) denote the vector formed by vertically concatenating \( \theta_{ij} \) for \( 1 \leq i \neq j \leq K \). By dualizing all such constraints in (2.7), we form the partial Lagrangian function:

\[
\mathcal{L}_\gamma (y, \xi, \theta) \triangleq \frac{1}{2} \sum_{i \in K} \left( \| y_i - \bar{y}_i \|_2^2 + \gamma \| \xi_i \|_2^2 \right) - \sum_{(i,j) \in G} \langle \theta_{ij}, A_1^{ij} y + A_2^{ij} \xi \rangle. \tag{2.8}
\]

and obtain the following partial dual function

\[
g_\gamma (\theta) \triangleq \min_{y, \xi} \left\{ \mathcal{L}_\gamma (y, \xi, \theta) : A_1^{ij} y + A_2^{ij} \xi \geq 0, i \in K \right\}. \tag{2.9}
\]

Hence, the dual problem corresponding to (2.7) is given as

\[
\Theta^*_\gamma \triangleq \arg\max \{ g_\gamma (\theta) : \theta \geq 0 \}, \quad \text{and} \quad p^*_\gamma \triangleq g_\gamma (\Theta^*_\gamma) \quad \text{for} \quad \theta^*_\gamma \in \Theta^*_\gamma. \tag{2.10}
\]

Since strong-duality trivially holds between the primal-dual problem pair, (2.7) and (2.10), we have

\[
p^*_\gamma = r_\gamma (y^*_\gamma, \xi^*_\gamma) = \frac{1}{2} \| y^*_\gamma - \bar{y} \|_2^2 + \frac{\gamma}{2} \| \xi^*_\gamma \|_2^2. \tag{2.11}
\]

For any given regularization parameter \( \gamma \geq 0 \) and dual variable \( \theta \), the partial Lagrangian
function $\mathcal{L}_\gamma$ is separable in $\{(y_i, \xi_i)\}_{i \in \mathcal{K}}$, and can be written as

$$\mathcal{L}_\gamma (y, \xi, \theta) = \sum_{i \in \mathcal{K}} \mathcal{L}_\gamma^i (y_i, \xi_i, \theta)$$

for some very simple quadratic function, $\mathcal{L}_\gamma^i$, of $(y_i, \xi_i)$ for each $i \in \mathcal{K}$. Moreover, after partially dualizing some of the constraints as shown in (2.8), the remaining ones in (2.9) define a superset, $\mathcal{Q}$, of the original feasible region. Indeed,

$$\mathcal{Q} = \{ (y, \xi) : A_1^{y} y + A_2^{\xi} \xi \geq 0, \ i \in \mathcal{K} \} = \{ (y, \xi) : \bar{A}_1^{y} y_i + \bar{A}_2^{\xi} \xi_i \geq 0, \ i \in \mathcal{K} \} - \bar{A}_2^{\xi} \xi_i = (\mathcal{Q}_i)_{i \in \mathcal{K}} - \bar{A}_2^{\xi} \xi_i = (\mathcal{Q}_i)_{i \in \mathcal{K}}$$

Consequently, since $\mathcal{L}_\gamma$ is separable as shown in (2.12), computing the partial dual function $g_\gamma (\theta)$ in (2.9) is equivalent to solving $K$ quadratic subproblems, i.e., one for each $i \in \mathcal{K}$,

$$\min_{y_i, \xi_i \in \mathbb{R}^{\bar{N}_i}, \xi \in \mathbb{R}^{\bar{k}_i}} \left\{ \mathcal{L}_\gamma^i (y_i, \xi_i, \theta) : \bar{A}_1^{y} y_i + \bar{A}_2^{\xi} \xi_i \geq 0 \right\}.$$  

(2.13)

Given the dual variables $\theta$, since all $K$ subproblems can be computed in parallel, one can take advantage of the computing power of multi-core processors. In the rest of the chapter, we discuss how to compute a solution to (2.3) via solving the dual problem: 

$$\max \{ g_\gamma (\theta) : \theta \geq 0 \}.$$  

2.2.2 Projected Subgradient Method for Dual

Clearly, for $\gamma = 0$, $g_0$ defined in (2.9) is the dual function for the original problem (2.4); and the projected subgradient method can be adopted for solving the dual problem 

$$\max \{ g_0 (\theta) : \theta \geq 0 \}.$$  

Let $\theta = 0$, i.e., $\theta^0 = 0$ for all $(i, j) \in \mathcal{G}$. Given the $k$-th dual iterate $\theta^k$, let $(y^k, \xi^k)$ denote an optimal solution to the minimization problem in (2.9) when $\gamma = 0$ and $\theta$ is set to $\theta^k$; and let $\theta^k_i$ denote an optimal dual associated with constraints $A_1^{y} y + A_2^{\xi} \xi \geq 0$ in (2.9). The next dual iterate $\theta^{k+1}$ is computed for an appropriately chosen step size $t_k > 0$:

$$\theta^{k+1}_{ij} = \Pi_{S_{ij}} \left( \theta^k_{ij} - t_k \left( A_1^{y} y^k + A_2^{\xi} \xi^k \right) \right),$$

(2.14)
where \( \Pi_{S^k_{ij}}(\cdot) \) denotes the Euclidean projection on to

\[
S^k_{ij} \triangleq \left\{ \theta_{ij} \geq 0 : \theta_{ij}^T A_{ij} + \theta_{ii}^T A_{ii} = 0 \right\}.
\]

Since the Lagrangian function \( L_0 \) is linear in \( \xi \) when \( \gamma = 0 \), \( \text{dom} g_0 \) is non-trivial; hence the projection on to the Cartesian product \( \bigotimes_{(i,j) \in G} S^k_{ij} \) ensures \( \theta^{k+1} \in \text{dom} g_0 \).

The projected subgradient method is guaranteed to converge in function value for a diminishing step size sequence \( \{t_k\}_{k=1}^\infty \), and it requires \( O(1/\epsilon^2) \) iterations to obtain an \( \epsilon \)-optimal solution – see [25]. On the other hand, even if the dual iterates converge to an optimal dual solution \( \theta^* \), the primal feasibility of the corresponding primal iterate sequence \( \{(y^k, \xi^k)\} \) cannot be guaranteed in the limit as it might converge to a stationary point of the Lagrangian \( L_0(\cdot, \cdot, \theta^*) \) that is primal infeasible, mainly due to lack of strict convexity, jointly in \( (y, \xi) \), of the objective in (2.4).

### 2.2.3 Tikhonov Regularization Approach

In order to ensure feasibility in the limit, which cannot be guaranteed by the subgradient method discussed above, we employ Tikhonov regularization as in (2.7) for \( \gamma > 0 \), of which convergence properties in general were investigated in [26]. In particular, as \( \gamma \) decreases to zero from above, the minimizer \( (y^*_\gamma, \xi^*_\gamma) \), as a function of \( \gamma \), converges to \( (y^*, \xi^*) \in \chi^* \) defined in (2.5), i.e., \( \xi^* \) has the least norm among all \( (y^*, \xi) \in \chi^* \).

**Lemma 2.4.** The minimizer of (2.7), \( y^*_\gamma \), as a function of the regularization parameter \( \gamma \), is Hölder continuous from right at \( \gamma = 0 \). In particular,

\[
\|y^*_\gamma - y^*\|_2 \leq \|\xi^*\|_2 \sqrt{\gamma}, \quad \forall \gamma \geq 0.
\]  

**Proof:** Let \( (y^*_\gamma, \xi^*_\gamma) \) be the optimal solution to (2.7) and \( (y^*, \xi^*) \) be defined as in (2.5). Note that \( (y^*, \xi^*) \) and \( (y^*_\gamma, \xi^*_\gamma) \) are feasible to (2.7) and (2.4), respectively; hence, from the first-order optimality conditions of (2.7) and (2.4), we have

\[
\begin{pmatrix}
(y^*_\gamma - \bar{y})^T \\
\gamma \xi^*_\gamma
\end{pmatrix} 
\begin{pmatrix}
y^* - y^*_\gamma \\
\xi^* - \xi^*_\gamma
\end{pmatrix} \geq 0, \quad \begin{pmatrix}
y^* - \bar{y} \\
0
\end{pmatrix}^T 
\begin{pmatrix}
y^*_\gamma - y^* \\
\xi^*_\gamma - \xi^*
\end{pmatrix} \geq 0.
\]  

Moreover, since \( (y^*, \xi^*) \) and \( (y^*_\gamma, \xi^*_\gamma) \) are optimal to (2.4) and (2.7), respectively; we also
have

\[ \frac{1}{2} \| y^* - \bar{y} \|_2^2 \leq \frac{1}{2} \| y^*_\gamma - \bar{y} \|_2^2, \quad \frac{1}{2} \| y^* - \bar{y} \|_2^2 + \frac{3}{2} \| \xi^*_\gamma \|_2^2 \leq \frac{1}{2} \| y^* - \bar{y} \|_2^2 + \frac{3}{2} \| \xi^* \|_2^2. \]

These two inequalities imply \( \| \xi^*_\gamma \|_2 \leq \| \xi^* \|_2 \). Finally, summing the two inequalities in (2.16) and using Cauchy-Schwarz, we obtain

\[ \| y^*_\gamma - y^* \|_2^2 \leq \gamma \xi^*_\gamma^T ( \xi^* - \xi^*_\gamma ) \leq \gamma \left( \| \xi^* \|_2^2 - \| \xi^*_\gamma \|_2^2 \right) \leq \gamma \| \xi^* \|_2^2, \]

which implies the desired result.

Since the objective function in (2.7) is strongly convex, jointly in \( y \) and \( \xi \), when \( \gamma > 0 \), Danskin’s theorem (see [27]) implies that \( g_{\gamma} \), i.e., the Lagrangian dual function corresponding to (2.7), is differentiable; therefore, one can use gradient type methods to solve the corresponding dual problem \( \max \{ g_{\gamma}(\theta) : \theta \geq 0 \} \). Moreover, strong convexity ensures that, one can solve the regularized primal problem in (2.7) by solving the associated dual problem in (2.10). Indeed, let \( \theta^*_\gamma \) be an optimal solution to (2.10), we can recover \( (y^*_\gamma, \xi^*_\gamma) \) by computing the primal minimizers in (2.9) when the dual is set to \( \theta^*_\gamma \). In particular, achieving primal feasibility in the limit for the primal iterate sequence is not an issue provided that we can construct a dual iterate sequence that is asymptotically optimal to (2.10). We complete this section by formally stating this result.

**Theorem 2.5.** Let \( \gamma > 0 \), and \( \{ \theta^k \} \) be some dual sequence such that \( \theta^k \geq 0 \) for \( k \geq 1 \) and \( \lim_{k \in \mathbb{Z}_+} g_{\gamma}(\theta^k) = p^*_\gamma \). Moreover, let \( (y^k, \xi^k) \) denote the unique optimal solution to the minimization problem in (2.9) when \( \theta \) is set to \( \theta^k \) for \( k \geq 1 \). Then \( (y^*_\gamma, \xi^*_\gamma) \) is the unique limit point of the primal sequence \( \{ (y^k, \xi^k) \} \). More specifically, for all \( k \geq 1 \), we have

\[ \| y^k - y^*_\gamma \|_2^2 + \gamma \| \xi^k - \xi^*_\gamma \|_2^2 \leq 2 \left( p^*_\gamma - g_{\gamma}(\theta^k) \right) \to 0. \]  

(2.17)

**Proof:** Let \( Q = \{(y, \xi) : A^i_1 y + A^i_2 \xi \geq 0, \ i \in \mathcal{K} \} \). Given \( \theta^k \geq 0 \) for any \( k \geq 1 \), since \( L_{\gamma}(y, \xi, \theta^k) \) is a quadratic function in \( (y, \xi) \), we can compute \( L_{\gamma}(y^*_\gamma, \xi^*_\gamma, \theta^k) \) by using second-order Taylor expansion of around \( (y^k, \xi^k) \):

\[
\begin{align*}
L_{\gamma}(y^*_\gamma, \xi^*_\gamma, \theta^k) &= L_{\gamma}(y^k, \xi^k, \theta^k) + \nabla_y L_{\gamma}(y^k, \xi^k, \theta^k)^T (y^*_\gamma - y^k) + \frac{1}{2} (y^*_\gamma - y^k)^T \nabla^2 L_{\gamma}(y^k, \xi^k, \theta^k) (y^*_\gamma - y^k) + \nabla_\xi L_{\gamma}(y^k, \xi^k, \theta^k)^T (\xi^*_\gamma - \xi^k) + \frac{1}{2} (\xi^*_\gamma - \xi^k)^T \nabla^2 L_{\gamma}(y^k, \xi^k, \theta^k) (\xi^*_\gamma - \xi^k).
\end{align*}
\]
Note that \( g_\gamma(\theta^k) = L(y^k, \xi^k, \theta^k) \), and since \((y^*_\gamma, \xi^*_\gamma) \in Q\), the first-order optimality condition for \((y^k, \xi^k)\) implies that the second term on the right-hand side of the above equality is non-negative. Therefore,

\[
p_* \gamma \geq p_* \gamma - \sum_{(i,j) \in G} \langle \theta^k_{ij}, A^i_1 y^*_\gamma + A^i_2 \xi^*_\gamma \rangle \]

\[
= L_\gamma(y^*_\gamma, \xi^*_\gamma, \theta^k) \geq g_\gamma(\theta^k) + \frac{1}{2} \left( \|y^*_\gamma - y^k\|_2^2 + \gamma \|\xi^*_\gamma - \xi^k\|_2^2 \right),
\]

where the first inequality above follows from \( \theta^k \geq 0 \) and \( A^i_1 y^*_\gamma + A^i_2 \xi^*_\gamma \geq 0 \) for all \((i, j) \in G\) – since \((y^*_\gamma, \xi^*_\gamma)\) satisfies all constraints in (2.7).

**Corollary 2.6.** Let \( \gamma = 0 \), and \( \{\theta^k\} \) be some dual sequence such that \( \theta^k \geq 0 \) for \( k \geq 1 \) and \( \lim_{k \to \infty} g_0(\theta^k) = p_0^* \), i.e., \( p_0^* = \frac{1}{2} \|y^* - \bar{y}\|_2^2 \), where \( y^* \) is the unique optimal solution defined in (2.5). Moreover, let \((y^k, \xi^k)\) denote an optimal solution to the minimization problem in (2.9) when \( \theta \) is set to \( \theta^k \) for \( k \geq 1 \). \( y^* \) is the unique limit point of the primal sequence \( \{y^k\} \). More specifically, we have

\[
\|y^k - y^*\|_2 \leq \left( p_0^* - g_0(\theta^k) \right) \to 0. \tag{2.18}
\]

In the rest of the chapter, we design methods based on dual decomposition to solve the convex regression problem in (2.3) or its regularized version in (2.7) when \( N \) is large. Suppose \( N \) is so large that solving either (2.3), or (2.7) using IPM is infeasible due to high memory requirements caused by \( O(N^2) \) shape constraints. In this scenario, using dual decomposition methods, including the methods proposed in this chapter, reduces the memory overhead; but, this will come at the cost of considerable increase in the run time if a high-accuracy solution is desired. That being said, in many applications, low-to-moderate-accuracy approximate solutions usually have significant value to the practitioner; this is when dual decomposition based first-order methods become attractive. Therefore, it is important to understand how the approximation quality of iterate sequence \( \{(y^k, \xi^k)\} \) changes as the algorithm runs, in order to better assess the trade of between memory requirement and convergence rate of the method chosen.

Our first objective is to study the rate of convergence in more detail. In particular, Corollary 2.6 implies that the projected subgradient method discussed in Section 2.2.2 guarantees \( \|y^k - y^*\|_2 \approx O(1/\sqrt{k}) \) rate. On the other hand, inspired by Nesterov’s smoothing for solving structured non-smooth problems in [28], we can improve the convergence rate. Indeed, combining the result of Lemma 2.4 with Theorem 4.11 we see
that the convergence rate in function values for the smoothed dual problem in (2.10) implies $y^k \to y^*$, and an $\epsilon$-optimal solution $y_\epsilon$, i.e., $\|y_\epsilon - y^*\|_2^2 \leq \epsilon$, can be computed in $O(1/\epsilon)$ iterations.

Our second objective is to study the convergence behavior of $\{\xi^k\}$ sequence. As discussed before in Section 2.2.2, when $\gamma = 0$, using the projected subgradient method cannot guarantee the asymptotic feasibility of $\{(y^k, \xi^k)\}$; in particular, although $y^k_\ell \to y^*_\ell = \hat{f}_N(\bar{x}_\ell)$ for all $\ell \in \mathcal{N}$, $\{\xi^k_\ell\} \subset \mathbb{R}^n$ may not converge to a point in $\partial \hat{f}_N(\bar{x}_\ell)$ for some $\ell \in \mathcal{N}$. This might be an issue to consider when designing algorithms for convex regression, as for some applications having error bounds on how $\{\xi^k_\ell\}$ approximates a subgradient at $\bar{x}_\ell$ might be as important as having error bounds on how $\{y^k_\ell\}$ approximates the function value at $\bar{x}_\ell$. For instance, when the objective is to fit concave utility functions to consumer data, subgradients can be used to infer consumers’ marginal utilities.

These two objectives motivate the next section, where we briefly state a first-order algorithm to efficiently solve the smoothed dual problem in (2.9).

Algorithm \text{APG}(\theta^0)

Iteration 0: Take $\bar{\theta}^1 = \theta^0$, $t_1 = 1$

Iteration $k$: ($k \geq 1$) Compute

1. $\theta^k \leftarrow \Pi_{\mathcal{Q}} \left( \bar{\theta}^k + \frac{1}{L} \nabla \rho(\bar{\theta}^k) \right)$
2. $t_{k+1} \leftarrow (1 + \sqrt{1 + 4t_k^2})/2$
3. $\bar{\theta}^{k+1} \leftarrow \theta^k + \frac{t_{k+1} - 1}{t_{k+1}} (\theta^k - \theta^{k-1})$

Figure 2.1: Accelerated Proximal Gradient Algorithm

2.2.4 Parallel Accelerated Proximal Gradient (P-APG) Algorithm

Let $\rho : \mathbb{R}^d \to \mathbb{R}$ be a concave function such that $\nabla \rho$ is Lipschitz continuous on $\mathbb{R}^d$ with constant $L$, and $\mathcal{Q} \subset \mathbb{R}^d$ be a convex set. Given an initial iterate $\theta^0$, let $\{\theta^k\}$ be the iterate sequence generated using the gradient ascent method as follows: $\theta^{k+1} = \theta^k + \nabla \rho(\theta^k)/L$ for $k \geq 0$. According to Corollary 2.1.2 in [25], the error bound is given by

$$0 \leq \rho^* - \rho(\theta^k) \leq \frac{2L}{k+4} \|\theta^0 - \theta^*\|_2^2,$$

(2.19)
for all $k \geq 1$ and for any $\theta^* \in \arg\min_{\theta \in \mathcal{Q}} \{\rho(\theta) : \theta \in \mathcal{Q}\}$, where $\rho^* = \rho(\theta^*)$. On the other hand, the APG algorithm, [29,30], displayed in Fig. 2.1 is based on Nesterov's accelerated gradient method [25,28]. Corollary 3 in [30], and Theorem 4.4 in [29] show that for all $k \geq 1$ the error bound for APG is given by

$$0 \leq \rho^* - \rho(\theta^k) \leq \frac{2L}{(k+1)^2} \|\theta^0 - \theta^*\|^2_2,$$

(2.20)

where $\theta^0$ is the initial APG iterate and $\theta^* \in \arg\min_{\theta \in \mathcal{Q}} \rho(\theta)$. Hence, using APG one can compute an $\delta$-optimal solution within at most $O\left(\sum L/\delta\right)$ APG iterations. Next, we will customize APG algorithm for solving (2.7) when $\gamma > 0$.

**Definition 2.7.** Let $A_3$ and $A_4$ denote the matrices formed by vertically concatenating $A_1^{ij}$ and $A_2^{ij}$, respectively, for all $(i,j) \in \mathcal{G}$. Define $C \triangleq \begin{bmatrix} A_3 & A_4 \end{bmatrix}$, the decision variable vector $\eta \triangleq \begin{bmatrix} y^T & \xi^T \end{bmatrix}$, and the following elements related to the regularized problem in (2.7). For $i \in \mathcal{K}$, $Q_i = \{ (y_i, \xi_i) : \bar{A}^{ii} y_i + \bar{A}_2^{ii} \xi_i \geq 0 \}$ and

$$Q \triangleq \{ \eta = (y, \xi) : (y_i, \xi_i) \in Q_i, \ i \in \mathcal{K} \}.$$

Now, consider the equivalent representation of (2.7):

$$\min_{\eta \in Q} \left\{ \frac{1}{2} \|y - \bar{y}\|^2_2 + \frac{\gamma}{2} \|\xi\|^2_2 \right\} \quad \text{s.t.} \quad C \eta \geq 0. \quad (2.21)$$

The objective function in (2.9) for the dual problem in (2.10), i.e., $\max \{ g_\gamma(\theta) : \theta \geq 0 \}$, can be written as

$$g_\gamma(\theta) = \min_{\eta \in Q} \left\{ \frac{1}{2} \|y - \bar{y}\|^2_2 + \frac{\gamma}{2} \|\xi\|^2_2 - \langle \theta, C \eta \rangle \right\}. \quad (2.22)$$

Theorem 7.1 in [31] and Danskin’s theorem imply that

$$\nabla g_\gamma(\theta) = -C \eta(\theta), \quad (2.23)$$

where $\eta(\theta)$ is the unique minimizer in (2.22), and $\nabla g_\gamma(\theta)$ is Lipschitz continuous with constant $L_\gamma$ in (2.24), where $\|C\|$ denotes the spectral norm of $C$.

$$L_\gamma = \frac{1}{\gamma} \sigma^2_{\max}(C) = \frac{1}{\gamma} \|C\|^2. \quad (2.24)$$

17
Algorithm P-APG($\gamma, \theta^0$)

Iteration 0: Set $\tilde{\theta}^1 = \theta^0$, $t_1 = 1$ and $L_\gamma = \frac{1}{\gamma} \sigma_\text{max}(C)$

Iteration $k$: ($k \geq 1$) Compute

1: $\eta^k \leftarrow \arg\min_{\eta \in Q} \left\{ \frac{1}{2} \|y - \bar{y}\|_2^2 + \frac{\gamma}{2} \|\xi\|_2^2 - \langle C^T \tilde{\theta}^k, \eta \rangle \right\}$

2: $\theta^k = \left( \tilde{\theta}^k - \frac{1}{t_k} C \eta^k \right)_+$

3: $t_{k+1} \leftarrow (1 + \sqrt{1 + 4t_k^2})/2$

4: $\tilde{\theta}^{k+1} \leftarrow \theta^k + t_{k+1}^{-1} (\theta^k - \tilde{\theta}^{k-1})$

Figure 2.2: Parallel APG Algorithm (P-APG)

Parallel APG algorithm (P-APG), displayed in Fig. 2.2, is the customized version of APG algorithm in Fig. 2.1 to solve (2.10). Note that the computation in Step-1 can be carried out in parallel using $K$ processors, each solving a small-size QP. Later in Section 2.2.6, we discuss the computational complexity of one P-APG iteration in detail.

Adaptive Step Size Strategy: One important property of APG methods is the ability to adopt an adaptive step-size sequence. Note $L_\gamma$, the Lipschitz constant of $\nabla g_\gamma(\theta)$, may not be known in advance or may be too conservative in practice – leading to very small steps. Instead of constant step size $1/L_\gamma$ in Step-2 of P-APG, if one uses an adaptive step sequence $\{1/s_k\}$, the $O(L_\gamma/k^2)$ rate shown in [29] still holds as long as

\[ g_\gamma(\theta^k) \geq g_\gamma(\tilde{\theta}^k) + \langle \nabla g_\gamma(\tilde{\theta}^k), \theta^k - \tilde{\theta}^k \rangle - \frac{s_k}{2} \|\theta^k - \tilde{\theta}^k\|_2^2, \tag{2.25} \]

holds for all $k$ where $\theta^k$ is computed using $s_k$ instead of $1/L_\gamma$. Clearly, one can choose $s_k \leq L_\gamma$; possibly take longer steps compared to constant step size $1/L_\gamma$ and still has a convergence guarantee with the same rate. We adopted the following rule in our numerical tests: let $\nu > 1$, for $k \geq 1$ we set $s_k = s_{k-1} \ell_k$ where $\ell_k \geq 0$ is the smallest integer such that (5.58) holds, and $s_0 = L_\gamma$.

In the rest of the chapter, other than the numerical section, for the sake of simplicity we assume $s_k = L_\gamma$ for all $k$. To better understand the convergence rate of P-APG, next, we provide a bound on $\|\theta^*_\|_2$ for all $\theta^*_\in \Theta^*_\gamma$. 

18
Lemma 2.8. Given \( \gamma \geq 0 \) and \( \delta \geq 0 \), let \( \theta_{\gamma, \delta} \geq 0 \) be a \( \delta \)-optimal solution to (2.10), i.e.,
\[
0 \leq p_{\gamma}^* - g_{\gamma}(\theta_{\gamma, \delta}) \leq \delta.
\]
Given \( \{\tilde{x}_\ell, \tilde{y}_\ell\}_{\ell \in \mathcal{N}} \), for all \( \ell \in \mathcal{N} \), define \( \tilde{y}_\ell \triangleq \tilde{y} + \frac{\alpha}{2} \|\tilde{x}_\ell - \hat{x}\|^2_2 \) for some given \( \alpha > 0 \), where \( \hat{x} \triangleq \frac{1}{N} \sum_{\ell \in \mathcal{N}} \tilde{x}_\ell \) and \( \tilde{y} \triangleq \frac{1}{N} \sum_{\ell \in \mathcal{N}} \tilde{y}_\ell \). Then
\[
\|\theta_{\gamma, \delta}\|_1 \leq \frac{2}{\alpha v} \left( \delta - p_{\gamma}^* + \frac{1}{2} \sum_{\ell \in \mathcal{N}} (\tilde{y}_\ell - \tilde{\eta}_\ell)^2 + \gamma \alpha^2 \|\tilde{x}_\ell - \hat{x}\|_2^2 \right) \triangleq B(\gamma, \delta, \alpha), \tag{2.26}
\]
where \( v \triangleq \min_{(i,j) \in V} \{|\|\tilde{x}_{\ell_1} - \tilde{x}_{\ell_2}\|^2_2 : \ell_1 \in \mathcal{C}_i, \ell_2 \in \mathcal{C}_j\} \).

Proof: For given \( \alpha > 0 \), define \( h : \mathbb{R}^n \to \mathbb{R} \) such that \( h(x) \triangleq \tilde{y} + \frac{\alpha}{2} \|x - \hat{x}\|^2_2 \). Note that for all \( \ell \in \mathcal{N} \), we have \( \tilde{y}_\ell = h(\tilde{x}_\ell) \), and \( \tilde{\eta}_\ell \triangleq \nabla h(\tilde{x}_\ell) = \alpha(\tilde{x}_\ell - \hat{x}) \). Since \( h \) is strongly convex with modulus \( \alpha > 0 \), for any \( (\ell_1, \ell_2) \in \mathcal{N} \times \mathcal{N} \), it follows that
\[
\tilde{y}_{\ell_2} - \tilde{y}_{\ell_1} + \langle \tilde{\xi}_{\ell_1}, \tilde{x}_{\ell_1} - \tilde{x}_{\ell_2}\rangle \geq \frac{\alpha}{2} \|\tilde{x}_{\ell_2} - \tilde{x}_{\ell_1}\|^2_2 \geq 0. \tag{2.27}
\]
Let \( \tilde{\eta} = [\tilde{y}^\top \tilde{\xi}^\top]^\top \) such that \( \tilde{y} = [\tilde{y}_i]_{i \in \mathcal{K}} \) and \( \tilde{\xi} = [\tilde{\xi}_i]_{i \in \mathcal{K}} \), where \( \tilde{y}_i = [\tilde{y}_\ell]_{\ell \in \mathcal{C}_i} \) and \( \tilde{\xi}_i = [\tilde{\xi}_\ell]_{\ell \in \mathcal{C}_i} \). Hence, \( \tilde{\eta} \in Q \) is a Slater point for the problem in (2.7), or equivalently (2.21). Since \( C\tilde{\eta} \geq \frac{\alpha v}{2} \mathbf{1} > 0 \), it follows from (2.22) that
\[
\frac{\alpha v}{2} \|\theta_{\gamma, \delta}\|_1 \leq \langle \theta_{\gamma, \delta}, C\tilde{\eta} \rangle \leq \frac{1}{2} \|\tilde{y} - \tilde{\eta}\|^2_2 + \frac{\gamma}{2} \|\tilde{\xi}\|^2_2 - g_{\gamma}(\theta_{\gamma, \delta}), \tag{2.28}
\]
and the result follows from \( \delta \)-optimality, i.e., \( p_{\gamma}^* - g_{\gamma}(\theta_{\gamma, \delta}) \leq \delta \).

Remark 2.9. When \( \gamma = 0 \), for any \( \theta_0^* \in \Theta_0^* \), it follows that
\[
\|\theta_0^*\|_1 \leq B(0, 0, \alpha) = \frac{2}{\alpha v} \left( \frac{1}{2} \sum_{\ell \in \mathcal{N}} (\tilde{y}_\ell - \tilde{\eta}_\ell)^2 - p_0^* \right) \leq \frac{1}{\alpha v} \sum_{\ell \in \mathcal{N}} (\tilde{y}_\ell - \tilde{\eta}_\ell)^2 \triangleq B_0(\alpha).
\]
Note \( p_0^* \leq p_{\gamma}^* \) for all \( \gamma \geq 0 \); hence, when \( \gamma > 0 \), for any \( \theta_{\gamma}^* \in \Theta_{\gamma}^* \), it follows that
\[
\|\theta_{\gamma}^*\|_1 \leq B(\gamma, 0, \alpha) \leq B(0, 0, \alpha) + \frac{\gamma \alpha}{v} \sum_{\ell \in \mathcal{N}} \|\tilde{x}_\ell - \hat{x}\|^2_2
\]
\[
\leq B_0(\alpha) + \frac{\gamma \alpha}{v} \sum_{\ell \in \mathcal{N}} \|\tilde{x}_\ell - \hat{x}\|^2_2 \triangleq B_0(\gamma, \alpha).
\]

The bound on \( \|\theta_{\gamma, \delta}\|_1 \) given in (2.26) holds for all \( \alpha > 0 \). Therefore, by choosing \( \alpha > 0 \) depending on \( \gamma \geq 0 \), we optimize the upper bounds \( B_0(\alpha) \) and \( B_0(\gamma, \alpha) \) defined in Remark 2.9.
Lemma 2.10. Given \( \gamma \geq 0 \), let \( \alpha^*_\gamma \triangleq \arg\min \{ B_\theta(\gamma, \alpha) : \alpha > 0 \} \), and \( \alpha^* \triangleq \alpha^*_0 \) for \( \gamma = 0 \), i.e., \( \alpha^* = \arg\min \{ B_\theta(\alpha) : \alpha > 0 \} \). For any \( \gamma \geq 0 \), \( \alpha^*_\gamma \) can be computed as follows

\[
\alpha^*_\gamma = 4 \left( \frac{\sum_{\ell \in \mathcal{N}} (\tilde{y}_\ell - \hat{y})^2}{\sum_{\ell \in \mathcal{N}} \| \hat{x}_\ell - \tilde{x}_\ell \|_2^2 \left( \| \hat{x}_\ell - \tilde{x}_\ell \|_2^2 + 8 \gamma \right)} \right)^{1/2},
\]

leading to tight upper bounds \( B^*_\theta \triangleq B_\theta(\alpha^*) \) and \( B^*_\theta(\gamma) \triangleq B_\theta(\gamma, \alpha^*_\gamma) \).

Proof: According to definition of \( \{ \tilde{y}_\ell \}_{\ell \in \mathcal{N}} \) given in Lemma 2.8 \( B_\theta(\gamma, \alpha) \) can be explicitly stated as follows:

\[
B_\theta(\gamma, \alpha) = \frac{1}{\alpha \nu} \sum_{\ell \in \mathcal{N}} \left( \tilde{y} - \bar{y}_\ell + \frac{\alpha}{2} \| \hat{x} - \tilde{x}_\ell \|_2^2 \right)^2 + \frac{\gamma \alpha}{\nu} \sum_{\ell \in \mathcal{N}} \| \hat{x} - \tilde{x}_\ell \|_2^2.
\]

To simplify the notation, let \( \eta_\ell \triangleq \tilde{y} - \bar{y}_\ell \), and \( q_\ell \triangleq \frac{\alpha}{2} \| \hat{x} - \tilde{x}_\ell \|_2^2 \) for \( \ell \in \mathcal{N} \). Via the change of variables \( \beta = \sqrt{\alpha} \), we obtain the following equivalent problem:

\[
\min_{\beta} \left\{ w(\beta) \triangleq \sum_{\ell \in \mathcal{N}} \left( \frac{1}{\beta} \eta_\ell + \beta q_\ell \right)^2 + 2 \gamma \beta^2 \sum_{\ell \in \mathcal{N}} q_\ell : \beta > 0 \right\}.
\]

Clearly, we have

\[
w'(\beta) = \sum_{\ell \in \mathcal{N}} q_\ell (q_\ell + 4 \gamma) \beta - \frac{q_\ell^2}{\beta^2}, \quad w''(\beta) = \sum_{\ell \in \mathcal{N}} q_\ell (q_\ell + 4 \gamma) + \frac{3 q_\ell^2}{\beta^4}.
\]

Since \( w''(\beta) \geq 0 \) for \( \beta > 0 \), \( w(\beta) \) is a convex function and first-order necessary optimality condition, i.e., \( w'(\beta^*) = 0 \), is also sufficient. In particular, solving for \( \beta^* \) and setting \( \alpha^*_\gamma = \sqrt{\beta^*} \) gives the desired result in (2.29). \( \square \)

Let constants \( B^*_\theta \) and \( B^*_\theta(\gamma) \) be as defined in Lemma 2.10. Now, using (2.24) and the bounds given in Remark 2.9, we can customize the generic rate results in (2.19) for gradient ascent and those in (2.20) for APG methods. In particular, for any \( \gamma > 0 \), in order to compute a \( \delta \)-optimal solution to the problem in (2.10), i.e., \( \theta_{\gamma, \delta} \geq 0 \) such that \( 0 \leq p^*_\gamma - g_\gamma(\theta_{\gamma, \delta}) \leq \delta \), the gradient ascent method requires \( O(L_\gamma/\delta) = O(B^*_\theta^2/(\gamma \delta)) \) iterations. On the other hand, P-APG in Fig. 2.2 can compute a \( \delta \)-optimal solution to (2.10) within \( O(\sqrt{L_\gamma/\delta}) \) iterations. More precisely, (2.24) implies \( O(B^*_\theta(\gamma)/(\gamma \delta)^{1/2}) \) iteration complexity for P-APG when applied to (2.10).

The \( O(1) \) constant depends on \( \sigma_{\max}(C) \), and to better have a better understanding of how it grows with the problem size, we provide some bounds for \( \sigma_{\max}(A_1) \), \( \sigma_{\max}(A_2) \),
and \( \sigma_{\text{max}}(C) \).

**Lemma 2.11.** Let \( A_1 \in \mathbb{R}^{(N-1) \times N} \) and \( A_2 \in \mathbb{R}^{(N-1) \times N} \) be the matrices in (2.4), i.e., corresponding to the constraints in (2.3); and let \( A_3, A_4, \) and \( C \) be the matrices as given in Definition 2.7. Then, \( \sigma_{\text{max}}(A_3) \leq \sigma_{\text{max}}(A_1) = \sqrt{2N} \), \( \sigma_{\text{max}}(A_4) \leq \sigma_{\text{max}}(A_2) \leq B_x N \), and \( \sigma_{\text{max}}(C) \leq \sqrt{2N} + B_x N \).

**Proof:** It is easy to observe that \( A_1^T A_1 = 2\Omega \), where \( \Omega \in \mathbb{R}^{N \times N} \) denotes the Laplacian matrix of a complete graph with \( N \) vertices, i.e., for each \( i = 1, \ldots, N \), \( \Omega_{ii} = N - 1 \), and \( \Omega_{ij} = -1 \) for all \( j \neq i \). It is known that \( \Omega \) has two distinct eigenvalues: 0 (with multiplicity 1) and \( N \) (with multiplicity \( N - 1 \)). Therefore, \( \sigma_{\text{max}}(A_1) = \sqrt{2N} \); and since \( A_3 \) is a submatrix of \( A_1 \), one immediately has \( \sigma_{\text{max}}(A_3) \leq \sigma_{\text{max}}(A_1) \).

Since \( A_2 \) is block-diagonal, we have \( \sigma_{\text{max}}(A_2) = \max_{\ell \in N} \{ \sigma_{\text{max}}(X_\ell) \} \), where \( X_\ell = -T_\ell \bar{X} \) (see Definition 2.1 for \( T_\ell \) and \( \bar{X} \)). Hence, we have \( \sigma_{\text{max}}(A_2) \leq \| \bar{X} \| \max \{ \| T_\ell \| : \ell \in N \} \). For \( \ell \in N \), let \( \Omega_\ell = T_\ell^T T_\ell \); it is easy to observe that \( \Omega_\ell \in \mathbb{R}^{N \times N} \) is the Laplacian matrix of a star-tree with \( N - 1 \) leaves (\( \ell \) is the internal node). It is known that \( \Omega \) has three distinct eigenvalues: 0 (with multiplicity 1), 1 (with multiplicity \( N - 2 \)), and \( N \) (with multiplicity 1). Therefore, \( \| T_\ell \| = \sqrt{N} \) for all \( \ell \in N \). On the other hand, since \( \| \bar{x}_\ell \|_2 \leq B_x \) for \( \ell \in N \), \( \| \bar{X} \| \leq B_x \sqrt{N} \). Therefore, \( \sigma_{\text{max}}(A_2) \leq B_x N \); and since \( A_4 \) is a submatrix of \( A_2 \), one immediately has \( \sigma_{\text{max}}(A_4) \leq \sigma_{\text{max}}(A_2) \). Finally, since \( C = [A_3 A_4] \), clearly \( \| C \| \leq \| A_3 \| + \| A_4 \| \).

Next, we study the error bounds for inexact solutions. Given \( \gamma > 0 \), let \( \theta_{\gamma, \delta} \) be a \( \delta \)-optimal solution to (2.10), and \( (y_{\gamma, \delta}, \xi_{\gamma, \delta}) \) be the optimal solution to the minimization problem in (2.9), or equivalently to (2.22), when \( \theta \) is set to \( \theta_{\gamma, \delta} \). In Theorem 2.12 we establish error bounds on the suboptimality \( \| y_{\gamma, \delta} - y^* \|_2 \), and on the infeasibility \( \| (A_1 y_{\gamma, \delta} + A_2 \xi_{\gamma, \delta}) - \|_2 \).

**Theorem 2.12.** Given \( \gamma > 0 \), let \( (y^*_\gamma, \xi^*_\gamma) \) and \( \theta^*_{\gamma, \delta} \) denote the optimal solutions to (2.7) and (2.10), respectively. Let \( \theta_{\gamma, \delta} \) be a \( \delta \)-optimal solution to (2.10), and \( (y_{\gamma, \delta}, \xi_{\gamma, \delta}) \) be the minimizer in (2.22) when \( \theta \) is set to \( \theta_{\gamma, \delta} \). For all \( \gamma, \delta > 0 \), the following bounds hold:

\[
\| y_{\gamma, \delta} - y^* \|_2 \leq \| x^* \|_2 \sqrt{\gamma} + \sqrt{2\delta},
\]

(2.33)

\[
\| (A_1 y_{\gamma, \delta} + A_2 \xi_{\gamma, \delta}) - \|_2 \leq 2\sqrt{N\delta} + B_x N \sqrt{2\delta \over \gamma}.
\]

(2.34)

Moreover, both starting from the initial iterate \( \theta^0 = 0 \), \( P\text{-APG} \) can compute \( (y_{\gamma, \delta}, \xi_{\gamma, \delta}) \)
within $K(\delta, \gamma) = \sigma_{\max}(C)B_{\delta}^*(\gamma)\sqrt{2/(\gamma\delta)}$ iterations while gradient ascent requires $2(B_{\delta}^*\sigma_{\max}(C))^2/(\gamma\delta)$ iterations, where $\sigma_{\max}(C) = O(N)$.

Proof: Given $\theta_{\gamma,\delta} \geq 0$ and the corresponding minimizer, $(y_{\gamma,\delta}, \xi_{\gamma,\delta})$, to the problem in (2.22) when $\theta$ is set to $\theta_{\gamma,\delta}$, Theorem 4.11 implies that

$$\|y_{\gamma,\delta} - y^*_\gamma\|^2 + \gamma\|\xi_{\gamma,\delta} - \xi^*_{\gamma}\|^2 \leq 2\left(p^*_{\gamma} - g_{\gamma}(\theta_{\gamma,\delta})\right) \leq 2\delta. \tag{2.35}$$

Hence, Lemma 2.4 and (2.35) together imply that

$$\|y_{\gamma,\delta} - y^*_\gamma\|_2 \leq \|y_{\gamma,\delta} - y^*_\gamma\|_2 + \|y^*_\gamma - y^*\|_2 \leq \|\xi^*\|_2\sqrt{\gamma} + \sqrt{2\delta}.$$ 

Moreover, since $(y_{\gamma,\delta}, \xi_{\gamma,\delta}) \in Q$ and $(y^*_\gamma, \xi^*_\gamma)$ is feasible to (2.7), i.e., $A_1y^*_\gamma + A_2\xi^*_\gamma \geq 0$, we have

$$\|(A_1y_{\gamma,\delta} + A_2\xi_{\gamma,\delta})_\gamma\|_2 \leq \|(A_3y_{\gamma,\delta} + A_4\xi_{\gamma,\delta})_\gamma\|_2 + \|(A_3y^*_\gamma + A_4\xi^*_\gamma)_\gamma\|_2 \leq \sigma_{\max}(A_3)\|y_{\gamma,\delta} - y^*_\gamma\|_2 + \sigma_{\max}(A_4)\|\xi_{\gamma,\delta} - \xi^*_\gamma\|_2 \tag{2.36}$$

where the first inequality follows from the fact that $\|x - y\|_2 \geq \|(x)_\gamma - (y)_\gamma\|_2$ for any $x$ and $y$. The infeasibility result in (2.34) immediately follows from (2.35) and (2.36). The iteration complexity bounds can be obtained using the arguments immediately after Remark 2.9.

As for some applications having an error bound on how $\xi_{\gamma,\delta}$ approximates $\xi^*$, i.e., the subgradients at $\{\bar{x}_\ell\}_{\ell \in \mathcal{N}}$, is crucial. Next, we show that $\|\xi_{\gamma,\delta} - \xi^*\|_2$ is indeed small.

**Theorem 2.13.** There exists $K > 0$ such that $\|\xi^*_\gamma - \xi^*\|_2 \leq K\|A_1(y^*_\gamma - y^*)\|_2$; hence, $\|\xi^*_\gamma - \xi^*\|_2 \leq K\sigma_{\max}(A_1)\|\xi^*\|_2\sqrt{\gamma}$, which implies

$$\|\xi_{\gamma,\delta} - \xi^*\|_2 \leq K\|\xi^*\|_2\sqrt{2N\gamma} + \sqrt{2\delta \over \gamma}.$$ 

Proof: Since $y^*$ is the unique optimal solution to (2.4), (2.5) implies that $\xi^* = \text{argmin}\{\|\xi\|_2 : A_1y^* + A_2\xi \geq 0\}$. Similarly, (2.7) implies that $\xi^*_\gamma = \text{argmin}\{\|\xi\|_2 : A_1y^*_\gamma + A_2\xi \geq 0\}$. Define $h(\gamma) \triangleq -A_1y^*_\gamma$ for $\gamma \geq 0$. Note from Lemma 2.4 we have $h(0) = -A_1y^*_0 = -A_1y^*$.
Therefore, for $\gamma \geq 0$,
\[
\xi_\gamma^* = \arg\min \left\{ \|\xi\|_2 : A_2 \xi \geq h(\gamma) \right\}. \tag{2.37}
\]

Note that for $\gamma = 0$, $\xi_0^* = \xi^*$. Sensitivity of metric projection onto parametric polyhedral sets is studied in [32]. According to Theorem 2.1 in [32], there exists $K > 0$ such that
\[
\|\xi_\gamma^* - \xi_{\gamma'}^*\|_2 \leq K \|h(\gamma) - h(\gamma')\|_2 \leq K\sigma_{\text{max}}(A_1)\|y_\gamma^* - y_{\gamma'}^*\|_2, \quad \forall \gamma, \gamma' \geq 0. \tag{2.38}
\]

Therefore, given $\gamma > 0$, setting $\gamma' = 0$, and using Lemma 2.4, we have
\[
\|\xi_\gamma^* - \xi^*\|_2 \leq K\sigma_{\text{max}}(A_1)\|y_\gamma^* - y^*\|_2 \leq K\sigma_{\text{max}}(A_1)\|\xi^*\|\sqrt{\gamma}. \tag{2.39}
\]

Moreover, (2.35) implies that $\|\xi_{\gamma,\delta}^* - \xi_{\gamma}^*\|_2 \leq \sqrt{\frac{2\sigma}{\gamma}}$. Hence, combining this with (2.39) gives the desired result since $\sigma_{\text{max}}(A_1) = \sqrt{2N}$.

We can summarize Theorem 2.12 and Theorem 2.13 briefly as follows. If the main objective is the function value approximation and estimating the subgradients are not crucial, then according to Theorem 2.12, for any given $\epsilon > 0$, setting $\gamma = \delta = \epsilon$ implies that $y_{\gamma,\delta} \in \mathbb{R}^N$ satisfies $\|y_{\gamma,\delta} - y^*\|_2^2 = O(\epsilon)$ and it can be computed within $O(N^2B^2_\delta/\epsilon^2)$ iterations of the gradient ascent method on (2.10) (which is the same as the iteration complexity of the projected subgradient method applied to (2.10) for $\gamma = 0$), and within $O(NB^2_\delta(\epsilon)/\epsilon)$ iterations of P-APG in Fig. 2.2 on (2.10). On the other hand if the subgradient approximation is important too, then according to Lemma 2.4, Theorem 2.12 and Theorem 2.13, for any given $\epsilon > 0$, by setting $\gamma = \epsilon$ and $\delta = \epsilon^2$ implies that $y_{\gamma,\delta} \in \mathbb{R}^N$ satisfies $\|y_{\gamma,\delta} - y^*\|_2^2 = O(\epsilon)$, $\|\xi_{\gamma,\delta} - \xi^*\|_2^2 = O(\epsilon)$ and $\|(A_1y_{\gamma,\delta} + A_2\xi_{\gamma,\delta}) - y^*\|_2^2 \leq O(\epsilon)$ within $O(N^2B^2_\delta/\epsilon^3)$ iterations using the gradient ascent method on (2.10), and within $O(NB^2_\delta(\epsilon)/\epsilon^{3/2})$ iterations using P-APG in Fig. 2.2 on (2.10).

### 2.2.5 Continuation Method for Convex Regression

Let $\theta_{\gamma,\delta}$ be a $\delta$-optimal solution to (2.10), and $(y_{\gamma,\delta}, \xi_{\gamma,\delta})$ be the minimizer in (2.22) when $\theta$ is set to $\theta_{\gamma,\delta}$. In Section 2.2.4, we have seen that for any fixed $\epsilon$, setting $\gamma = \delta = \epsilon$ implies that $y_{\gamma,\delta}$ can be computed within $O(NB^2_\delta(\epsilon)/\epsilon)$ iterations of P-APG and it satisfies $\|y_{\gamma,\delta} - y^*\|_2^2 = O(\epsilon)$. In this section, we describe a continuation method to solve (2.4). In particular, we would like to generate an iterate sequence $\{y^t\}_{t \in \mathbb{Z}_+}$ such that $y^{(t)} \to y^*$ as $t \to +\infty$ with the following properties:

i) for any $\epsilon > 0$, $y^{(t)}$ satisfies $\|y^{(t)} - y^*\|_2 = O(\epsilon)$ for all $t \geq T_\epsilon = O(\log(1/\epsilon))$;
ii) moreover, \( T_t \) iterations of the continuation require at most \( \mathcal{O}(1/\epsilon) \) P-APG iterations in total, i.e., the algorithm generates an asymptotically optimal iterate sequence with \( \mathcal{O}(1/\epsilon) \) rate without fixing the algorithmic parameters depending on the tolerance \( \epsilon > 0 \).

Let \( \beta > 1 \) and define \( \{\epsilon_t\}_{t \in \mathbb{Z}_+} \) such that \( \epsilon_t = \epsilon_0/\beta^t \) for some \( \epsilon_0 > 0 \). Also define \( \{\gamma_t\}_{t \in \mathbb{Z}_+} \) and \( \{\delta_t\}_{t \in \mathbb{Z}_+} \) such that \( \gamma_t = \kappa_0 \epsilon_t \) and \( \delta_t = \kappa_\delta \epsilon_t \) for \( t \geq 1 \) for some \( \kappa_0, \kappa_\delta > 0 \).

Next, for all \( t \geq 1 \), let \( \theta^{(t)} \triangleq \theta_{\gamma_t, \delta_t} \) be a \( \delta_t \)-optimal solution to (2.10) when \( \gamma = \gamma_t \), such that it is computed using P-APG in Fig. 2.2 starting from the initial iterate \( \theta^{(1)} \), where \( \theta^{(0)} = 0 \), and \( (y^{(t)}, \xi^{(t)}) \) be the minimizer in (2.22) when \( \theta \) is set to \( \theta^{(t)} \) and \( \gamma = \gamma_t \). Then clearly from (2.33), we have

\[
\|y^{(t)} - y^*\|_2 \leq \|\xi^*\|_2 \sqrt{\eta_t} + \sqrt{2\delta_t} = \Gamma \beta^{-\frac{t}{2}}, \quad t \geq 1,
\]

(2.40)

where \( \Gamma \triangleq \sqrt{\eta_0} \left( \|\xi^*\|_2 \sqrt{\kappa_0 + \sqrt{2\kappa_\delta}} \right) \). Therefore, \( \|y^{(t)} - y^*\|_2 \leq \epsilon \) for all \( t \geq T_\epsilon \triangleq \lceil \log_\beta (\Gamma^2/\epsilon) \rceil \). Let \( y_t \triangleq y^{(T_\epsilon)} \); hence, \( \|y_t - y^*\|_2 \leq \epsilon \).

Note that for all \( t \geq 1 \), starting from \( \theta^{(t-1)} \), P-APG can compute \( \theta^{(t)} \) within \( K_t \triangleq \sigma_{\max}(C) ||\theta^*_{\gamma_t} - \theta^{(t-1)}||_2 \sqrt{2/(\delta_t \gamma_t)} \) iterations. From Lemma 2.8 and Remark 2.9, it follows that \( \|\theta^{(t-1)}\|_1 \leq B_{0}(\gamma_{t-1}, \delta_{t-1}, \alpha) \leq B_{0}(\gamma_{t-1}, \alpha) + \frac{2\delta_{t-1}}{\alpha_{t-1}} \), and \( ||\theta^*_{\gamma_t}\|_1 \leq B_{0}(\gamma_t, \alpha) \) for all \( \alpha > 0 \).

Therefore, Lemma 2.10 and \( \gamma_{t-1} > \gamma_t \) imply

\[
||\theta^*_{\gamma_t} - \theta^{(t-1)}||_2 \leq ||\theta^{(t-1)}||_1 + ||\theta^*_{\gamma_t}||_1 \leq 2B_{0}(\gamma_{t-1}) + \frac{2\delta_{t-1}}{\alpha_{t-1}} \cdot \frac{1}{\epsilon_t}. \quad (2.41)
\]

Hence, \( K_t \), the number of P-APG iterations to compute \( \theta^{(t)} \) can be bounded above as follows

\[
K_t \leq \tilde{K}_t \triangleq ||C|| \left( 2B_{0}(\gamma_{t-1}) + \frac{2\delta_{t-1}}{\alpha_{t-1}} \right) \sqrt{\frac{2}{\kappa_{\gamma} \kappa_{\delta}}} \frac{1}{\epsilon_t}. \quad (2.42)
\]

From Lemma 2.10, we have \( \alpha_{\gamma_0}^* \leq \alpha_{\gamma_{t-1}}^* \leq \alpha_{\gamma_t}^* \leq \alpha^* \) for \( t \geq 1 \); hence, for \( \gamma > 0 \),

\[
B_{0}(\gamma) = B_{0}(\alpha_{\gamma_0}^*) \leq \frac{\gamma \alpha_{\gamma_t}^*}{v} \sum_{\ell \in \mathcal{N}} \|\bar{x}_{\ell} - \hat{x}\|_2^2 \leq B_{0}(\alpha_{\gamma_0}^*) + \frac{\gamma \alpha_{\gamma_0}^*}{v} \sum_{\ell \in \mathcal{N}} \|\bar{x}_{\ell} - \hat{x}\|_2^2.
\]

Using this upper bound in (2.42), we can bound the total number of P-APG iterations needed to compute \( y_\epsilon \). In particular, \( y_\epsilon \) can be computed within

\[
\sum_{t=1}^{T_\epsilon} K_t \leq ||C|| \sqrt{\frac{2}{\kappa_{\gamma} \kappa_{\delta}}} \left[ \frac{2\beta}{v} \left( \kappa_{\gamma} \alpha_{\gamma_0}^* \sum_{\ell \in \mathcal{N}} \|\bar{x}_{\ell} - \hat{x}\|_2^2 + \kappa_{\delta} \right) \frac{\kappa_{\delta}}{\alpha_{\gamma_0}^*} \right] T_\epsilon + \frac{2B_{0}(\alpha_{\gamma_0}^*)}{\epsilon_0} \sum_{t=1}^{T_\epsilon} \beta^t \quad (2.43)
\]

24
P-APG iterations. Note that \( \sum_{t=1}^{T_t} \beta^k = \frac{1}{\beta-1} (\beta^{T_t} - 1) \leq \frac{\beta}{\beta-1} \frac{\Gamma^2}{\epsilon} \) since \( T_t = \lceil \log_\beta(\Gamma^2/\epsilon) \rceil \). Therefore, \( \sum_{t=1}^{T_t} \bar{K}_t = \mathcal{O}(1/\epsilon) \).

To implement this scheme, for each outer iteration \( t \geq 1 \), verifiable sufficient conditions for \( \delta_t \)-optimality can be used to terminate inner P-APG iterations. In fact, the number of P-APG iterations to compute \( \theta^{(t)} \) is bounded above by \( \bar{K}_t \), which can be computed a-priori; hence, giving us a stopping condition for the inner iterations. Moreover, one can also use other stopping conditions for inner iterations based on \( \nabla g_{\gamma t} \) which are also sufficient for \( \delta_t \) optimality; thus, making it possible to proceed to the next outer iteration before waiting for \( \bar{K}_t \) inner iterations – see Section 3.3 in [33] for a similar discussion.

### 2.2.6 Computational complexity of P-APG iterations

In Section 2.2.4, we have seen that for any fixed \( \epsilon \), setting \( \gamma = \delta = \epsilon \) implies that \( \| y_{\gamma,\delta} - y^* \|_2^2 = \mathcal{O}(\epsilon) \) and \( y_{\gamma,\delta} \) can be computed within \( \mathcal{O}(1/\epsilon) \) iterations of P-APG – see Theorem 2.12. In Section 2.2.5, we discussed that using continuation one can generate an iterate sequence \( \{ y^t \}_{t \in \mathbb{Z}_+} \) such that \( y^{(t)} \to y^* \) as \( t \to +\infty \), and \( \| y^{(t)} - y^* \|_2^2 = \mathcal{O}(\epsilon) \) for all \( t \geq T_\epsilon = \mathcal{O}(1/\epsilon) \) for all \( \epsilon > 0 \); moreover, computing \( y^{(T_\epsilon)} \) require at most \( \mathcal{O}(1/\epsilon) \) iterations of P-APG in total – see (2.43).

The bottleneck operations at each P-APG iteration, displayed in Fig. 2.2, are i) evaluating the matrix-vector multiplications with \( C \) and \( C^T \), and ii) computing Step 1, which requires solving \( K \) small-size QPs. Matrix-vector multiplications with \( C \) and \( C^T \) requires evaluating multiplications with \( A_3 \), \( A_3^T \), \( A_4 \) and \( A_4^T \). Moreover, since \( A_3 \) is a submatrix of \( A_1 \in \mathbb{R}^{N(N-1) \times N} \), and \( A_4 \) is a submatrix of \( A_2 \in \mathbb{R}^{N(N-1) \times N_n} \), as long as left and right vector multiplications with \( A_1 \) and \( A_2 \) can be done efficiently, one can do same operations with \( C \) easily. Due to specific structures of \( A_1 \) and \( A_2 \), without forming \( A_1 \) and \( A_2 \) explicitly, one can compute \( A_1 y \) and \( A_1^T z \) with \( \mathcal{O}(N^2 - N) \) complexity for all \( y \) and \( z \); \( A_2 \xi \) and \( A_2^T \omega \) with \( \mathcal{O}(n(N^2 - N)) \) complexity for all \( \xi \) and \( \omega \). More importantly, neither \( A_1 \) nor \( A_2 \) is stored in the memory; storing only \( \{ \bar{x}_t \}_{t=1}^{N} \) is sufficient to be able to compute these matrix-vector multiplications.

First, we will consider the bottleneck step while solving (2.7) using a primal-dual IPM alone, without P-APG. This result will also help us understand the complexity of computing Step 1, which requires solving \( K \) small size QPs as shown in (2.13), which are in a similar form with the QP in (2.7).

Let \( c \in \mathbb{R}^{N(n+1)} \) be an arbitrary vector, \( G = \begin{bmatrix} I_N & \mathbf{0} \\ \mathbf{0}^\top & \gamma I_n \end{bmatrix} \), and \( A = [A_1 \ A_2] \) where
\( A_1 \in \mathbb{R}^{N(N-1) \times N} \) and \( A_2 \in \mathbb{R}^{N(N-1) \times N_n} \) are defined in (2.6). Consider the generic QP

\[
\min_{\eta} \frac{1}{2} \eta^T G \eta + c^T \eta \quad \text{s.t.} \quad A \eta \geq 0 : \theta,
\]

(2.44)

where \( \theta \in \mathbb{R}^{N(N-1)} \) is the vector of dual variables. Note that for appropriately chosen \( c \in \mathbb{R}^{N(n+1)} \), (2.7) is a special case of (2.44). Let \( s \in \mathbb{R}^{N(N-1)} \) represent the slack variables such that \( s = [s_{\ell}]_{\ell \in \mathcal{N}}, \) where \( s_{\ell} = [s_{\ell \ell'}]_{\ell' \in \mathcal{N} \setminus \{\ell\}} \in \mathbb{R}^{N-1}. \) Given some \( \tau > 0, \) the perturbed KKT system is given as

\[
G \eta - A^T \theta + c = 0, \quad A \eta - s = 0, \quad s_{\ell \ell'} \theta_{\ell \ell'} = \tau, \quad (\ell, \ell') \in \mathcal{P};
\]

(2.45)

\( s \geq 0, \quad \theta \geq 0. \)

Instead of directly solving the KKT system (for \( \tau = 0 \)), the primal-dual path following IPM methods inexactly solve the perturbed KKT conditions as \( \tau \searrow 0. \) Given \( \tau > 0 \) and some point \((\eta, s, \theta)\) such that \( s > 0 \) and \( \theta > 0, \) the major operation is to compute the Newton direction for the nonlinear equation system in (2.45) from the given point. The Newton direction can be computed by solving the following system

\[
\begin{bmatrix}
G & -A^T \\
A & \Theta^{-1} \mathcal{S}
\end{bmatrix}
\begin{bmatrix}
\Delta \eta \\
\Delta \theta
\end{bmatrix} =
\begin{bmatrix}
-r_d \\
-r_p - s + \tau \Theta^{-1} \mathbf{1}
\end{bmatrix},
\]

(2.46)

and setting \( \Delta s = A \Delta \eta + r_p, \) where \( \mathcal{S} = \text{diag}(s), \Theta = \text{diag}(\theta), \) \( r_p = A \eta - s, \) \( r_d = G \eta - A^T \theta + c. \) (2.46) implies that \( \Delta \eta \) can be computed by solving

\[
\left(G + A^T S^{-1} \Theta A\right) \Delta \eta = -r_d + A^T S^{-1} \Theta \left(-r_p - s + \tau \Theta^{-1} \mathbf{1}\right).
\]

(2.47)

It is easy to see that \( M \triangleq G + A^T S^{-1} \Theta A \) is indeed a block arrowhead matrix. Indeed, let \( d = S^{-1} \Theta \mathbf{1} \in \mathbb{R}^{N(N-1)}, \) i.e., \( d_{\ell \ell'} = \theta_{\ell \ell'}/s_{\ell \ell'} \) for \((\ell, \ell') \in \mathcal{P}, \) and define \( d_{\ell} = [d_{\ell \ell'}]_{\ell' \in \mathcal{N} \setminus \{\ell\}} \in \mathbb{R}^{N-1} \) for each \( \ell \in \mathcal{N}. \) Since \( A = [A_1 A_2], \) from the definition of \( A_1 \) and \( A_2 \) in (2.6), it follows that \( M \) can be written as

\[
M = \\
\begin{pmatrix}
M_{00} & M_{01} & M_{02} & \cdots & M_{0N} \\
M_{01} & M_{11} & 0 & \cdots & 0 \\
M_{02} & 0^T & M_{22} & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
M_{0N} & 0^T & 0^T & \cdots & M_{NN}
\end{pmatrix} \in \mathbb{R}^{N(n+1) \times N(n+1)}, \quad \text{where}
\]

(2.48)

26
\[ M_{00} = I_N + \sum_{\ell \in \mathcal{N}} T_{\ell}^T \text{diag}(d_{\ell}) T_{\ell}, \quad M_{\ell\ell} = \gamma I_n + X_{\ell}^T \text{diag}(d_{\ell}) X_{\ell}, \quad M_{0\ell} = T_{\ell}^T \text{diag}(d_{\ell}) X_{\ell}, \]

for \( \ell \in \mathcal{N} \). Define \( R_{\ell} \triangleq T_{\ell}^T \text{diag}(d_{\ell}) T_{\ell} \) for \( \ell \in \mathcal{N} \). Since \( X_{\ell} = -T_{\ell}X \), we have

\[ M_{00} = \sum_{\ell \in \mathcal{N}} R_{\ell}, \quad M_{\ell\ell} = \gamma I_n + \bar{X}_{\ell}^T R_{\ell} \bar{X}, \quad M_{0\ell} = -R_{\ell} \bar{X}, \quad \ell \in \mathcal{N}. \quad (2.49) \]

Moreover, due to structure of \( T_{\ell} \) (see Definition \[2.1\]), \( R_{\ell} \) is a symmetric sparse matrix with a very special structure. In particular, it has only \( 3N - 2 \) nonzero elements, and \( R_{\ell}X \) can be computed in \( \mathcal{O}(Nn) \) flops. Hence, forming \( M_{\ell\ell} \) and \( M_{0\ell} \) require \( \mathcal{O}(Nn(n + 1)) \) and \( \mathcal{O}(Nn) \) flops, respectively. It is easy to show that forming \( M_{00} \) can be done in \( \mathcal{O}(N^2) \) flops; therefore, constructing \( M \) requires \( \mathcal{O}(N^2n(n + 2)) \) flops in total.

In the next lemma, we show that given an arbitrary \( b \in \mathbb{R}^{N(n + 1)} \), the solution to the system \( M \Delta \eta = b \) for \( M \) given in \((2.48)\) can be directly computed as in \((2.50)\). Alternatively, one can also compute the Cholesky factorization of \( M \) first, and then use forward-backward substitution to compute the solution, which requires roughly the same amount of work that computing \((2.50)\) requires. In the proof of Theorem \[2.14\] we also show as a side result that the Cholesky factorization of a generic block arrowhead matrix as in \((2.48)\) can be computed very efficiently, compared to factorization of a dense matrix.

**Theorem 2.14.** Let \( M \in \mathbb{R}^{N(n+1) \times N(n+1)} \) be a symmetric positive definite matrix with the generic block arrowhead structure given as in \((2.48)\), where \( M_{00} \in \mathbb{R}^{N \times N} \), \( M_{0\ell} \in \mathbb{R}^{N \times n} \) and \( M_{\ell\ell} \in \mathbb{R}^{n \times n} \) for \( \ell \in \mathcal{N} \). Given arbitrary \( b \in \mathbb{R}^{N(n+1)} \) such that \( b^T = [b_0^T b_1^T \cdots b_N^T]^T \), the system \( M \Delta \eta = b \) can be efficiently solved requiring \( \mathcal{O}(N^3 + N^2n^2 + 2Nn^3) \) flops, where \( b_0 \in \mathbb{R}^N \), \( b_\ell \in \mathbb{R}^n \) for \( \ell \in \mathcal{N} \), and \( \Delta \eta^T = [\Delta y^T \Delta \xi_1^T \cdots \Delta \xi_N^T] \). The solution is given as

\[ \Delta y = M_{00}^{-1} \left( b_0 - \sum_{\ell \in \mathcal{N}} M_{0\ell} M_{\ell\ell}^{-1} b_\ell \right), \quad \Delta \xi_\ell = M_{\ell\ell}^{-1} \left( b_\ell - M_{0\ell}^T \Delta y \right), \quad \ell \in \mathcal{N}. \quad (2.50) \]

**Proof:** In order to compute the Cholesky decomposition, we appropriately permute \( M \) and consider the following equation system:

\[
\begin{pmatrix}
M_{11} & 0 & \cdots & 0 & M_{10}^T \\
0^T & M_{22} & \cdots & 0 & M_{20}^T \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0^T & 0^T & \cdots & M_{NN} & M_{0N}^T \\
M_{01} & M_{02} & \cdots & M_{0N} & M_{00}
\end{pmatrix}
\begin{pmatrix}
\Delta \xi_1 \\
\vdots \\
\Delta \xi_N \\
\Delta y
\end{pmatrix}
= \begin{pmatrix}
b_1 \\
\vdots \\
b_N \\
b_0
\end{pmatrix}
\]
Let $M_{\text{per}}$ be the matrix on the left hand side of \((2.51)\). Compared to $M$, Cholesky decomposition of $M_{\text{per}}$ can be computed much more efficiently. Indeed, diagonal blocks are factorized first: $M_{00} = F_0 F_0^\top$, and $M_{\ell \ell} = F_\ell F_\ell^\top$ for $\ell \in \mathcal{N}$. Since $M$ is positive definite, all the blocks on the diagonal are also positive definite; hence, $F_0$ and $F_\ell$ for $\ell \in \mathcal{N}$ are invertible. The Cholesky factorization $M_{\text{per}} = L_{\text{per}} L_{\text{per}}^\top$ can be easily verified:

$$L_{\text{per}} = \begin{pmatrix} F_1 & 0 & \cdots & 0 & 0 \\ 0^\top & F_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0^\top & 0^\top & \cdots & F_N & 0 \\ L_1 & L_2 & \cdots & L_N & F_0 \end{pmatrix},$$

(2.52)

where $L_\ell = M_{0\ell} \left(F_{\ell}^{-1}\right)^\top$ for $\ell \in \mathcal{N}$. Note that Cholesky factorization of $M_{\ell \ell}$ can be computed with $\mathcal{O}(n^3)$ complexity for each $\ell \in \mathcal{N}$, and with $\mathcal{O}(N^3)$ for $\ell = 0$. Hence, the total complexity for computing $\{F_{\ell}\}_{\ell \in \mathcal{N} \cup \{0\}}$ is $\mathcal{O}(N^3 + N n^3)$. Moreover, for each $\ell \in \mathcal{N}$, computing $L_\ell$ requires $\mathcal{O}(n^3)$ flops for inverting the lower diagonal matrix $F_\ell$, and $\mathcal{O}(N n^2)$ for the multiplication; thus, the total complexity of computing $L_{\text{per}}$ is $\mathcal{O}(N^3 + N^2 n^2 + 2 N n^3)$. Moreover, storing $L_{\text{per}}$ requires roughly $N(n^2 + N)/2 + N^2 n$ memory locations. Finally, computing $\Delta \eta$ requires one forward and one backward substitution which will roughly add another $\mathcal{O}(N n^2 + N^2 n)$ flops to the complexity.

Instead computing Cholesky factorization $M_{\text{per}} = L_{\text{per}} L_{\text{per}}^\top$ explicitly, we will derive a closed form update rule for $\Delta \eta$. This will save us from storing $L_{\text{per}}$ and from doing additional forward-backward substitutions. First, we solve $L_{\text{per}} \Delta r = b$ via forward substitution, where $\Delta r^\top = [\Delta r_1^\top \cdots \Delta r_N^\top \Delta r_0^\top]^\top$. From (2.52), it clearly follows that

$$\Delta r_0 = F_0^{-1}(b_0 - \sum_{\ell \in \mathcal{N}} L_\ell \Delta r_\ell), \quad \Delta r_\ell = F_\ell^{-1} b_\ell, \quad \ell \in \mathcal{N}. \quad (2.53)$$

Next, we solve $L_{\text{per}}^\top \Delta \eta = \Delta r$ for $\Delta \eta$ via backward substitution:

$$\Delta y = \left(F_0^\top\right)^{-1} \Delta r_0, \quad \Delta \xi_\ell = \left(F_\ell^\top\right)^{-1} \left(\Delta r_\ell - L_\ell^\top \Delta y\right), \quad \ell \in \mathcal{N}. \quad (2.54)$$

Note that for each $\ell \in \mathcal{N}$, from the definitions of $F_\ell$ and $L_\ell$, it follows that

$$L_\ell F_\ell^{-1} = M_{0\ell} \left(F_{\ell}^{-1}\right)^\top F_\ell^{-1} = M_{0\ell} M_{\ell \ell}^{-1}. \quad (2.55)$$
Therefore, using (2.53), (2.54), and (2.55), we can solve for $\Delta \eta$ in closed form as shown in (2.50).

As we discussed before, the bottleneck step while solving (2.7) using a primal-dual path following IPM is to solve either the augmented system in (2.46) or the normal equations in (2.47). This reduces to computing the Cholesky decomposition of $M$ in (2.48) with components defined in (2.49) and using forward-backward substitution to compute $\Delta \eta$. Alternatively, according to Theorem 2.14, one can also directly compute the solution as in (2.50). Both alternatives have roughly the same complexity requiring $O(N^3 + N^2 n^2 + 2Nn^3)$ flops. Clearly, when $N$ is large, i.e., $N \geq 10^5$, this bottleneck step becomes impractical. On the other hand, combining P-APG and IPM, leaves the form of the bottleneck step unchanged, while making it more manageable by dividing it into smaller subsystem solves. In particular, the total complexity of computing Step 1 in P-APG consists of the complexity of solving $K$ small size QPs as shown in (2.13).

Consider the problem in Step 1 of P-APG, and let $c = -C^T \tilde{\theta}^i \in \mathbb{R}^{N(n+1)}$. For each $i \in K$, define $c_i \in \mathbb{R}^{\tilde{N}(n+1)}$ such that $c_i$ is the subvector of $c$ corresponding to the indices of $\eta_i = [y_i^T \xi_i^T]^T$, i.e., $\langle c, \eta \rangle = \sum_{i \in K} \langle c_i, \eta_i \rangle$ for any $\eta$. Moreover, let $\tilde{G} = \begin{pmatrix} I_{\tilde{N}} & 0 \\ 0 & I_{\tilde{N}n} \end{pmatrix}$, and $\tilde{A} = [\tilde{A}_1^i \tilde{A}_2^i]$ where $\tilde{A}_1^i \in \mathbb{R}^{\tilde{N} (\tilde{N} - 1) \times \tilde{N}}$ and $\tilde{A}_2^i \in \mathbb{R}^{\tilde{N} (\tilde{N} - 1) \times \tilde{N}n}$ are defined in Definition 2.3.

Hence, the problem in Step 1 of Fig. 2.1 can be equivalently written as

$$\min_{\eta_i} \frac{1}{2} \eta_i^T \tilde{G} \eta_i + c_i^T \eta_i \quad \text{s.t.} \quad \tilde{A}^i \eta_i \geq 0 : \theta_{ii}, \quad i \in K,$$

(2.56)

where $\theta_{ii} \in \mathbb{R}^{\tilde{N} (\tilde{N} - 1)}$ is the vector of dual variables. For each $i \in K$, (2.56) is in a similar form with the QP in (2.7). Therefore, we immediately have the following result as a corollary of Theorem 2.14.

**Corollary 2.15.** For each $i \in K$, the normal equations corresponding to the QP in (2.56) are in the same form with (2.47) leading to a system with a block-arrowhead matrix as in (2.48) with much smaller dimensions. Thus, Newton direction computations require $O(N^3 + \tilde{N}^2 n^2 + 2\tilde{N}n^3)$ flops for each $i \in K$.

Suppose that we have $K$ parallel processors. It is worth noting that thanks to the separability of the problem in Step 1 of P-APG, i.e., (2.56), one can do this computation in parallel, running a primal-dual path following IPM on each one of the $K$ processors, or sequentially running the primal-dual path following IPM on a single processor $K$ times.
Remark 2.16. The total number of IPM iterations until P-APG terminates can be analyzed using the iteration complexity results on inexact accelerated proximal gradient algorithms [34], where Schmidt et al. analyzed APG in Fig. 2.1 when $\nabla \rho$ in Step 1 is computed inexactly. In particular, one does not need to solve QP-subproblems exactly in each P-APG iteration. Given a tolerance sequence $\{\tau_k\} \subset \mathbb{R}^+$ such that $\tau_k \searrow 0$, the number of primal path-following IPM iterations to compute a $\tau_k$-optimal solutions to QP-subproblems in the $k$-th iteration of P-APG is bounded above by $O(\bar{N} \ln(\frac{1}{\tau_k}))$ – see Section 4.3.2 in [25] (similar bounds can be driven for primal-dual path-following IPMs as well). Moreover, since QP-subproblems are strongly convex, $\tau_k$-optimality in function values implies an error bound on gradient evaluations in Step 2 of P-APG.

Recall that under Assumption 2.2, we have $N = K\bar{N}$ such that $\bar{N} > n + 1$. Below we consider the bottleneck memory requirement for solving (2.7) in 2 cases: running a) P-APG with a primal-dual IPM computing Step-1 in Fig. 2.2 and b) IPM alone on (2.7). For case a), the memory bottleneck in each iteration is due to solution of $K$ Newton systems corresponding to (2.56); on the other hand, for case b), the memory bottleneck is due to solution of a much larger Newton system using the normal equations in (2.47). In a naive implementation of case b), one stores the non-zero components of the Cholesky factor $L_{\text{per}}$ in (2.52) corresponding to the block arrowhead matrix in (2.48) after permuting as in (2.51), which requires storing $O(N^2(n + 1) + Nn^2) = KO(K\bar{N}^2(n + 1) + \bar{N}n^2)$ entries; while for case a), for each $i \in K$, one stores the non-zero components of a Cholesky factor, analogous to (2.52), for the QP in (2.56) – see Corollary 2.15; hence, this naive implementation requires storing $KO(\bar{N}^2(n + 1) + \bar{N}n^2)$ entries for all the Cholesky factors in total, in addition to storing $\bar{N}^2K(K - 1)$ dual variables, i.e., $\theta = [\theta_{ij}]_{(i,j) \in \mathcal{G}} \in \mathbb{R}^{\bar{N}^2K(K - 1)}$. Furthermore, for case b), in a more memory efficient implementation, (2.50) in Theorem 2.14 implies that $\Delta \xi_\ell$ can be computed sequentially after computing $\Delta \gamma$, which requires to store $O(N^2)$ at any time at the expense of forming $M_{\ell \ell}$ and $M_0\ell$ twice. Similarly, one can exploit this fact for case a) as well while solving normal equations for each $i \in \mathcal{N}$, which requires $O(K\bar{N}^2)$ memory in total if $K$ processors run in parallel, and $O(N^2)$ if $K$ QPs in (2.56) are solved sequentially on a single processor. Therefore, running IPM within P-APG reduces the memory requirement significantly at least by a factor of $K$ in comparison to running IPM alone, e.g., if we partition $N$ observations into $K = 10$ subsets and each subproblem requires 1GB of memory, then running IPM alone requires roughly 100GB, while IPM within P-APG requires only 10GB in total. This discussion is summarized in Table 2.1. Finally, recall the discussion at the beginning of
Section 2.2.6: neither $A_1$ nor $A_2$ needs to be stored in the memory; storing only $\{\bar{x}_\ell\}_{\ell=1}^N$ is sufficient to be able to compute matrix-vector multiplications with $A_1$ and $A_2$.

Table 2.1: Comparison of Memory Usage

<table>
<thead>
<tr>
<th></th>
<th>IPM alone</th>
<th>P-APG with IPM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive</td>
<td>$O(K^2N^2n)$</td>
<td>$O(KN^2(n + K))$</td>
</tr>
<tr>
<td>Memory Efficient</td>
<td>$O(K^2N^2)$</td>
<td>$O(KN^2)$</td>
</tr>
</tbody>
</table>

2.3 Competitive Methods

In this section, we discuss an active set method for solving (2.7), and a multi-block ADMM method recently proposed by [24] to solve problem (2.4).

2.3.1 Active Set Method (ASM)

Although the number of constraints is $O(N^2)$ in (2.7), one expects that only few of them will be potentially active at the optimal solution; furthermore, this indeed turned out to be the case based on our numerical results for the test problems we considered in this chapter – the number of active constraints was roughly $O(N)$. Therefore, in this section, we briefly state a primal active set method to solve the regularized convex regression problem in (2.7) as an immediate alternative to P-APG method, and compare it with our P-APG method. One issue with primal active set methods is to determine an initial feasible point such that only very few constraints are active; and usually to overcome this problem one can use either “Phase I” or “big M” techniques. However, as we have already seen in the proof of Lemma 2.8, it is easy to construct an interior point for the polyhedron in (2.7) in spite of $O(N^2)$ constraints defining the set. In particular, let $\hat{x} \triangleq \frac{1}{N} \sum_{\ell \in \mathcal{N}} \bar{x}_\ell$ and $\hat{y} \triangleq \frac{1}{N} \sum_{\ell \in \mathcal{N}} \bar{y}_\ell$, and for an arbitrary $\alpha > 0$, define $\tilde{y} = [\bar{y}_\ell]_{\ell \in \mathcal{N}}$ and $\tilde{\xi} = [\xi_\ell]_{\ell \in \mathcal{N}}$ such that

$$\tilde{y}_\ell \triangleq \hat{y} + \frac{\alpha}{2} \| \bar{x}_\ell - \hat{x} \|_2^2, \quad \tilde{\xi}_\ell \triangleq \alpha (\bar{x}_\ell - \hat{x}), \quad \ell \in \mathcal{N}. \quad (2.57)$$

According to (2.27), $\tilde{\eta} \triangleq [\tilde{y}^\top \tilde{\xi}^\top]^\top$ is a Slater point such that $A\tilde{\eta} = A_1\hat{y} + A_2\tilde{\eta} \geq \frac{\alpha v}{2} \mathbf{1}$, where $A = [A_1 A_2]$ and $v > 0$ is defined in Lemma 2.8. Hence, no constraint is active at $\tilde{\eta}$.
Algorithm ASM

Iteration 0: Set $\tilde{\eta}^0$ as in (2.57), and $W^0 \leftarrow \emptyset$
Iteration $k$: $(k \geq 0)$

1: $\Delta \eta^k \leftarrow \arg\min_{\Delta \eta} \frac{1}{2} \Delta \eta^\top G \Delta \eta + (G \eta^k + c)^\top \Delta \eta$ s.t. $a_{\ell \ell'}^\top \Delta \eta = 0$, $(\ell, \ell') \in W^k$

2: if $\Delta \eta^k = 0$, then
3: Compute $\theta_{\ell \ell'}(\ell, \ell') \in W^k \subset \mathbb{R}$ such that $\sum_{(\ell, \ell') \in W^k} a_{\ell \ell'} \theta_{\ell \ell'} = G \eta^k + c$
4: if $\theta_{\ell \ell'} \geq 0$ for all $(\ell, \ell') \in W^k$, STOP with solution $\eta^* = \eta^k$;
5: else $(\bar{\ell}, \bar{\ell}') \leftarrow \arg\min_{(\ell, \ell') \in W^k} \theta_{\ell \ell'}$, $\eta^{k+1} \leftarrow \eta^k$, $W^{k+1} \leftarrow W^k \setminus \{(\bar{\ell}, \bar{\ell}')\}$;
6: else $\Delta \eta^k \neq 0$
7: $t^k \leftarrow \min \left\{ 1, \min \left\{ \frac{-a_{\ell \ell'}^\top \eta^k}{a_{\ell \ell'}^\top \Delta \eta^k} : (\ell, \ell') \notin W^k \text{ s.t. } a_{\ell \ell'}^\top \Delta \eta^k < 0 \right\} \right\}$,
8: $\eta^{k+1} \leftarrow \eta^k + t^k \Delta \eta^k$, $I \leftarrow \{(\ell, \ell') \notin W^k : a_{\ell \ell'}^\top \eta^{k+1} = 0\}$
9: if $I \neq \emptyset$, then $W^{k+1} \leftarrow W^k \cup \{(\ell, \ell')\}$ for some $(\ell, \ell') \in I$;
10: else set $W^{k+1} \leftarrow W^k$;

Figure 2.3: Active Set Algorithm (ASM)

Consider (2.7), which can be restated in a more compact form: the QP in (2.44) with $c = [y^\top 0^\top]^\top$. We will show that the primal active set algorithm shown in Fig. 2.3 can be efficiently implemented. In the rest, $a_{\ell \ell'}^\top$ denotes the row of $A$ corresponding to index $(\ell, \ell') \in P$ – recall that the rows of $A$ are sorted according to increasing lexicographic order on the index set $P$.

Definition 2.17. For $k \geq 1$, let $W^k \subset P$ denote the working set at iteration $k$, which is a subset of active constraint indices, i.e., $a_{\ell \ell'} \eta^k = 0$ for $(\ell, \ell') \in W^k$, and $m_k = |W^k|$. We form $A^k = [a_{\ell \ell'}^\top]_{(\ell, \ell') \in W^k} \in \mathbb{R}^{m_k \times N(n+1)}$ concatenating the rows vertically, and define $A_1^k \in \mathbb{R}^{m_k \times N}$ and $A_2^k \in \mathbb{R}^{m_k \times Nn}$ as the submatrices of $A^k$ such that $A_1^k$ and $A_2^k$ consist of columns of $A^k$ corresponding to $y$ and $\xi$, respectively.

The working set update strategy given in Fig. 2.3 ensures that $\{a_{\ell \ell'}\}_{(\ell, \ell') \in W^k}$ are linearly independent for all $k \geq 1$ – see [35] for details on this property, which we assume in the rest of this section.

Note that in each iteration $k \geq 1$, we need to solve a subproblem to determine the
direction $\Delta \eta^k$ as follows

$$
\Delta \eta^k = \arg\min_{\Delta \eta} \frac{1}{2} \Delta \eta^T G \Delta \eta + (G \eta^k + c)^T \Delta \eta \quad \text{s.t.} \quad A^k \Delta \eta = 0 : \theta^k, \quad (2.58)
$$

where $\theta^k = [\theta_{k(\ell,\ell')}]_{(\ell,\ell') \in W^k} \in \mathbb{R}^{m_k}$ denotes an optimal dual solution. Hence, $(\Delta \eta^k, \theta^k)$ satisfies the KKT system corresponding to (2.58):

$$
\begin{bmatrix}
G & A^k\gamma \\
A^k & 0
\end{bmatrix}
\begin{bmatrix}
-\Delta \eta^k \\
\theta^k
\end{bmatrix}
= \begin{bmatrix}
G \eta^k + c \\
0
\end{bmatrix} \quad \Rightarrow \quad A^k G^{-1} A^k\gamma \theta^k = A^k(\eta^k + c) \quad (2.59)
$$

since $G^{-1}c = c$. Therefore, $\theta^k$ can be computed via forward and backward substitution after computing the Cholesky factorization of $A^k G^{-1} A^k\gamma$; next one can compute $\Delta \eta^k$ according to the first row in the KKT system as follows: $\Delta \eta^k = G^{-1} A^k\gamma \theta^k - (\eta^k + c)$.

**Remark 2.18.** It is worth noting that $A^k G^{-1} A^k\gamma = A^k_1 A^k_1\gamma + \frac{1}{\gamma} A^k_2 A^k_2\gamma$. For any $k \geq 1$, $A^k_1 A^k_1\gamma$ and $A^k_2 A^k_2\gamma$ computations require $\mathcal{O}(m_k^2)$ and $\mathcal{O}(m_k^2 n)$ flops, respectively; and given $a_{k\ell'}$ for some $(\ell, \ell') \in P \setminus W^k$, $A^k G^{-1} a_{k\ell'}$ can be computed in $\mathcal{O}(m_k(n + 1))$ flops. These complexity bounds can be easily verified by observing the structure in $A^k$ after ordering its rows according to increasing lexicographic order on the index set $W^k$.

Naively, the majority of total computational complexity at iteration $k$ is mainly due to forming $A^k G^{-1} A^k\gamma = A^k_1 A^k_1\gamma + \frac{1}{\gamma} A^k_2 A^k_2\gamma$ in $\mathcal{O}(m_k^2(n + 1))$ flops, and computing its Cholesky factorization in $\mathcal{O}(m_k^3)$ flops – the factorization exists since $A^k G^{-1} A^k\gamma$ is positive definite due to $\text{rank}(A^k) = m_k$. That said, at the end of each iteration the working set changes by at most one index; thus, one does not need to compute Cholesky factorizations from scratch. In particular, because at most one row (constraint) is added or deleted from $A^k$, Cholesky factorization for $A^{k+1} G^{-1} A^{k+1\gamma}$ can be updated very efficiently by using $A^{k+1} G^{-1} A^{k+1\gamma} = L^k L^k\gamma$ from the previous iteration. Also, note $\theta^k$ is a byproduct of this approach, so we don’t need to compute it again in the following step if $\eta^k = 0$. Next, we will briefly discuss how to utilize the information from the previous iteration to solve the subproblems much more efficiently.

**Lemma 2.19.** For some $m \geq 1$, let $B \in \mathbb{R}^{m \times N(n+1)}$ and $b \in \mathbb{R}^{N(n+1)}$ such that $\text{rank}(B) = m$ and $b$ is not in the row-space of $B$. Suppose $LL\gamma$ represent the Cholesky factorization of $BG^{-1} B\gamma$ for some symmetric positive definite matrix $G$. Define $\bar{B} = \begin{bmatrix}
B \\
b^\top
\end{bmatrix} \in \mathbb{R}^{(m+1) \times N(n+1)}$. Then given $L$, Cholesky factorization for $\bar{B} G^{-1} \bar{B} \gamma = \bar{L} \bar{L}\gamma$ can
be computed as
\[
\bar{L} = \begin{bmatrix} L & 0 \\ h^\top & d \end{bmatrix}, \quad h = L^{-1}B^{-1}b, \quad d = \sqrt{b^\top G^{-1}b}. \tag{2.60}
\]

**Proof:** Since \(\text{rank}(\bar{B}) = m + 1\), trivially \(\bar{B}G^{-1}\bar{B}^\top\) is positive definite, and it has a Cholesky factorization \(\bar{L}\bar{L}^\top\). Moreover, it is easy to verify that \(\bar{L}\) given in (2.60) is the Cholesky factor. Assume that we already know Cholesky factorization \(A^k G^{-1} A^k = L^k L^k\top\), and \(W^{k+1} = W^k \cup \{(\ell, \ell')\}\) for some \((\ell, \ell') \in \mathcal{P} \setminus W^k\). Suppose \(a_{\ell\ell'}^\top\) is appended to \(A^k\) as the last row to form \(A^{k+1}\). Since \(\text{rank}(A^{k+1}) = m_k + 1\), setting \(B = A^k\) and \(b = a_{\ell\ell'}\) satisfies the conditions in Lemma 2.19. Thus, according to (2.60), the new factorization for \(A^{k+1} G^{-1} A^{k+1}_\ell\) can be computed as \(L^{k+1} = \begin{bmatrix} L^k & 0 \\ h^k & d^k \end{bmatrix}\), which only requires to solve \(L^k h_k = A^k G^{-1} a_{\ell\ell'}\) for \(h_k\), and to compute \(d^k = \sqrt{a_{\ell\ell'}^\top G^{-1} a_{\ell\ell'}}\). Note computing \(h_k\) requires forming \(A^k G^{-1} a_{\ell\ell'}\), which can be computed in \(O(m_k n)\) flops according to Remark 2.18 and implementing one forward substitution, which can be done in \(O(m_k^2)\) flops.

Now consider the case \(W^{k+1} = W^k \setminus \{(\ell, \ell')\}\) for some \((\ell, \ell') \in W^k\). Note that \(a_{\ell\ell'}^\top\) is an arbitrary row of \(A^k\) (not necessarily the last one). The following lemma will help us update the factorization corresponding to \(W^{k+1}\) efficiently when we are given \(L^k\).

**Lemma 2.20.** Let \(B_1 \in \mathbb{R}^{s_1 \times N(n+1)}\), \(B_2 \in \mathbb{R}^{s_2 \times N(n+1)}\), and \(b \in \mathbb{R}^N(n+1)\) such that \(\text{rank}(B) = s_1 + s_2 + 1\), where \(B = [B_1^\top \ b \ B_2^\top]^\top\). Suppose \(L L^\top\) represent the Cholesky factorization of \(B G^{-1} B^\top\) for some symmetric positive definite matrix \(G\), where \(L = \begin{bmatrix} L_1 & 0_{s_1 \times 1} & 0_{s_1 \times s_2} \\ h_1^\top & d & 0_{1 \times s_2} \\ F & h_2 & L_2 \end{bmatrix}\). Define \(\bar{B} = [B_1^\top \ B_2^\top]^\top \in \mathbb{R}^{(s_1 + s_2) \times N(n+1)}\). Then given \(L\), Cholesky factorization for \(\bar{B} G^{-1} \bar{B}^\top = \bar{L} \bar{L}^\top\) can be computed as
\[
\bar{L} = \begin{bmatrix} L_1 & 0_{s_1 \times s_2} \\ F & \bar{L}_2 \end{bmatrix}, \quad \text{s.t.} \quad \bar{L}_2 \bar{L}_2^\top = L_2 L_2^\top + h_2 h_2^\top. \tag{2.61}
\]

Moreover, given \(L_2\) and \(h_2\), computing \(\bar{L}_2\) requires \(O(s_2^2)\) flops.

**Proof:** It is easy to verify that \(\bar{L}\) given in (2.61) is the lower-triangular Cholesky factor of \(\bar{B} G^{-1} \bar{B}^\top\). For details of computing \(\bar{L}_2\), refer to [36]. Moreover, MATLAB routine
cholupdate($L_2^T, h_2$) can be called to compute $\tilde{L}_2^T$.

**Algorithm Multi-Block ADMM**

Iteration 0: $\Delta_{ij} \leftarrow \bar{x}_i - \bar{x}_j$ for $(i, j) \in \mathcal{N} \times \mathcal{N}$ and $\tilde{\Delta}_j \leftarrow (\sum_{i \in \mathcal{N}} \Delta_{ij} \Delta_{ij}^T)^{-1}$ for $j \in \mathcal{N}$

Iteration $k$: $(k \geq 0)$

1. $\xi_{ij}^{k+1} \leftarrow \tilde{\Delta}_j (\sum_{i \in \mathcal{N}} \Delta_{ij} (\theta_{ij}^k / \rho + \nu_{ij}^k + y_{ij}^k - y_{ij}^f))$ for $j \in \mathcal{N}$
2. $\tilde{\nu}_{ij}^{k+1} \leftarrow \nu_{ij}^k - \Delta_{ij}^T \xi_{ij}^{k+1}$ for $(i, j) \in \mathcal{N} \times \mathcal{N}$
3. $\bar{x}_{ij}^{k+1} \leftarrow \bar{y} + D^T \theta^k + \rho D^T \tilde{\nu}^{k+1}$
4. $y_{ij}^{k+1} \leftarrow \frac{1}{1 + 2 \rho \sum_{j \in \mathcal{N}} w_j} \left( w_i + 2 \rho \sum_{j \in \mathcal{N}} w_j \right)$ for $i \in \mathcal{N}$
5. $\nu_{ij}^{k+1} \leftarrow \min \left\{ y_{ij}^{k+1} + \Delta_{ij}^T \xi_{ij}^{k+1} - y_{ij}^{k+1} - \theta_{ij}^k / \rho, 0 \right\}$ for $(i, j) \in \mathcal{N} \times \mathcal{N}$
6. $\theta_{ij}^{k+1} \leftarrow \theta_{ij}^k + \rho \left( \nu_{ij}^{k+1} + y_{ij}^{k+1} - y_{ij}^{f} - \Delta_{ij}^T \xi_{ij}^{k+1} \right)$ for $(i, j) \in \mathcal{N} \times \mathcal{N}$

Figure 2.4: Multi-block ADMM (ADMM)

### 2.3.2 Multi-block ADMM

Recently, Mazumder et al. [24] proposed a multi-block ADMM to solve problem (2.4). Although, the authors report that it works well in practice, to our best knowledge, the convergence property of the method is still unknown. In fact, it is recently shown that ADMM does not necessarily converge when the number of primal variable blocks are three or more [37]; and the ADMM algorithm in [24], displayed in Fig. 5.5, alternatingly updates three blocks of primal variables: $\mathbf{x} = [\xi_{ij}]_{i \in \mathcal{N}}, \mathbf{y} = [y_{ij}]_{i \in \mathcal{N}}$ and $\mathbf{v} = [\nu_{ij}]_{(i,j) \in \mathcal{N} \times \mathcal{N}}$.

The matrix $D \in \mathbb{R}^{N^2 \times N}$ is similar to our matrix $A_2$ defined in Definition 2.1 except $D$ also contains rows corresponding to $(i, i) \in \mathcal{N} \times \mathcal{N}$, i.e., $D \mathbf{y} = \mathbf{z} \in \mathbb{R}^{N^2}$ such that $z_{ij} = y_{ij} - y_i$ for $(i, j) \in \mathcal{N} \times \mathcal{N}$, and the long-vector $\mathbf{z}$ obtained by sorting its elements according to increasing lexicographic order on the index set $\mathcal{N} \times \mathcal{N}$. Similarly, the elements of the auxiliary variable $\tilde{\nu} \in \mathbb{R}^{N^2}$ is also sorted according to increasing lexicographic order on the index set $\mathcal{N} \times \mathcal{N}$. During initialization, the ADMM algorithm requires computing $\tilde{\Delta}_j = (\sum_{i \in \mathcal{N}} \Delta_{ij} \Delta_{ij}^T)^{-1}$ for all $i \in \mathcal{N}$, where $\Delta_{ij} = \bar{x}_i - \bar{x}_j$. Although it is required only one time, this computation costs $O(N^2n^2 + Nn^3)$ flops. Based on our numerical tests, as $N$ increases, this preprocessing time becomes substantial compared to overall runtime. At each iteration, the algorithm needs to update five different variables: $\mathbf{\xi}^k, \mathbf{w}^k, \mathbf{y}^k, \mathbf{v}^k$ and $\mathbf{\theta}^k$. The cost for updating subgradient vector $\mathbf{\xi}^k$ is $O(N^2n + Nn^2)$ flops, updating $\mathbf{w}^k$ takes $O(N^2)$ flops, and given $\mathbf{w}^{k-1}$ updating the function value-vector $\mathbf{y}^k$ takes $O(N)$ flops,
and updating residuals $\nu^k$ and dual variables $\theta^k$ both take $O(N^2)$ flops separately. Thus, the overall per iteration complexity is $O(N^2 n + N n^2)$ with $O(N^2 n^2 + N n^3)$ one-time cost at the beginning. Note that ADMM needs to store not only matrix $D \in \mathbb{R}^{N^2 \times N}$ and vectors $\Delta_{ij} \in \mathbb{R}^n$ for all $(i,j) \in \mathcal{N}$, that are comparable to our $A_1$ and $A_2$, but also $\bar{\Delta}_i \in \mathbb{R}^{n \times n}$ for all $i \in \mathcal{N}$, which are the matrices inverted during pre-processing; hence, the number of non-zeros stored in the RAM for ADMM is roughly $(N^2 - N)(n + 2) + N n^2$, which is $O(K^2 N^2 n)$. When compared to Table 2.1 clearly P-APG leads to significant memory savings.

### 2.4 Numerical Study

Here we demonstrate the scalability of P-APG, and compare its performance against other competitive methods: an interior point method, an active set method (ASM), and a multi-block ADMM. To solve the convex regression problem, we implemented P-APG, ASM and ADMM in MATLAB, and used the stand-alone version MOSEK as an interior point solver for benchmarking purposes. Moreover, for P-APG, we also use MOSEK together with the Parallel Computing Toolbox, in order to solve K QP-subproblems in parallel using $K$ cores in each iteration of P-APG in Fig. 2.2. MOSEK is a commercial off-the-shelf software which has a state-of-the-art interior-point optimizer for quadratic problems. Note that MOSEK also comes with CVX, which is a popular MATLAB-based modeling system for convex optimization; but this version of MOSEK is not compatible with Parallel Computing Toolbox in MATLAB, i.e., even though one calls MOSEK through CVX formulations within a parfor loop, the $K$ subproblems are still solved in a sequential manner. In order to take advantage of the computing power in a cluster of computers for long-running jobs, one has to adopt batch processing in MATLAB to be able to better exploit the processor cores in multiple machines. On the other hand, matrix operations in MATLAB leverage multi-core and multi-threading framework by default. Hence, ASM and ADMM are coded without using the parallel toolbox, as they only contain matrix operations in every iteration and these operations are executed in parallel automatically. To eliminate factors that might have an influence on the runtime to the best extent, we carried out all numerical tests comparing P-APG against other methods on high performance computing cluster by executing a single script, so that they all run on exactly the same processor cores and memory modules. Numerical tests are carried out on a single node at a research computing cluster. The node is composed of one 24-core processor, each having 1GB RAM (24GB RAM in
total). We determine the number of core processors and the amount of RAM allocated depending on the size of the problem solved – see Sections 2.4.1 and 2.4.2.

**Experimental setup:** Our problem setup adopted in the following sections involve two different test functions: 1) \( f_0(x) = \frac{1}{2} x^T Q x \) and 2) \( f_0(x) = \exp(p^T x) \), where \( Q \in \mathbb{R}^{n \times n} \) is a symmetric matrix, \( p \in \mathbb{R}^n \), and they are randomly generated as follows. We first set \( \bar{Q} = \Lambda^T \Lambda \) such that \( \Lambda \in \mathbb{R}^{n \times n} \) is generated randomly with all components being i.i.d. with \( \mathcal{N}(0, 1) \), where \( \mathcal{N}(\mu, \sigma^2) \) denotes Normal distribution with mean \( \mu \) and variance \( \sigma^2 \); next, without changing left and right singular vectors of \( \bar{Q} \), we transform its singular values such that the resulting condition number is 15 and we call the resulting matrix as \( Q \); and \( p \in \mathbb{R}^n \) is generated using uniform distribution on the hypercube \([0, 0.2]^n\). The noisy observations \( \{y_\ell\}_{\ell \in \mathcal{N}} \) are generated according to (2.1), where the locations \( \{x_\ell\}_{\ell = 1}^N \subset \mathbb{R}^n \) and additive noise \( \{\epsilon_\ell\}_{\ell = 1}^N \subset \mathbb{R} \) are generated randomly with all components being i.i.d. with \( \mathcal{N}(0, 4) \) and \( \mathcal{N}(0, 100) \) respectively. In addition, we moved 30% of randomly chosen location/observation pairs into the interior of the epigraph of the test function \( f_0 \) by replacing \((\bar{x}_\ell, \bar{y}_\ell)\) with \((x_\ell, 1.3\bar{y}_\ell)\). In all the experiments involving P-APG and ASM, we set \( \gamma = 10^{-4} \) in (2.7).

### 2.4.1 Convergence behavior of P-APG on the regularized problem

We compare i) running MOSEK alone and ii) running it within P-APG on the regularized problem (2.7) with increasing dimension. The numerical study is mainly aimed to demonstrate how the performance of each method scales for solving the regularized problem as its dimension increases. First, we start with a small size problem: \( n = 10, N = 100 \), and use the test function \( f_0(x) = \frac{1}{2} x^T Q x \). We compare the quality of the solutions computed by P-APG and dual gradient ascent (as the dual function \( g_\gamma \) in (2.22) is differentiable). In order to compute dual gradient, \( \nabla g_\gamma \), one needs to solve \( K \) quadratic subproblems. To exploit this parallel structure, we partition the data into two sets, i.e., \( K = 2 \). Within both dual gradient ascent and P-APG, we called MOSEK to compute the dual gradients via solving \( K \) QP-subproblems. Since we allow violations for the relaxed constraints, we define the “duality gap” at the \( k \)-th iteration as \( \theta_k^T C \eta_k \) – recall that \( \eta_k = [y_k^T \xi_k^T]^T \). Fig. 2.5(left) represents how the duality gap for both methods change at each iteration. In order to better understand the behavior of P-APG, we report in Fig. 2.5(right) the duality gap of P-APG in a larger scale. Fig. 2.6 reports the infeasibility of iterates, i.e., \( \| (A_1 y_k + A_2 \xi_k) \|_2 \).

A primal-dual iterate \((\eta, \theta)\) is optimal if the duality gap and infeasibility are both
Figure 2.5: Duality Gap for P-APG and Dual Gradient Ascent: (left) P-APG and Dual Gradient Ascent, (right) Zoom-in for P-APG Method

Figure 2.6: Distance to Feasible Region for P-APG and Dual Gradient Ascent.

zero. As the feasibility happens in the limit, the duality gap in Fig. 2.5(right) can go below 0, which can be explained by the infeasibility of iterates. Therefore, observing a decrease in duality gap only tells one part of the story; without convergence to feasibility, it is not valuable alone as a measure. As shown in the Fig. 2.5, the duality gap converges quickly to zero for both methods. On the other hand, as shown in Fig. 2.6, constraint violation for P-APG iterates decreases to 0 much faster than it does for the dual gradient ascent iterates. Hence, P-APG iterate sequence converges to the unique optimal solution considerably faster.

As shown in Tables 2.2, 2.3, 2.4 and 2.5, the dimension of variables $n \in \{20, 80\}$, and the number of observations $N \in \{200, 400, 800, 1600, 2400\}$. Since the number of constraints increases at the rate of $O(N^2)$, as the size of problem increases in $N$, we reported the normalized infeasibility $\|A_1y + A_2\xi\|_2/\sqrt{N^2 - N}$ and normalized duality gap $|\theta^TC\eta|/(N^2 - N)$. We partition the set of observations $\mathcal{N}$ into $K$ subsets.
Each one of them consists of 100 points; therefore, we set $K = 2, 4, 8, 16, 24$ for $N = 200, 400, 800, 1600, 2400$, and we reserve $2/2, 4/4, 8/8, 16/16, 24/24$ number of Cores/RAM, respectively, depending on $N$ so that for each job submitted to the computing cluster, an instance of (2.7) is solved using P-APG on the node such that each subproblem in (2.56) for $i \in \mathcal{K}$ is computed on a different core. We tested both the adaptive step and constant step version of P-APG, which we abbreviate as PAPG_A and PAPG_C, respectively. Both PAPG_A and PAPG_C are terminated whenever they compute a primal-dual iterate, $\eta = [y^T \xi^T]^T$ and $\theta$, satisfying the stopping criteria: $\| (A_1 y + A_2 \xi) \| / \sqrt{N^2 - N} \leq 1e^{-1}$ and $|\theta^T C \eta|/(N^2 - N) \leq 5e^{-7}$, or at the end of 2 hours, which are reported as Infeasibility and DualGap respectively in the tables. Moreover, we also report relative suboptimality, i.e., $\text{SubOpt}_\text{Reg} = |p - p^*_\gamma|/p^*_\gamma$, where $p^*_\gamma$ denotes the optimal value to (2.7) and $p$ denotes the objective value of (2.7) at termination. Preprocess for P-APG method is the wall-clock time elapsed during the computation of the maximum singular value for the matrix $A_4$. Additionally, in all the tables, N/A means that the wall clock time exceeded 2 hours for the job, and Wall-time stands for wall-clock time in seconds for the whole job including Preprocess.

The numerical results reported in Tables 2.2, 2.3, 2.4 and 2.5 show that P-APG solution is very close to the optimal solution of the regularized problem in (2.7). Note that MOSEK using interior point optimizer starts working slowly beyond $N = 1600$ due to $\mathcal{O}(N^2 n)$ memory requirement – see Table 2.1. Numerical results show that advantages

<table>
<thead>
<tr>
<th>n, N</th>
<th>Algorithm</th>
<th>Cores/RAM</th>
<th>Preprocess</th>
<th>Wall-time</th>
<th>Infeasibility</th>
<th>SubOpt_Reg</th>
<th>DualGap</th>
</tr>
</thead>
<tbody>
<tr>
<td>20, 200</td>
<td>Mosk.Reg</td>
<td>2/2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PAPG.A</td>
<td>2/2</td>
<td>0.2</td>
<td>30</td>
<td>9.5E-02</td>
<td>3.86E-04</td>
<td>6.80E-08</td>
</tr>
<tr>
<td></td>
<td>PAPG.C</td>
<td>2/2</td>
<td>0.2</td>
<td>16</td>
<td>9.75E-02</td>
<td>1.22E-03</td>
<td>2.63E-07</td>
</tr>
<tr>
<td>20, 400</td>
<td>Mosk.Reg</td>
<td>4/4</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PAPG.A</td>
<td>4/4</td>
<td>1.1</td>
<td>47</td>
<td>9.54E-02</td>
<td>1.92E-03</td>
<td>2.56E-07</td>
</tr>
<tr>
<td></td>
<td>PAPG.C</td>
<td>4/4</td>
<td>1.0</td>
<td>44</td>
<td>9.94E-02</td>
<td>7.45E-05</td>
<td>3.23E-08</td>
</tr>
<tr>
<td>20, 800</td>
<td>Mosk.Reg</td>
<td>8/8</td>
<td>0</td>
<td>42</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PAPG.A</td>
<td>8/8</td>
<td>4.3</td>
<td>92</td>
<td>9.92E-02</td>
<td>6.14E-03</td>
<td>4.79E-07</td>
</tr>
<tr>
<td></td>
<td>PAPG.C</td>
<td>8/8</td>
<td>4.2</td>
<td>101</td>
<td>9.87E-02</td>
<td>2.06E-03</td>
<td>1.75E-07</td>
</tr>
<tr>
<td>20, 1600</td>
<td>Mosk_Reg</td>
<td>16/16</td>
<td>0</td>
<td>311</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PAPG.A</td>
<td>16/16</td>
<td>20.3</td>
<td>259</td>
<td>9.50E-02</td>
<td>1.10E-02</td>
<td>4.80E-07</td>
</tr>
<tr>
<td></td>
<td>PAPG.C</td>
<td>16/16</td>
<td>20.0</td>
<td>408</td>
<td>9.99E-02</td>
<td>5.89E-03</td>
<td>2.63E-07</td>
</tr>
<tr>
<td>20, 2400</td>
<td>Mosk.Reg</td>
<td>24/24</td>
<td>0</td>
<td>987</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PAPG.A</td>
<td>24/24</td>
<td>56.1</td>
<td>323</td>
<td>9.77E-02</td>
<td>1.43E-02</td>
<td>4.61E-07</td>
</tr>
<tr>
<td></td>
<td>PAPG.C</td>
<td>24/24</td>
<td>59.7</td>
<td>723</td>
<td>1.00E-01</td>
<td>7.42E-03</td>
<td>2.43E-07</td>
</tr>
</tbody>
</table>
Table 2.3: Comparison with test function $\frac{1}{2}x^TQx$ for $n = 80$

<table>
<thead>
<tr>
<th>$n, N$</th>
<th>Algorithm</th>
<th>Cores/RAM</th>
<th>Preprocess</th>
<th>Wall-time</th>
<th>Infeasibility</th>
<th>SubOpt_Reg</th>
<th>DualGap</th>
</tr>
</thead>
<tbody>
<tr>
<td>80, 200</td>
<td>Mosek_Reg</td>
<td>2/2</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PAPG_A</td>
<td>2/2</td>
<td>0.5</td>
<td>89</td>
<td>9.46E-02</td>
<td>6.08E-04</td>
<td>1.88E-07</td>
</tr>
<tr>
<td></td>
<td>PAPG_C</td>
<td>2/2</td>
<td>0.5</td>
<td>54</td>
<td>9.93E-02</td>
<td>1.05E-03</td>
<td>3.28E-07</td>
</tr>
<tr>
<td>80, 400</td>
<td>Mosek_Reg</td>
<td>4/4</td>
<td>0</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PAPG_A</td>
<td>4/4</td>
<td>2.5</td>
<td>324</td>
<td>8.92E-02</td>
<td>5.79E-04</td>
<td>9.54E-08</td>
</tr>
<tr>
<td></td>
<td>PAPG_C</td>
<td>4/4</td>
<td>2.6</td>
<td>287</td>
<td>9.87E-02</td>
<td>1.09E-03</td>
<td>1.96E-07</td>
</tr>
<tr>
<td>80, 800</td>
<td>Mosek_Reg</td>
<td>8/8</td>
<td>0</td>
<td>97</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PAPG_A</td>
<td>8/8</td>
<td>12.2</td>
<td>383</td>
<td>9.79E-02</td>
<td>1.43E-03</td>
<td>1.29E-07</td>
</tr>
<tr>
<td></td>
<td>PAPG_C</td>
<td>8/8</td>
<td>12.5</td>
<td>379</td>
<td>9.92E-02</td>
<td>1.02E-03</td>
<td>9.12E-08</td>
</tr>
<tr>
<td>80, 1600</td>
<td>Mosek_Reg</td>
<td>16/16</td>
<td>0</td>
<td>661</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PAPG_A</td>
<td>16/16</td>
<td>55.9</td>
<td>597</td>
<td>9.59E-02</td>
<td>1.61E-03</td>
<td>7.67E-08</td>
</tr>
<tr>
<td></td>
<td>PAPG_C</td>
<td>16/16</td>
<td>57.5</td>
<td>1145</td>
<td>9.94E-02</td>
<td>1.61E-03</td>
<td>7.70E-08</td>
</tr>
<tr>
<td>80, 2400</td>
<td>Mosek_Reg</td>
<td>24/24</td>
<td>0</td>
<td>1966</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PAPG_A</td>
<td>24/24</td>
<td>128.5</td>
<td>897</td>
<td>9.91E-02</td>
<td>1.17E-03</td>
<td>3.87E-08</td>
</tr>
<tr>
<td></td>
<td>PAPG_C</td>
<td>24/24</td>
<td>133.8</td>
<td>1947</td>
<td>1.00E-01</td>
<td>2.57E-03</td>
<td>9.19E-08</td>
</tr>
</tbody>
</table>

Table 2.4: Comparison with test function $\exp(p^Tx)$ for $n = 20$

<table>
<thead>
<tr>
<th>$n, N$</th>
<th>Algorithm</th>
<th>Cores/RAM</th>
<th>Preprocess</th>
<th>Wall-time</th>
<th>Infeasibility</th>
<th>SubOpt_Reg</th>
<th>DualGap</th>
</tr>
</thead>
<tbody>
<tr>
<td>20, 200</td>
<td>Mosek_Reg</td>
<td>2/2</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PAPG_A</td>
<td>2/2</td>
<td>0.2</td>
<td>35</td>
<td>8.97E-02</td>
<td>1.01E-03</td>
<td>3.58E-07</td>
</tr>
<tr>
<td></td>
<td>PAPG_C</td>
<td>2/2</td>
<td>0.2</td>
<td>17</td>
<td>9.62E-02</td>
<td>1.16E-03</td>
<td>4.09E-07</td>
</tr>
<tr>
<td>20, 400</td>
<td>Mosek_Reg</td>
<td>4/4</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>NaN</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PAPG_A</td>
<td>4/4</td>
<td>1.0</td>
<td>67</td>
<td>9.41E-02</td>
<td>1.36E-03</td>
<td>2.15E-07</td>
</tr>
<tr>
<td></td>
<td>PAPG_C</td>
<td>4/4</td>
<td>1.0</td>
<td>71</td>
<td>9.93E-02</td>
<td>6.50E-04</td>
<td>7.11E-08</td>
</tr>
<tr>
<td>20, 800</td>
<td>Mosek_Reg</td>
<td>8/8</td>
<td>0</td>
<td>66</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PAPG_A</td>
<td>8/8</td>
<td>4.2</td>
<td>194</td>
<td>7.04E-02</td>
<td>5.56E-03</td>
<td>4.62E-07</td>
</tr>
<tr>
<td></td>
<td>PAPG_C</td>
<td>8/8</td>
<td>4.1</td>
<td>266</td>
<td>9.91E-02</td>
<td>5.39E-03</td>
<td>4.45E-07</td>
</tr>
<tr>
<td>20, 1600</td>
<td>Mosek_Reg</td>
<td>16/16</td>
<td>0</td>
<td>558</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PAPG_A</td>
<td>16/16</td>
<td>18.7</td>
<td>553</td>
<td>9.75E-02</td>
<td>6.14E-02</td>
<td>3.66E-07</td>
</tr>
<tr>
<td></td>
<td>PAPG_C</td>
<td>16/16</td>
<td>18.8</td>
<td>886</td>
<td>9.72E-02</td>
<td>1.65E-03</td>
<td>4.51E-07</td>
</tr>
<tr>
<td>20, 2400</td>
<td>Mosek_Reg</td>
<td>24/24</td>
<td>0</td>
<td>2155</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PAPG_A</td>
<td>24/24</td>
<td>128.5</td>
<td>797</td>
<td>9.97E-02</td>
<td>5.79E-02</td>
<td>1.29E-07</td>
</tr>
<tr>
<td></td>
<td>PAPG_C</td>
<td>24/24</td>
<td>133.8</td>
<td>1347</td>
<td>1.00E-01</td>
<td>2.92E-03</td>
<td>3.59E-08</td>
</tr>
</tbody>
</table>

of P-APG over running IPM alone on (2.7) become more and more evident as the number of observations, $N$, increases.

### 2.4.2 Comparison with ASM and ADMM

In this section, we compare P-APG with ASM and multi-block ADMM. It is worth noting that multi-block ADMM solves the original problem (2.3), while P-APG and the active set method (ASM) solve the regularized problem (2.7). All algorithms are
Table 2.5: Comparison with test function \( \exp(p^T x) \) for \( n = 80 \)

<table>
<thead>
<tr>
<th>n, N</th>
<th>Algorithm</th>
<th>Cores/RAM</th>
<th>Preprocess</th>
<th>Walltime</th>
<th>Infeasibility</th>
<th>SubOpt_Reg</th>
<th>DualGap</th>
</tr>
</thead>
<tbody>
<tr>
<td>80, 200</td>
<td>Mosek_Reg</td>
<td>2/2</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PAPG_A</td>
<td>2/2</td>
<td>0.4</td>
<td>53</td>
<td>9.21E-02</td>
<td>6.68E-03</td>
<td>4.25E-07</td>
</tr>
<tr>
<td></td>
<td>PAPG_C</td>
<td>2/2</td>
<td>0.4</td>
<td>34</td>
<td>9.27E-02</td>
<td>1.60E-03</td>
<td>1.01E-07</td>
</tr>
<tr>
<td>80, 400</td>
<td>Mosek_Reg</td>
<td>4/4</td>
<td>0</td>
<td>21</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PAPG_A</td>
<td>4/4</td>
<td>2.6</td>
<td>209</td>
<td>9.39E-02</td>
<td>3.00E-03</td>
<td>1.15E-07</td>
</tr>
<tr>
<td></td>
<td>PAPG_C</td>
<td>4/4</td>
<td>2.6</td>
<td>183</td>
<td>9.63E-02</td>
<td>2.92E-03</td>
<td>1.12E-07</td>
</tr>
<tr>
<td>80, 800</td>
<td>Mosek_Reg</td>
<td>8/8</td>
<td>0</td>
<td>143</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PAPG_A</td>
<td>8/8</td>
<td>12.8</td>
<td>299</td>
<td>9.12E-02</td>
<td>9.44E-04</td>
<td>1.90E-08</td>
</tr>
<tr>
<td></td>
<td>PAPG_C</td>
<td>8/8</td>
<td>13.0</td>
<td>272</td>
<td>9.87E-02</td>
<td>1.65E-04</td>
<td>5.01E-10</td>
</tr>
<tr>
<td>80, 1600</td>
<td>Mosek_Reg</td>
<td>16/16</td>
<td>0</td>
<td>687</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PAPG_A</td>
<td>16/16</td>
<td>55.3</td>
<td>465</td>
<td>9.72E-02</td>
<td>3.07E-03</td>
<td>3.59E-08</td>
</tr>
<tr>
<td></td>
<td>PAPG_C</td>
<td>16/16</td>
<td>54.7</td>
<td>753</td>
<td>9.95E-02</td>
<td>1.28E-04</td>
<td>1.53E-09</td>
</tr>
<tr>
<td>80, 2400</td>
<td>Mosek_Reg</td>
<td>24/24</td>
<td>0</td>
<td>2789</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>PAPG_A</td>
<td>24/24</td>
<td>138.2</td>
<td>774</td>
<td>9.69E-02</td>
<td>7.82E-03</td>
<td>6.39E-08</td>
</tr>
<tr>
<td></td>
<td>PAPG_C</td>
<td>24/24</td>
<td>139.3</td>
<td>1379</td>
<td>9.94E-02</td>
<td>2.23E-03</td>
<td>2.01E-08</td>
</tr>
</tbody>
</table>

terminated whenever they compute an iterate, \((y, \xi)\), satisfying the following stopping criteria: \(\|y - y^*\|/\sqrt{N} \leq 5e-3\) and \(\|(A_1 y + A_2 \xi)\|/\sqrt{N^2 - N} \leq 1e-1\), where the first one is the relative sub-optimality with respect to the original problem in (2.3) and the second one is the normalized infeasibility. The initial point for ASM is set by using (2.57), where \(\alpha = 1/N\). This choice of \(\alpha\) works consistently well based on our test.

Figure 2.7: P-APG vs ASM \((n, N) = (10, 100)\): (left) Relative Suboptimality, (right) Zoom-in for P-APG method

Experiments comparing convergence behaviors of P-APG and ASM on a small-size problem were carried out for \((n, N) = (10, 100)\). As shown in Fig. 2.7, active set algorithm spends quite long in a warm-up phase before making noticeable progress in terms of function value; and this behavior becomes more and more apparent as the size of the
Figure 2.8: P-APG vs ASM \((n, N) = (10, 100)\): (left) Number of Active Constraints for ASM, (right) Normalized infeasibility for P-APG

Figure 2.9: Wall-time Ratio between ADMM and P-APG

In Tables 2.6 and 2.7 besides the statistics reported in Section 2.4.1 we also report
Accuracy which measures the solution quality with respect to the original problem in (2.3). In particular, given an approximate solution \( \hat{y} \), obtained by solving either (2.3) or (2.7) depending on the algorithm chosen, Accuracy is computed as \( \| \hat{y} - y^* \| / \sqrt{N} \).

As in Section [2.4.1], Preprocess for P-APG method denotes the wall-clock time used for computing the maximum singular value for matrix \( A_4 \), and Preprocess for ADMM accounts for \( \bar{\Delta}_j \) computation for all \( j \in \mathcal{N} \) as shown in Figure 5.5. The performance comparison is shown in Table 2.6 and Table 2.7, which clearly display that as the number of observations \( N \) increases, ASM starts struggling to finish the job within 2 hours beyond \( N=800 \), and the gap between P-APG and ADMM closes rapidly, and eventually P-APG outperforms ADMM at \( N = 2400 \), which is also demonstrated in Fig. 2.9.

Table 2.6: Comparison of PAPG and other methods \( \frac{1}{2}x^TQx \)

<table>
<thead>
<tr>
<th>n, N</th>
<th>Algorithms</th>
<th>Cores/RAM</th>
<th>Preprocess</th>
<th>Wall-time</th>
<th>Infeasibility</th>
<th>Accuracy</th>
<th>SubOpt_Reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>80, 200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mosek</td>
<td>2/2</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Mosek_Reg</td>
<td>2/2</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>1.30E-03</td>
<td>0</td>
<td>–</td>
</tr>
<tr>
<td>ADMM</td>
<td>2/2</td>
<td>3</td>
<td>4</td>
<td>5.48E-04</td>
<td>4.32E-03</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>ASM</td>
<td>2/2</td>
<td>0</td>
<td>111</td>
<td>0</td>
<td>1.31E-03</td>
<td>9.33E-06</td>
<td>–</td>
</tr>
<tr>
<td>PAPG_A</td>
<td>2/2</td>
<td>1</td>
<td>105</td>
<td>7.29E-02</td>
<td>1.31E-03</td>
<td>6.54E-05</td>
<td>–</td>
</tr>
<tr>
<td>PAPG_C</td>
<td>2/2</td>
<td>0</td>
<td>43</td>
<td>8.43E-02</td>
<td>1.31E-03</td>
<td>3.23E-06</td>
<td>–</td>
</tr>
<tr>
<td>80, 400</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mosek</td>
<td>4/4</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Mosek_Reg</td>
<td>4/4</td>
<td>0</td>
<td>36</td>
<td>0</td>
<td>2.60E-03</td>
<td>0</td>
<td>–</td>
</tr>
<tr>
<td>ADMM</td>
<td>4/4</td>
<td>11</td>
<td>20</td>
<td>4.46E-04</td>
<td>4.71E-03</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>ASM</td>
<td>4/4</td>
<td>0</td>
<td>665</td>
<td>0</td>
<td>2.57E-03</td>
<td>5.10E-07</td>
<td>–</td>
</tr>
<tr>
<td>PAPG_A</td>
<td>4/4</td>
<td>3</td>
<td>319</td>
<td>9.38E-02</td>
<td>2.56E-03</td>
<td>1.17E-03</td>
<td>–</td>
</tr>
<tr>
<td>PAPG_C</td>
<td>4/4</td>
<td>3</td>
<td>176</td>
<td>9.79E-02</td>
<td>2.56E-03</td>
<td>5.72E-04</td>
<td>–</td>
</tr>
<tr>
<td>80, 800</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mosek</td>
<td>8/8</td>
<td>0</td>
<td>109</td>
<td>0</td>
<td>0</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Mosek_Reg</td>
<td>8/8</td>
<td>0</td>
<td>188</td>
<td>0</td>
<td>3.80E-03</td>
<td>0</td>
<td>–</td>
</tr>
<tr>
<td>ADMM</td>
<td>8/8</td>
<td>50</td>
<td>121</td>
<td>3.78E-04</td>
<td>4.98E-03</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>ASM</td>
<td>8/8</td>
<td>0</td>
<td>7006</td>
<td>0</td>
<td>3.82E-03</td>
<td>1.47E-05</td>
<td>–</td>
</tr>
<tr>
<td>PAPG_A</td>
<td>8/8</td>
<td>12</td>
<td>391</td>
<td>7.55E-02</td>
<td>3.81E-03</td>
<td>3.73E-04</td>
<td>–</td>
</tr>
<tr>
<td>PAPG_C</td>
<td>8/8</td>
<td>12</td>
<td>281</td>
<td>9.89E-02</td>
<td>3.81E-03</td>
<td>2.17E-04</td>
<td>–</td>
</tr>
<tr>
<td>80, 1600</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mosek</td>
<td>16/16</td>
<td>0</td>
<td>544</td>
<td>0</td>
<td>0</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Mosek_Reg</td>
<td>16/16</td>
<td>0</td>
<td>918</td>
<td>0</td>
<td>2.40E-03</td>
<td>0</td>
<td>–</td>
</tr>
<tr>
<td>ADMM</td>
<td>16/16</td>
<td>221</td>
<td>537</td>
<td>2.93E-04</td>
<td>4.96E-03</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>ASM</td>
<td>16/16</td>
<td>0</td>
<td>&gt;2 hours</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>–</td>
</tr>
<tr>
<td>PAPG_A</td>
<td>16/16</td>
<td>50</td>
<td>844</td>
<td>9.68E-02</td>
<td>3.16E-03</td>
<td>2.22E-03</td>
<td>–</td>
</tr>
<tr>
<td>PAPG_C</td>
<td>16/16</td>
<td>50</td>
<td>802</td>
<td>9.96E-02</td>
<td>3.17E-03</td>
<td>2.06E-03</td>
<td>–</td>
</tr>
<tr>
<td>80, 2400</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mosek</td>
<td>24/24</td>
<td>0</td>
<td>2537</td>
<td>0</td>
<td>0</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Mosek_Reg</td>
<td>24/24</td>
<td>0</td>
<td>4576</td>
<td>0</td>
<td>3.40E-03</td>
<td>0</td>
<td>–</td>
</tr>
<tr>
<td>ADMM</td>
<td>24/24</td>
<td>678</td>
<td>2332</td>
<td>2.16E-04</td>
<td>4.99E-03</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>ASM</td>
<td>24/24</td>
<td>0</td>
<td>&gt;2 hours</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>–</td>
</tr>
<tr>
<td>PAPG_A</td>
<td>24/24</td>
<td>155</td>
<td>1040</td>
<td>9.69E-02</td>
<td>3.26E-03</td>
<td>9.24E-04</td>
<td>–</td>
</tr>
<tr>
<td>PAPG_C</td>
<td>24/24</td>
<td>155</td>
<td>1184</td>
<td>9.88E-02</td>
<td>3.26E-03</td>
<td>1.30E-03</td>
<td>–</td>
</tr>
</tbody>
</table>
Table 2.7: Comparison of PAPG and other methods $\exp(p^T x)$

<table>
<thead>
<tr>
<th>n, N</th>
<th>Algorithms</th>
<th>Cores/RAM</th>
<th>Preprocess</th>
<th>Wall-time</th>
<th>Infeasibility</th>
<th>Accuracy</th>
<th>SubOpt_Reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>80, 200</td>
<td>Mosek</td>
<td>2/2</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>Mosek_Reg</td>
<td>2/2</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>4.41E-04</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>ADMM</td>
<td>2/2</td>
<td>3</td>
<td>4</td>
<td>6.41E-04</td>
<td>4.96E-03</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>ASM</td>
<td>2/2</td>
<td>0</td>
<td>155</td>
<td>0</td>
<td>4.41E-04</td>
<td>2.51E-05</td>
</tr>
<tr>
<td></td>
<td>PAPG_A</td>
<td>2/2</td>
<td>0</td>
<td>64</td>
<td>9.21E-02</td>
<td>4.43E-04</td>
<td>6.68E-03</td>
</tr>
<tr>
<td></td>
<td>PAPG_C</td>
<td>2/2</td>
<td>0</td>
<td>41</td>
<td>9.27E-02</td>
<td>4.42E-04</td>
<td>1.60E-03</td>
</tr>
<tr>
<td>80, 400</td>
<td>Mosek</td>
<td>4/4</td>
<td>0</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>Mosek_Reg</td>
<td>4/4</td>
<td>0</td>
<td>33</td>
<td>0</td>
<td>9.28E-04</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>ADMM</td>
<td>4/4</td>
<td>11</td>
<td>18</td>
<td>5.51E-04</td>
<td>4.95E-03</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>ASM</td>
<td>4/4</td>
<td>0</td>
<td>827</td>
<td>0</td>
<td>9.29E-04</td>
<td>2.90E-05</td>
</tr>
<tr>
<td></td>
<td>PAPG_A</td>
<td>4/4</td>
<td>3</td>
<td>167</td>
<td>9.39E-02</td>
<td>9.24E-04</td>
<td>3.00E-03</td>
</tr>
<tr>
<td></td>
<td>PAPG_C</td>
<td>4/4</td>
<td>3</td>
<td>143</td>
<td>9.63E-02</td>
<td>9.24E-04</td>
<td>2.92E-03</td>
</tr>
<tr>
<td>80, 800</td>
<td>Mosek</td>
<td>8/8</td>
<td>0</td>
<td>136</td>
<td>0</td>
<td>0</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>Mosek_Reg</td>
<td>8/8</td>
<td>0</td>
<td>175</td>
<td>0</td>
<td>9.74E-04</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>ADMM</td>
<td>8/8</td>
<td>50</td>
<td>97</td>
<td>3.50E-04</td>
<td>4.08E-03</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>ASM</td>
<td>8/8</td>
<td>0</td>
<td>&gt; 2 hours</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>PAPG_A</td>
<td>8/8</td>
<td>13</td>
<td>236</td>
<td>9.12E-02</td>
<td>9.72E-04</td>
<td>9.44E-04</td>
</tr>
<tr>
<td></td>
<td>PAPG_C</td>
<td>8/8</td>
<td>13</td>
<td>209</td>
<td>9.87E-02</td>
<td>9.69E-04</td>
<td>1.65E-04</td>
</tr>
<tr>
<td>80, 1600</td>
<td>Mosek</td>
<td>16/16</td>
<td>0</td>
<td>843</td>
<td>0</td>
<td>0</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>Mosek_Reg</td>
<td>16/16</td>
<td>0</td>
<td>1107</td>
<td>0</td>
<td>1.40E-03</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>ADMM</td>
<td>16/16</td>
<td>266</td>
<td>787</td>
<td>2.66E-04</td>
<td>4.98E-03</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>ASM</td>
<td>16/16</td>
<td>0</td>
<td>&gt; 2 hours</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>PAPG_A</td>
<td>16/16</td>
<td>62</td>
<td>461</td>
<td>9.72E-02</td>
<td>1.45E-03</td>
<td>3.07E-03</td>
</tr>
<tr>
<td></td>
<td>PAPG_C</td>
<td>16/16</td>
<td>59</td>
<td>742</td>
<td>9.95E-02</td>
<td>1.44E-03</td>
<td>1.28E-04</td>
</tr>
<tr>
<td>80, 2400</td>
<td>Mosek</td>
<td>24/24</td>
<td>0</td>
<td>2522</td>
<td>0</td>
<td>0</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>Mosek_Reg</td>
<td>24/24</td>
<td>0</td>
<td>3365</td>
<td>0</td>
<td>1.80E-03</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>ADMM</td>
<td>24/24</td>
<td>846</td>
<td>2486</td>
<td>2.12E-04</td>
<td>4.95E-03</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>ASM</td>
<td>24/24</td>
<td>0</td>
<td>&gt; 2 hours</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>PAPG_A</td>
<td>24/24</td>
<td>138</td>
<td>774</td>
<td>9.69E-02</td>
<td>1.82E-03</td>
<td>7.82E-03</td>
</tr>
<tr>
<td></td>
<td>PAPG_C</td>
<td>24/24</td>
<td>139</td>
<td>1379</td>
<td>9.94E-02</td>
<td>1.81E-03</td>
<td>2.23E-03</td>
</tr>
</tbody>
</table>

2.5 Conclusion

In this chapter, we proposed P-APG method to efficiently compute the least squares estimator for large-scale convex regression problems. By relaxing constraints partially, we obtained the separability on the corresponding Lagrangian dual problem. Using Tikhonov regularization, we ensured the feasibility of iterates in the limit, and we provided error bounds on 1) the distance between the inexact solution to the regularized problem and the optimal solution to the original problem, 2) the constraint violation of the regularized solution. We also proposed a continuation scheme which directly solves the (unregularized) original problem (without any negative impact on the iteration complexity), and it does not require a parameter input depending on the desired solution tolerance $\epsilon$. The comparison in the numerical section demonstrates the efficiency of
P-APG method on memory usage compared to IPM. Furthermore, our numerical tests show that P-APG becomes the method of choice for large $N$ values when compared to ASM and ADMM.
Chapter 3  
Distributed Consensus Optimization

3.1 Introduction and Motivation

Distributed optimization and control in collaborative multi-agent systems have attracted much attention recently due to the rise of large-scale networks such as mobile cellular, electricity, traffic networks and internet, etc \[39,40\]. Due to sheer size of these networks, more than often, it would not be possible to set up a centralized storage for data and information. In other cases, agents (also referred as nodes or processors in other literature) may not want to disclose or share their local private information due to privacy concerns or regulations. Consequently, data are stored in a distributed fashion with each agent locally storing a fraction of it. Furthermore, usually these multi-agent systems are likely to fail more when they grow larger and larger in size – even if each component has only a slight chance of failing. Note that the possible failures on nodes and linkages in these networks lead to topology changes within the network. The goal is to design such a framework that it collaboratively optimizes a global objective by only using the local information, and highly scalable to the size of the network. It is important to emphasize that there is an active research on distributed optimization and control over dynamic (time-varying) communication networks; that said, in this dissertation we focus on the distributed optimization problems over static communication networks and consider both synchronous and asynchronous agents.

Some good example of such distributed optimization problems are stated in an expository article \[40\] by Nedic. For the sake of coherence and convenience to the reader, we summarize one such problem here. In machine learning, one of most common tasks is classification, which deals with classifying un-labeled observations using models that was trained with a number of labeled sample data, such as email spam detector and
document classifier. Support vector machine (SVM) is a suitable common technique for the task. It not only finds a separating hyperplane which divides the labeled sample into predefined number of clusters, but also does identifies one such hyperplane that it separates the data with the maximal margin, $H_3$ as shown in Fig 3.1.

![Three Different Classifiers](image)

Figure 3.1: Three Different Classifiers

Now consider a large network with labeled samples stored at each local database, and each agent $i$ has a set of observations $\chi_i = \{(x^j_i, y^j_i)\}_{j=1}^{m_i}$, where $m_i$ is often large, $x^j_i \in \mathbb{R}^n$ is the data point, and $y^j_i \in \{0, 1\}$ is a scaler, the corresponding label. Thus, each agent has their own objective function $f_i$ which is given by,

$$f_i(w) = \sum_{j=1}^{m_i} \max \left\{ 0, 1 - y^j_i (w^T x^j_i) \right\} + \frac{\lambda}{2} \|w\|_2^2$$

where the first term is the hinge loss function, also known as soft-margin formulation, the second term is ridge regularization to overcome the issue of overfitting model, and $\lambda$ is the regularization parameter that balances the trade-off between fidelity and generalization properties of the model, and it usually can be determined by cross-validation. The goal here is to design an algorithm to solve

$$\min_{w} \sum_{i \in \mathcal{N}} f_i(w)$$

in order to find the hyperplane separating all the data to the best extent by collaborative work among all the agents without any of the agents sharing its local data $\chi_i$ with the
other nodes $j \in \mathcal{N} \setminus \{i\}$, where $\mathcal{N}$ denotes the set of agents.

One important approach to address distributed optimization is to use consensus algorithm. That is each agent maintains its own copy decision variable and agrees on a common value at the end of the process by only communicating to its neighbors. Specifically, in the network $\mathcal{G}(\mathcal{N}, \mathcal{E})$, agent $i$ starts from its initial decision variable $x_i^0$, and updates using a weighted average of its neighbors given by,

$$x_i^{k+1} = \sum_{(i,j) \in \mathcal{E}} a_{ij} x_j^k,$$

where $a_{ij}$ is a nonnegative weight that agent $i$ assigns to its neighbors’ local decisions. It is formally known as consensus protocol \[41\]–\[43\], and has been shown to convergence the common vector $\hat{x}$ asymptotically. Using this technique allowed researchers to extend the existing optimization algorithms designed for conventional centralized optimization to run in a distributed way and make them highly adaptable to other problem settings. First-order methods combined with consensus protocol naturally suit this class of problems, due to the fact that they only use gradient information, have simple iterations, and they have desirable computational complexity when low-to-medium accuracy solutions are required. In this dissertation, using the consensus framework, we propose two distinct first-order algorithms to address decentralized optimization problems. We will take a closer look at the proposed methodologies in the next two chapters. Before going into the details, we briefly explain the following four major areas of extension for distributed optimization:

**Global and side constraints:** We have only mentioned unconstraint problems till now. However, in a large complex system, we may encounter situations that agents have their own constraints on resource, distribution preference, and so on. Dealing with constraints in centralized problems is straightforward, while it poses challenges in distributed optimization. Additionally, upper-stream decision-maker could also impose global constraints on the network \[39\]–\[47\].

**Asynchronous implementation:** Generally speaking, when the algorithm works under a global clock in the synchronous setting, agents have to sync up on progress during the process of convergence. In other words, the algorithm advances to the next iteration only if all the agents complete their computation at the current iteration. In this case, the overall speed would depend on the slowest agent in terms of computational capability. It is likely to have different types or generation of computational infrastructures within
the network in real life – showing highly nonhomogeneous computational speeds. In this case, it is more desirable to have an algorithm that can be implemented in asynchronous setting. That is a random agent wakes up, computes the updates, and broadcast its new information to its neighbors, while the others remain “asleep” (do not compute their updates) [44,45,48–51].

Noisy gradient computation: In some cases, the objective function might not be explicitly available to the agents, but instead one can get an estimation of its gradient. Under the assumption that the estimator is unbiased and has bounded second moment, one can design a distributed algorithm with special diminishing stepsize sequence, see [52–54].

Time varying topology: As mentioned previously, it is expected to have some link failures in a large network. It results to a dynamic network topology as the links between agents disappear and appear at different time periods. The idea is that as long as the network is connected in the longrun, we can expect consensus among the agents. There are distributed algorithms with provable convergence guarantees under such dynamic environment [39,41,55,56].

We will discuss more in detail on the first three areas, and focus on the scope of static networks in this dissertation. We will leave the dynamic network setup as one direction of future research. In the next section, we normally describe the problem statement.

### 3.2 Problem Statement

Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ denote a connected undirected graph of $N$ computing nodes where nodes $i$ and $j$ can communicate information only if $(i, j) \in \mathcal{E}$. Each node $i \in \mathcal{N} := \{1, \ldots, N\}$ has a private (local) cost function

$$F_i(x) := \rho_i(x) + \gamma_i(x),$$  \hspace{1cm} (3.1)

where $\rho_i : \mathbb{R}^n \to \mathbb{R}$ is a possibly non-smooth convex function, and $\gamma_i : \mathbb{R}^n \to \mathbb{R}$ is a smooth convex function. We assume that the proximal map

$$\text{prox}_{\rho_i}(x) := \arg \min_{y \in \mathbb{R}^n} \left\{ \rho_i(y) + \frac{1}{2} \|y - x\|^2 \right\}$$  \hspace{1cm} (3.2)

is efficiently computable for $i \in \mathcal{N}$.  

49
Consider the following convex problem and the underlying graph $G$:

\[
F^* := \min_{x \in \mathbb{R}^n} F(x) := \sum_{i \in \mathcal{N}} F_i(x). \tag{3.3}
\]

Clearly, (3.3) can also be solved in a “centralized” fashion, which is to communicate all the private functions $F_i$ to a central node, and solve the overall problem at that node. However, such an approach can be very expensive both from communication and computation perspectives. In particular, suppose $(A_i, b_i) \in \mathbb{R}^{m \times (n+1)}$ and $F_i(x) = \|A_i x - b_i\|_2^2 + \lambda \|x\|_1$ for some given trade-off parameter $\lambda > 0$ for $i \in \mathcal{N}$ such that $m \ll n$ and $N \gg 1$. Hence, (3.3) is a very large scale LASSO problem with distributed data. To solve (3.3) in a centralized fashion, the data $\{(A_i, b_i) : i \in \mathcal{N}\}$ needs to be communicated to and consolidated at the central node. This can be prohibitively expensive or not even possible, and may also violate privacy constraints – in case some node $i$ does not want to reveal the details of its private data. Furthermore, it requires that the central node has large enough memory and storage to be able to accommodate all the data.

On the other hand, at the expense of slower convergence, one can completely do away with a central node, and seek for consensus among all the nodes on one optimal decision using “local” decisions communicated by the neighboring nodes. That is each node $i \in \mathcal{N}$ in the network works with their own local decision variable $x_i$, and communicate $x_i$ and other related information with its neighborhood to reach consensus on the decision, i.e. $x_i = x_j, \ \forall i, j \in \mathcal{N}$. From computational perspective, for certain cases, computing partial gradients locally can be more computationally efficient when compared to computing the entire gradient at a central node. With these considerations in mind, we propose decentralized algorithms that can compute solutions to (3.3) using only local computations without explicitly requiring the nodes to communicate the functions $\{F_i : i \in \mathcal{N}\}$; thereby, circumventing all privacy, communication and memory issues. To facilitate the design of such decentralized algorithms, we take advantage of the fact that graph $G$ is connected, and reformulate (3.3) as

\[
\min_{x_i \in \mathbb{R}^n, i \in \mathcal{N}} \left\{ \sum_{i=1}^{N} F_i(x_i) : x_i = x_j, \ \forall (i, j) \in \mathcal{E} \right\}. \tag{3.4}
\]

This computational setting, i.e., decentralized consensus optimization, appears as a generic model for various applications in signal processing, e.g., [57–61], machine learning, e.g., [62–64] and statistical inference, e.g., [65,66].
### 3.3 Related work

A number of different distributed optimization algorithms have been proposed to solve (3.3) — see Table 3.1 that displays some recent work. Duchi et al. [67] proposed a dual averaging algorithm to solve (3.3) in a distributed fashion over \( G \) when each \( F_i \) is convex. This algorithm computes \( \epsilon \)-optimal solution in \( \mathcal{O}(1/\epsilon^2) \) iterations; however, they do not provide any guarantees on the consensus violation \( \max_{(i,j) \in \mathcal{E}} \| \bar{x}_i - \bar{x}_j \|_2 \leq \epsilon \).

Nedić and Ozdaglar [68] developed a subgradient method with constant step size \( c > 0 \) for distributed minimization of (3.3) where the network topology is time-varying. Setting the subgradient stepsize \( c = \mathcal{O}(\epsilon) \) in their method guarantees to compute a solution \( \bar{x} = [\bar{x}_i]_{i \in \mathcal{N}} \) such that its consensus violation \( \max_{(i,j) \in \mathcal{E}} \| \bar{x}_i - \bar{x}_j \|_2 \leq \epsilon \) within \( \mathcal{O}(1) \) iterations; and its suboptimality is bounded from above as \( \sum_{i \in \mathcal{N}} F_i(\bar{x}_i) - F^* \leq \epsilon \) within \( \mathcal{O}(1/\epsilon^2) \) iterations; however, since the step size is constant, neither suboptimality nor consensus errors are guaranteed to decrease further. Although these algorithms are for more general problems and assume mere convexity on each \( F_i \), this generality comes at the cost of \( \mathcal{O}(1/\epsilon^2) \) complexity bounds, and they also tend to be very slow in practice. On the other extreme, under much stronger conditions: assuming each \( F_i \)

#### Table 3.1: Comparison of our method with the previous work

<table>
<thead>
<tr>
<th>Reference</th>
<th>Assumption on ( F_i )</th>
<th>operation / iter</th>
<th>iter # for ( \epsilon )-feas.</th>
<th>iter # for ( \epsilon )-opt.</th>
<th>comm. steps ( \epsilon )-opt.</th>
<th>Single loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duchi et al. (2012) [67]</td>
<td>composite, Lipschitz cont.</td>
<td>prox subgrad.</td>
<td>( \mathcal{O}(1/\epsilon^2) )</td>
<td>( \mathcal{O}(1/\epsilon^2) )</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Nedic &amp; Ozdaglar (2009) [68]</td>
<td>convex</td>
<td>subgrad.</td>
<td>( \mathcal{O}(1) )</td>
<td>( \mathcal{O}(1/\epsilon^2) )</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Wei &amp; Ozdaglar (2012) [69]</td>
<td>strictly convex</td>
<td>prox</td>
<td>unknown</td>
<td>( \mathcal{O}(1/\epsilon) )</td>
<td>( \mathcal{O}(1/\epsilon) )</td>
<td>yes</td>
</tr>
<tr>
<td>Makhdoumi &amp; Ozdaglar (2014) [70]</td>
<td>convex</td>
<td>prox</td>
<td>( \mathcal{O}(1/\epsilon) )</td>
<td>( \mathcal{O}(1/\epsilon) )</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Wei &amp; Ozdaglar (2013) [71]</td>
<td>convex</td>
<td>prox</td>
<td>( \mathcal{O}(1/\epsilon) )</td>
<td>( \mathcal{O}(1/\epsilon) )</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>Jakovetic et al. (2014) <a href="D-NC">72</a></td>
<td>smooth convex bound. ( \nabla \Phi_i )</td>
<td>prox, ( \nabla \Phi_i )</td>
<td>( \mathcal{O}(1/\sqrt{\epsilon}) )</td>
<td>( \mathcal{O}(1/\sqrt{\epsilon}) )</td>
<td>( \mathcal{O}(\log(1/\epsilon)) )</td>
<td>no</td>
</tr>
<tr>
<td>Chen &amp; Ozdaglar (2012) [73]</td>
<td>composite convex ( F_i = \rho + \gamma_i ) bounded ( \nabla \Phi_i )</td>
<td>prox, ( \nabla \Phi_i )</td>
<td>( \mathcal{O}(1/\sqrt{\epsilon}) )</td>
<td>( \mathcal{O}(1/\sqrt{\epsilon}) )</td>
<td>( \mathcal{O}(1/\epsilon) )</td>
<td>no</td>
</tr>
<tr>
<td>Shi et al. (2015) [74] (PG-EXTRA)</td>
<td>composite convex ( F_i = \rho_i + \gamma_i )</td>
<td>prox, ( \nabla \Phi_i )</td>
<td>( \mathcal{O}(1/\epsilon^2) )</td>
<td>( \mathcal{O}(1/\epsilon^2) )</td>
<td>( \mathcal{O}(1/\epsilon^2) )</td>
<td>yes</td>
</tr>
<tr>
<td>Our work (DFAL)</td>
<td>composite convex ( F_i = \rho_i + \gamma_i )</td>
<td>prox, ( \nabla \Phi_i )</td>
<td>( \mathcal{O}(1/\epsilon) )</td>
<td>( \mathcal{O}(1/\epsilon) )</td>
<td>( \mathcal{O}(1/\epsilon) )</td>
<td>no</td>
</tr>
<tr>
<td>Our work (DPGA-I, DPGA-II)</td>
<td>composite convex ( F_i = \rho_i + \gamma_i )</td>
<td>prox, ( \nabla \Phi_i )</td>
<td>( \mathcal{O}(1/\epsilon) )</td>
<td>( \mathcal{O}(1/\epsilon) )</td>
<td>( \mathcal{O}(1/\epsilon) )</td>
<td>yes</td>
</tr>
</tbody>
</table>
is smooth and has bounded gradients, Jakovetic et al. [72] developed a fast distributed gradient method D-NC (the distributed Nesterov gradient with consensus iterations) with $O(\log(1/\epsilon)/\sqrt{\epsilon})$ convergence rate in communication rounds. For the quadratic loss, which is one of the most commonly used loss functions, bounded gradient assumption does not hold. In terms of distributed applicability, D-NC requires all the nodes $N$ to agree on a doubly stochastic weight matrix $W \in \mathbb{R}^{|N| \times |N|}$; it also assumes that the second largest eigenvalue of $W \in \mathbb{R}^{|N| \times |N|}$ is known globally among all the nodes – this is not attainable for very large scale fully distributed networks. D-NC is a two-loop algorithm: for each outer loop $k$, each node computes their gradients once, and it is followed by $O(\log(k))$ communication rounds. In the rest of this section, we briefly discuss those algorithms that balance the trade-off between the iteration complexity and the required assumptions on $\{F_i\}_{i \in N}$.

Wei & Ozdaglar [69,71], and recently Makhdoumi & Ozdaglar [70] proposed distributed ADMM algorithms that can compute an $\epsilon$-optimal and $\epsilon$-feasible solution in $O(1/\epsilon)$ prox map evaluations for each $F_i$. These algorithms have superior iteration complexity compared to the subgradient methods discussed above. That said, there are many practical problems where one can compute the prox map for $\rho_i$ efficiently; but, computing the prox map for $F_i = \rho_i + \gamma_i$ is not easy - see Section 4.3 for an example. One can overcome this limitation of ADMM by locally splitting variables, i.e., setting $F_i(x_i, y_i) := \rho_i(x_i) + \gamma_i(y_i)$, and adding a constraint $x_i = y_i$ in (3.4) – we call this implementation of ADMM as SADMM. However, this approach more than doubles local memory requirement; and in order for ADMM to be efficient, the prox maps for both $\rho_i$ and $\gamma_i$ still must be simple.

When node functions $F_i$ are composite convex, i.e., $F_i = \rho + \gamma_i$, assuming that the non-smooth term $\rho$ is the same at all nodes, and $\nabla \gamma_i$ is bounded for all $i \in N$, Chen & Ozdaglar [73] proposed an inexact proximal-gradient method, which exploits the function structure, for distributed minimization of (3.3) over a time-varying network topology. Their method also consists of two loops; it can compute $\epsilon$-feasible and $\epsilon$-optimal solution in $T = O(1/\sqrt{\epsilon})$ iterations which require $k$ communication rounds with neighbors during the $k$-th iteration for each $1 \leq k \leq T$ – hence, leading to $\sum_{k=1}^{T} k = O(\epsilon^{-1})$ communications per node in total. Note that there are also many practical problems where nodes in the network have different non-smooth components in their objective and/or have different preference when choosing non-smooth regularizers. In contrast, the methods proposed in Chapter [4] and Chapter [5] allow node specific non-smooth functions $\rho_i$, do not assume bounded $\nabla \gamma_i$ for any $i \in N$, and are still able to compute an $\epsilon$-optimal
\( \epsilon \)-feasible solution in \( \mathcal{O}(\epsilon^{-1}) \) iterations, where each iteration requires computing \( \text{prox}_{\rho_i} \) and \( \nabla \gamma_i \) for \( i \in \mathcal{N} \), and one or two communication rounds among the neighbors – hence, \( \mathcal{O}(\epsilon^{-1}) \) communications per node in total.

The DFAL algorithm in Chapter 4, DPGA algorithms in Chapter 5, and the one proposed by Shi et al. in [74] are all proximal gradient based distributed algorithms that can solve (3.3) over a static connected network when \( F_i = \rho_i + \gamma_i \) as in (3.1); hence, all these algorithms can handle node specific non-smooth terms. In [74], the proximal gradient exact first-order algorithm (PG-EXTRA) is proposed to solve composite optimization in a distributed fashion – PG-EXTRA is an extension of the algorithm EXTRA [75] to handle the non-smooth terms \( \{\rho_i\}_{i \in \mathcal{N}} \). They showed \( \mathcal{O}(1/t) \) convergence for the ergodic average of squared residuals for the consensus and KKT violations. That said, we consider their rate result as \( \mathcal{O}(1/\sqrt{t}) \) because their result on consensus violation, 
\[
\frac{1}{t} \sum_{k=1}^{t} \|Ux^k\|^2 = \mathcal{O}(1/t),
\]
only guarantees \( \|U\bar{x}^t\| = \mathcal{O}(1/\sqrt{t}) \), where \( x^k = [x^k_i]_{i \in \mathcal{N}} \in \mathbb{R}^{n|\mathcal{N}|} \) denotes PG-EXTRA iterate at iteration \( k \geq 1 \), \( \bar{x}^t := \sum_{k=1}^{t} x^k / t \) denotes their ergodic average, and \( U \in \mathbb{R}^{n|\mathcal{N}| \times n|\mathcal{N}|} \) such that null space of \( U \) only contains \( 1 \in \mathbb{R}^{n|\mathcal{N}|} \), i.e., \( Ux = 0 \) implies \( x_1 = \ldots = x_N \). PG-EXTRA is a node-based distributed algorithm, and each node \( i \in \mathcal{N} \) stores four different copies of local variable \( x_i \) at each iteration \( k \), say \( x_i^{k+3/2}, x_i^{k+1}, x_i^{k+1/2} \) and \( x_i^k \), and requires two rounds of local communications to be able to compute next iterates – one can reduce the information exchange to one round per iteration at the expense of increasing the storage at node \( i \) from \( 4n \) to \( (4 + d_i)n \). In terms of applicability, PG-EXTRA requires all the nodes \( \mathcal{N} \) to agree on two symmetric mixing matrices: \( W, \tilde{W} \in \mathbb{R}^{n|\mathcal{N}| \times n|\mathcal{N}|} \) such that \( \tilde{W} \succ 0 \) and \( \frac{W + I}{2} \succeq \tilde{W} \succeq W \) – see Assumption 1 in [74]. The gradient stepsize, \( c \), is the same for all nodes, and should satisfy \( c < \sigma_{\min}(\tilde{W})/L_{\max} \). If \( W, \tilde{W} \) are chosen such that \( W \succeq 0 \) (this requires coordination among all the agents), and \( \tilde{W} = \frac{W + I}{2} \), then \( c \) can be chosen \( c \in (0, \frac{1}{L_{\max}}) \), which is independent of the global topology, and only depends on \( L_{\max} := \max_{i \in \mathcal{N}} L_i \) – some max-consensus algorithm is needed to compute \( L_{\max} \), and this may not be feasible for very large scale fully distributed networks.

Table 3.2: Comparison on data amount stored and communicated per node per communication round (iteration)

<table>
<thead>
<tr>
<th></th>
<th>DPGA-I</th>
<th>DPGA-II</th>
<th>ADMM</th>
<th>SADMM</th>
<th>DFAL</th>
<th>PG-EXTRA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage</td>
<td>3n</td>
<td>3n</td>
<td>3n</td>
<td>7n</td>
<td>4n</td>
<td>4n or (4 + d_i)n</td>
</tr>
<tr>
<td>Communication</td>
<td>2n</td>
<td>n</td>
<td>2n</td>
<td>4n</td>
<td>n</td>
<td>2n or (n)</td>
</tr>
</tbody>
</table>
Table 3.2 summarizes the storage and communication requirements for the algorithms discussed above; we illustrate their practical performance in the following two chapters. We recently became aware of three recent works proposed in \cite{53,57,76} for solving (3.3) over a connected graph $\mathcal{G}$. These methods are very closely related to our proximal gradient ADMM (PG-ADMM), which we discuss in Chapter 5, and are also based on linearized ADMM method. Suppose $F_i(x) = \rho_i(x) + \gamma(A_ix)$. When compared to our smooth convexity assumption on $\{\gamma_i\}_{i \in \mathcal{N}}$, Chang et al. \cite{57} showed the convergence of their distributed method under a far more stringent assumption: $\gamma_i$ is strongly convex with a Lipschitz continuous gradient for $i \in \mathcal{N}$. Moreover, under this stronger assumption, they were able to show linear convergence rate only when the non-smooth terms are absent, i.e., $\rho_i \equiv 0$, and $A_i$ has full column rank for all $i \in \mathcal{N}$. Finally, their distributed algorithm requires the global knowledge of $\sigma_{\min}(\Omega + W)$ of the graph $\mathcal{G}$, where $\Omega$ is the graph Laplacian, and $W$ is the adjacency matrix – this assumption is not attainable for very large scale fully distributed networks. On the other hand, the algorithms we propose in Chapter 4 and Chapter 5 are fully distributed, i.e., the agents do not require the knowledge of some global parameters depending on the entire network topology; instead, we only assume that agents know who their neighbors are. Ling et al. \cite{76} were able to show the convergence of their distributed method without strong convexity when penalty parameter is chosen sufficiently large; however, no rate has been shown for this setting – again, as in \cite{57} determining whether the parameter is large enough requires the global knowledge of $\sigma_{\min}(\Omega)$. The algorithm in \cite{76} is similar to our DPGA-II algorithm, and in contrast to sublinear rate result shown in this manuscript, Ling et al. were able to establish a rate result only under strong convexity assumption. Bianchi et al. \cite{53} also proposed a distributed algorithm based on linearized ADMM for solving (3.3), where each node computes proximal gradient steps; the authors proved its convergence and also show almost sure convergence for its randomized version, where a random set of agents become active and compute their proximal gradient steps and broadcast the recently updated local variables to their neighbors. However, no rate has been shown for neither the deterministic nor the randomized versions. The methods in \cite{53} run on edge-based formulations of the decentralized problem; due to the nature of edge-based distributed algorithms, they require each node to store a dual variable for each edge it is linked to, and to memorize all its neighbor’s local variables in addition to its own local variable; and this information is being exchanged at every iteration. Consequently, information exchange, computational effort and memory requirement are far more expensive than
node-based algorithms proposed in the following chapters.
In this chapter, we propose a distributed first-order augmented Lagrangian (DFAL) algorithm, establish the following main result for the synchronous case in Section 4.2.3, and extend it to an asynchronous setting in Section 4.2.4 and Section 4.2.5.

**Main Theorem.** Let \( \{x^{(k)}\}_{k \in \mathbb{Z}_+} \) denote the sequence of DFAL iterates. Then \( F^* = \lim_{k \to \infty} \sum_{i \in \mathcal{N}} F_i(x_i^{(k)}) \). Furthermore, \( x^{(k)} \) is \( \epsilon \)-optimal and \( \epsilon \)-feasible within \( O(\log(\epsilon^{-1})) \) DFAL iterations, requiring \( O\left( \frac{\psi_{\max} \log(\epsilon^{-1})}{d_{\min}} \right) \) communications per node, and \( O(\epsilon^{-1}) \) gradient and proximal map computations for \( \gamma_i \) and \( \rho_i \), respectively, where \( \psi_{\max} \) denotes the largest eigenvalue of the Laplacian of \( \mathcal{G} \), and \( d_{\min} \) denotes the minimum degree over all nodes.

Aybat and Iyengar [77] proposed an efficient first-order augmented Lagrangian (FAL) algorithm for the basis pursuit problem \( \min_{x \in \mathbb{R}^n} \{ \|x\|_1 : Ax = b \} \) to compute an \( \epsilon \)-optimal and \( \epsilon \)-feasible solution to within \( O(\kappa^2(A)/\epsilon) \) matrix-vector multiplications, where \( A \in \mathbb{R}^{m \times n} \) such that \( \text{rank}(A) = m \), and \( \kappa(A) := \sigma_{\max}(A)/\sigma_{\min}(A) \) denotes the condition number of \( A \). In this chapter, we extend their FAL algorithm to solve a more general version of (3.4) in Section 4.2.1 and 4.2.2, and establish the Main Result for (3.4) in Section 4.2.3. In Section 4.2.4, we propose an asynchronous version of DFAL. It is important to emphasize that DFAL can be easily extended to solve (3.4) when there are global constraints on network resources of the form \( Ex - q \in \mathcal{K} \), where \( \mathcal{K} \) is a proper cone, and none of the algorithms discussed in Chapter 3 can accommodate such global conic constraints efficiently. We do not discuss this extension in the dissertation; however, the analysis would be similar to [78, 79].
Definition 4.1. (a) Let \( \Gamma \) be the set of convex functions \( \gamma : \mathbb{R}^n \rightarrow \mathbb{R} \) such that \( \nabla \gamma \) is Lipschitz continuous with constant \( L_\gamma \), and \( \gamma(x) \geq \gamma \) for all \( x \in \mathbb{R}^n \) for some \( \gamma \in \mathbb{R} \).

(b) Let \( \mathcal{R} \) be the set of convex functions \( \rho : \mathbb{R}^n \rightarrow \mathbb{R} \) such that subdifferential of \( \rho \) is uniformly bounded on \( \mathbb{R}^n \), i.e., there exists \( B > 0 \) such that \( \|q\|_2 \leq B \) for all \( q \in \partial \rho(x) \), \( x \in \mathbb{R}^n \); and \( \tau \|x\|_2 \leq \rho(x) \) for all \( x \in \mathbb{R}^n \) for some \( \tau > 0 \).

Assumption 4.2. For all \( i \in \mathcal{N} \), we assume that \( \gamma_i \in \Gamma \) and \( \rho_i \in \mathcal{R} \) with corresponding constants \( L_{\gamma_i}, \gamma_i, B_i \) and \( \tau_i \).

Most of the important regularizers and loss functions used in machine learning and statistics literature lie in \( \mathcal{R} \) and \( \Gamma \), respectively. In particular, any norm, e.g., \( \| \cdot \|_\alpha \) with \( \alpha \in \{1, 2, \infty\} \), group norm (see Section 4.3), nuclear norm, etc., weighted sum of these norms, e.g., sparse group norm (see Section 4.3), all belong to \( \mathcal{R} \). Given \( A \in \mathbb{R}^{m \times n} \) and \( b \in \mathbb{R}^m \), quadratic-loss \( \|Ax - b\|_2^2 \), Huber-loss \( \sum_{i=1}^m h(a_i^T x - b_i) \) (see Section 4.3), logistic-loss \( \sum_{i=1}^m \log (1 + e^{-b_i a_i^T x}) \), or fair-loss \([80]\) functions all belong to \( \Gamma \).

Throughout the chapter, we adopt the notation \( x = (x_i; x_{-i}) \) with \( x_{-i} = (x_j)_{j \neq i} \) to denote a vector where \( x_i \) and \( x_{-i} \) are treated as variable and parameter sub-vectors of \( x \), respectively. Given \( f : \mathbb{R}^{n \mathcal{N}} \rightarrow \mathbb{R}, \nabla_{x_i} f(x) \in \mathbb{R}^n \) denotes the sub-vector of \( \nabla f(x) \in \mathbb{R}^{n \mathcal{N}} \) corresponding to components of \( x_i \in \mathbb{R}^n \).

### 4.1 APG Algorithm for the Centralized Model

Consider the centralized version (3.3) where all the functions \( F_i \) are available at a central node, and all computations are carried out at this node. Suppose \( \{\rho_i\}_{i \in \mathcal{N}} \) and \( \{\gamma_i\}_{i \in \mathcal{N}} \) satisfy Assumption 4.2. Let \( \rho(x) := \sum_{i=1}^N \rho_i(x) \) and \( \gamma(x) := \sum_{i=1}^N \gamma_i(x) \). Lipschitz continuity of each \( \nabla \gamma_i \) with constant \( L_{\gamma_i} \) implies that \( \nabla \gamma \) is also Lipschitz continuous with constant \( L_{\gamma} = \sum_{i=1}^N L_{\gamma_i} \). When \( \text{prox}_{\rho/L_{\gamma}} \) can be computed efficiently, the accelerated proximal gradient (APG) algorithm proposed in \([29,30]\) guarantees that

\[
0 \leq F(x^{(t)}) - F^* \leq \frac{2L_\gamma}{(t + 1)^2} \|x^{(0)} - x^*\|^2_2, \tag{4.1}
\]

where \( x^{(0)} \) is the initial iterate and \( x^* \in \arg\min_{x \in \mathbb{R}^n} F(x) \) —see Corollary 3 in \([30]\), and Theorem 4.4 in \([29]\). Thus, APG can compute an \( \epsilon \)-optimal solution to (3.3) within \( \mathcal{O}(\sqrt{L_\gamma} \epsilon^{-\frac{1}{2}}) \) iterations.

As discussed above, the centralized APG algorithm cannot be applied when the nodes are unwilling or unable to communicate the privately known functions \( \{F_i\}_{i \in \mathcal{N}} \).
to a central node. There are many other settings where one may want to solve \((3.3)\) as a “distributed” problem. For instance, although \(\text{prox}_{t_\rho_i}\) can be computed efficiently for all \(t > 0\) and \(i \in \mathcal{N}\), \(\text{prox}_{\rho/L_\gamma}\) may be hard to compute. As an example, consider a problem with \(\rho_1(X) = \sum_{i,j} |X_{ij}| \) and \(\rho_2 = \sum_{i=1}^{\text{rank}(X)} \sigma_i(X)\), where \(\sigma(X)\) denotes the vector of singular values for \(X \in \mathbb{R}^{n_1 \times n_2}\). Here, \(\text{prox}_{t_\rho_i}\) is easy to compute for all \(t > 0\) and \(i \in \{1, 2\}\); however, \(\text{prox}_{t(\rho_1 + \rho_2)}\) is hard to compute. Thus, the “centralized” APG algorithm cannot be applied. In the rest of this chapter, we focus on decentralized algorithms.

### 4.2 DFAL Algorithm for the Decentralized Model

Let \(\mathbf{x} = (x_1^T, \ldots, x_N^T) \in \mathbb{R}^{nN}\) denotes a vector formed by concatenating \(\{x_i\}_{i \in \mathcal{N}} \subset \mathbb{R}^n\) as a long column vector. Consider the following optimization problem of the form:

\[
\bar{F}^* := \min_{\mathbf{x} \in \mathbb{R}^{nN}} \left\{ \bar{F}(\mathbf{x}) := \bar{\rho}(\mathbf{x}) + \bar{\gamma}(\mathbf{x}) \text{ s.t. } A \mathbf{x} = \mathbf{b} \right\},
\]

where \(\bar{\rho}(\mathbf{x}) := \sum_{i=1}^{N} \rho_i(x_i), \bar{\gamma}(\mathbf{x}) := \sum_{i=1}^{N} \gamma_i(x_i)\), and \(A \in \mathbb{R}^{m \times nN}\) has \(\text{rank}(A) = m\), i.e., the linear map is surjective. In Section 4.2.3, we show that the distributed optimization problem in \((3.4)\) is a special case of \((4.2)\), i.e., for all connected \(\mathcal{G}\), there exists a surjective \(A\) such that \((3.4)\) is equivalent to \((4.2)\). In the rest of the section, we will use the following notation: Let \(\{A_i\}_{i \in \mathcal{N}} \subset \mathbb{R}^{m \times n}\) such that \(A = [A_1, A_2, \ldots, A_N]; \tilde{L} := \max_{i \in \mathcal{N}} L_{\gamma_i},\bar{\tau} := \min_{i \in \mathcal{N}} \tau_i\).

We propose to solve \((4.2)\) by inexact \(\lceil\text{ly}\rceil\) solving the following sequence of subproblems in a distributed manner:

\[
x^{(k)}_* \in \arg\min_{\mathbf{x} \in \mathbb{R}^{nN}} P^{(k)}(\mathbf{x}) := \lambda^{(k)} \bar{\rho}(\mathbf{x}) + f^{(k)}(\mathbf{x}),
\]

\[
f^{(k)}(\mathbf{x}) := \lambda^{(k)} \bar{\gamma}(\mathbf{x}) + \frac{1}{2} \|A\mathbf{x} - \mathbf{b} - \lambda^{(k)} \theta^{(k)}\|^2,
\]

for appropriately chosen sequences of penalty parameters \(\{\lambda^{(k)}\}\) and dual variables \(\{\theta^{(k)}\}\) such that \(\lambda^{(k)} \searrow 0\). In particular, given \(\{\alpha^{(k)}, \xi^{(k)}\}\) satisfying \(\alpha^{(k)} \searrow 0\) and \(\xi^{(k)} \searrow 0\), the iterate sequence \(\{\mathbf{x}^{(k)}\}\) is constructed such that every \(\mathbf{x}^{(k)}\) satisfies one of the following conditions:

\[
\begin{align*}
(a) \quad & P^{(k)}(\mathbf{x}^{(k)}) - P^{(k)}(\mathbf{x}_*^{(k)}) \leq \alpha^{(k)}, \\
(b) \quad & \exists g_i^{(k)} \in \partial_{\mathbf{x}_i} P^{(k)}(\mathbf{x}) |_{\mathbf{x} = \mathbf{x}^{(k)}} \text{ s.t. } \max_{i \in \mathcal{N}} \|g_i^{(k)}\|_2 \leq \frac{\xi^{(k)}}{\sqrt{\mathcal{N}}},
\end{align*}
\]

\footnote{\(\mathcal{G}\) can contain cycles.}
\[ \partial_x P^{(k)}(x)|_{x=\bar{x}} := \lambda^{(k)} \partial \rho_i(x_i)|_{x_i=\bar{x}} + \nabla_x f^{(k)}(\bar{x}). \]

Note that \( \nabla f^{(k)}(x) \) is Lipschitz continuous in \( x \in \mathbb{R}^{nN} \) with constant \( \lambda^{(k)} L + \sigma_{\text{max}}^2(A) \). Given \( \{x^{(0)}, \lambda^{(0)}, \alpha^{(0)}, \xi^{(0)}\} \) and \( c \in (0,1) \), we choose the sequence \( \{\lambda^{(k)}, \alpha^{(k)}, \xi^{(k)}, \theta^{(k)}\} \) as shown in Fig. 4.1.

**Algorithm DFAL** \( (\lambda^{(1)}, \alpha^{(1)}, \xi^{(1)}) \)

Step 0: Set \( \theta^{(1)} = 0, k = 1 \)

Step \( k \geq 1 \)

1. Compute \( x^{(k)} \) such that (4.5)(a) or (4.5)(b) holds
2. \( \theta^{(k+1)} = \theta^{(k)} - \frac{A\bar{x}^{(k)} - b}{\lambda^{(k)}} \)
3. \( \lambda^{(k+1)} = c\lambda^{(k)}, \quad \alpha^{(k+1)} = c^2 \alpha^{(k)}, \quad \xi^{(k+1)} = c^2 \xi^{(k)} \)

**Figure 4.1:** First-order Augmented Lagrangian algorithm

In Section 4.2.1 we show that DFAL can compute an \( \epsilon \)-optimal and \( \epsilon \)-feasible \( x_\epsilon \) to (4.2), i.e., \( \|Ax_\epsilon - b\|_2 \leq \epsilon \) and \( |\bar{F}(x_\epsilon) - F^*| \leq \epsilon \), in at most \( O(\log(1/\epsilon)) \) iterations.

Next, in Section 4.2.2 we show that computing an \( \epsilon \)-optimal, \( \epsilon \)-feasible solution \( x_\epsilon \) requires at most \( O\left(\frac{\sigma_{\text{max}}^2(A)}{\min_{i \in N} \sigma_{\text{min}}^2(A_i)} \epsilon^{-1}\right) \) floating point operations. Using this result, in Section 4.2.3 we establish that DFAL can compute \( x_\epsilon \) in a distributed manner within \( O(\epsilon^{-1}) \) communication steps, i.e., the Main Result stated in Section 3.1. Finally, in Section 4.2.4 we show how to modify DFAL for an asynchronous computation setting.

### 4.2.1 DFAL iteration complexity

We first show that \( \{x^{(k)}\} \) is a bounded sequence, and then argue that this also implies boundedness of \( \{\theta^{(k)}\} \). First, we start with a technical lemma that will be used in establishing the main results of this section.

**Lemma 4.3.** Let \( \tilde{\rho} : \mathbb{R}^{nN} \to \mathbb{R} \) be defined as \( \tilde{\rho}(x) = \sum_{i \in N} \rho_i(x_i) \), where \( \rho_i \in \mathcal{R} \) with uniform bound \( B_i \) on its subdifferential for all \( x \in \mathbb{R}^n \) and for all \( i \in \mathcal{N} \). Let \( f : \mathbb{R}^{nN} \to \mathbb{R} \) denote a convex function such that there exist constants \( \{L_i\}_{i=1}^N \subseteq \mathbb{R}_{++} \) that satisfy

\[ f(y) \leq f(\bar{y}) + \nabla f(\bar{y})^T(y - \bar{y}) + \sum_{i=1}^N \frac{L_i}{2} \|y_i - \bar{y}_i\|^2 \]

for all \( y, \bar{y} \in \mathbb{R}^{nN} \). Given \( \alpha, \lambda \geq 0 \), and \( \bar{x} \in \mathbb{R}^{nN} \) such that \( \lambda \rho(\bar{x}) + f(\bar{x}) \) is \( \lambda \rho(\bar{x}) + f(\bar{x}) \) \( \alpha \)-lower semi-continuous at \( \bar{x} \), it follows that \( \|\nabla x_i f(\bar{x})\|_2 \leq \sqrt{2L_i \alpha} + \lambda B_i \) for all \( i \in \mathcal{N} \).
Proof: Let \( x \in \mathbb{R}^{nN} \) and \( g_i \in \partial \rho_i(x_i) \) for all \( i \in \mathcal{N} \). From convexity of \( \rho_i \) and Cauchy-Schwarz, it follows that \( \rho_i(x_i) \leq \rho(\bar{x}_i) + \|g_i\|_2 \|x_i - \bar{x}_i\|_2 \) for all \( i \in \mathcal{N} \). Hence, we have

\[
\lambda \bar{\rho}(x) + f(x) \leq \lambda \rho(\bar{x}) + f(\bar{x}) + \sum_{i \in \mathcal{N}} \left( \lambda B_i \|x_i - \bar{x}_i\|_2^2 + \nabla x_i f(\bar{x})^T (x_i - \bar{x}_i) + \frac{L_i}{2} \|x_i - \bar{x}_i\|_2^2 \right).
\]

Minimizing on both sides and using the separability of the right side, we have \( \min_{x \in \mathbb{R}^{nN}} \lambda \bar{\rho}(x) + f(x) \leq \lambda \bar{\rho}(\bar{x}) + f(\bar{x}) + \sum_{i \in \mathcal{N}} \min_{x_i \in \mathbb{R}^n} h_i(x_i) \), where \( h_i(x_i) := \nabla x_i f(\bar{x})^T (x_i - \bar{x}_i) + \lambda B_i \|x_i - \bar{x}_i\|_2^2 + \frac{L_i}{2} \|x_i - \bar{x}_i\|_2^2 \). Let \( \bar{x}_i^* := \text{argmin}_{x_i \in \mathbb{R}^n} h_i(x_i) \). Then the first-order optimality conditions imply that \( 0 \in \nabla x_i f(\bar{x}) + L_i (\bar{x}_i^* - \bar{x}_i) + \lambda B_i \partial \|x_i - \bar{x}_i\|_2 \bigg|_{x_i = \bar{x}_i^*} \) for all \( i \in \mathcal{N} \).

Let \( \mathcal{I} := \{ i \in \mathcal{N} : \|\nabla x_i f(\bar{x})\|_2 \leq \lambda B_i \} \). For each \( i \in \mathcal{N} \), there are two possibilities.

**Case 1:** Suppose that \( i \in \mathcal{I} \), i.e., \( \|\nabla x_i f(\bar{x})\|_2 \leq \lambda B_i \). Since \( \min_{x_i \in \mathbb{R}^n} h_i(x_i) \) has a unique solution, and \(-\nabla x_i f(\bar{x}) \in \lambda B_i \partial \|x_i - \bar{x}_i\|_2 \bigg|_{x_i = \bar{x}_i} \) when \( \|\nabla x_i f(\bar{x})\|_2 \leq \lambda B_i \), it follows that \( \bar{x}_i^* = \bar{x}_i \) if and only if \( \|\nabla x_i f(\bar{x})\|_2 \leq \lambda B_i \). Hence, \( h_i(\bar{x}_i^*) = 0 \).

**Case 2:** Suppose that \( i \in \mathcal{I}^c := \mathcal{N} \setminus \mathcal{I} \), i.e., \( \|\nabla x_i f(\bar{x})\|_2 > \lambda B_i \). In this case, \( \bar{x}_i^* \neq \bar{x}_i \).

From the first-order optimality condition, we have \( \nabla x_i f(\bar{x}) + L_i (\bar{x}_i^* - \bar{x}_i) + \lambda B_i \bar{x}_i^* - \bar{x}_i \bigg|_{\bar{x}_i = \bar{x}_i^*} = 0 \). Let \( s_i := \frac{\bar{x}_i^* - \bar{x}_i}{\|\bar{x}_i^* - \bar{x}_i\|_2} \) and \( t_i := \|\bar{x}_i^* - \bar{x}_i\|_2 \), then \( s_i = \frac{-\nabla x_i f(\bar{x})}{L_i t_i + \lambda B_i} \). Since \( \|s_i\|_2 = 1 \), it follows that \( t_i = \|\nabla x_i f(\bar{x})\|_2 - \lambda B_i > 0 \), and \( s_i = \frac{-\nabla x_i f(\bar{x})}{\|\nabla x_i f(\bar{x})\|_2} \). Hence, \( \bar{x}_i^* = \bar{x}_i - \frac{\|\nabla x_i f(\bar{x})\|_2 - \lambda B_i}{L_i} \nabla x_i f(\bar{x}) \nabla x_i f(\bar{x})_2 \)

From the \( \alpha \)-optimality of \( \bar{x} \), it follows that

\[
\sum_{i \in \mathcal{I}} \left( \|\nabla x_i f(\bar{x})\|_2^2 - \lambda B_i \right) \leq \sum_{i \in \mathcal{I}} \left( \|\nabla x_i f(\bar{x})\|_2^2 - \lambda B_i \right) \leq \frac{\alpha}{2} \sum_{i \in \mathcal{I}} \lambda \bar{\rho}(\bar{x}) + f(\bar{x}) - \min_{x \in \mathbb{R}^{nN}} \lambda \bar{\rho}(x) + f(x) \leq \alpha,
\]

which implies that \( \|\nabla x_i f(\bar{x})\|_2 \leq \sqrt{2L_i} \alpha + \lambda B_i \) for all \( i \in \mathcal{I} \). Moreover, \( \|\nabla x_i f(\bar{x})\|_2 \leq \lambda B_i \) for all \( i \in \mathcal{I}^c \). Hence, the result follows from these two inequalities.

In Lemma 4.4, we show that function \( f^{(k)} \) defined in (4.4) satisfies the condition given in Lemma 4.3.

**Lemma 4.4.** The function \( f^{(k)} \) in (4.3) satisfies the condition in Lemma 4.3 with the constants \( L_i = L_i^{(k)} \), where \( L_i^{(k)} := \lambda^{(k)} L_{\gamma_i} + \sigma_{\text{max}}^2(A) \) for all \( i \in \mathcal{N} \).

**Proof:** For all \( i \in \mathcal{N} \), since \( \nabla \gamma_i \) is Lipschitz continuous with constant \( L_{\gamma_i} \), for any
\(\mathbf{x}, \bar{\mathbf{x}} \in \mathbb{R}^{n_N}\), we have \(\gamma_i(x_i) \leq \gamma_i(\bar{x}_i) + \nabla \gamma_i(\bar{x}_i)^T (x_i - \bar{x}_i) + \frac{L_{\gamma_i}}{2} \|x_i - \bar{x}_i\|^2\). Then, we have

\[
\gamma(\mathbf{x}) \leq \sum_{i=1}^{N} \gamma_i(\bar{x}_i) + \nabla \gamma_i(\bar{x}_i)^T (x_i - \bar{x}_i) + \frac{L_{\gamma_i}}{2} \|x_i - \bar{x}_i\|^2
\]

\[
\leq \gamma(\bar{\mathbf{x}}) + \nabla \gamma(\bar{\mathbf{x}})^T (\mathbf{x} - \bar{\mathbf{x}}) + \sum_{i=1}^{N} \frac{L_{\gamma_i}}{2} \|x_i - \bar{x}_i\|^2.
\]  \hspace{1cm} (4.6)

Let \(h^{(k)}(\mathbf{x}) = \frac{1}{2} \|A\mathbf{x} - b - \lambda^{(k)} \theta^{(k)}\|^2\). It follows that \(\nabla h^{(k)}\) is Lipschitz continuous with constant \(\sigma_{\text{max}}^2(A)\). Since \(f^{(k)} = \lambda^{(k)} \tilde{\gamma} + h^{(k)}\), the result follows from (4.6).

Lemma 4.3 and Lemma 4.4 allow us to bound \(\|\theta^{(k+1)}\|_2\) in terms of \(\{\|\nabla x_i \gamma(x_i^{(k)})\|_2\}_{i \in \mathcal{N}}\). We later use this bound in an inductive argument to establish that \(\{\mathbf{x}^{(k)}\}\) is bounded.

**Lemma 4.5.** Let \(\{\mathbf{x}^{(k)}\}\) be the DFAL iterate sequence, i.e., at least one of the conditions in (4.5) hold for all \(k \geq 1\). Define \(\Theta_i^{(k)} := \max \left\{\sqrt{2L_i^{(k)} \alpha_i^{(k)}} \frac{\xi_i^{(k)}}{\lambda^{(k)}}, \frac{1}{\sqrt{N}} \frac{\xi_i^{(k)}}{\lambda^{(k)}}\right\} + B_i + \|\nabla \gamma_i(x_i^{(k)})\|_2\). Then for all \(k \geq 1\), we have

\[
\|\theta^{(k+1)}\|_2 \leq \min_{i \in \mathcal{N}} \left\{ \frac{\Theta_i^{(k)}}{\sigma_{\text{min}}(A_i)} \right\}.
\]

**Proof:** Fix \(k \geq 1\). Suppose that \(\mathbf{x}^{(k)}\) satisfies (4.5)(a). Then Lemma 4.3 implies that for all \(i \in \mathcal{N}\),

\[
\|\nabla x_i f^{(k)}(\mathbf{x}^{(k)})\|_2 = \|\lambda^{(k)} \nabla \gamma_i(x_i^{(k)}) + A_i^T (A\mathbf{x}^{(k)} - b - \lambda^{(k)} \theta^{(k)})\|_2 \leq \sqrt{2L_i^{(k)} \alpha_i^{(k)} + \lambda^{(k)} B_i}.
\]

Now, suppose that \(\mathbf{x}^{(k)}\) satisfies (4.5)(b). Then triangular inequality immediately implies that \(\|\nabla x_i f^{(k)}(\mathbf{x}^{(k)})\|_2 \leq \frac{\xi_i^{(k)}}{\sqrt{N}} + \lambda^{(k)} B_i\) for all \(i \in \mathcal{N}\). Combining the two inequalities, and further using triangular Cauchy-Schwarz inequalities, it follows for all \(i \in \mathcal{N}\) that

\[
\|A\mathbf{x}^{(k)} - b - \lambda^{(k)} \theta^{(k)}\|_2 \leq \frac{\max \left\{\sqrt{2L_i^{(k)} \alpha_i^{(k)}}, \frac{\xi_i^{(k)}}{\sqrt{N}} + \lambda^{(k)} B_i + \|\nabla \gamma_i(x_i^{(k)})\|_2\right\}}{\sigma_{\text{min}}(A_i)}.\]

Hence, we conclude by diving the above inequality by \(\lambda^{(k)}\) and using the definition of \(\theta^{(k+1)}\).

Theorem 4.6 establishes that the DFAL iterate sequence \(\{\mathbf{x}^{(k)}\}\) is bounded whenever \(\{\rho_i, \gamma_i\}_{i \in \mathcal{N}}\) satisfy Assumption 4.2, therefore, the sequence of dual variables \(\{\theta^{(k)}\}\) is bounded according to Lemma 4.5.

**Theorem 4.6.** Suppose Assumption 4.2 holds. Then there exist constants \(B_x, B_\theta, \bar{\lambda} > 0\) such that \(\max\{\|\mathbf{x}^*_x\|_2, \|\mathbf{x}^{(k)}\|_2\} \leq B_x\) and \(\|\theta^{(k)}\|_2 \leq B_\theta\) for all \(k \geq 1\), whenever \(\lambda^{(1)}\) and \(\xi^{(1)}\) are chosen such that \(0 < \lambda^{(1)} \leq \bar{\lambda}\) and \(\frac{\xi^{(1)}}{\lambda^{(1)}} < \bar{\tau}\).
Proof: Let \( A = [A_1, A_2, \ldots, A_N] \in \mathbb{R}^{m \times n^N} \) such that \( A_i \in \mathbb{R}^{m \times n} \) for all \( i \in \mathcal{N} \). Throughout the proof we assume that \( \sigma_{\text{max}}(A) \geq \sqrt{\max_{i \in \mathcal{N}} d_i + 1} \), and \( \sigma_{\text{min}}(A_i) = \sqrt{d_i} \geq 1 \) for all \( i \in \mathcal{N} \), where \( d_i \geq 1 \) is the degree of \( i \in \mathcal{N} \). Indeed, when \( A \) is chosen as described in Section 4.2.3 corresponding to graph \( \mathcal{G} \), we showed that \( \sigma_{\text{max}}^2(A) = \psi_1 \), where \( \psi_1 \) is the largest eigenvalue of the Laplacian \( \Omega \) corresponding to \( \mathcal{G} \). Furthermore, it is also shown in [81] that when \( \mathcal{G} \) is connected, one has \( \psi_1 \geq \max_{i \in \mathcal{N}} d_i + 1 > 1 \). Hence, \( \sigma_{\text{max}}(A) \geq \sqrt{\max_{i \in \mathcal{N}} d_i + 1} > 1 \). Finally, for \( A \) chosen as described in Section 4.2.3 corresponding to graph \( \mathcal{G} \), we also showed that \( \sigma_{\text{min}}(A_i) = \sqrt{d_i} \) for all \( i \in \mathcal{N} \).

To keep notation simple, without loss of generality, we assume that \( \gamma_i = 0 \) for all \( i \in \mathcal{N} \). Hence, \( \gamma(x) \geq 0 \) for all \( x \in \mathbb{R}^{nN} \). Let \( x^* \) be a minimizer of \((4.2)\). By Lipschitz continuity of \( \nabla \gamma_i \), we have for all \( i \in \mathcal{N} \)

\[
\|\nabla \gamma(x_i)\|_2 \leq L_{\gamma_i} \|x_i - x_i^*\|_2 + \|\nabla \gamma_i(x_i^*)\|_2.
\] (4.7)

We prove the theorem using induction. We show that, for an appropriately chosen bound \( R \), \( \|x^{(k)} - x^*\|_2 \leq R \) implies \( \|x^{(k+1)} - x^*\|_2 \leq R \), for all \( k \geq 1 \). Fix \( k \geq 1 \). First, suppose that \( x^{(k+1)} \) satisfies \((4.5)\)(a), i.e. \( P^{(k+1)}(x^{(k+1)}) \leq P^{(k+1)}(x^*) + \alpha^{(k+1)} \). By dividing both sides by \( \lambda^{(k+1)} \), it follows from Assumption \(4.2\), \( Ax^* = b \), and \( f^{(k+1)}(\cdot) \geq 0 \) that

\[
\tilde{\tau} \|x^{(k+1)}\|_2 \leq \tilde{\rho}(x^*) + \tilde{\gamma}(x^*) + \frac{\lambda^{(k+1)}}{2} \left( \|\theta^{(k+1)}\|_2^2 + \frac{\alpha^{(k+1)}}{(\lambda^{(k+1)})^2} \right).
\] (4.8)

Next, suppose \( x^{(k+1)} \) satisfies \((4.5)\)(b). It follows from convexity of \( P^{(k+1)} \) and Cauchy-Schwarz inequality that \( P^{(k+1)}(x^{(k+1)}) \leq P^{(k+1)}(x^*) + \zeta^{(k+1)} \|x^{(k+1)} - x^*\|_2 \). Again, dividing both sides by \( \lambda^{(k+1)} \), we get

\[
\tilde{\tau} \|x^{(k+1)}\|_2 \leq \tilde{\rho}(x^*) + \tilde{\gamma}(x^*) + \frac{\lambda^{(k+1)}}{2} \|\theta^{(k+1)}\|_2^2 + \frac{\zeta^{(k+1)}}{\lambda^{(k+1)}} \|x^{(k+1)} - x^*\|_2.
\] (4.9)

Combining the bounds for both cases, \((4.8)\) and \((4.9)\), and using triangular inequality, we have

\[
\left( \tilde{\tau} - \frac{\zeta^{(k+1)}}{\lambda^{(k+1)}} \right) \|x^{(k+1)} - x^*\|_2 \leq \tilde{F}^* + \tilde{\tau} \|x^*\|_2 + \frac{\lambda^{(k+1)}}{2} \left( \|\theta^{(k+1)}\|_2^2 + \frac{\alpha^{(k+1)}}{(\lambda^{(k+1)})^2} \right),
\] (4.10)

for all \( k \geq 0 \). Note that \( \{\lambda^{(k)}, \alpha^{(k)}, \zeta^{(k)}\} \) is chosen in \textbf{DFAL} such that \( \frac{\alpha^{(k)}}{(\lambda^{(k)})^2} = \frac{\alpha^{(1)}}{(\lambda^{(1)})^2} \).
for all \( k > 1 \), and both \( \xi^{(k)} \frac{1}{\lambda^{(k)}} \searrow 0 \) and \( \lambda^{(k)} \searrow 0 \) monotonically. Since \( \sigma_{\min}(A_i) \geq 1 \) for all \( i \in \mathcal{N} \), the inductive assumption \( \| x^{(k)} - x^* \|_2 \leq R \), (4.7), and Lemma 4.5 together imply that

\[
\| \theta^{(k+1)} \|_2 \leq \min_{i \in \mathcal{N}} \left\{ \max \left\{ \frac{2L_i^{(1)}}{\lambda^{(1)}} \cdot \frac{\xi^{(1)}}{\lambda^{(1)}} \right\} + B_i + \| \nabla g_i(x^*_i) \|_2 + L_\gamma R \right\}. \tag{4.11}
\]

To simplify bounds further, choose \( \alpha^{(1)} = \frac{1}{4N} (\lambda^{(1)} \bar{\tau} )^2 \), and \( \xi^{(1)} = \frac{1}{2} \lambda^{(1)} \bar{\tau} \) for \( \lambda^{(1)} \leq \sigma_{\max}^2 (A)/\bar{L} \), where \( \bar{L} = \max_{i \in \mathcal{N}} \{ \lambda_i \} \). Let \( \bar{B} := \max_{i \in \mathcal{N}} B_i \) and \( \bar{G} := \max \{ \| \nabla g_i(x^*_i) \|_2 : i \in \mathcal{N} \} \). Together with (4.10), (4.11) and \( \sigma_{\max}(A) \geq 1 \), this choice of parameters implies that

\[
\frac{\bar{\tau}}{2} \| x^{(k+1)} - x^* \|_2 \leq \bar{F}^* + \bar{\tau} \| x^* \|_2 + \frac{\lambda^{(1)}}{2} \left[ \left( \frac{\bar{\tau} \sigma_{\max}(A)}{\sqrt{N}} + \bar{B} + \bar{G} + \bar{L}R \right)^2 + \frac{\bar{\tau}^2}{4N} \right].
\]

Define \( \beta_1 := \frac{2}{\bar{\tau}} \left( \bar{F}^* + \bar{\tau} \| x^* \|_2 \right) \), \( \beta_2 := \frac{\bar{\tau} \sigma_{\max}(A)/\sqrt{N} + \bar{B} + \bar{G}}{\sqrt{\bar{\tau}}} \), \( \beta_3 := \frac{\bar{L}}{\sqrt{\bar{\tau}}} \), and \( \beta_4 := \frac{\bar{\tau}^2}{4N} \). Then we have that \( \| x^{(k+1)} - x^* \|_2 \leq \beta_1 + \lambda^{(1)} \left[ \left( \beta_2 + \beta_3 R \right)^2 + \beta_4 \right] \).

Note that we are free to choose any \( \lambda^{(1)} > 0 \) satisfying \( \lambda^{(1)} \leq \sigma_{\max}^2 (A)/\bar{L} \). Our objective is to show that by appropriately choosing \( \lambda^{(1)} \), we can guarantee that \( \beta_1 + \lambda^{(1)} \left[ \left( \beta_2 + \beta_3 R \right)^2 + \beta_4 \right] \leq R \), which would then complete the inductive proof. This is indeed true if the above quadratic inequality in \( R \), has a solution, or equivalently if the discriminant

\[
\Delta = (2\lambda^{(1)} \beta_2 \beta_3 - 1)^2 - 4\lambda^{(1)} \beta_3^2 \left[ \lambda^{(1)} (\beta_2^2 + \beta_4) + \beta_3 \right]
\]

is non-negative. Note that \( \Delta \) is continuous in \( \lambda^{(1)} \), and \( \lim_{\lambda^{(1)} \to 0} \Delta = 1 \). Thus, for all sufficiently small \( \lambda^{(1)} > 0 \), we have \( \Delta \geq 0 \). Hence, we can set \( R = \frac{1 - 2\lambda^{(1)} \beta_2 \beta_3 - \sqrt{\Delta}}{2\lambda^{(1)} \beta_3^2} \) for some \( \lambda^{(1)} > 0 \) such that \( \Delta \geq 0 \), and this will imply that \( \| x^{(k+1)} - x^* \|_2 \leq R \) whenever \( \| x^{(k)} - x^* \|_2 \leq R \) for all \( k \geq 1 \).

The induction will be complete if we can show that \( \| x^{(1)} - x^* \|_2 \leq R \). Note that in \textbf{DFAL} we set \( \theta^{(1)} = 0 \). Hence, for \( k = 0 \), (4.10) implies that \( \| x^{(1)} - x^* \|_2 \leq \beta_1 + \lambda^{(1)} \beta_4 \). Hence, our choice of \( R \) guarantees that \( \| x^{(1)} - x^* \|_2 \leq R \). This completes the induction.

Following the same arguments leading to (4.10), it can also be shown that for all \( k \geq 0 \)

\[
\left( \frac{\bar{\tau}}{\lambda^{(k+1)}} \right) \| x^{(k+1)} - x^* \|_2 \leq \bar{F}^* + \bar{\tau} \| x^* \|_2 + \frac{\lambda^{(k+1)}}{2} \| \theta^{(k+1)} \|_2^2.
\]
Therefore, we can conclude that $\|x^{(k)}_* - x^*\| \leq R$ for all $k \geq 1$ holds for the same $R$ we selected above.

Note that $\Delta$ is a concave quadratic of $\lambda^{(1)}$ such that $\Delta = 1$ when $\lambda^{(1)} = 0$; hence, one of its roots is positive and the other one is negative. Moreover, $R \leq \frac{1}{2\lambda^{(1)}\beta_3^2} - \frac{\beta_2}{\beta_3}$ and the bound on $R$ is decreasing in $\lambda^{(1)} > 0$. Hence, in order to get a smaller bound on $R$, we will choose $\lambda^{(1)}$ as the positive root of $\Delta$. In particular, we set $\lambda^{(1)} = \sqrt{(\beta_2 + \beta_3\beta_4)^2 + \beta_4 - (\beta_2 + \beta_3\beta_4)}$.

We are now ready to state a key result that will imply the iteration complexity of DFAL.

**Theorem 4.7.** Suppose Assumption 4.2 holds and $\lambda^{(1)}$ and $\xi^{(1)}$ are chosen according to Theorem 4.6. Then the primal-dual iterate sequence $\{x^{(k)}, \theta^{(k)}\}$ generated by DFAL satisfy

(a) $\|Ax^{(k)} - b\|_2 \leq 2B_\theta \lambda^{(k)}$,

(b) $\tilde{F}(x^{(k)}) - \tilde{F}^* \geq -\lambda^{(k)}\frac{\|\theta^{(k+1)}\|_2 + B_\theta}{2}$

(c) $\bar{F}(x^{(k)}) - \bar{F}^* \leq \lambda^{(k)}\left(\frac{B_\theta^2}{2} + \frac{\max\{\alpha^{(1)}, \xi^{(1)}B_\theta\}}{(\lambda^{(1)})^2}\right)$,

where $\theta^*$ denotes any optimal dual solution to (4.2).

**Proof:** The proof directly follows from Theorem 3.3 in [77]. For the sake of completeness, we also provide the proof here. Let $x^*$ denote an optimal solution to (1.2).

Note that (a) follows immediately from Cauchy-Schwarz and the definition of $\theta^{(k+1)}$.

First, we prove the second inequality in (b). Suppose that $x^{(k)}$ satisfies (4.5)(a), which implies that $	ilde{F}(x^{(k)}) + \frac{\lambda^{(k)}}{2}\|\theta^{(k+1)}\|_2^2 \leq \bar{F}(x^*) + \frac{\lambda^{(k)}}{2}\|\theta^{(k)}\|_2^2 + \frac{\alpha^{(1)}}{\lambda^{(1)}}$. Now, suppose that $x^{(k)}$ satisfies (4.5)(b). From the convexity of $P^{(k)}$ and Cauchy-Schwarz, it follows that $P^{(k)}(x^{(k)}) \leq P^{(k)}(x^*) + \xi^{(k)}\|x^{(k)} - x^*\|_2$. Hence, dividing it by $\lambda^{(k)}$, we have $\bar{F}(x^{(k)}) + \frac{\lambda^{(k)}}{2}\|\theta^{(k+1)}\|_2^2 \leq \bar{F}(x^*) + \frac{\lambda^{(k)}}{2}\|\theta^{(k)}\|_2^2 + \frac{\xi^{(k)}}{\lambda^{(1)}}$. Therefore, for all $k \geq 1$, $x^{(k)}$ satisfies the second inequality in (b) since it also satisfies

$$\bar{F}(x^{(k)}) - \bar{F}^* \leq \lambda^{(k)}\left(\frac{\|\theta^{(k)}\|_2^2}{2} + \frac{\|\theta^{(k+1)}\|_2^2}{2} + \frac{\alpha^{(k)}, \xi^{(k)}\|x^{(k)} - x^*\|_2^2}{(\lambda^{(1)})^2}\right).$$

Now, in order to prove the first inequality in (b), we will exploit the primal-dual relations
of the following two pairs of problems:

\[(P) : \min_{x \in \mathbb{R}^{nN}} \{ \tilde{F}(x) : Ax = b \}, \quad (D) : \max_{\theta \in \mathbb{R}^m} b^T \theta - \tilde{F}^*(A^T \theta),\]

\[(P_k) : \min_{x \in \mathbb{R}^{nN}} \lambda(k) \tilde{F}(x) + \frac{1}{2} \| A x - b_k \|^2, \quad (D_k) : \max_{\theta \in \mathbb{R}^m} \lambda(k) (b^T \theta - \tilde{F}^*(A^T \theta)) - \frac{(\lambda(k))^2}{2} h(\theta),\]

where \(b_k := b + \lambda(k) \theta(k), h(\theta) := \| \theta - \theta(k) \|^2 - \| \theta(k) \|^2_2,\) and \(\tilde{F}^*\) denotes the convex conjugate of \(\tilde{F}\). Note that problem \((P_k)\) is nothing but the subproblem in (4.3). Therefore, from weak-duality between \((P_k)\) and \((D_k)\), it follows that

\[P^{(k)}(x^{(k)}) = \lambda(k) \tilde{F}(x^{(k)}) + \frac{1}{2} \| A x^{(k)} - b_k \|^2 \geq \lambda(k) (b^T \theta^* - \tilde{F}^*(A^T \theta^*)) - \frac{(\lambda(k))^2}{2} h(\theta^*).\]

Note that from strong duality between \((P)\) and \((D)\), it follows that \(\tilde{F}^* = \tilde{F}(\theta^*) = b^T \theta^* - \tilde{F}^*(A^T \theta^*)\). Therefore, dividing the above inequality by \(\lambda(k)\), we obtain

\[
\tilde{F}(x^{(k)}) - \tilde{F}^* \geq \frac{-\lambda(k)}{2} \left( \| \theta^* \|^2_2 - 2(\theta^*)^T \theta^{(k)} + \| \theta^{(k+1)} \|^2_2 \right) \geq -\lambda(k) h(\theta^*) - B \theta^*.
\]

\[
\square
\]

**Corollary 4.8.** The **DFAL** iterates \(x^{(k)}\) are \(\epsilon\)-feasible, i.e., \(\| A x^{(k)} - b \|_2 \leq \epsilon,\) and \(\epsilon\)-optimal, i.e., \(|\tilde{F}(x^{(k)}) - \tilde{F}^*| \leq \epsilon,\) for all \(k \geq N(\epsilon)\) and \(N(\epsilon) = \log \frac{C}{\epsilon} \) for some \(C > 0\).

### 4.2.2 Overall computational complexity

Efficiency of **DFAL** depends on the complexity of the oracle for Step 1 in Fig. 4.1. In this section, we construct an oracle **MS-APG** that computes an \(x^{(k)}\) satisfying (4.5) within \(O(1/\lambda(k))\) gradient and prox computations. This result together with Theorem 4.7 guarantees that for any \(\epsilon > 0, \) **DFAL** can compute an \(\epsilon\)-optimal and \(\epsilon\)-feasible iterate within \(O(\epsilon^{-1})\) floating point operations. Following lemma gives the iteration complexity of the oracle **MS-APG** displayed in Fig. 4.2.

**Lemma 4.9.** Let \(\tilde{\rho} : \mathbb{R}^{nN} \to \mathbb{R}\) such that \(\tilde{\rho}(x) = \sum_{i \in N} \rho_i(x_i),\) where \(\rho_i : \mathbb{R}^n \to \mathbb{R}\) is a convex function for all \(i \in N,\) and \(f : \mathbb{R}^{nN} \to \mathbb{R}\) be a convex function such that it satisfies the condition in Lemma 4.3 for some constants \(\{L_i\}_{i=1}^N \subset \mathbb{R}^{++}.\) Suppose that \(y^* \in \arg\min \Phi(y) := \tilde{\rho}(y) + f(y).\) Then the **MS-APG** iterate sequence \(\{y^{(\ell)}\}_{\ell \in \mathbb{Z}^+}\), computed as in Fig. 4.2, satisfies

\[
0 \leq \Phi(y^{(\ell)}) - \min_{y \in \mathbb{R}^{nN}} \Phi(y) \leq \frac{\sum_{i=1}^N 2L_i y_i^{(0)} - y_i^*}{(\ell + 1)^2}.
\]
\[ \text{Proof:} \] (4.12) follows from adapting the proof of Theorem 4.4 in [29] for the case here. 

**Algorithm MS-APG (\(u, f, y^{(0)}\))**

Step 0: Take \(y^{(0)} = t^{(0)} = 1\)

Step \(\ell\): (\(\ell \geq 1\))

1. \(y^{(\ell)}_i = \text{prox}_{\rho_i/L_i} \left( y^{(\ell)}_i - \nabla y_i f(y^{(\ell)}) / L_i \right) \quad \forall i \in \mathcal{N}\)
2. \(t^{(\ell+1)} = (1 + \sqrt{1 + 4 \left( t^{(\ell)} \right)^2}) / 2\)
3. \(y^{(\ell+1)} = y^{(\ell)} + \frac{t^{(\ell)} - 1}{t^{(\ell+1)}} \left( y^{(\ell)} - y^{(\ell-1)} \right)\)

Figure 4.2: Multi Step - Accelerated Prox. Gradient (MS-APG) algorithm

Consider the problem \(\Phi^* = \min \Phi(y) = \bar{\rho}(y) + f(y)\) defined in Lemma 4.9. Note that \(\nabla f\) is Lipschitz continuous with constant \(L = \max_{i \in \mathcal{N}} L_i\). In MS-APG algorithm, the step length \(1/L_i \geq 1/L\) is different for each \(i \in \mathcal{N}\). Instead, if one were to use the APG algorithm [29, 30], then the step length would have been \(1/L\) for all \(i \in \mathcal{N}\). When \(\{L_i\}_{i \in \mathcal{N}}\) are close to each other, the performances of MS-APG and APG are on par; however, when \(\frac{\max_{i \in \mathcal{N}} L_i}{\min_{i \in \mathcal{N}} L_i} \gg 1\), APG can only take very tiny steps for all \(i \in \mathcal{N}\); hence, MS-APG is likely to converge much faster in practice.

Since the subproblem (4.3) is in the form given in Lemma 4.9, the following result immediately follows.

**Lemma 4.10.** The iterate sequence \(\{y^{(\ell)}\}_{\ell \in \mathbb{Z}_+}\) generated by MS-APG(\(\lambda^{(k)} \bar{\rho}, f^{(k)}, x^{(k-1)}\)) satisfies \(P^{(k)}(y^{(\ell)}) - P^{(k)}(x^{(k)}_{\ast}) \leq \alpha^{(k)}\), for all \(\ell \geq \sqrt{\sum_{i=1}^{\mathcal{N}} 2L^{(k)}_i \|x^{(k-1)}_i - x^{(k)}_i\|_2^2} - 1\), where \(L^{(k)}_i\) is defined in Lemma 4.4 and \(x^{(k)}_{\ast}\) represents the \(i\)-th block of \(x^{(k)}\). Hence, one can compute \(x^{(k)}\) satisfying (4.5) within \(\mathcal{O}(1/\lambda^{(k)})\) **MS-APG** iterations.

Theorem 4.7 and Lemma 4.10 together imply that DFAL can compute an \(\epsilon\)-feasible, and \(\epsilon\)-optimal solution to (4.2) within \(\mathcal{O}(1/\epsilon)\) **MS-APG** iterations. Due to space considerations, we will only state and prove this result for the case where \(\nabla \bar{\gamma}\) is bounded in \(\mathbb{R}^{n\mathcal{N}}\) since the bounds \(B\gamma\) and \(Bx\) are more simple for this case. Note that Huber-loss, logistic-loss, and fair-loss functions indeed have bounded gradients.

**Theorem 4.11.** Suppose that \(\exists G_i > 0\) such that \(\|\nabla \gamma_i(x)\|_2 \leq G_i\) for all \(x \in \mathbb{R}^n\) and for all \(i \in \mathcal{N}\). Let \(N^{DFAL}_{\epsilon}(\mathcal{E})\) and \(N^{I}_{\epsilon}(\mathcal{E})\) denote the number of DFAL-iterations to compute an \(\epsilon\)-optimal, and an \(\epsilon\)-feasible solutions to (4.2), respectively. Let \(N^{(k)}\)
denote \textbf{MS-APG} iteration number required to compute \(x^{(k)}\) satisfying at least one of the conditions in (4.5). Then

\[
\sum_{k=1}^{N^\text{DFAL}(\epsilon)} N(k) = O\left(\Theta^2 \sigma_{\max}(A) \epsilon^{-1}\right),
\]
\[
\sum_{k=1}^{N^\text{DFAL}(\epsilon)} N(k) = O\left(\Theta \sigma_{\max}(A) \epsilon^{-1}\right),
\]

where \(\Theta = \frac{\sigma_{\max}(A)}{\min_{i \in \mathcal{N}} \sigma_{\min}(A_i)}\).

\textbf{Proof:} We assume that \(\sigma_{\max}(A) \geq \sqrt{\max_{i \in \mathcal{N}} d_i + 1}\), and \(\sigma_{\min}(A_i) = \sqrt{d_i} \geq 1\) for all \(i \in \mathcal{N}\), where \(d_i\) denotes the degree of \(i \in \mathcal{N}\). As discussed in the proof of Theorem 4.6 this is a valid assumption for distributed optimization problem in (3.4). Let \(\theta^*\) denote an optimal dual solution to (4.2). Note that from the first-order optimality conditions for (4.2), we have \(0 \in \nabla \gamma_i(x^*_i) + A_i \theta^* + \partial \rho_i(x_i)|_{x_i = x^*_i};\) hence, \(|A_i \theta^*|_2 \leq B_i + G_i\). Therefore, \(|\theta^*|_2 \leq \min_{i \in \mathcal{N}} B_i + G_i\).

Given \(0 < \lambda^{(1)} \leq \sigma_{\max}(A)/\tilde{L}\), choose \(\alpha^{(1)}, \xi^{(1)} > 0\) such that \(\alpha^{(1)} = \frac{1}{4\tau} \left(\lambda^{(1)} \tilde{\tau}\right)^2\), and \(\xi^{(1)} = \frac{1}{2} \lambda^{(1)} \tilde{\tau}\). Then Lemma 4.5 and \(\sigma_{\max}(A) \geq 1\) together imply that for all \(k \geq 1\)

\[
|\theta^{(k)}|_2 \leq \min_{i \in \mathcal{N}} \left\{\frac{\tilde{\tau} \sigma_{\max}(A) \sqrt{N} + B_i + G_i}{\sigma_{\min}(A_i)}\right\} := B_{\theta}.
\]

Hence, note that \(|\theta^*|_2 \leq B_{\theta}\).

To simplify notation, suppose that \(\lambda^{(1)} = \min\left\{1, \sigma_{\max}(A)/\tilde{L}\right\} = 1\). (4.10) implies that for all \(k \geq 1\)

\[
|x^{(k)} - x^*|_2 \leq \frac{2}{\tilde{\tau}} \left[\tilde{F}^* + \tilde{\tau} \|x^*\|_2 + \frac{1}{\tilde{\tau}} \left(B_{\tilde{\theta}} + \frac{\tilde{\tau}^2}{4N}\right)\right] := B_x.
\]

Note that (4.14) implies that \(\frac{\xi^{(1)}}{(\tilde{\tau})^2} B_x = \frac{1}{\tilde{\tau}^2} B_x \geq \frac{1}{\tilde{\tau}} B_{\tilde{\theta}}^2 + \frac{\tilde{\tau}^2}{8N} \geq \tilde{\tau}^2 \geq \frac{\xi^{(1)}}{(\tilde{\tau})^2},\) where we used the fact \(B_{\tilde{\theta}} \geq \frac{\sigma_{\max}(A)}{\max_{i \in \mathcal{N}} \left\{\sigma_{\min}(A_i)\right\}} \frac{\tilde{\tau}}{\sqrt{N}} \geq \frac{\tilde{\tau}}{\sqrt{N}}\). Note that the last inequality follows from our assumption on \(A\) stated at the beginning of the proof, i.e. \(\sigma_{\max}(A) \geq \sqrt{\max_{i \in \mathcal{N}} d_i + 1}\) and \(\sigma_{\min}(A_i) = d_i\) for all \(i \in \mathcal{N}\). Hence, Theorem 4.7 \(\lambda^{(1)} = 1\), and \(|\theta^*|_2 \leq B_{\theta}\) imply that

\[
N_f^{\text{DFAL}}(\epsilon) \leq \log_{\frac{1}{c}} \left(\frac{2B_{\tilde{\theta}}}{\epsilon}\right) = \log_{\frac{1}{c}} \left(\frac{2}{\min_{i \in \mathcal{N}} \left\{\frac{\tilde{\tau} \sigma_{\max}(A) \sqrt{N} + B_i + G_i}{\sigma_{\min}(A_i) \epsilon}\right\}}\right) := \tilde{N}_f,
\]
\[
N_0^{\text{DFAL}}(\epsilon) \leq \log_{\frac{1}{c}} \left(\frac{1}{\epsilon} \max\left\{\frac{1}{2} \left(\|\theta^*\|_2 + B_0\right)^2, B_{\tilde{\theta}}^2 + \tilde{F}^* + \tilde{\tau} \|x^*\|_2 + \frac{\tilde{\tau}^2}{8N}\right\}\right),
\]

67
We also show that the steps in \( C \) where
\[ \in \Omega_0 \]
to the edge \( (4.2) \) of
In this section, we show that the decentralized optimization problem
\[ \text{compute an } \epsilon \text{-feasible solution can be bounded above:} \]
\[ N^{(k)} \leq \frac{2}{\epsilon} \left( \frac{2 \lambda^{(k)} + \lambda}{\sigma_{\max}^2(A)} \right) \leq \frac{8 B_x \sqrt{N}}{\tau} \sigma_{\max}(A) c^{-k}. \] (4.17)
Hence, (4.15) and (4.17) imply that the total number of MS-APG iterations to compute an \( \epsilon \)-feasible solution can be bounded above:
\[
\sum_{k=1}^{N_{\text{DFAL}}(\epsilon)} N^{(k)} \leq \frac{8 B_x \sqrt{N}}{\tau} \sigma_{\max}(A) \sum_{k=1}^{N_f} c^{-k} \leq \frac{8 B_x \sqrt{N}}{c(1-c)\tau} \sigma_{\max}(A) \left( \frac{1}{c} \right)^{N_f},
\]
\[
\leq \frac{16 B_x \sqrt{N}}{c(1-c)\tau} \min_{i \in N} \left\{ \frac{\tau \sigma_{\max}(A)/\sqrt{N} + B_i + G_i}{\sigma_{\min}(A_i)\epsilon} \right\} \sigma_{\max}(A) = \mathcal{O}\left( \frac{\sigma_{\max}^2(A)}{\min_{i \in N} \sigma_{\min}(A_i) \epsilon} \right).
\] (4.18)
Similarly, (4.16) and (4.17) imply that the total number of MS-APG iterations to compute an \( \epsilon \)-optimal solution can be bounded above:
\[
\sum_{k=1}^{N_{\text{DFAL}}(\epsilon)} N^{(k)} \leq \frac{8 B_x \sqrt{N}}{c(1-c)\tau} \sigma_{\max}(A) \left( \frac{1}{c} \right)^{N_0} \sigma_{\max}^3 \left( \frac{1}{\min_{i \in N} \sigma_{\min}(A_i) \epsilon} \right).
\]

4.2.3 Synchronous algorithm for distributed optimization

In this section, we show that the decentralized optimization problem (3.4) is a special case of (4.2); therefore, Theorem 4.11 establishes the Main Result stated in the Introduction.

We also show that the steps in DFAL can be further simplified in this context.

Construct a directed graph by introducing an arc \((i, j)\) where \( i < j \) for every edge \((i, j)\) in the undirected graph \( G = (N, E) \). Then the constraints \( x_i - x_j = 0 \) for all \((i, j) \in E\) in the distributed optimization problem (3.4) can be reformulated as \( Cx = 0 \), where \( C \in \mathbb{R}^{n|E| \times nN} \) is a block matrix such that the block \( C_{(i,j),l} \in \mathbb{R}^{n \times n} \) corresponding to the edge \((i, j) \in E\) and node \( l \in N\), i.e., \( C_{(i,j),l} \) is equal to \( I_n \) if \( l = i \), \( -I_n \) if \( l = j \), and \( 0_n \) otherwise, where \( I_n \) and \( 0_n \) denote \( n \times n \) identity and zero matrices, respectively. Let \( \Omega \in \mathbb{R}^{N \times N} \) be the Laplacian of \( G \), i.e., for all \( i \in N \), \( \Omega_{ii} = d_i \), and for all \((i, j) \in N \times N \) such that \( i \neq j \), \( \Omega_{ij} = -1 \) if either \((i, j) \in E\) or \((j, i) \in E\), where \( d_i \) denotes the degree of
\( i \in \mathcal{N} \). Then it follows that

\[
\Psi := C^T C = \Omega \otimes I_n,
\]

where \( \otimes \) denotes the Kronecker product. Let \( \psi_{\text{max}} := \psi_1 \geq \psi_2 \geq \ldots \geq \psi_N \) be the eigenvalues of \( \Omega \). Since \( \mathcal{G} \) is connected, \( \text{rank}(\Omega) = N - 1 \), i.e., \( \psi_{N-1} > 0 \) and \( \psi_N = 0 \). From the structure of \( \Psi \) it follows that that \( \{ \psi_i \}_{i=1}^N \) are also the eigenvalues of \( \Psi \), each with algebraic multiplicity \( n \). Hence, \( \text{rank}(C) = n(N - 1) \).

Let \( C = U \Sigma V^T \) denote the reduced singular value decomposition (SVD) of \( C \), where \( U \in \mathbb{R}^{n|\mathcal{E}| \times n(N-1)} \), \( \Sigma = \text{diag}(\sigma) \), \( \sigma \in \mathbb{R}^{n(N-1)} \), and \( V \in \mathbb{R}^{nN \times n(N-1)} \). Note that \( \sigma_{\text{max}}^2(C) = \psi_{\text{max}} \), and \( \sigma_{\text{min}}^2(C) = \psi_{N-1} \). Define \( A := \Sigma V^T \). \( A \in \mathbb{R}^{n(N-1) \times nN} \) has linearly independent rows; more importantly, \( A^T A = C^T C = \Psi \); hence, \( \sigma_{\text{max}}(A) = \psi_{\text{max}} \), and \( \sigma_{\text{min}}^2(A) = \psi_{N-1} \). We also have \( \{x \in \mathbb{R}^{nN} : Ax = 0\} = \{x \in \mathbb{R}^{nN} : Cx = 0\} \). Hence, the general problem in (4.2) with \( A := \Sigma V^T \) and \( b = 0 \in \mathbb{R}^{n(N-1)} \) is equivalent to (4.4). Let \( A_i \in \mathbb{R}^{n(N-1) \times n} \) and \( C_i \in \mathbb{R}^{n|\mathcal{E}| \times n} \) be the submatrices of \( A \) and \( C \), respectively, corresponding to \( x_i \), i.e., \( A = [A_1, A_2, \ldots, A_N] \), and \( C = [C_1, C_2, \ldots, C_N] \). Clearly, it follows from the definition of \( C \) that \( \sigma_{\text{max}}(C_i) = \sigma_{\text{min}}(C_i) = \sqrt{d_i} \) for all \( i \in \mathcal{N} \). Using the property of SVD, it can also be shown for \( A = \Sigma V^T \) that \( \sigma_{\text{max}}(A_i) = \sigma_{\text{min}}(A_i) = \sqrt{d_i} \) for all \( i \in \mathcal{N} \). Thus, Theorem 4.11 establishes the Main Result.

We now show that we do not have to compute the SVD of \( C_i \) or \( A \), or even the dual multipliers \( \theta^{(k)} \) when \( \text{DFAL} \) is used to solve (3.4). In \( \text{DFAL} \) the matrix \( A \) is used in Step 1 (i.e. within the oracle \( \text{MS-APG} \)) to compute \( \nabla f^{(k)} \), and in Step 2 to compute \( \theta^{(k+1)} \). Since \( \theta^{(1)} = \mathbf{0} \), Step 2 in \( \text{DFAL} \) and (4.4) imply that \( \theta^{(k+1)} = -\sum_{t=1}^k A_{x_{(t)}} \), and

\[
\nabla f^{(k)}(x) = \lambda^{(k)} \nabla \bar{g}(x) + A^T (Ax - \lambda^{(k)} \theta^{(k)}) = \lambda^{(k)} \nabla \bar{g}(x) + \Psi (x + \lambda^{(k)} \sum_{t=1}^{k-1} \frac{1}{\lambda^{(t)}} x_{(t)}).
\]

Moreover, from the definition of \( \Psi \), it follows that

\[
\nabla x_i f^{(k)}(x) = \lambda^{(k)} \nabla \gamma_i(x_i) + d_i \left( x_i + \bar{x}_{i}^{(k)} \right) - \sum_{j \in \mathcal{O}_i} \left( x_j + \bar{x}_{j}^{(k)} \right),
\]

where \( \bar{x}^{(k)} := \sum_{t=1}^{k-1} \frac{\lambda^{(k)}}{\lambda^{(t)}} x_{(t)} \), and \( \mathcal{O}_i \) denotes the set of nodes adjacent to \( i \in \mathcal{N} \). Thus, it follows that Step 1 of \( \text{MS-APG} \) can be computed in a distributed manner by only communicating with the adjacent nodes without explicitly computing \( \theta^{(k)} \) in Step 2 of \( \text{DFAL} \).

In particular, for the \( k \)-th \( \text{DFAL} \) iteration, each node \( i \in \mathcal{N} \) stores \( \bar{x}^{(k)}_i \) and \( \{ \bar{x}^{(k)}_{j} \}_{j \in \mathcal{O}_i} \), which can be easily computed locally if \( \{ x^{(t)}_j \}_{j \in \mathcal{O}_i} \) is transmitted to \( i \) at the end of Step 1 of the previous \( \text{DFAL} \) iterations \( 1 \leq t \leq k - 1 \). Hence, during the \( \ell \)-th iteration of
MS-APG\(\left(\lambda^{(k)}\bar{\rho}, f^{(k)}, x^{(k-1)}\right)\) call, each node \(i \in \mathcal{N}\) can compute \(\nabla_{y_i} f^{(k)}(\bar{x}^{(t)})\) locally if \(\{\bar{y}_j^{(t)}\}_{j \in \mathcal{O}_i}\) is transmitted to \(i\) at the end of Step 3 in MS-APG. It is important to note that every node can independently check (4.5)(b), i.e., \(\exists g_i^{(k)} \in \partial \rho_i (x_i) |_{x_i=x_i^{(k)}} + \nabla_{x_i} f^{(k)}(x_i^{(k)})\) for all \(i \in \mathcal{N}\) such that \(\max_{i \in \mathcal{N}} \|g_i^{(k)}\|_2 \leq \frac{\epsilon^{(k)}}{\sqrt{N}}\). Hence, nodes can reach a consensus to move to the next DFAL iteration without communicating their private information. If (4.5)(b) does not hold for \(\ell_{\max}^{(k)} := B_x \sqrt{\sum_{i \in \mathcal{N}} \frac{L_i^{(k)}}{\alpha_i}}\) MS-APG iterations, then Lemma 4.10 implies that (4.5)(a) must be true. Hence, all the nodes move to next DFAL iteration after \(\ell_{\max}^{(k)}\) many MS-APG updates. For implementational version of DFAL, see Figure 4.3 where \(B_x\) is the bound in Theorem 4.6 and \(\mathcal{N}_i := \mathcal{O}_i \cup \{i\}\).

Algorithm DFAL \((x^{(0)}, \lambda^{(1)}, \alpha^{(1)}, \xi^{(1)}, B_x, \psi_{\text{max}})\)

1: \(k \leftarrow 1, \quad \bar{x}_i^{(1)} \leftarrow 0, \quad \forall i \in \mathcal{N}\)
2: while \(k \geq 1\) do
3: \(\ell \leftarrow 1, \quad t^{(1)} \leftarrow 1, \quad \text{STOP} \leftarrow \text{false}\)
4: \(y_i^{(0)} \leftarrow x_i^{(k-1)}, \quad \bar{y}_i^{(1)} \leftarrow x_i^{(k-1)}, \quad \forall i \in \mathcal{N}\)
5: \(L_i^{(k)} \leftarrow \lambda^{(k)} L_i + \psi_{\max}, \forall i \in \mathcal{N}\)
6: \(\ell_{\max}^{(k)} \leftarrow B_x \sqrt{\sum_{i \in \mathcal{N}} \frac{L_i^{(k)}}{\alpha_i}}\)
7: while STOP = false do
8: for \(i \in \mathcal{N}\) do
9: \(q_i^{(t)} \leftarrow \lambda^{(k)} \nabla_{y_i} \left(\bar{y}_i^{(t)} \right) + \sum_{j \in \mathcal{N}_i} \Omega_{ij} \left(\bar{y}_j^{(t)} + \bar{x}_j^{(k)}\right)\)
10: \(y_i^{(t)} \leftarrow \text{prox}_{\lambda^{(k)} \rho_i / L_i^{(k)}} \left(\bar{y}_i^{(t)} - q_i^{(t)} / L_i^{(k)}\right)\)
11: end for
12: if \(\exists g_i \in q_i^{(t)} + \lambda^{(k)} \partial \rho_i (\bar{y}_i^{(t)})\) s.t. \(\max_{i \in \mathcal{N}} \|g_i\|_2 \leq \frac{\epsilon^{(k)}}{\sqrt{N}}\) then
13: \(\text{STOP} \leftarrow \text{true}, \quad x_i^{(k)} \leftarrow \bar{y}_i^{(t)}, \quad \forall i \in \mathcal{N}\)
14: else if \(\ell = \ell_{\max}^{(k)}\) then
15: \(\text{STOP} \leftarrow \text{true}, \quad x_i^{(k)} \leftarrow y_i^{(t)}, \quad \forall i \in \mathcal{N}\)
16: end if
17: \(t^{(t+1)} \leftarrow (1 + \sqrt{1 + 4 \left(t^{(t)}\right)^2}) / 2\)
18: \(\bar{y}_i^{(t+1)} \leftarrow y_i^{(t)} + \frac{t^{(t)}-1}{t^{(t+1)}} \left(y_i^{(t)} - y_i^{(t-1)}\right), \quad \forall i \in \mathcal{N}\)
19: \(\ell \leftarrow \ell + 1\)
20: end while
21: \(\lambda^{(k+1)} \leftarrow c \lambda^{(k)}, \quad \alpha^{(k+1)} \leftarrow c^2 \alpha^{(k)}, \quad \xi^{(k+1)} \leftarrow c^2 \xi^{(k)}\)
22: \(\bar{x}_i^{(k+1)} \leftarrow \lambda^{(k+1)} \frac{\xi^{(k+1)}}{\lambda^{(k+1)}} \left(\bar{x}_i^{(k)} + x_i^{(k)}\right), \quad \forall i \in \mathcal{N}\)
23: \(k \leftarrow k + 1\)
24: end while

Figure 4.3: Dist. First-order Aug. Lagrangian (DFAL) alg.
4.2.4 Asynchronous implementation

Here we propose an asynchronous version of DFAL. For the sake of simplicity of the exposition, in this section we only consider a simple randomized block coordinate descent (RBCD) method, which will lead to an asynchronous implementation of DFAL that can compute an $\epsilon$-optimal and $\epsilon$-feasible solution to (3.4) with probability $1 - p$ within $O\left(\frac{1}{\epsilon^2} \log \left(\frac{1}{p}\right)\right)$ RBCD iterations. In Section 4.2.5, we discuss how to improve this rate to $O\left(\frac{1}{\epsilon} \log \left(\frac{1}{p}\right)\right)$ using an accelerated RBCD.

Nesterov [82] proposed an RBCD method for solving $\min_{y \in \mathbb{R}^n} f(y)$, where $f$ is convex with block Lipschitz continuous gradient, i.e., $\nabla_y f(y; y_{-i})$ is Lipschitz continuous in $y_i$ with constant $L_i$ for all $i$. Later, [83] extended the convergence rate results to $\min_{y \in \mathbb{R}^n} \Phi(y) := \sum_{i=1}^{N} \rho_i (y_i) + f(y)$, such that $\text{prox}_{\rho_i}$ can be computed efficiently for all $t > 0$ and $i \in \mathcal{N}$, and established that given $\alpha > 0$, and $p \in (0, 1)$, for $\ell \geq \frac{2NC}{\alpha} \left(1 + \log \frac{1}{p}\right)$, the iterate sequence $\{y^{(\ell)}\}$ computed by RBCD displayed in Fig. 4.4 satisfies

$$\mathbb{P}(\Phi(y^{(\ell)}) - \Phi^* \leq \alpha) \geq 1 - p,$$

where $C := \max \{\mathcal{R}_{\alpha}^1(y^{(0)}), \Phi(y^{(0)}) - \Phi^*\}$, $\mathcal{R}_{\alpha}^2(y^{(0)}) := \max_{y \in \mathcal{Y}^*} \{\sum_{i=1}^{N} L_i \|y_i - y_i^*\|^2_2 : \Phi(y) \leq \Phi(y^{(0)}), y^* \in \mathcal{Y}^*\}$, and $\mathcal{Y}^*$ denotes the set of optimal solutions. RBCD is significantly faster in practice for very large scale problems, particularly when the partial gradient $\nabla_y f(y)$ can be computed more efficiently as compared to the full gradient $\nabla f(y)$. The RBCD algorithm can be implemented for the distributed minimization problem when the nodes in $\mathcal{G}$ work asynchronously. Assume that for any $y = (y_i)_{i \in \mathcal{N}} \in \mathbb{R}^{nN}$, each node $i$ is equally likely to be the first to complete computing $\text{prox}_{\rho_i/L_i} (y_i - \nabla_{y_i} f(y)/L_i)$, i.e., each node has an exponential clock with equal rates. Suppose node $i \in \mathcal{N}$ is the first node to complete Step 2 of RBCD. Then, instead of waiting for the other nodes to finish, node $i$ sends a message to its neighbors $j \in \mathcal{O}_i$ to terminate their computations, and shares $y_i^{(\ell+1)}$ with them. Note that RBCD can be easily incorporated into DFAL as an oracle to solve subproblems in (4.3) by replacing (4.5) with

$$\mathbb{P}\left(P^{(k)}(x^{(k)}) - P^{(k)}(x_i^{(k)}) \leq \alpha^{(k)}\right) \geq \left(1 - p\right)^{\frac{1}{N(\epsilon)}},$$

where $N(\epsilon) = \log_{\frac{\epsilon}{\alpha}} \left(\frac{\epsilon}{\tau}\right)$ defined in Corollary 4.8. Since $\left(1 - p\right)^{\frac{1}{N(\epsilon)}} \leq 1 - \frac{p}{N(\epsilon)}$ for $p \in (0, 1)$, the total number of RBCD iterations for the $k$-th subproblem is bounded: $N^{(k)} \leq O\left(\frac{1}{\alpha \epsilon \tau} \log \left(\frac{N(\epsilon)}{p}\right)\right) = O\left(\frac{1}{\alpha \epsilon \tau} \left(\log \left(\frac{1}{p}\right) + \log \left(\frac{1}{\tau}\right)\right)\right)$. Hence, Corollary 4.8 and
 imply that asynchronous DFAL, i.e., \((4.5)\text{(a)}\) replaced with \((4.20)\), can compute an \(\epsilon\)-optimal and \(\epsilon\)-feasible solution to \((3.4)\) with probability \(1 - p\) within \(\mathcal{O}\left(\frac{1}{\epsilon^2 \log \left(\frac{1}{p}\right)}\right)\) RBCD iterations. These results can be extended to the case where each node has different clock rates using \([84]\).

**Algorithm RBCD \((\bar{\rho}, f, y^{(0)})\)**

Step \(\ell\) (\(\ell \geq 0\))
1. \(i \in \mathcal{N}\) is realized with probability \(\frac{1}{N}\)
2. \(y_i^{(\ell+1)} = \text{prox}_{\rho_i/L_i}\left(y_i^{(\ell)} - \nabla y_i f(y^{(\ell)})/L_i\right)\)
3. \(y_{-i}^{(\ell+1)} = y_{-i}^{(\ell)}\)

Figure 4.4: Randomized Block Coordinate Descent (RBCD) alg.

### 4.2.5 Improved rate for asynchronous DFAL

Let \(\mathcal{R}\) denote a discrete random variable uniformly distributed over the set \(\mathcal{N}\). Let \([U_1, U_2, \ldots, U_N]\) denote a partition of the \(nN\)-dimensional identity matrix where \(U_i \in \mathbb{R}^{nN \times n}, i = 1, \ldots, N\). In the rest, given \(h \in \mathbb{R}^{nN}\), we denote \(h_{[\mathcal{R}]} := U_{\mathcal{R}} U_{\mathcal{R}}^\top h\). Consider the composite convex optimization problem

\[
\Phi^* := \min_{y \in \mathbb{R}^{nN}} \Phi(y) := \sum_{i=1}^{N} \rho_i(y_i) + f(y),
\]

where \(\rho_i : \mathbb{R}^n \to \mathbb{R}\) is a closed convex function for all \(i \in \mathcal{N}\) such that \(\text{prox}_{t \rho_i}\) can be computed efficiently for all \(t > 0\) and \(i \in \mathcal{N}\), and \(f : \mathbb{R}^{nN} \to \mathbb{R}\) is a differentiable convex function such that for some \(\{L_i\}_{i \in \mathcal{N}} \subset \mathbb{R}_{++}\), \(f\) satisfies

\[
\mathbb{E}[f(y + h_{[\mathcal{R}]})] \leq f(y) + \frac{1}{N} \left(\langle \nabla f(y), h \rangle + \frac{1}{2} \sum_{i \in \mathcal{N}} L_i \|h_i\|_2^2\right)
\]

for all \(y, h \in \mathbb{R}^{nN}\). \([85]\) proposed the accelerated proximal coordinate descent algorithm \(\text{ARBBCD}\) (see Figure 4.5) to solve \((4.21)\). They showed that for a given \(\alpha > 0\), the iterate sequence \(\{z^{(\ell)}, u^{(\ell)}\}\) computed by \(\text{ARBBCD}\) satisfies

\[
\mathbb{E}\left[\Phi\left(\left(\frac{1}{Nt^{(\ell)}}\right)^2 u^{(\ell+1)} + z^{(\ell+1)}\right) - \Phi^*\right] \leq \alpha, \quad \forall \ell \geq 2N\sqrt{\frac{C}{\alpha}},
\]

(4.23)
where
\[ C := \min_{y^* \in Y^*} \left( 1 - \frac{1}{N} \right) \left( \Phi(z^{(0)}) - \Phi^* \right) + \frac{1}{2} \sum_{i \in \mathcal{N}} L_i \| z_i^{(0)} - y_i^* \|_2^2, \tag{4.24} \]

and \( Y^* \) denotes the set of optimal solutions.

**Algorithm ARBCD** \((z^{(0)})\)

1: \( \ell \leftarrow 0, \ t^{(0)} \leftarrow 1, \ u_i^{(1)} \leftarrow 0, \ \forall i \in \mathcal{N} \)
2: **while** \( \ell \geq 0 \) **do**
3: \( i \) is a sample of \( \mathcal{R} \)
4: \( z_i^{(\ell+1)} \leftarrow \text{prox}_{t^{(\ell)} \rho_i / L_i} \left( z_i^{(\ell)} - \frac{t^{(\ell)}}{L_i} \nabla y_i f \left( \frac{1}{Nt^{(\ell)}} \right) \right) \)
5: \( u_i^{(\ell+1)} \leftarrow u_i^{(\ell)} + N^2 t^{(\ell)} (1 - t^{(\ell)}) \left( z_i^{(\ell+1)} - z_i^{(\ell)} \right) \)
6: \( z_{-i}^{(\ell+1)} \leftarrow z_{-i}^{(\ell)}, \ u_{-i}^{(\ell+1)} \leftarrow u_{-i}^{(\ell)} \)
7: \( t^{(\ell+1)} \leftarrow 1 + \sqrt{1 + \frac{2N t^{(\ell)}}{2}} \)
8: **end while**

**Figure 4.5:** Accelerated Randomized Proximal Block Coordinate Descent (ARBCD) algorithm

In the following result, we establish that the bound \((4.22)\) can be exploited for designing an accelerated version of asynchronous DFAL.

**Lemma 4.12.** Fix \( \alpha > 0 \), and \( p \in (0, 1) \). Let \( \{z_k^{(\ell)}, u_k^{(\ell)}\}_{\ell \in \mathbb{Z}_+} \), \( k = 1, \ldots, K \), denote the iterate sequence corresponding to \( K := \log(1/p) \) independent calls to ARBCD\((y^{(0)})\).

Define \( y_k := \left( \frac{1}{Nt^{(T)}} \right)^2 u_k^{(T+1)} + z_k^{(T+1)} \) for \( k = 1, \ldots, K \), and \( T := 2N \sqrt{\frac{2C}{\alpha}} \). Then
\[
\mathbb{P} \left( \min_{k=1,\ldots,K} \Phi(y_k) - \Phi^* \leq \alpha \right) \geq 1 - p.
\]

**Proof:** Since the sequence \( \{y_k\}_{k=1}^K \) is i.i.d., and each \( y_k \) satisfies \( \mathbb{E}[\Phi(y_k) - \Phi^*] \leq \frac{\alpha}{2} \), Markov’s inequality implies that \( \mathbb{P}(\Phi(y_k) - \Phi^* > \alpha) \leq \mathbb{E}[\Phi(y_k) - \Phi^*] / \alpha \leq \frac{1}{2} \) for \( 1 \leq k \leq K \). Therefore, we have
\[
\mathbb{P} \left( \min_{k=1,\ldots,K} \Phi(y_k) - \Phi^* \leq \alpha \right) = 1 - \prod_{k=1}^K \mathbb{P}(\Phi(y_k) - \Phi^* > \alpha) \leq \left( \frac{1}{2} \right)^K = 1 - p.
\]

From Lemma 4.12 it follows that we can compute \( y_\alpha \) such that \( \mathbb{P}(\Phi(y_\alpha) - \Phi^* \leq \alpha) \geq 1 - p \) in at most \( 2N \sqrt{\frac{2C}{\alpha}} \log(\frac{1}{p}) \) ARBCD iterations. This new oracle can be used to construct an asynchronous version of DFAL algorithm with \( \mathcal{O}(1/\epsilon) \) complexity.
Theorem 4.13. Fix $\epsilon > 0$ and $p \in (0, 1)$. Consider an asynchronous variant of DFAL where $\left[4.5\right](a)$ in Figure 4.1 is replaced by

$$\mathbb{P} \left( P^{(k)}(x^{(k)}) - P^{(k)}(x_{\ast}^{(k)}) \leq \alpha^{(k)} \right) \geq (1 - p) \frac{1}{\sqrt{\epsilon}}, \quad (4.25)$$

where $N(\epsilon) = \log_2 \left( \frac{C}{\epsilon} \right)$ is defined in Corollary 4.8. Then $\{x_i^{(N(\epsilon))}\}_{i \in \mathcal{N}}$ satisfies

$$\mathbb{P}(\epsilon) := \mathbb{P} \left( \left\lvert \sum_{i \in \mathcal{N}} F_i \left( x_i^{(N(\epsilon))} \right) - F^{\ast} \right\rvert \leq \epsilon, \quad \max_{(i,j) \in E} \left\{ \|x_i^{(N(\epsilon))} - x_j^{(N(\epsilon))}\|_2 \right\} \leq \epsilon \right) \geq 1 - p,$$

and $O \left( \frac{1}{\epsilon} \log \left( \frac{1}{p} \right) \right)$ ARBCD iterations are required to compute $\{x_i^{(N(\epsilon))}\}_{i \in \mathcal{N}}$.

Proof: Consider the $k$-th DFAL subproblem $\min P^{(k)}(x) := \lambda^{(k)} \sum_{i \in \mathcal{N}} \rho_i(x_i) + f^{(k)}(x)$, where $f^{(k)}$ is defined in (4.4). Let $\hat{L}_i^{(k)} := \lambda^{(k)} L_{\gamma_i} + d_i$ for all $i \in \mathcal{N}$. Then it can be easily shown that $f^{(k)}$ satisfies $\left[4.22\right]$ with constants $\{\hat{L}_i^{(k)}\}_{i \in \mathcal{N}}$ for all $1 \leq k \leq N(\epsilon)$. Hence, ARBCD algorithm can be used to solve $\min P^{(k)}(x)$ with the iteration complexity given in Lemma 4.12. Consider the random event

$$\Delta := \bigcap_{k=1}^{N(\epsilon)} \left\{ P^{(k)}(x^{(k)}) - P^{(k)}(x_{\ast}^{(k)}) \leq \alpha^{(k)} \right\} \quad \text{or}$$

$$\exists g_i^{(k)} \in \partial x_i P^{(k)}(x) \big|_{x=x^{(k)}} \text{ s.t. } \max_{i \in \mathcal{N}} \|g_i^{(k)}\|_2 \leq \frac{\xi^{(k)}}{\sqrt{N}}, \quad (4.26)$$

Clearly, for all random sequences $\{x_i^{(k)}\}_{k=1}^{N(\epsilon)}$ satisfying random event $\Delta$, Corollary 4.8 implies that $\left\lvert \sum_{i \in \mathcal{N}} F_i \left( x_i^{(N(\epsilon))} \right) - F^{\ast} \right\rvert \leq \epsilon$ and $\max_{(i,j) \in E} \left\{ \|x_i^{(N(\epsilon))} - x_j^{(N(\epsilon))}\|_2 \right\} \leq \epsilon$. Hence, we have

$$\mathbb{P}(\epsilon) \geq \mathbb{P}(\Delta) \geq \prod_{k=1}^{N(\epsilon)} \mathbb{P} \left( P^{(k)}(x^{(k)}) - P^{(k)}(x_{\ast}^{(k)}) \leq \alpha^{(k)} \right) \geq 1 - p.$$

In the rest, we bound the total number of ARBCD iterations required by asynchronous variant of DFAL to compute $x^{(N(\epsilon))}$. Note that $(1 - p) \frac{1}{\sqrt{\epsilon}}$ is a concave function for $p \in (0, 1)$, and we have $(1 - p) \frac{1}{\sqrt{\epsilon}} \leq 1 - \frac{p}{N(\epsilon)}$. Therefore, Lemma 4.12 and the discussion after Lemma 4.12 together imply that the number of ARBCD iterations, $N^{(k)}$, to compute $x^{(k)}$ satisfying either $\left[4.25\right]$ or $\left[4.5\right](b)$ is bounded above for $1 \leq k \leq N(\epsilon)$ as follows

$$N^{(k)} \leq 2N \sqrt{\frac{2C^{(k)}}{\alpha^{(k)}}} \log \left( \frac{N(\epsilon)}{p} \right) = 2N \left( \log \left( \frac{1}{p} \right) + \log \log \left( \frac{C'}{\epsilon} \right) \right) \sqrt{\frac{2C^{(k)}}{\alpha^{(k)}}}, \quad (4.27)$$
with $C^{(k)} = P^{(k)}(x^{(k-1)}) - P^{(k)}(x^{(k)}) + \sum_{i \in N} \frac{L_i^{(k)}}{2} \|x_i^{(k-1)} - x_{*i}^{(k)}\|^2$.

Convexity of $\{\rho_i\}_{i \in N}$, and Lemma 4.4 imply that

$$P^{(k)}(x^{(k-1)}) - P^{(k)}(x^{(k)}) \leq \left< \lambda^{(k)} s^{(k)} + \nabla f^{(k)}(x^{*}), x^{(k-1)} - x^{*} \right> + \sum_{i \in N} \frac{L_i^{(k)}}{2} \|x_i^{(k-1)} - x_{*i}^{(k)}\|^2,$$

where $s^{(k)} \in \partial \lambda^{(k)} \tilde{\rho}(x)|_{x=x^{(k-1)}}$, and $\tilde{\rho}(x) = \sum_{i \in N} \rho_i(x_i)$. Note that optimality conditions imply that $-\nabla f^{(k)}(x^{*}) \in \partial \lambda^{(k)} \tilde{\rho}(x)|_{x=x^{*}}$. Assumption 4.2 implies that $\|\nabla x_i f^{(k)}(x^{*})\|_2 \leq \lambda^{(k)} B_i$ and $\|s_i^{(k)}\|_2 \leq \lambda^{(k)} B_i$ for all $i \in N$. Hence, for some $\tilde{C} > 0$, we have $C^{(k)} \leq \sum_{i \in N} \left( \frac{L_i^{(k)} + L_i^{(k)}}{2} + 2\lambda^{(k)} B_i \right) \|x_i^{(k-1)} - x_{*i}^{(k)}\|^2 \leq \tilde{C} B^2$ for all $k \geq 1$. Consequently, we can bound the total number of ARBCD iterations to compute $x^{N(\epsilon)}$ as follows:

$$\sum_{k=1}^{N(\epsilon)} N^{(k)} \leq 2NBx \sqrt{\frac{2\tilde{C}}{\alpha(0)}} \left( \log \left( \frac{1}{\epsilon} \right) + \log \log \frac{\tilde{C}}{\epsilon} \right) \sum_{k=1}^{N(\epsilon)} c^{-k}.$$ 

Since $N(\epsilon) = \log \frac{1}{\epsilon}$, and $\sum_{k=1}^{N(\epsilon)} c^{-k} = \frac{(\frac{1}{\epsilon})^{N(\epsilon)} - 1}{1-c} = \tilde{C}^{-1}/(1-c)$. Hence, we can conclude that $\sum_{k=1}^{N(\epsilon)} N^{(k)} = O\left( \frac{\tilde{C}}{\epsilon} \left( \log \left( \frac{1}{\epsilon} \right) + \log \log \left( \frac{1}{\epsilon} \right) \right) \right)$. 

### 4.3 Numerical results

In this section, we compared DFAL with an ADMM method proposed in [70] on the sparse group LASSO problem with Huber loss:

$$\min_{x \in \mathbb{R}^n} \sum_{i=1}^{N} \beta_1 \|x\|_1 + \beta_2 \|x\|_{G_i} + \sum_{j=1}^{m_i} h_{\delta} (a_i^T (j) x - b_i),$$

where $\beta_1, \beta_2 > 0$, $A_i \in \mathbb{R}^{m_i \times n}$, $b_i \in \mathbb{R}^{m_i}$, $a_i^T (j)$ denotes the $j$-th row of $A_i$, the Huber loss function $h_{\delta}(x) := \max\{tx - t^2/2 : t \in [-\delta, \delta]\}$, and $\|x\|_{G_i} := \sum_{k=1}^{K} \|x_{g_i(k)}\|_2$ denotes the group norm with respect to the partition $G_i$ of $[1, n] := \{1, \cdots, n\}$ for all $i \in N$, i.e., $G_i = \{g_i(k)\}_{k=1}^{K}$ such that $\bigcup_{k=1}^{K} g_i(k) = [1, n]$, and $g_i(j) \cap g_i(k) = \emptyset$ for all $j \neq k$. In this case, $\gamma_i(x) := \sum_{j=1}^{m_i} h_{\delta} (a_i^T (j) x - b_i)$ and $\rho_i(x) := \beta_1 \|x\|_1 + \beta_2 \|x\|_{G_i}$. Next, we briefly describe the ADMM algorithm in [70], and propose a more efficient variant, SADMM, for (5.39).
Algorithm SADMM \( (c, x^{(0)}) \)

Initialization: \( y^{(0)} = x^{(0)}, p^{(k)}_i = \tilde{p}^{(k)}_i = 0, \quad i \in \mathcal{N} \)

Step \( \ell \): (\( \ell \geq 0 \)) For \( i \in \mathcal{N} \) compute
1. \( x^{(k+1)}_i = \text{prox}_{\frac{1}{c(d_i^2 + d_i + 1)}}(\tilde{x}^{(k)}_i) \)
2. \( y^{(k+1)}_i = \text{prox}_{\frac{1}{c(d_i^2 + d_i + 1)}}(\tilde{y}^{(k)}_i) \)
3. \( s^{(k+1)}_i = \sum_{j \in \mathcal{N}_i} \Omega_{ij} x^{(k+1)}_j / (d_i + 1) \)
4. \( p^{(k+1)}_i = p^{(k)}_i + s^{(k+1)}_i \)
5. \( \tilde{s}^{(k+1)}_i = \sum_{j \in \mathcal{N}_i} \Omega_{ij} y^{(k+1)}_j / (d_i + 1) \)
6. \( \tilde{p}^{(k+1)}_i = \tilde{p}^{(k)}_i + \tilde{s}^{(k+1)}_i \)

Figure 4.6: Split ADMM algorithm

4.3.1 A distributed ADMM algorithm

As defined in Section 4.2.3, given a connected graph \( G = (\mathcal{N}, \mathcal{E}) \), let \( \Omega \in \mathbb{R}^{\mathcal{N} \times \mathcal{N}} \) denote the Laplacian of the graph \( G = (\mathcal{N}, \mathcal{E}) \), \( \mathcal{O}_i \) denote the set of neighboring nodes of \( i \in \mathcal{N} \), and define \( \mathcal{N}_i := \mathcal{O}_i \cup \{i\} \). Let \( \mathcal{Z}_i := \{ z_i \in \mathbb{R}^{d_i + 1} : \sum_{j \in \mathcal{N}_i} z_{ij} = 0 \} \). [70] show that (3.4) can be equivalently written as

\[
\min_{x_i \in \mathbb{R}^n, z_i \in \mathcal{Z}_i} \sum_{i=1}^N F_i(x_i) := \rho_i(x_i) + \gamma_i(x_i)
\text{s.t.} \quad \Omega_{ij} x_j = z_{ij}, \quad i \in \mathcal{N}, \quad j \in \mathcal{N}_i. \tag{4.29}
\]

Only (4.29) is penalized when forming the augmented Lagrangian, which is alternatingly minimized in \( x \in \mathbb{R}^n, z^T = [z_1^T, \ldots, z_N^T] \), where \( \mathcal{Z}_i \supseteq z_i = [z_{ij}]_{j \in \mathcal{N}_i} \in \mathbb{R}^{d_i + 1} \). [70] establish that suboptimality and consensus violation converge to 0 with a rate \( O(1/k) \), and in each iteration every node communicates 3n scalars. From now on, we refer to this algorithm that directly works with \( F_i \) as ADMM. Computing \( \text{prox}_{F_i} \) for each \( i \in \mathcal{N} \) is the computational bottleneck in each iteration of ADMM. Note that computing \( \text{prox}_{F_i} \) for (5.59) is almost as hard as solving the problem. To deal with this issue, we considered the following reformulation:

\[
\min_{x_i, y_i \in \mathbb{R}^n, z_i, \tilde{z}_i \in \mathcal{Z}_i} \sum_{i \in \mathcal{N}} \rho_i(x_i) + \gamma_i(y_i)
\text{s.t.} \quad \Omega_{ij} x_j = z_{ij}, \quad i \in \mathcal{N}, \quad j \in \mathcal{N}_i
\quad \Omega_{ij} y_j = \tilde{z}_{ij}, \quad i \in \mathcal{N}, \quad j \in \mathcal{N}_i
\quad x_i = q_i, \quad y_i = q_i, \quad i \in \mathcal{N}. \]
ADMM algorithm for this formulation is displayed in Fig. 5.6 where \( c > 0 \) denotes the penalty parameter. Steps of \textbf{SADMM} can be derived by minimizing the augmented Lagrangian alternatingly in \((x, y)\), and in \((z, \bar{z}, q)\) while fixing the other. As in [70], computing \((z, \bar{z}, q)\) can be avoided by exploiting the structure of optimality conditions. Prox centers in \textbf{SADMM} are

\[
\bar{x}_i^{(k)} = x_i^{(k)} - \frac{\sum_{j \in N_i} \Omega_{ji}(s_j^{(k)} + p_j^{(k)}) + p_i^{(k)} + (x_i^{(k)} - y_i^{(k)})}{d_i^2 + d_i + 1},
\]

\[
\bar{y}_i^{(k)} = y_i^{(k)} - \frac{\sum_{j \in N_i} \Omega_{ji}(s_j^{(k)} + p_j^{(k)}) - p_i^{(k)} - (x_i^{(k)} - y_i^{(k)})}{d_i^2 + d_i + 1},
\]

respectively; and \( r_i^{(k+1)} = r_i^{(k)} + (x_i^{(k+1)} - y_i^{(k+1)})/2 \).

### 4.3.2 Implementation details and numerical results

The following lemma shows that in \textbf{DFAL} implementation, each node \( i \in \mathcal{N} \) can check (4.5)(b) very efficiently. For \( x \in \mathbb{R} \), define \( \text{sgn}(x) \) as -1, 0 and 1 when \( x < 0 \), \( x = 0 \), and \( x > 0 \), respectively; and for \( x \in \mathbb{R}^n \), define \( \text{sgn}(x) = [\text{sgn}(x_1), \text{sgn}(x_2), \ldots, \text{sgn}(x_n)]^T \).

**Lemma 4.14.** Let \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) be a differentiable function, \( G = \{g(k)\}_{k=1}^K \) be a partition of \([1, n]\). For \( \beta_1, \beta_2 > 0 \), define \( P = \lambda \rho + f \), where \( \rho(x) := \beta_1 \|x\|_1 + \beta_2 \|x\|_G \). Then, for all \( \bar{x} \in \mathbb{R}^n \) and \( \xi > 0 \), there exists \( \nu \in \partial P(x)|_{x=\bar{x}} \) such that \( \|\nu\|_2 \leq \xi \) if and only if \( \|\pi^* + \omega^* + \nabla f(\bar{x})\|_2 \leq \xi \) for \( \pi^*, \omega^* \) such that for each \( k \), if \( \bar{x}_{g(k)} \neq \mathbf{0} \), then \( \pi^*_{g(k)} = \lambda \beta_1 \text{sgn}(\bar{x}_{g(k)}) + \left(1 - \text{sgn}(|\bar{x}_{g(k)}|)\right) \odot \eta_{g(k)}, \) and \( \omega^*_{g(k)} = \lambda \beta_2 \frac{\bar{x}_{g(k)}}{\|\bar{x}_{g(k)}\|_2} \); otherwise, if \( \bar{x}_{g(k)} = \mathbf{0} \), then \( \pi^*_{g(k)} = \eta_{g(k)} \), and \( \omega^*_{g(k)} \) equals

\[
-\left(\pi^*_{g(k)} + \nabla x_{g(k)} f(\bar{x})\right) \min \left\{1, \frac{\lambda \beta_2}{\|\pi^*_{g(k)} + \nabla x_{g(k)} f(\bar{x})\|_2}\right\},
\]

where \( \eta_{g(k)} = -\text{sgn}\left(\nabla x_{g(k)} f(\bar{x})\right) \odot \min \left\{||\nabla x_{g(k)} f(\bar{x})||, \lambda \beta_1\right\} \), and \( \odot \) denotes component-wise multiplication.

**Proof:** Given any convex function \( \rho : \mathbb{R}^n \rightarrow \mathbb{R} \) and \( \bar{x} \in \mathbb{R}^n \), in order to simplify the notation throughout the proof, \( \partial \rho(x)|_{x=\bar{x}} \subset \mathbb{R}^n \), the subdifferential of \( \rho \) at \( \bar{x} \), will be written as \( \partial \rho(\bar{x}) \). Given \( \bar{x} \in \mathbb{R}^n \), there exists \( \nu \in \partial P(\bar{x}) \) such that \( \|\nu\|_2 \leq \xi \), if and only if \( \|\nu^*\| \leq \xi \), where \( \nu^* = \text{argmin}\{\|\nu\|_2 : \nu \in \partial P(\bar{x})\} \). Note that \( \partial P(\bar{x}) = \lambda \partial \rho(\bar{x}) + \nabla f(\bar{x}), \) and

\[
\partial \rho(\bar{x}) = \beta_1 \prod_{k=1}^K \partial \|\bar{x}_{g(k)}\|_1 + \beta_2 \prod_{k=1}^K \partial \|\bar{x}_{g(k)}\|_2,
\]

(4.30)
where \( \prod \) denotes the Cartesian product. Since the groups \( \{g(k)\}_{k=1}^K \) are not overlapping with each other, the minimization problem is separable in groups. Hence, for all \( k \in [1, K] \), we have

\[
\nu^*_g(k) = \pi^*_g(k) + \omega^*_g(k) + \nabla_{x_g(k)} f(\bar{x}) \text{ such that }
\]

\[
(\pi^*_g(k), \omega^*_g(k)) = \arg\min_{\pi_g(k) + \omega_g(k) + \nabla_{x_g(k)} f(\bar{x})} \| \pi_g(k) + \omega_g(k) + \nabla_{x_g(k)} f(\bar{x}) \|^2_2
\]

s.t. \( \pi_g(k) \in \lambda \beta_1 \partial \| x_g(k) \|_1, \omega_g(k) \in \lambda \beta_2 \partial \| x_g(k) \|_2 \). \hfill (4.31)

Fix \( k \in [1, K] \). We will consider the solution to above problem in two cases. Suppose that \( \bar{x}_g(k) = 0 \). Since \( \partial \| 0 \|_1 \) is the unit \( \ell_\infty \)-ball, and \( \partial \| 0 \|_2 \) is the unit \( \ell_2 \)-ball, (4.31) can be equivalently written as

\[
(\pi^*_g(k), \omega^*_g(k)) = \arg\min_{\pi_g(k) + \omega_g(k) + \nabla_{x_g(k)} f(\bar{x})} \| \pi_g(k) + \omega_g(k) + \nabla_{x_g(k)} f(\bar{x}) \|^2_2
\]

s.t. \( \| \pi_g(k) \|_\infty \leq \lambda \beta_1, \| \omega_g(k) \|_2 \leq \lambda \beta_2 \). \hfill (4.32)

Clearly, it follows from Euclidean projection on to \( \ell_2 \)-ball that

\[
\omega^*_g(k) = - (\pi^*_g(k) + \nabla_{x_g(k)} f(\bar{x})) \min \left\{ 1, \frac{\lambda \beta_2}{\| \pi^*_g(k) + \nabla_{x_g(k)} f(\bar{x}) \|^2_2} \right\}.
\]

Hence, \( \| \pi^*_g(k) + \omega^*_g(k) + \nabla_{x_g(k)} f(\bar{x}) \|_2 = \max\{0, \| \pi^*_g(k) + \nabla_{x_g(k)} f(\bar{x}) \|_2 - \lambda \beta_2 \} \). Therefore,

\[
\pi^*_g(k) = \arg\min_{\| \pi_g(k) + \nabla_{x_g(k)} f(\bar{x}) \|_2 : \| \pi_g(k) \|_\infty \leq \lambda \beta_1} \| \pi_g(k) + \nabla_{x_g(k)} f(\bar{x}) \|_2
\]

\[
= - \text{sgn}(\nabla_{x_g(k)} f(\bar{x})) \odot \min \{ | \nabla_{x_g(k)} f(\bar{x}) |, \lambda \beta_1 \}.
\]

Now, suppose that \( \bar{x}_g(k) \neq 0 \). This implies that \( \partial \| \bar{x}_g(k) \|_2 = \{ \bar{x}_g(k)/\| \bar{x}_g(k) \|_2 \} \). Hence, when \( \bar{x}_g(k) \neq 0 \), we have \( \omega^*_g(k) = \lambda \beta_2 \bar{x}_g(k)/\| \bar{x}_g(k) \|_2 \), and the structure of \( \partial \| \cdot \|_1 \) implies that \( \pi^*_j = \lambda \beta_1 \text{sgn}(\bar{x}_j) \) for all \( j \in g(k) \) such that \( | \bar{x}_j | > 0 \); and it follows from (4.31) that for all \( j \in g(k) \) such that \( \bar{x}_j = 0 \), we have

\[
\pi^*_j = \arg\min \left\{ (\pi_j + \frac{\partial}{\partial x_j} f(\bar{x}))^2 : | \pi_j | \leq \lambda \beta_1 \right\} = - \text{sgn}(\frac{\partial}{\partial x_j} f(\bar{x})) \min \{ | \frac{\partial}{\partial x_j} f(\bar{x}) |, \lambda \beta_1 \}.
\]

Both DFAL and SADMM call for \( \text{prox}_{\rho} \). In Lemma 4.15, we show that it can be computed in closed form. On the other hand, when ADMM, and SADMM are implemented on (5.59), one needs to compute \( \text{prox}_{F_1} \) and \( \text{prox}_{\gamma_1} \), respectively; however, these proximal operations do not assume closed form solutions. Therefore, in order to be
fair, we computed them using an efficient interior point solver MOSEK (ver. 7.1.0.12).

**Lemma 4.15.** Let \( \rho(x) = \beta_1 \|x\|_1 + \beta_2 \|x\|_G \). For \( t > 0 \) and \( \bar{x} \in \mathbb{R}^n \), \( x^p = \text{prox}_{t \rho}(\bar{x}) \) is given by \( x^p_{g(k)} = \eta_{g(k)} \max\left\{1 - \frac{u_{12}}{\|u_{g(k)}\|_2}, 0\right\} \), for \( 1 \leq k \leq K \), where \( \eta' = \text{sgn}(\bar{x}) \odot \max\{\|\bar{x}\| - t \beta_1, 0\} \).

**Proof:** Since the groups are not overlapping with each other, the proximal problem becomes separable in groups. Let \( n_k := |g(k)| \) for all \( k \). Thus, it suffices to show that \( \min_{x_{g(k)} \in \mathbb{R}^{n_k}} \{\beta_1 \|x\|_1 + \beta_2 \|x_{g(k)}\|_2 + \frac{1}{2t} \|x_{g(k)} - \bar{x}_{g(k)}\|_2^2\} \) has a closed form solution as shown in the statement for some fixed \( k \). By the definition of dual norm, we have

\[
\min_{x_{g(k)} \in \mathbb{R}^{n_k}} \beta_1 \|x_{g(k)}\|_1 + \beta_2 \|x_{g(k)}\|_2 + \frac{1}{2t} \|x_{g(k)} - \bar{x}_{g(k)}\|_2^2,
\]

(4.33)

\[
= \min_{x_{g(k)} \in \mathbb{R}^{n_k}} \max_{\|u_1\|_\infty \leq \beta_1} u_1^T x_{g(k)} + \max_{\|u_2\|_2 \leq \beta_2} u_2^T x_{g(k)} + \frac{1}{2t} \|x_{g(k)} - \bar{x}_{g(k)}\|_2^2,
\]

(4.34)

\[
= \max_{\|u_1\|_\infty \leq \beta_1} \min_{\|u_2\|_2 \leq \beta_2} (u_1 + u_2)^T x_{g(k)} - \frac{t}{2} \|u_1 + u_2\|_2^2.
\]

(4.35)

Let \((u_1^*, u_2^*)\) be the optimal solution of (4.35). Since \( x^p_{g(k)} \) is optimal to (4.33), it follows from (4.34) that

\[ x^p_{g(k)} = \bar{x}_{g(k)} - t(u_1^* + u_2^*). \]

(4.36)

Note that (4.35) can be equivalently written as \( \min\{|u_1 + u_2 - 1/t \bar{x}_{g(k)}|_2 : \|u_1\|_\infty \leq \beta_1, \|u_2\|_2 \leq \beta_2\} \). Minimizing over \( u_2 \), we have

\[ u_2^*(u_1) = \left(\frac{1}{t} \bar{x}_{g(k)} - u_1\right) \min\left\{\beta_2, \frac{1}{\|1/t \bar{x}_{g(k)} - u_1\|_2}, 1\right\}. \]

(4.37)

Hence, we have

\[ u_1^* = \arg\min_{\|u_1\|_\infty \leq \beta_1} \|u_1 - 1/t \bar{x}_{g(k)}\|_2 \max\left\{1 - \frac{\beta_2}{\|u_1 - 1/t \bar{x}_{g(k)}\|_2}, 0\right\} \]

\[ = \arg\min_{\|u_1\|_\infty \leq \beta_1} \{\|u_1 - 1/t \bar{x}_{g(k)}\|_2 - \beta_2, 0\}. \]

Clearly, \( u_1^* = \arg\min_{\|u_1\|_\infty \leq \beta_1} \{(u_1 - 1/t \bar{x}_{g(k)})\|_2 = \text{sgn}(\bar{x}_{g(k)}) \min\left\{\frac{1}{t} \bar{x}_{g(k)} \min\{1/t \bar{x}_{g(k)} \|_1, \beta_1\}. \) The final result follows from combining (4.36) and (4.37). \( \square \)
Table 4.1: Comparison of DFAL, AFAL (Asynchronous DFAL), ADMM, and SADMM. (Termination time \( T=1800 \) sec.)

<table>
<thead>
<tr>
<th>Size</th>
<th>Alg.</th>
<th>Size ( n_g )</th>
<th>Case 1</th>
<th>Case 2</th>
<th>Case 1</th>
<th>Case 2</th>
<th>CPU Time (sec.)</th>
<th>Case 1</th>
<th>Case 2</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SDPT3 (C)</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>28</td>
<td>24</td>
<td>22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>APG (C)</td>
<td>5</td>
<td>1E-3</td>
<td>N/A</td>
<td>85</td>
<td>N/A</td>
<td>2173</td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DFAL (D)</td>
<td>4E-4, 7E-4</td>
<td>6E-4</td>
<td>6E-4</td>
<td>6E-4</td>
<td>1E-3</td>
<td>1E-3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AFAL (D)</td>
<td>3E-4, 8E-4</td>
<td>3E-4</td>
<td>6E-4</td>
<td>6E-4</td>
<td>1E-3</td>
<td>1E-3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADMM (D)</td>
<td>6E-5, 3E-5</td>
<td>7E-5</td>
<td>7E-5</td>
<td>1E-4</td>
<td>1E-4</td>
<td>1E-4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SADMM (D)</td>
<td>1E-4, 3E-4</td>
<td>1E-4</td>
<td>3E-4</td>
<td>1E-4</td>
<td>1E-4</td>
<td>1E-4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SDPT3 (C)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>28</td>
<td>24</td>
<td>22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>APG (C)</td>
<td>1E-3</td>
<td>N/A</td>
<td>0</td>
<td>N/A</td>
<td>10</td>
<td>N/A</td>
<td>2173</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DFAL (D)</td>
<td>4E-4, 7E-4</td>
<td>6E-4</td>
<td>6E-4</td>
<td>6E-4</td>
<td>1E-3</td>
<td>1E-3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AFAL (D)</td>
<td>3E-4, 8E-4</td>
<td>3E-4</td>
<td>6E-4</td>
<td>6E-4</td>
<td>1E-3</td>
<td>1E-3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ADMM (D)</td>
<td>6E-5, 3E-5</td>
<td>7E-5</td>
<td>7E-5</td>
<td>1E-4</td>
<td>1E-4</td>
<td>1E-4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SADMM (D)</td>
<td>1E-4, 3E-4</td>
<td>1E-4</td>
<td>3E-4</td>
<td>1E-4</td>
<td>1E-4</td>
<td>1E-4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In our experiments, the network was either a star tree or a clique with either 5 or 10 nodes. The remaining problem parameters defining \( \{\rho_i, \gamma_i\} \) were set as follows. We set \( \beta_1 = \beta_2 = \frac{1}{N}, \delta = 1, \) and \( K = 10. \) Let \( n = Kn_g \) for \( n_g \in \{100, 300\}, \) i.e., \( n \in \{1000, 300\}. \) We generated partitions \( \{G_i\} \) in two different ways. For test problems in CASE 1, we created a single partition \( G = \{g(k)\}_{k=1}^{K} \) by generating \( K \) groups uniformly at random such that \( |g(k)| = n_g \) for all \( k \); and set \( G_i = G \) for all \( i \in \mathcal{N}, \) i.e., \( \rho_i(x) = \rho(x) := \beta_1 \|x\|_1 + \beta_2 \|x\|_G \) for all \( i \in \mathcal{N}. \) For the test problems in CASE 2, we created a different partition \( G_i \) for each node \( i, \) in the same manner as in CASE 1. For all \( i \in \mathcal{N}, \) \( m_i = \frac{n_i}{N}, \) and the elements of \( A_i \in \mathbb{R}^{m_i \times n} \) are i.i.d. with standard Gaussian, and we set \( b_i = A_i \bar{x} \) for \( \bar{x}_j = (-1)^j e^{-(j-1)/n_g} \) for \( j \in [1, n]. \)

We solved the distributed optimization problem (3.4) using DFAL, AFAL (asynchronous version of DFAL with accelerated RBCD -see Section 4.2.5 for details), ADMM, and SADMM for both cases, on both star trees, and cliques, and for \( N \in \{5, 10\} \) and \( n_g \in \{100, 300\}. \) For each problem setting, we randomly generated 5 instances. For benchmarking, we solved the centralized problem (3.3) using SDPT3 for both cases. Note that for CASE 1, \( \sum_{i \in \mathcal{N}} \rho_i(x) = \|x\|_1 + \|x\|_G \) and its prox mapping can be computed efficiently, while for CASE 2, \( \sum_{i \in \mathcal{N}} \rho_i(x) \) does not assume a simple prox map. Therefore, for the first case we were also able to use APG, described in Section 4.1, to solve...
by exploiting the result of Lemma 4.15. All the algorithms are terminated when the relative suboptimality, $|F(k) - F^*|/|F^*|$, is less than $10^{-3}$, and consensus violation, $CV(k)$, is less than $10^{-4}$, where $F(k)$ equals to $\sum_{i \in N} F_i(x_i^{(k)})$ for DFAL and ADMM, and to $\sum_{i \in N} F_i\left(x_i^{(k)} + y_i^{(k)}\right)$ for SADMM; $CV(k)$ equals to $\max_{(ij) \in E} \|x_i^{(k)} - x_j^{(k)}\|_2/\sqrt{n}$ for DFAL, and ADMM, and to $\max\{\max_{(ij) \in E} \|x_i^{(k)} - x_j^{(k)}\|_2, \max_{i \in N} \|x_i^{(k)} - y_i^{(k)}\|_2\}/\sqrt{n}$ for SADMM. If the stopping criteria are not satisfied in 30min., we terminated the algorithm and report the statistics corresponding to the iterate at the termination.

In Table 5.1, 'xxx (C)' stands for “algorithm xxx is used to solve the centralized problem”. Similarly, 'xxx (D)' for the decentralized one. For the results separated by comma, the left and right ones are for the star tree and clique, resp. Table 5.1 displays the means over 5 replications for each case. The number of iterations in each case clearly illustrates the topology of the network plays an important role in the convergence speed of DFAL, which coincides to our analysis in Section 4.2.2.
Chapter 5  
Distributed Linearized ADMM for Composite Convex Consensus Optimization

In this chapter, we consider the same problem setting in Chapter 3; however, because of certain notational benefits, we adopted new notations different from the previous chapters. Below we briefly summarize the notational convention of this chapter. Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ denote a connected undirected graph of $N$ computing nodes where nodes $i$ and $j$ can communicate information only if $(i,j) \in \mathcal{E}$. Each node $i \in \mathcal{N} := \{1, \ldots, N\}$ has a private (local) cost function $\Phi_i : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ such that

$$\Phi_i(x) := \xi_i(x) + f_i(x),$$

where $\xi_i : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is a possibly non-smooth convex function, and $f_i : \mathbb{R}^n \to \mathbb{R}$ is a smooth convex function. We assume that $f_i$ is differentiable on an open set containing $\text{dom} \xi_i$ with a Lipschitz continuous gradient $\nabla f_i$, of which Lipschitz constant is $L_i$; and the prox map of $\xi_i$,

$$\text{prox}_{\xi_i}(x) := \arg\min_{y \in \mathbb{R}^n} \left\{ \xi_i(y) + \frac{1}{2} \|y - x\|^2 \right\},$$

is efficiently computable for $i \in \mathcal{N}$. In this chapter, we study a distributed consensus problem \[86\]; in particular, we consider solving a multi-agent consensus optimization problem of minimizing the sum of privately known composite convex functions in (5.1):

$$F^* := \min_{x \in \mathbb{R}^n} \sum_{i \in \mathcal{N}} \Phi_i(x).$$

We consider the setting where only local information exchange is allowed, i.e., there is no central node such that the data can be collected, and only neighboring nodes can
exchange data; and we focus on the following equivalent formulation:

\[
\min_{\mathbf{x}} \left\{ \sum_{i \in \mathcal{N}} F(x) := \Phi_i(x_i) : x_i = x_j, \forall (i, j) \in \mathcal{E} \right\}, \tag{5.4}
\]

where \( \mathbf{x} = [x_i]_{i \in \mathcal{N}} \in \mathbb{R}^{n|\mathcal{N}|} \). As in Chapter 3, we call \( \bar{\mathbf{x}} = [\bar{x}_i]_{i \in \mathcal{N}} \), \( \epsilon \)-feasible if the consensus violation satisfies \( \max_{(i, j) \in \mathcal{E}} \|\bar{x}_i - \bar{x}_j\|_2 \leq \epsilon \) and \( \epsilon \)-optimal if \( \left| \sum_{i \in \mathcal{N}} \Phi_i(\bar{x}_i) - F^* \right| \leq \epsilon \).

In Chapter 4, we proposed a distributed first-order augmented Lagrangian (DFAL) algorithm to solve (5.3), where each \( \Phi_i \) is a composite convex function as in (3.1). Assuming \( \xi_i \) is bounded below by some norm, i.e., \( \xi_i(.) \geq \|\| \), and it has a uniformly bounded subdifferential for each \( i \in \mathcal{N} \), it is shown that any limit point of DFAL iterates is optimal; and for any \( \epsilon > 0 \), an \( \epsilon \)-optimal and \( \epsilon \)-feasible solution can be computed within \( \mathcal{O}(\log(\epsilon^{-1})) \) DFAL iterations, which require \( \mathcal{O}(\frac{\sigma_{\max}(\Omega)}{d_{\min}} \epsilon^{-1}) \) gradient computations and communications per node in total, where \( \Omega \) denotes the Laplacian of \( \mathcal{G} \), and \( d_{\min} \) is the degree of the smallest degree node. Note that many regularizers used in the machine learning and statistics, e.g., \( \ell_p \)-norms, group norm, nuclear norm, and positive weighted sum of these norms, satisfy these assumptions on \( \xi_i \). Based on our tests and the results reported in [87] the algorithm DFAL performs very well in practice; however, its implementation on a network of distributed agents requires a more complex network protocol. Specifically, checking the subgradient stopping criterion for inner iterations requires evaluating a logical conjunction over \( \mathcal{G} \), which may not be easy for large networks.

The contribution of this chapter can be summarized as follows: 1) we propose a proximal gradient alternating direction method of multipliers (PG-ADMM) and its stochastic gradient variant SPG-ADMM to solve composite convex problems – for related work in stochastic ADMM, see [88,89], while it is yet to determine if they are suitable for distributed implementations. We only assume that the prox map of \( \xi_i \) can be computed efficiently, while other distributed ADMM based algorithms are efficient when either \( \Phi_i = \xi_i + f_i \) or both \( \xi_i \) and \( f_i \) have simple prox maps that can be computed efficiently; 2) we show ergodic convergence of both suboptimality and consensus violation bounds for PG-ADMM with the rate of \( \mathcal{O}(1/t) \), and for SPG-ADMM with the rate of \( \mathcal{O}(1/\sqrt{t}) \); 3) we implement PG-ADMM and SPG-ADMM on two different, but equivalent, consensus formulations of (5.3) – this gives rise to two different node-based distributed algorithms: DPGA-I and DPGA-II and their stochastic gradient variants SDPGA-I and SDPGA-II – and we examine the effect of the underlying network topology on their convergence rate. 4) The proposed algorithms DPGA-I, DPGA-II, SDPGA-I and SDPGA-II are fully
distributed, i.e., the agents are not required to know any global parameters depending on the entire network topology, e.g., the second smallest eigenvalue of the Laplacian; instead, we only assume that agents know who their neighbors are. Using only local communication, our node-based distributed algorithms require less communication burden and memory storage compared to edge-based distributed algorithms. Moreover, proposed algorithms consist of a single loop, i.e., there are no outer and inner iteration loops; therefore, they are easy and practical to be implemented over distributed networks.

To sum up, there are many practical problems where one can compute the prox map for \( \xi_i \) efficiently; however, computing the prox map for \( \Phi_i = \xi_i + f_i \) is not easy. The methods proposed in this chapter can compute an \( \epsilon \)-optimal \( \epsilon \)-feasible solution in \( O(\epsilon^{-1}) \) iterations without assuming bounded \( \nabla f_i \) for any \( i \in \mathcal{N} \); each iteration of these methods requires computing \( \text{prox}_{\xi_i} \) and \( \nabla f_i \) for \( i \in \mathcal{N} \), and one or two communication rounds among the neighbors – hence, \( O(\epsilon^{-1}) \) communications per node in total.

To sum up, unlike two-loop methods, e.g., [72, 73, 87], DPGA algorithms proposed here have only single-loop, and they are very easy to implement - see Fig. 5.1 and Fig. 5.2. These surprisingly simple algorithms can compute an \( \epsilon \)-feasible and \( \epsilon \)-optimal solution to (5.3) within \( O(1/\epsilon) \) communication rounds among neighboring nodes for all \( \epsilon > 0 \) (this is the best rate known for our setting) under much weaker assumptions on \( \xi_i \) and \( f_i \) compared to DFAL and with much simpler set of instructions compared to all the algorithms discussed above. To the best of our knowledge, in terms of storage and communication requirements per communication round, and convergence rate in terms of communication rounds, DPGA-II achieves the best guarantees known in the literature for problem (5.3) when \( \Phi_i \) is as in (5.1) for \( i \in \mathcal{N} \).

### 5.1 Proximal Gradient ADMM

Let \( \mathcal{N} = \{1, \ldots, N\} \), and \( \{\Phi_i\}_{i \in \mathcal{N}} \) be a collection of composite convex functions such that \( \Phi_i(x_i) = \xi_i(x_i) + f_i(x_i) \), where for all \( i \in \mathcal{N} \) both \( \xi_i : \mathbb{R}^{n_i} \to \mathbb{R} \cup \{+\infty\} \), and \( f_i : \mathbb{R}^{n_i} \to \mathbb{R} \) are convex, and \( f_i \) is differentiable on an open set containing \( \text{dom} \xi_i \) with a Lipschitz continuous gradient \( \nabla f_i \), of which Lipschitz constant is \( L_i \). Let \( g : \mathbb{R}^{n_x} \to \mathbb{R} \cup \{+\infty\} \) be a possibly non-smooth convex function. Suppose \( f : \mathbb{R}^{n_x} \to \mathbb{R} \) denotes \( f(x) = \sum_{i \in \mathcal{N}} f_i(x_i) \), where \( n_x := \sum_{i \in \mathcal{N}} n_i \) and \( x = [x_i]_{i \in \mathcal{N}} \in \mathbb{R}^{n_x} \). Throughout the chapter, we adopt the notation \( x = [x_i]_{i \in \mathcal{N}} = (x_i; x_{-i}) \in \mathbb{R}^{n_x} \) with \( x_i \in \mathbb{R}^{n_i} \) and \( x_{-i} = [x_j]_{j \in \mathcal{N} \setminus \{i\}} \in \mathbb{R}^{n_{x-n_i}} \) to denote a vector where \( x_i \) and \( x_{-i} \) are treated as variable and parameter sub-vectors of \( x \),
respectively; and \( \nabla_x f(x) \in \mathbb{R}^{n_i} \) denotes the sub-vector of \( \nabla f(x) \in \mathbb{R}^{n_x} \) corresponding to components of \( x_i \in \mathbb{R}^{n_i} \). In this section we consider the following problem:

\[
\min_{x,y} \quad F(x,y) := g(y) + \sum_{i=1}^{N} \Phi_i(x_i) \\
\text{s.t.} \quad A_i x_i + B_i y = b_i : \ i \in \mathcal{N},
\]

where \( A_i \in \mathbb{R}^{m_i \times n_i}, \ B_i \in \mathbb{R}^{m_i \times n_y}, \ b_i \in \mathbb{R}^{m_i} \) for all \( i \in \mathcal{N} \), and \( \lambda_i \in \mathbb{R}^{m_i} \) denotes the dual variable corresponding to the \( i \)-th constraint. We assume that \( \{\xi_i\}_{i \in \mathcal{N}} \) and \( g \) have easy prox maps. Define the augmented Lagrangian function as

\[
\mathcal{L}_\gamma(x,y,\lambda) = g(y) + \sum_{i=1}^{N} \xi_i(x_i) + \phi_\gamma(x,y,\lambda),
\]

where \( \lambda = [\lambda_i]_{i \in \mathcal{N}} \), and \( \phi_\gamma \) denotes the smooth part of the augmented Lagrangian, i.e.,

\[
\phi_\gamma(x,y,\lambda) = \sum_{i=1}^{N} f_i(x_i) + \sum_{i=1}^{N} \lambda_i^T (A_i x_i + B_i y - b_i) + \frac{\gamma}{2} \sum_{i=1}^{N} \|A_i x_i + B_i y - b_i\|^2.
\]

Note setting \( \gamma = 0 \) gives the standard Lagrangian function \( \mathcal{L}(x,y,\lambda) = \mathcal{L}_0(x,y,\lambda) \).

Consider the algorithm PG-ADMM stated below for solving (5.5): for \( k \geq 0 \) compute

\[
\begin{align*}
x_i^{k+1} &:= \text{prox}_{c_i \xi_i} \left( x_i^k - c_i \nabla_x \phi_\gamma(x_i^k, y^k, \lambda^k) \right), \quad i \in \mathcal{N} \\
y^{k+1} &:= \arg \min_{y} \mathcal{L}_\gamma(x^{k+1}, y, \lambda^k) \\
\lambda_i^{k+1} &:= \lambda_i^k + \gamma (A_i x_i^{k+1} + B_i y^{k+1} - b_i), \quad i \in \mathcal{N}
\end{align*}
\]

where \( c_i > 0 \) is the gradient step size for \( i \in \mathcal{N} \), which should be related to \( L_i \) and \( \sigma_{\max}(A_i) \).

In Section 5.1.2, we study the convergence properties of (5.6) given a deterministic first-order oracle which returns \( \nabla f_i \); and we also consider the effect of using stochastic first-order oracle, which returns noisy observations of \( \nabla f_i \), on the convergence rate.

**Definition 5.1.** Given \( f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\} \) such that \( \text{dom} f \) is open, and \( \nabla f \) exists on \( \text{dom} f \). \( G : \mathbb{R}^n \times \Xi \to \mathbb{R}^n \) is called a stochastic first-order oracle (SFO) for \( \nabla f \) if there exists \( \sigma \in [0, +\infty) \) such that for all \( x \in \text{dom} f \), \( \mathbb{E}_\zeta [G(x, \zeta)] = \nabla f(x) \), and \( \mathbb{E}_\zeta [\|G(x, \zeta) - \nabla f(x)\|^2] \leq \sigma^2 \), where \( \zeta \in \Xi \) is a random variable following a certain distribution.
Definition 5.2. Let $G_i$ denote an SFO for $\nabla f_i$ for $i \in \mathcal{N}$ with common parameter $\sigma$. Let $G_i^\tau$ be an SFO for $\nabla_x \phi_\gamma(\cdot, x_{-i}, y, \lambda)$ defined as $G_i^\tau(x_i, \zeta_i; x_{-i}, y, \lambda) = G_i(x_i, \zeta_i) + A_i^T(\lambda_i + \gamma(A_i x_i + B_i y - b_i))$.

Assumption 5.3. $\{\zeta_i^k \in \Xi : i \in \mathcal{N}, k \in \mathbb{Z}_+\}$ is a set of independent identically distributed random variables.

Under Assumption 5.3 consider the algorithm SPG-ADMM stated below for solving (5.5): for $k \geq 0$ compute

$$x_i^{k+1} := \text{prox}_{c_i \xi_i}(x_i^k - c_i G_i^\tau(x_i^k, \zeta_i^k; x_{-i}^k, y^k, \lambda^k)), \quad i \in \mathcal{N}$$

$$y^{k+1} := \text{argmin}_y \mathcal{L}_\gamma(x^{k+1}, y, \lambda^k)$$

$$\lambda_i^{k+1} := \lambda_i^k + \gamma(A_i x_i^{k+1} + B_i y^{k+1} - b_i), \quad i \in \mathcal{N}$$

(5.7)

where $c_i^k > 0$ is the stochastic-gradient step size for $i \in \mathcal{N}$ at the $k$-th iteration.

Remark 5.4. In the extreme case that $\xi_i = 0, i \in \mathcal{N}$, PG-ADMM (5.6) and SPG-ADMM (5.7) reduce to G-ADMM and SG-ADMM that take gradient steps for $x_i$-subproblems and have been studied in [90] and [91]. Specifically, [91] proves the $O(1/t)$ convergence rate of G-ADMM and $O(1/\sqrt{t})$ convergence rate of SG-ADMM. Our PG-ADMM and SPG-ADMM can be viewed as extensions of G-ADMM and SG-ADMM where general $\xi_i$’s are allowed. Our proofs of convergence rate results for PG-ADMM and SPG-ADMM in Section 5.1.3 are closely related to and inspired by [91].

5.1.1 Preliminaries, Notations and Simple Identities

Given a symmetric positive definite matrix $Q \in \mathbb{S}^n$, i.e., $Q > 0$, define $\|v\|_Q := \sqrt{v^T Q v}$ for all $v \in \mathbb{R}^n$. In our analysis, we used the two well-known identities frequently:

$$(v_2 - v_1)^T Q(v_3 - v_1) = \frac{1}{2} \left(\|v_2 - v_1\|_Q^2 + \|v_3 - v_1\|_Q^2 - \|v_2 - v_3\|_Q^2\right), \quad (5.8)$$

$$(v_1 - v_2)^T Q(v_3 - v_1) = \frac{1}{2} \left(\|v_1 - v_4\|_Q^2 - \|v_1 - v_3\|_Q^2\right) + \frac{1}{2} \left(\|v_2 - v_3\|_Q^2 - \|v_2 - v_4\|_Q^2\right). \quad (5.9)$$

For simplicity, we adopted the following notation to denote the stacked vectors or tuples: $x = [x_1^T, \ldots, x_N^T]^T$, $x^k = [x_1^k^T, \ldots, x_N^k^T]^T$, and $x^* = [x_1^{*T}, \ldots, x_N^{*T}]^T$. Let $u = [x^T y^T]^T$, $\lambda = [\lambda_1^T \ldots \lambda_N^T]^T$, and $w = [u^T \lambda^T]^T$; and let $u^k$, $u^*$, $\lambda^k$, $\lambda^*$, $w^k$, $w^*$ be defined similarly. For the sake of simplifying the notational burden, we also adopted $(x, y, \lambda) := [x^T, y^T, \lambda^T]^T$.
Let \( m := \sum_{i \in \mathcal{N}} m_i \), \( n_x := \sum_{i \in \mathcal{N}} n_i \), and \( F : \mathbb{R}^{n_x+n_y} \rightarrow \mathbb{R} \cup \{+\infty\} \) such that \( F \) denotes the objective function of problem \((5.5)\), i.e., \( F(u) := g(y) + \sum_{i=1}^{N} \Phi_i(x_i) \). Throughout the chapter, \( \|x\| \) denotes the Euclidean norm of \( x \), and \( I_n \) denotes \( n \)-by-\( n \) identity matrix.

We denote the set of optimal primal-dual pairs for \((5.5)\) as \( \chi^* \), i.e., \( w^* = (x^*, y^*, \lambda^*) \in \chi^* \) if and only if \( w^* \) is a saddle point of the Lagrangian function \( L = L_0 \) corresponding to \((5.5)\). The following assumption is made in the rest of the discussions.

**Assumption 5.5.** The optimal primal-dual pair set \( \chi^* \) for \((5.5)\) is non-empty, i.e., there exists \( (x^*, y^*, \lambda^*) \in \chi^* \) such that \( L(x, y, \lambda^*) \geq L(x^*, y^*, \lambda^*) \geq L(x^*, y^*, \lambda) \) for all \( x, y, \lambda \).

According to the first-order optimality conditions for \((5.5)\), solving \((5.5)\) is equivalent to finding \( (x^*, y^*, \lambda^*) \in \chi^* \) such that the following holds:

\[
\begin{align*}
-A_i^T \lambda_i^* - \nabla f_i(x_i^*) & \in \partial \xi_i(x_i^*), \quad i \in \mathcal{N},
- \sum_{i=1}^{N} B_i^T \lambda_i^* & \in \partial g(y^*),
A_i x_i^* + B_i y^* - b_i & = 0, \quad i \in \mathcal{N}.
\end{align*}
\]

(5.10)

We analyze both PG-ADMM and SPG-ADMM, stated in \((5.6)\) and in \((5.7)\), respectively, under Assumption 5.5.

### 5.1.2 Convergence Rate of PG-ADMM and SPG-ADMM

In this section, we prove the \( O(1/t) \) and \( O(1/\sqrt{t}) \) ergodic convergence rates for PG-ADMM and SPG-ADMM stated in \((5.6)\) and \((5.7)\), respectively, under Assumption 5.5. Note PG-ADMM and SPG-ADMM produce the same iterate sequence with probability 1 when \( \sigma = 0 \) in Definition 5.2. Therefore, we will first analyze SPG-ADMM, and then derive the bounds for PG-ADMM by sharpening the SPG-ADMM bounds for the case \( \sigma = 0 \). After some constant terms are discarded, SPG-ADMM for solving \((5.5)\) can be stated as: for \( k \geq 0 \),

\[
\begin{align*}
x_i^{k+1} & := \text{prox}_{\epsilon_i \xi_i} \left( x_i^k - c_i \left( G_i(x_i^k, \zeta_i^k) + \gamma A_i^T (A_i x_i^k + B_i y_i^k - b_i + \frac{1}{\gamma} \lambda_i^k) \right) \right), \quad i \in \mathcal{N},
y^{k+1} & := \arg\min_y g(y) + \frac{\gamma}{2} \sum_{i=1}^{N} \left\| A_i x_i^{k+1} + B_i y - b_i + \frac{1}{\gamma} \lambda_i \right\|^2, \quad (5.12)
\lambda_i^{k+1} & := \lambda_i^k + \gamma \left( A_i x_i^{k+1} + B_i y_i^{k+1} - b_i \right), \quad i \in \mathcal{N}. \quad (5.13)
\end{align*}
\]

The first-order optimality conditions for \((5.11)-(5.12)\) are given respectively as follows

87
\[ \partial \xi(x_i) \geq \frac{1}{c_i} (x_i - x_i^{k+1}) + \left[ G_i(x_i, \zeta_i^k) + A_i^T \left( \lambda_i^k + \gamma (A_i x_i^k + B_i y_i^k - b_i) \right) \right], \quad i \in \mathcal{N} \]

\[ \partial g(y^{k+1}) \geq - \sum_{i=1}^{N} B_i^T \left( \lambda_i^k + \gamma (A_i x_i^{k+1} + B_i y_i^{k+1} - b_i) \right) \]

Since \( \xi \) and \( g \) are convex, using the subgradients above with (5.13), we have

\[ \xi_i(x_i) - \xi_i(x_i^{k+1}) \geq (x_i - x_i^{k+1})^T \left[ \frac{1}{c_i} (x_i - x_i^{k+1}) - \left( G_i(x_i^k, \zeta_i^k) + A_i^T \lambda_i^{k+1} + \gamma A_i^T (A_i x_i^k - x_i^{k+1}) + B_i (y^k - y^{k+1}) \right) \right], \quad (5.14) \]

\[ g(y) - g(y^{k+1}) \geq - (y - y^{k+1})^T \sum_{i=1}^{N} B_i^T \lambda_i^{k+1}. \quad (5.15) \]

for any \( x_i \in \mathbb{R}^{n_i} \) and \( y \in \mathbb{R}^{n_y} \).

Let \( Q_i^k = I - \gamma c_i^2 A_i^T A_i \), \( \delta_i^k := G_i(x_i^k, \zeta_i^k) - \nabla_i f_i(x_i^k) \) denote the error in the noisy gradient generated by SFO, and \( D := \max_{i \in \mathcal{N}} \sup_{x, x' \in \text{dom}(\xi_i)} \| x - x' \| \) if the domain of each \( \xi_i \) for \( i \in \mathcal{N} \) is compact, then \( D < +\infty \).

**Lemma 5.6.** Let \( \{w^k\}_{k \geq 1} \) be the SPG-ADMM iterate sequence generated as in (5.11)-(5.13) starting from given \( u^0 \). If \( \{c_i^k\}_k \) satisfies \( c_i^k \leq \left( L_i + \gamma \sigma_i^2 A_i + 1(\delta_i^k)(1 + \sqrt{k}) \right) \) for all \( i \in \mathcal{N} \) and \( k \geq 0 \), then for any \( \lambda \) and \( u^* = (x^*, y^*) \in \chi^* \), the following inequality holds for all \( k \geq 0 \),

\[
\sum_{i=1}^{N} (x_i^* - x_i^{k+1})^T A_i^T \lambda_i^{k+1} + (y^* - y^{k+1})^T \sum_{i=1}^{N} B_i^T \lambda_i^{k+1} + \gamma \sum_{i=1}^{N} (\lambda_i - \lambda_i^{k+1})^T (b_i - A_i x_i^{k+1} - B_i y^{k+1})
\]

\[ F(u^*) - F(u^{k+1}) + \frac{1}{2} \sum_{i=1}^{N} \left[ \frac{1}{\gamma} \| \lambda_i - \lambda_i^k \|^2 - \frac{1}{\gamma} \| \lambda_i - \lambda_i^{k+1} \|^2 + \frac{1}{c_i^k} \| x_i^* - x_i^k Q_i^k \|^2 \right. 
\]

\[ - \frac{1}{c_i^k} \| x_i^* - x_i^{k+1} \|^2_{Q_i^k} + \gamma \| A_i x_i^* + B_i y^* - b_i \|^2 - \| A_i x_i^* + B_i y^{k+1} - b_i \|^2 
\]

\[ + 2(x_i^* - x_i^k)^T \delta_i^k + \frac{\| \delta_i^k \|^2}{1 + \sqrt{k}} \geq 0, \quad (5.16) \]

where \( \delta_i^k := G_i(x_i^k, \zeta_i^k) - \nabla_i f_i(x_i^k) \), and \( 1(\delta_i^k) \) is 1 if \( \| \delta_i^k \| > 0 \), and equal to 0 otherwise, i.e., when \( \delta_i^k = 0 \).
Proof: Note that (5.14) implies that,

\[ \xi_i(x_i) - \xi_i(x_i^{k+1}) \]

\[ \geq \frac{1}{c_i^2} \left( x_i - x_i^{k+1} \right)^T \left( I - c_i \gamma A_i^T A_i \right) \left( x_i - x_i^{k+1} \right) - \]

\[ \left( x_i - x_i^{k+1} \right)^T \left( G_i(x_i^k, \zeta_i^k) + A_i^T \lambda_i^{k+1} + \gamma A_i^T B_i(y^k - y^{k+1}) \right) \]

\[ = \frac{1}{2c_i^2} \left\| x_i - x_i^{k+1} \right\|^2_{Q_i} + \frac{1}{2} \left\| x_i^k - x_i^{k+1} \right\|^2_{Q_i} - \]

\[ \frac{1}{2c_i^2} \left\| x_i - x_i^k \right\|^2_{Q_i} - \left( x_i - x_i^{k+1} \right)^T \nabla f_i(x_i^k) \]

\[ - \left( x_i - x_i^{k+1} \right)^T \delta_i^k - \left( x_i - x_i^{k+1} \right)^T A_i^T \lambda_i^{k+1} \]

\[ - \gamma \left( A_i x_i - A_i x_i^{k+1} \right)^T \left( B_i y^k - B_i y^{k+1} \right), \quad (5.17) \]

where in the equality we applied the identity (5.8) with \( Q = I_{n_i} - c_i \gamma A_i^T A_i \), and use the definition \( \delta_i^k := G_i(x_i^k, \zeta_i^k) - \nabla f_i(x_i^k) \). Since each \( f_i \) is convex with Lipschitz continuous \( \nabla f_i \) for all \( x_i \in \text{dom} \xi_i \), we have

\[ 0 \leq f_i(x_i) - f_i(x_i^k) - \nabla f_i(x_i^k)^T (x_i - x_i^k) \leq \frac{L_i}{2} \left\| x_i - x_i^k \right\|^2. \]

Hence, it is easy to see that

\[ \left( x_i - x_i^{k+1} \right)^T \nabla f_i(x_i^k) \]

\[ = \left( x_i - x_i^k \right)^T \nabla f_i(x_i^k) + \left( x_i^k - x_i^{k+1} \right)^T \nabla f_i(x_i^k) \]

\[ \leq f_i(x_i) - f_i(x_i^k) + f_i(x_i^k) - f_i(x_i^{k+1}) + \frac{L_i}{2} \left\| x_i^k - x_i^{k+1} \right\|^2 \]

\[ = f_i(x_i) - f_i(x_i^{k+1}) + \frac{L_i}{2} \left\| x_i^k - x_i^{k+1} \right\|^2. \quad (5.18) \]

Moreover, it also trivially follows from the definition of \( I(\delta_i^k) \) that

\[ \left( x_i - x_i^{k+1} \right)^T \delta_i^k \leq \left( x_i - x_i^k \right)^T \delta_i^k + \frac{1}{2(1 + \sqrt{k})} \delta_i^k \]

\[ + \frac{I(\delta_i^k)(1 + \sqrt{k})}{2} \left\| x_i^k - x_i^{k+1} \right\|^2. \quad (5.19) \]

Finally, applying identity (5.9) with \( Q = I_{m_i}, v_1 = A_i x_i - b_i, v_2 = A_i x_i^{k+1} - b_i, v_3 = \)
\(-B_iy^{k+1}\), and \(v_4 = -B_iy^k\), we have

\[
(A_i x_i - A_i x_i^{k+1})^\top (B_i y^k - B_i y^{k+1})
\]

\[
= \frac{1}{2} \left( \|A_i x_i + B_i y^k - b_i\|^2 - \|A_i x_i + B_i y^{k+1} - b_i\|^2 \right)
\]

\[
+ \frac{1}{2} \left( \|A_i x_i^{k+1} + B_i y^{k+1} - b_i\|^2 - \|A_i x_i^{k+1} + B_i y^k - b_i\|^2 \right)
\]

\[
\leq \frac{1}{2} \left( \|A_i x_i + B_i y^k - b_i\|^2 - \|A_i x_i + B_i y^{k+1} - b_i\|^2 \right) + \frac{1}{2\gamma^2} \|\lambda_i^k - \lambda_i^{k+1}\|^2 .
\]

(5.20)

Therefore, using (5.18), (5.19) and (5.20) within (5.17), it follows that

\[
\xi_i(x_i) - \xi_i(x_i^{k+1}) + f_i(x_i) - f_i(x_i^{k+1}) + (x_i - x_i^{k+1})^\top A_i^\top \lambda_i^{k+1}
\]

\[
+ \frac{1}{2\gamma} \|\lambda_i^k - \lambda_i^{k+1}\|^2 + (x_i - x_i^k)^\top \delta_i^k + \|\delta_i^k\|^2
\]

\[
\geq \frac{1}{2\gamma_i} \|x_i - x_i^{k+1}\|_{Q_i^k}^2 - \frac{1}{2\gamma_i} \|x_i - x_i^k\|_{Q_i^k}^2
\]

\[
+ \frac{1}{2\gamma_i} \|x_i^k - x_i^{k+1}\|_{Q_i^k}^2 - \frac{1}{2\gamma_i} \|x_i - x_i^k\|_{Q_i^k}^2
\]

\[
+ \frac{1}{2\gamma_i} \|A_i x_i + B_i y^{k+1} - b_i\|^2 - \gamma \|A_i x_i + B_i y^k - b_i\|^2
\]

\[
\geq \frac{1}{2\gamma_i} \|x_i - x_i^{k+1}\|_{Q_i^k}^2 - \frac{1}{2\gamma_i} \|x_i - x_i^k\|_{Q_i^k}^2 + \frac{\gamma}{2} \|A_i x_i + B_i y^{k+1} - b_i\|^2 - \frac{\gamma}{2} \|A_i x_i + B_i y^k - b_i\|^2 ,
\]

(5.21)

where the second inequality follows from the condition

\[
0 < c_i^k \leq \left( L_i + \gamma \sigma_{max}(A_i) + \mathbb{1}(\delta_i^k)(1 + \sqrt{k}) \right)^{-1} .
\]

Indeed, this implies \(\frac{1}{c_i^k} \|x_i^k - x_i^{k+1}\|_{Q_i^k}^2 \geq \left( L_i + \mathbb{1}(\delta_i^k)(1 + \sqrt{k}) \right) \|x_i^k - x_i^{k+1}\|^2\). Next, combining (5.13) and (5.15) together with the above inequality implies for any \(\lambda \in \mathbb{R}^m\):

\[
\begin{pmatrix}
(x_1^* - x_1^{k+1}) \\
\vdots \\
(x_N^* - x_N^{k+1}) \\
y^* - y^{k+1} \\
\lambda_1 - \lambda_1^{k+1} \\
\vdots \\
\lambda_N - \lambda_N^{k+1}
\end{pmatrix}^\top
\begin{pmatrix}
A_1^\top \lambda_1^{k+1} \\
\vdots \\
A_N^\top \lambda_N^{k+1} \\
\sum_{i=1}^N B_i^\top \lambda_i^{k+1} \\
b_1 - A_1 x_1^{k+1} - B_1 y^{k+1} \\
\vdots \\
b_N - A_N x_N^{k+1} - B_N y^{k+1}
\end{pmatrix} + F(u^*) - F(u^{k+1})
\]

90
Using (5.8) with \( Q = I_{m_t} \), we have
\[
\frac{1}{2\gamma} \left( \| \lambda^k - \lambda^k+1 \|^2 - \| A_i x_i^* + B_i y^k - b_i \|^2 - \gamma \left( \| x_i^* - x_i \|^2 + \| y^k - y \|^2 \right) \right) \geq 0,
\]
(5.22)
Setting \( u = u^* \) in (5.22) and using the above equality, we obtain the desired inequality (5.16).

\textbf{Lemma 5.7.} Under Assumption 5.3, let \( \{w^k\}_{k \geq 1} \) be the SPG-ADMM iterate sequence generated as in (5.11)-(5.13) starting from given \( w^0 = (x^0, y^0, \lambda^0) \). Define \( \bar{w}^t = (\bar{x}^t, \bar{y}^t) = \frac{1}{t} \sum_{k=1}^{t} x^k \) and \( \bar{y}^t = \frac{1}{t} \sum_{k=1}^{t} y^k \) for \( t \geq 1 \). Fix arbitrary \( \lambda \) and \( u^* = (x^*, y^*) \in \chi^* \).

Suppose \( \sigma > 0 \). When \( \{c_i^k\}_{k \geq 1} \) sequence is chosen as \( c_i^k = \left( L_i + \gamma \sigma^2 \max(A_i) \right)^{-1} + \sqrt{\frac{1}{t}} \) for \( i \in N \), the following bound holds for all \( t \geq 1 \),
\[
\mathbb{E} \left[ F(\bar{u}^t) - F(u^*) + \sum_{i=1}^{N} \lambda_i^T \left( A_i x_i^T + B_i y^T - b_i \right) \right] \leq \frac{N(D^2 + 2\sigma^2)}{2\sqrt{t}} + \frac{1}{2t} \sum_{i=1}^{N} \left( \frac{1}{\gamma} \left( \| \lambda_i - \lambda_i^0 \|^2 + \frac{1}{c_i} \| x_i^* - x_i \|^2 \right) + \frac{\gamma}{2t} \left( \| y^* - y \|^2 \right) \right) + \frac{\gamma}{2t} \left( \| y^* - y \|^2 \right) \sum_{i=1}^{N} B_i^T B_i.
\]
Suppose \( \sigma = 0 \), i.e., \( G_i(x_i) = \nabla f_i(x_i) \) w.p.1 for any \( x_i \) and \( i \in N \). When \( c_i^k = c_i \) for all \( k \geq 0 \) for some \( c_i \in (0, \frac{1}{L_i + \gamma \sigma^2 \max(A_i)}] \), the following bound holds w.p.1 for all \( t \geq 1 \),
\[
F(\bar{u}^t) - F(u^*) + \sum_{i=1}^{N} \lambda_i^T \left( A_i x_i^T + B_i y^T - b_i \right) \leq \frac{1}{2t} \sum_{i=1}^{N} \left( \frac{1}{\gamma} \left( \| \lambda_i - \lambda_i^0 \|^2 + \frac{1}{c_i} \| x_i^* - x_i \|^2 \right) + \frac{\gamma}{2t} \left( \| y^* - y \|^2 \right) \right) \sum_{i=1}^{N} B_i^T B_i.
\]

\textbf{Proof:} Since \( u^* \) is a solution to (5.5), we have \( A_i x_i^* + B_i y^* = b_i \) for \( i \in N \). Hence, for
any $\{\lambda_i\}_{i \in \mathcal{N}}$, and $\hat{w}$ we have

$$\sum_{i=1}^{N} \lambda_i^T (b_i - A_i \hat{x}_i - B_i \hat{y}) = \sum_{i=1}^{N} (x_i^* - \hat{x}_i)^T A_i^T \hat{\lambda}_i + (y^* - \hat{y})^T \sum_{i=1}^{N} B_i^T \hat{\lambda}_i + \sum_{i=1}^{N} (\lambda_i - \hat{\lambda}_i)^T (b_i - A_i \hat{x}_i - B_i \hat{y}).$$

Hence, invoking the convexity of $F(\cdot)$ and using the above equality with $\hat{w} = w^{k+1}$ justify the first inequality and the first equality, respectively, below:

$$F(\bar{u}^t) - F(u^*) + \sum_{i \in \mathcal{N}} \lambda_i^T \left( A_i \bar{x}_i^t + B_i \bar{y}_i^t - b_i \right)$$

$$\leq -\frac{1}{t} \sum_{k=0}^{t-1} \left[ F(u^*) - F(u^{k+1}) + \sum_{i \in \mathcal{N}} \lambda_i^T (b_i - A_i x_i^{k+1} - B_i y^{k+1}) \right]$$

$$\leq \frac{1}{2t} \sum_{k=0}^{t-1} \sum_{i=1}^{N} \left[ \frac{1}{\gamma} \||\lambda_i - \lambda_k^0\||^2 + \frac{1}{\gamma^0} \|x_i^* - x_i^0\|^2 + \frac{1}{\gamma^0} \|x_i^* - x_i^{k+1}\|^2_{Q_i^k} + \frac{1}{\gamma} \|x_i^* - x_i^{k+1}\|^2_{Q_i^k} + \frac{1}{\gamma} \|x_i^* - x_i^{k+1}\|^2_{Q_i^k} \right.$$ 

$$\left. \gamma \|A_i x_i^* + B_i y^{k+1} - b_i\|^2 - \gamma \|A_i x_i^* + B_i y^{k+1} - b_i\|^2 + 2(x_i - x_i^k)^T \delta_i^k + \frac{\|\delta_i^k\|^2}{1 + \sqrt{k}} \right] =: \Gamma^t,$$

where the second inequality follows from Lemma 5.6.

First, consider the PG-ADMM iterate sequence generated using $G_i(x_i^k, \zeta_i^k) = \nabla f_i(x_i^k)$, i.e., $\delta_i^k = 0$ for all $i \in \mathcal{N}$ and $k \geq 0$. In this setting, according to Lemma 5.6, we are allowed to fix a constant step size $c_i$ for each $i \in \mathcal{N}$. Indeed, $c_i^k = c_i := (L_i + \gamma \sigma_{\max}(A_i))^{-1}$ for $k \geq 1$. Hence, one obtains the desired result in (5.24) by showing

$$\Gamma^t \leq \frac{1}{2t} \sum_{i=1}^{N} \left( \frac{1}{\gamma} \||\lambda_i - \lambda_0^0\||^2 + \frac{1}{\gamma^0} \|x_i^* - x_i^0\|^2 + \gamma \|y^* - y_0^0\|^2_{B_{i}^T B_i} \right),$$

which follows from dropping the non-positive terms after the telescoping sum in the definition of $\Gamma^t$ is evaluated, and using the fact that $A_i x_i^* + B_i y_0^0 = b_i$ for all $i \in \mathcal{N}$.

Next, consider the SPG-ADMM iterate sequence for which $\delta_i^k > 0$ with probability 1 for all $i \in \mathcal{N}$ and $k \geq 0$. In this case, according to Lemma 5.6, if one sets $c_i^k = \left( L_i + \gamma \sigma_{\max}(A_i) + 1 + \sqrt{k} \right)^{-1} > 0$ for all $i \in \mathcal{N}$ and $k \geq 0$, then $F(\bar{u}^t) - F(u^*) + \sum_{i \in \mathcal{N}} \lambda_i^T (A_i \bar{x}_i + B_i \bar{y}_i - b_i) \leq \Gamma^t$ holds for all $t \geq 1$. Moreover, we also have

$$\frac{1}{2t} \sum_{k=0}^{t-1} \sum_{i=1}^{N} \frac{1}{c_i^k} \|x_i^* - x_i^{k+1}\|^2_{Q_i^k} - \frac{1}{c_i^k} \|x_i^* - x_i^{k+1}\|^2_{Q_i^k} \leq \frac{1}{2t} \sum_{i=1}^{N} \left( \frac{1}{c_i^0} \|x_i^* - x_i^0\|^2_{Q_i^k} - \frac{1}{c_i^{k+1}} \|x_i^* - x_i^{k+1}\|^2_{Q_i^k} + \sum_{k=0}^{t-1} \left( \sqrt{k+1} - \sqrt{k} \right) \|x_i^* - x_i^{k+1}\|^2_{Q_i^k} \right).$$

92
This implies all defined as in Lemma 5.7. Fix an arbitrary point where we obtain the desired result by taking the expectation of the upper bound on (5.11) SPG-ADMM.

Hence, it follows that

\[ \Gamma' \leq \frac{ND^2}{2\sqrt{t}} + \frac{1}{2t} \sum_{i=1}^{N} \frac{1}{c_i} \left\| x_i^* - x_i^0 \right\|_{Q_i^*}^2. \]

Let \( \zeta^\ell = [\zeta^\ell_i]_{i \in \mathcal{N}} \) for \( \ell \geq 0 \). Note from (5.7) it follows that \( (x^k, y^k, \lambda^k) \) are random variables depending only on \( \Upsilon^k := \{ \zeta^\ell \}_{\ell=0}^{k-1} \) for all \( k \geq 1 \); hence, \( \mathbb{E}_{\zeta^k} [\delta^k | \Upsilon^k] = 0 \) and \( \mathbb{E}_{\zeta^k} [\| \delta^k \|^2 | \Upsilon^k] \leq \sigma^2 \) because \( \delta^k_i := G_i(x^k_i, \zeta^k_i) - \nabla_i f_i(x^k_i) \). Therefore, for all \( k \geq 0 \), we have

\[
\mathbb{E}_{\Upsilon_{k+1}} [(x^k_i)^\top \delta^k_i] = \mathbb{E}_{\Upsilon_k} [\mathbb{E}_{\zeta^k} [(x^k_i)^\top \delta^k_i | \Upsilon^k]] = \mathbb{E}_{\Upsilon_k} [(x^k_i)^\top \mathbb{E}_{\zeta^k} [\delta^k_i | \Upsilon^k]] = 0,
\]

\[
\mathbb{E}_{\Upsilon_{k+1}} [\| \delta^k_i \|^2] = \mathbb{E}_{\Upsilon_k} [\mathbb{E}_{\zeta^k} [\| \delta^k_i \|^2 | \Upsilon^k]] \leq \sigma^2.
\]

We obtain the desired result by taking the expectation of the upper bound on \( \Gamma' \) above, and using \( \sum_{k=0}^{t-1} \frac{1}{1+\sqrt{k}} \leq \int_0^{t-1} \frac{1}{1+\sqrt{s}} \, ds \leq \sqrt{t} \). \( \square \)

Now we are ready to prove the \( O(1/t) \) and \( O(1/\sqrt{t}) \) convergence rates of PG-ADMM stated in (5.6) and SPG-ADMM stated in (5.7) in an ergodic sense for solving (5.5).

**Theorem 5.8.** Under Assumption 5.3, let \( \{w^k\}_{k \geq 1} \) be the iterate sequence generated by SPG-ADMM (5.11)- (5.13) starting from given \( w^0 \); and \( \{\bar{w}^k\}_{k \geq 1} \) be the average sequence defined as in Lemma 5.7. Fix an arbitrary point \( (u^*, \lambda^*_1, \ldots, \lambda^*_N) \in \chi^* \).

Suppose \( \sigma = 0 \), i.e., \( G_i(x_i) = \nabla f_i(x_i) \) w.p.1 for any \( x_i \) and \( i \in \mathcal{N} \). When \( c_i^k = c_i \) for all \( k \geq 0 \) for some \( c_i \in (0, \frac{1}{L_i + \gamma \sigma_{\max}(A_i)}) \), the following bounds hold for all \( t \geq 1 \)

\[
0 \leq F(u^*) - F(u^*) + 2 \sum_{i=1}^{N} \| \lambda^*_i \| \| A_i x_i^* + B_i y^* - b_i \| \leq \frac{C(c_1, \ldots, c_N)}{t},
\]

where \( C(c_1, \ldots, c_N) := \sum_{i=1}^{N} \left( \frac{1}{2} \| \lambda^*_i \|^2 + \| \lambda^*_i \|^2 + \frac{1}{2c_i} \| x_i^* - x_i^0 \|_{Q_i}^2 \right) + \frac{2}{\gamma} \| y^* - y^0 \| \frac{\sum_{i=1}^{N} |B_i|}{\max_{1 \leq i \leq N} (|B_i|)} \| \). This implies \( O(1/t) \) ergodic convergence rate, i.e., for all \( t \geq 1 \) both suboptimality and
infeasibility satisfy
\[ |F(\bar{u}^t) - F(u^*)| \leq C(c_1, \ldots, c_N)/t, \quad \text{and} \quad \sum_{i=1}^N \|\lambda_i^*\| \|A_i \bar{x}_i^t + B_i \bar{y}^t - b_i\| \leq C(c_1, \ldots, c_N)/t. \]

Suppose \( \sigma > 0 \). Fix \( c_i = (L_i + \gamma \sigma^2_{\max}(A_i) + 1)^{-1} \) for \( i \in \mathcal{N} \) and set \( C := C(c_1, \ldots, c_N) \).
Let \( C' := N(D^2/2 + \sigma^2) \), and \( D := \max_{i \in \mathcal{N}} \sup_{x,x' \in \text{dom}(\xi_i)} \|x - x'\| \). When \( \{c_i^k\}_k \) sequence is chosen as \( c_i^k = \left( \frac{1}{c_i} + \sqrt{k} \right)^{-1} \) for \( i \in \mathcal{N} \), the following bounds hold for all \( t \geq 1 \):
\[
0 \leq \mathbb{E}\left[ F(\bar{u}^t) - F(u^*) + 2 \sum_{i=1}^N \|\lambda_i^*\| \|A_i \bar{x}_i^t + B_i \bar{y}^t - b_i\| \right] \leq \frac{C'}{\sqrt{t}} + \frac{C}{t}, \quad (5.26)
\]
This implies \( O(1/\sqrt{t}) \) ergodic convergence rate when \( D < +\infty \), i.e., for all \( t \geq 1 \) both suboptimality and infeasibility satisfy
\[
\mathbb{E}\left[ |F(\bar{u}^t) - F(u^*)| \right] \leq \frac{C'}{\sqrt{t}} + \frac{C}{t}, \quad \text{and} \quad \sum_{i=1}^N \|\lambda_i^*\| \mathbb{E}\left[ \|A_i \bar{x}_i^t + B_i \bar{y}^t - b_i\| \right] \leq \frac{C'}{\sqrt{t}} + \frac{C}{t}. \quad (5.27)
\]

Proof: Let \( \rho_i := 2\|\lambda_i^*\| \) for \( i \in \mathcal{N} \). Since any \((u^*, \lambda^*) \in \chi^*\) is a saddle point of \( \mathcal{L} \), it satisfies \( \mathcal{L}(u, \lambda^*) \geq \mathcal{L}(u^*, \lambda^*) \) for any \( u = (x, y) \), which can be easily checked using the optimality condition (5.10). Thus, combining it with the fact that \( A_i x_i^* + B_i y^* = b_i \) for \( i \in \mathcal{N} \) and using Cauchy-Schwarz results in the following inequality for all \( u = (x, y) \)
\[
F(u) - F(u^*) + \sum_{i=1}^N 2\|\lambda_i^*\| \|A_i x_i + B_i y - b_i\| \geq F(u) - F(u^*) + \sum_{i=1}^N \lambda_i^* \top (A_i x_i + B_i y - b_i) \geq 0. \quad (5.28)
\]
Hence, by setting \( u = \bar{u}^t = (\bar{x}^t, \bar{y}^t) \), we immediately obtain the first inequalities in (5.25) and (5.26).

Note (5.24) holds for any \( \lambda = [\lambda_i]_{i \in \mathcal{N}} \); hence, by setting \( \lambda_i = \rho_i (A_i \bar{x}^t_i + B_i \bar{y}^t - b_i) / \|A_i \bar{x}^t_i + B_i \bar{y}^t - b_i\| \) in (5.24), noting that \( \|\lambda_i\| = \rho_i \), and using \( \|\lambda_i - \lambda_0^0\|^2/2 \leq \|\lambda_i\|^2 + \|\lambda_0^0\|^2 \), we obtain the second inequality in (5.25), i.e., \( F(\bar{u}^t) - F(u^*) + \sum_{i=1}^N \rho_i \|A_i \bar{x}_i^t + B_i \bar{y}^t - b_i\| \leq C(c_1, \ldots, c_N)/t \). Let \( \bar{\tau}_i := A_i \bar{x}_i^t + B_i \bar{y}^t - b_i \) for \( i = 1, 2, \ldots, N \). From (5.28) and the above inequality, it follows that
\[
- \sum_{i=1}^N \|\lambda_i^*\| \|\bar{\tau}_i\| \leq F(\bar{u}^t) - F(u^*) \leq C(c_1, \ldots, c_N)/t - \sum_{i=1}^N \rho_i \|\bar{\tau}_i\|. \]
Hence, since $\rho_i := 2\|\lambda_i\|$, it follows that $\sum_{i=1}^N \|\lambda_i\| \|\bar{t}_i\| \leq \frac{C(c_1,\ldots,c_N)}{\bar{t}}$. Therefore, one can conclude that $\max\{|F(\bar{u}^t) - F(u^*)|, \sum_{i=1}^N \|\lambda_i\| \|A_i \bar{x}_i + B_i \bar{y} - b_i\|\} \leq \frac{C(c_1,\ldots,c_N)}{\bar{t}}$.

Using similar arguments as above one can show $O(1/\sqrt{t})$ rate for the SPG-ADMM. Indeed, observe that (5.23) holds for any $\lambda = [\lambda_i]_{i \in \mathcal{N}}$; hence, by setting $\lambda_i = \rho_i (A_i \bar{x}_i + B_i \bar{y} - b_i)$ as a random vector in (5.23), noting that $\mathbb{E}[\|\lambda_i\|] = \rho_i$, and using $\|\lambda_i - \lambda_0\|^2/2 \leq \|\lambda_i\|^2 + \|\lambda_0\|^2$, we obtain the second inequality in (5.26), which clearly implies (5.27) as discussed above.

5.2 Distributed Proximal Gradient Methods

In this section, we provide two different formulations of the decentralized optimization problem in (5.4), and both of them are special cases of (5.5). Hence, we develop two different distributed implementations of PG-ADMM in (5.6), one for each formulation; and finally we derive the customized convergence bounds showing the effect of network topology for each implementation. Similarly, one can also implement SPG-ADMM in (5.7) on the two different decentralized formulations of (5.4) to obtain the stochastic gradient variants of these distributed algorithms. The error bounds for these stochastic variants can be driven as we obtain the bounds for the deterministic versions using Lemma (5.7)

Due to space considerations, we skip their proofs and only state the error bounds for these variants using stochastic gradients as corollaries of the deterministic error bound results shown in Theorem (5.9) and Theorem (5.10)

In the rest of this chapter, we adopt the following notation. Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be a connected graph, where $\mathcal{N} := \{1,\ldots,N\}$ denotes the set of computing nodes, and $\mathcal{E} \subset \mathcal{N} \times \mathcal{N}$ denote the set of (undirected) edges. Without loss of generality, assume that edges in $\mathcal{E}$ are oriented, i.e., $(i,j) \in \mathcal{E}$ implies that $i < j$. Let $\mathcal{O}_i := \{j \in \mathcal{N} : (i,j) \in \mathcal{E}\}$ denote the set of neighboring nodes of $i \in \mathcal{N}$, and $d_i := |\mathcal{O}_i|$ denote the degree of node $i \in \mathcal{N}$; we also define $\mathcal{N}_i := \mathcal{O}_i \cup \{i\}$. Let $\Omega \in \mathbb{R}^{[\mathcal{N}] \times [\mathcal{N}]}$ denote the Laplacian of the graph $\mathcal{G}$, and $M \in \mathbb{R}^{[\mathcal{E}] \times [\mathcal{N}]}$ denote the oriented edge-node incidence matrix, i.e., for $e = (i,j) \in \mathcal{E}$ and $k \in \mathcal{N}$, $M_{ek}$ is equal to 1 if $k = i$, equal to $-1$ if $k = j$, and equal to 0, otherwise.

Note that $\Omega = M^T M$. Let $\psi_{\max} := \psi_1 \geq \psi_2 \geq \ldots \geq \psi_N$ be the eigenvalues of $\Omega$. Since we assumed that $\mathcal{G}$ is connected, we have $\psi_{N-1} > \psi_N = 0$. Hence, $\text{rank}(M) = \text{rank}(\Omega) = N - 1$. Moreover, $(M \otimes I_n)^T (M \otimes I_n) = \Omega \otimes I_n$; and from the structure of $\Omega \otimes I_n$, it follows that $\{\psi_i\}_{i=1}^N$ are also the eigenvalues of $\Omega \otimes I_n$, each with algebraic
multiplicity $n$. Thus, $R := \text{rank}(M \otimes I_n) = n(N - 1)$. Let $M \otimes I_n = U \Sigma V^\top$ denote the reduced singular value decomposition (SVD) of $M \otimes I_n$, where $U = [u_1 \ldots u_R] \in \mathbb{R}^{n|\mathcal{N}| \times R}$, $V = [v_1 \ldots v_R] \in \mathbb{R}^{n|\mathcal{N}| \times R}$, $\Sigma = \text{diag}(\sigma)$, and $\sigma \in \mathbb{R}^R_{++}$. Note that $\sigma_{\max}(M \otimes I_n) = \max\{\sigma_r : r = 1, \ldots, R\} = \sqrt{\psi_1}$, and $\sigma_{\min}(M \otimes I_n) = \min\{\sigma_r : r = 1, \ldots, R\} = \sqrt{\psi_{N-1}}$.

### 5.2.1 DPGA-I Algorithm

The first formulation that is suitable for implementing PG-ADMM in (5.6) or SPG-ADMM in (5.7) follows from [70] using communication matrices. In particular, $C \in \mathbb{R}^{|\mathcal{N}| \times |\mathcal{N}|}$ is called a communication matrix if for all $i \in \mathcal{N}$, $C_{ij} = 0$ for all $j \notin \mathcal{N}_i$, $C_{ij} < 0$ for all $i \in \mathcal{O}_i$, and $C_{ii} = -\sum_{j \notin \mathcal{O}_i} C_{ij}$. It is easy to show that for $x = [x_i]_{i \in \mathcal{N}} \in \mathbb{R}^{n|\mathcal{N}|}$ satisfying $(C \otimes I_n) x = 0$, we have $x_i = \bar{x}$ for all $i \in \mathcal{N}$ for some $\bar{x} \in \mathbb{R}^n$, where $\otimes$ denotes the Kronecker product. Therefore, given $C$ with properties above, the feasible set in (5.4) can be equivalently represented as $\{x = [x_i]_{i \in \mathcal{N}} : (C \otimes I_n) x = 0\}$. Note that the Laplacian $\Omega$ of the graph $\mathcal{G}$ is a communication matrix. Hence, (5.4) can be equivalently written as

\[
\begin{aligned}
\min_{x \in \mathbb{R}^{n|\mathcal{N}|}} & \left\{ F(x) := \sum_{i \in \mathcal{N}} \Phi_i(x_i) : (\Omega \otimes I_n) x = 0 \right\}, \\
\text{s.t.} & \quad \Omega_{ij} x_j - y_{ij} = 0 : \lambda_{ij}, \quad \forall j \in \mathcal{N}_i, \; \forall i \in \mathcal{N},
\end{aligned}
\]

(5.29)

where $x^T = [x_1^T, \ldots, x_N^T]$ and $\Phi_i$ is defined in (5.1). Here, we derive the algorithm and its convergence bounds by setting $C = \Omega$ (it is worth emphasizing that the same arguments used here also work for any general communication matrix, resulting in the same error bounds with slightly different $O(1)$ constants). In the rest of this section, assume that (5.3) has a solution, and (5.29) satisfies Assumption 5.5.

For each $i \in \mathcal{N}$, define new set of primal variables $y_{ij} \in \mathbb{R}^n$ for $j \in \mathcal{N}_i$, and form $y_i := [y_{ij}]_{j \in \mathcal{N}_i} \in \mathbb{R}^{(d_i + 1)n}$. Let $\mathcal{Y}_i := \{y_i : \sum_{j \in \mathcal{N}_i} y_{ij} = 0\}$ for $i \in \mathcal{N}$, and define $g(y) := \sum_{i \in \mathcal{N}} \mathbb{1}_{\mathcal{Y}_i}(y_i)$, where $\mathbb{1}_{\mathcal{Y}_i}$ denotes the indicator function of the set $\mathcal{Y}_i$, i.e., $\mathbb{1}_{\mathcal{Y}_i}(y_i)$ is equal to 0 if $y_i \in \mathcal{Y}_i$, and to $+\infty$ otherwise, where $y^T = [y_1^T, \ldots, y_N^T]$. Hence, following the insight of [70], consider the following equivalent formulation:

\[
\begin{aligned}
\min_{y, x} & \quad g(y) + \sum_{i \in \mathcal{N}} \Phi_i(x_i) \\
\text{s.t.} & \quad \Omega_{ij} x_j - y_{ij} = 0 : \lambda_{ij}, \quad \forall j \in \mathcal{N}_i, \; \forall i \in \mathcal{N},
\end{aligned}
\]

(5.30)

where $\lambda_{ij} \in \mathbb{R}^n$ denote the Lagrange multiplier corresponding to the primal constraint $\Omega_{ij} x_j - y_{ij} = 0$ in (5.30). Similar to the definition of $y_i$, let $\lambda_i = [\lambda_{ij}]_{j \in \mathcal{N}_i} \in \mathbb{R}^{(d_i + 1)n}$, and $\lambda = [\lambda_i]_{i \in \mathcal{N}} \in \mathbb{R}^{n(2|\mathcal{E}| + |\mathcal{N}|)}$.
It is important to note that the formulation (5.30) is a special case of (5.5), and one can employ PG-ADMM or SPG-ADMM to solve (5.30). In the rest, we focus on the implementation details of PG-ADMM. Indeed, it can be easily observed that $A_i$ of (5.5) takes the following form for (5.30): $A_i = (\omega_i \otimes I_n) \in \mathbb{R}^{n(d_i+1) \times n}$, where $\omega_i \in \mathbb{R}^{d_i+1}$ is obtained from the $i$-th column of $\Omega$ after 0 entries are removed without changing the order of non-zero entries. Furthermore, it can also be easily observed that $\sum_{i=1}^{N} B_i^T B_i$ in Lemma 5.7 is equal to $I_{n(2|x|+|\Omega|)}$ for the constraint matrix structure in (5.30); therefore, $\|y^* - y^0\|_{\sum_{i=1}^{N} B_i^T B_i} = \|y^* - y^0\|$. For all $i \in \mathcal{N}$, we set the stepsize $c_i$ to its upper bound given in Lemma 5.7, i.e., $c_i = 1/(L_i + \gamma \sigma_{\max}(A_i))$. Since $\sigma_{\max}(A_i) = \sqrt{d_i^2 + d_i}$ for $i \in \mathcal{N}$ for the formulation (5.30), this corresponds to setting $c_i := 1/(L_i + \gamma d_i(d_i + 1))$.

The smooth part of the augmented Lagrangian $\phi_\gamma$ for the formulation (5.30) can be written as

$$
\phi_\gamma(x, y, \lambda) = \sum_{i \in \mathcal{N}} \left[ f_i(x_i) + \sum_{j \in \mathcal{N}_i} \lambda_{ij}^2 (\Omega_{ij} x_j - y_{ij}) + \frac{\gamma}{2} \sum_{j \in \mathcal{N}_i} \|\Omega_{ij} x_j - y_{ij}\|^2 \right],
$$

for a fixed penalty parameter $\gamma > 0$; hence, $\nabla_x \phi_\gamma$ can be computed as

$$
\nabla_x \phi_\gamma(x^k, y^k, \lambda^k) = \nabla f(x_j^k) + \sum_{i \in \mathcal{N}_j} \left[ \Omega_{ij} \lambda_{ij}^k + \gamma \Omega_{ij} (\Omega_{ij} x_j^k - y_{ij}^k) \right]
$$

and the steps of PG-ADMM in (5.6) take the following form:

$$
x_{j}^{k+1} = \text{prox}_{c_j \xi_j} \left( x_j^k - c_j \nabla_x \phi_\gamma(x^k, y^k, \lambda^k) \right), \quad j \in \mathcal{N}, \quad (5.31)
$$

$$
y_{i}^{k+1} = \text{argmin}_{y_{i}} \left\{ \frac{\gamma}{2} \sum_{j \in \mathcal{N}_i} \|\Omega_{ij} x_j^{k+1} - y_{ij} + \frac{1}{\gamma} \lambda_{ij}^k\|^2 : \sum_{j \in \mathcal{N}_i} y_{ij} = 0 \right\}, \quad i \in \mathcal{N}, \quad (5.32)
$$

$$
\lambda_{ij}^{k+1} = \lambda_{ij}^k + \gamma \left( \Omega_{ij} x_j^{k+1} - y_{ij}^{k+1} \right), \quad j \in \mathcal{N}_i, \quad i \in \mathcal{N}. \quad (5.33)
$$

Except for $x$-step in (5.31), the $y$-step in (5.32) and $\lambda$-step in (5.33) are exactly the same as those in [70]. Instead of taking proximal gradient step, $x_{j}^{k+1}$ is computed in [70] by solving $\min_{x_j \in \mathbb{R}^n} \Phi_j(x_j) + \sum_{i \in \mathcal{N}_j} \lambda_{ij}^k \top (\Omega_{ij} x_j - y_{ij}^k) + \frac{\gamma}{2} \sum_{j \in \mathcal{N}_i} \|\Omega_{ij} x_j - y_{ij}^k\|^2$, which is equivalent to computing $\text{prox}_{\xi_j + f_j}$. Note that even if both $\xi_j$ and $f_j$ have simple prox maps, the prox map of the sum is not necessarily simple and it can be impractical to compute.

Since $y$-step and $\lambda$-step are the same as those in [70], the results of this paragraph directly follow from Section III of [70]. Let $\{x^0_i\}_{i \in \mathcal{N}}$ denote the set of initial primal
Therefore, the steps in (5.31)-(5.33) can be simplified as shown in Figure 5.1 below. Indeed,\[ y^k_{ij} = \Omega_{ij} x^k_{ij} + \frac{1}{\gamma} (\lambda^k_{ij} - p^k_{ij}) \]

Moreover, since \( y^k_{ij} \) is nothing but PG-ADMM customized to the decentralized formulation, the optimal dual can be computed in closed form: \( p^k_{ij} = \frac{1}{d_i+1} \sum_{j \in \mathcal{N}_i} \lambda^k_{ij} + \frac{\gamma}{d_i+1} \sum_{j \in \mathcal{N}_i} \Omega_{ij} x^k_{ij} \) for \( k \geq 0 \). Suppose that we initialize \( \lambda^0_{ij} = p^0_i \) for all \( j \in \mathcal{N}_i \) for some given \( p^0_i \) for all \( i \in \mathcal{N} \). Thus, \( p^k_{ij} = p^k_i + \frac{\gamma}{d_i+1} \sum_{j \in \mathcal{N}_i} \Omega_{ij} x^k_{ij} \) for all \( k \geq 0 \). Finally, by defining \( s^0_i = 0 \) and \( s^k_i := \frac{1}{d_i+1} \sum_{j \in \mathcal{N}_i} \Omega_{ij} x^k_{ij} \) for \( k \geq 1 \), and initializing \( y^0_{ij} := \Omega_{ij} x^0_{ij} \) for all \( j \in \mathcal{N}_i \) and \( i \in \mathcal{N} \), the computation of \( \nabla x_i \phi \) in step (5.31) can be simplified.

\[
\nabla x_j \phi(x^k, y^k, \lambda^k) - \nabla f_j(x^k_j) = \sum_{i \in \mathcal{N}_j} \Omega_{ij} \left( \lambda^k_{ij} + \gamma (\Omega_{ij} x^k_{ij} - y^k_{ij}) \right) = \sum_{i \in \mathcal{N}_j} \Omega_{ij} \left( 2p^k_i - p^k_{ij} \right)
\]

holds for \( k \geq 0 \). Note that this is true for \( k = 0 \) because of how we initialize \( \lambda^0 \) and \( y^0 \).

Therefore, the steps in (5.31)- (5.33) can be simplified as shown in Figure 5.1 below.

\[
\begin{align*}
\text{Algorithm DPGA-I (} \gamma, x^0, p^0 ) \quad & \\
\text{Initialization: } & c_i = (L_i + \gamma d_i (d_i + 1))^{-1}, \quad s^0_i = 0, \quad i \in \mathcal{N} \\
\text{Step } & k := (k \geq 0) \text{ For } i \in \mathcal{N} \text{ compute} \\
1. & x^k_{ij} = \text{prox}_{c_i \xi_i} \left( x^k_i - c_i \left( \nabla f_i(x^k_i) + \sum_{j \in \mathcal{N}_i} \Omega_{ij} (p^k_j + \gamma s^k_j) \right) \right), \quad i \in \mathcal{N} \\
2. & s^k_i := \frac{1}{d_i+1} \sum_{j \in \mathcal{N}_i} \Omega_{ij} x^k_{ij}, \quad i \in \mathcal{N} \\
3. & p^k_i = p^k_i + \gamma s^k_i, \quad i \in \mathcal{N}
\end{align*}
\]

Figure 5.1: Distributed Proximal Gradient Algorithm I (DPGA-I)

The algorithm works in the following distributed fashion: i) each node stores three variables in \( \mathbb{R}^n \): \( x^k_i, s^k_i \) and \( p^k_i \); ii) each node sends out its \( p^k_i + \gamma s^k_i \) to all its neighbors, and computes the proximal step; iii) after the computation of proximal steps, each node broadcasts the updated variable \( x^k_{ij} \), and then updates \( s^k_{ij} \); iv) each node updates \( p^k_{ij} \), and then repeats.

### 5.2.2 Error Bounds for DPGA-I & Effect of Network Topology

In this section, we examine the effect of network topology on the convergence rate of DPGA-I, which is nothing but PG-ADMM customized to the decentralized formulation.
in (5.30) as discussed in Section 5.2.1. To obtain simple $O(1)$ constants in the error bounds, we set $p_i^0 = 0$ for all $i \in \mathcal{N}$.

**Theorem 5.9.** Given arbitrary $\mathbf{x}^0$ and $\gamma > 0$, let $p_i^0 = 0$ for all $i \in \mathcal{N}$, and $\{\mathbf{x}^k\}_{k \geq 1}$ be the DPGA-I iterate sequence, generated as shown in Figure 5.1. The average sequence $\{\bar{\mathbf{x}}^t\}_{t \geq 1}$, defined as $\bar{\mathbf{x}}_i^t = (\sum_{k=1}^t x_i^k)/t$ for $i \in \mathcal{N}$ and $t \geq 1$, converges to an optimal solution to (5.4) with the following suboptimality and infeasibility bounds satisfied for all $t \geq 1$,

$$
\frac{1}{t} \left( \frac{2(d_{\max} + 1)}{\gamma \sigma^2_{\min}(\Omega)} \sum_{i \in \mathcal{N}} \kappa_i^2 + \sum_{i \in \mathcal{N}} \frac{1}{2c_i} \|x^* - x_i^0\|^2 \right) \leq F(\bar{\mathbf{x}}^t) - F^* \leq \frac{1}{t} \left( \sum_{i \in \mathcal{N}} \frac{1}{2c_i} \|x^* - x_i^0\|^2 \right)
$$

where $\kappa_i > 0$ denotes an upper bound on the elements of $\partial \Phi_i(x^*)$, i.e., if $q \in \partial \Phi_i(x^*)$, then $\|q\| \leq \kappa_i$, for each $i \in \mathcal{N}$.

**Proof:** Suppose that a solution to (5.3) exists; then $x^* \in \mathbb{R}^n$ is an optimal solution to (5.3) if and only if $\mathbf{x}^* \in \mathbb{R}^{n|\mathcal{N}|}$ such that $x_i^* = x^*$ for all $i \in \mathcal{N}$ is optimal to (5.29), and $(\mathbf{x}^*, \mathbf{y}^*)$ such that $\mathbf{y}^* \in \mathbb{R}^{n(2|\mathcal{E}|+|\mathcal{N}|)}$ and $y_{ij}^* = \Omega_{ij} x^*$ for all $j \in \mathcal{N}_i$ for $i \in \mathcal{N}$ is optimal to (5.30). The formulation (5.29), which is equivalent to (5.30), can also be written as

$$
\min_{\mathbf{x}} \left\{ \sum_{i \in \mathcal{N}} \Phi_i(x_i) : \sum_{j \in \mathcal{N}_i} \Omega_{ij} x_j = p_i : \forall i \in \mathcal{N} \right\},
$$

(5.34)

where $p_i \in \mathbb{R}^n$ denotes the dual variable corresponding to the primal constraint for $i \in \mathcal{N}$. Recall that we assume (5.29) satisfies Assumption 5.5; hence, there exists a primal-dual optimal pair $(\mathbf{x}^*, \mathbf{p}^*)$ for (5.29), where $\mathbf{x}^* \in \mathbb{R}^{n|\mathcal{N}|}$ such that $x_i^* = x^*$ for $i \in \mathcal{N}$ and $\mathbf{p}^* := [p_i^*]_{i \in \mathcal{N}}$. Note that Assumption 5.5 is satisfied if Slater’s condition holds, i.e., $\cap_{i \in \mathcal{N}} \text{ri(} \text{dom} \xi_i \text{)} \neq \emptyset$.

From the first-order optimality conditions for (5.34), $\mathbf{p}^* := [p_i^*]_{i \in \mathcal{N}}$ is an optimal dual solution to (5.34) if and only if

$$
0 \in \partial \Phi_j(x^*) + \sum_{i \in \mathcal{N}_j} \Omega_{ij} p_i^*, \quad \forall j \in \mathcal{N},
$$

(5.35)

i.e., $-(\Omega \otimes I_n)^\top \mathbf{p}^* \in \partial F(\mathbf{x}^*)$ and $x_i^* = x^*$ for $i \in \mathcal{N}$, where $x^* \in \mathbb{R}^n$ is an optimal solution to (5.3). Moreover, according to the first-order optimality conditions for (5.30),

99
\( \lambda^* \in \mathbb{R}^{n(2|\mathcal{E}|+|\mathcal{N}|)} \) is dual optimal to (5.30) if and only if there exists some \( p = [p_i]_{i \in \mathcal{N}} \) such that \( \lambda^*_{ij} = p_i \) for all \( j \in \mathcal{N}_i \) and \( i \in \mathcal{N} \), and \( 0 \in \partial \Phi_j(x^*) + \sum_{i \in \mathcal{N}_j} \Omega_{ij} \lambda^*_{ij} \) for all \( j \in \mathcal{N} \). Therefore, given \( p^* \) an optimal dual solution to (5.34), one can construct a dual optimal solution \( \lambda^* \) to (5.30) by simply setting \( \lambda^*_{ij} = p_i^* \) for all \( j \in \mathcal{N}_i \) and \( i \in \mathcal{N} \). In the rest of the proof, fix \( x^* \in \mathbb{R}^n \) and \( p^* = [p_i^*]_{i \in \mathcal{N}} \) such that \( x^* \) is an optimal solution to (5.3), and \( p^* \) is a dual solution to (5.34); thus, \( (x^*, y^*, \lambda^*) \) is a primal-dual optimal solution to (5.30), where \( x_i^* = x^* \) for \( i \in \mathcal{N} \), \( y_{ij}^* = \Omega_{ij} x^* \) and \( \lambda^*_{ij} = p_i^* \) for \( j \in \mathcal{N}_i \) and \( i \in \mathcal{N} \).

Since \( A_i = (\omega_i \otimes I_n) \in \mathbb{R}^{n(d_i+1) \times n} \), where \( \omega_i \in \mathbb{R}^{d_i+1} \) is the \( i \)-th column of \( \Omega \) after 0 entries are removed, \( A_i^T A_i = d_i(d_i+1)I_n \); hence, \( c_i = 1/(L_i + \gamma d_i(d_i+1)) \) implies that \( \frac{1}{\alpha} ||x_i^0 - x^*||_{\Omega_i - \gamma c_i A_i^T A_i} = L_i ||x_i^0 - x^*||^2 \). We also have \( \sum_{i=1}^n B_i^T B_i = I_n(2|\mathcal{E}|+|\mathcal{N}|) \), and since we initialize \( y^0 \) such that \( y_{ij}^0 = \Omega_{ij} x_j^0 \), and \( y_{ij}^0 = \Omega_{ij} x^* \), for all \( j \in \mathcal{N}_i \) and \( i \in \mathcal{N} \), it trivially follows that \( ||y^*-y^0||^2 \sum_{i=1}^n B_i^T B_i = \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}_i} \Omega_{ij}^2 ||x^* - x_j^0||^2 = \sum_{i \in \mathcal{N}} d_i(d_i+1)||x^* - x_i^0||^2 \).

Furthermore, Lemma 5.7 holds for all \( \lambda = [\lambda_i]_{i \in \mathcal{N}} \in \mathbb{R}^{n(2|\mathcal{E}|+|\mathcal{N}|)} \), where \( \lambda_i = [\lambda_{ij}]_{j \in \mathcal{N}_i} \in \mathbb{R}^{n(d_i+1)} \) for \( i \in \mathcal{N} \). Hence, given \( p^T = [p_1^T, \ldots, p_n^T]^T \) for some \( p_i \in \mathbb{R}^n \) for all \( i \in \mathcal{N} \), we set \( \lambda_{ij} = p_i \) for all \( j \in \mathcal{N}_i \) and \( i \in \mathcal{N} \), and invoke Lemma 5.7 for this specific choice of \( \lambda \) to obtain the following inequality, customized for (5.30), for all \( t \geq 1 \):

\[
F(\bar{x}^t) - F^* + p^T(\Omega \otimes I_n)\bar{x}^t = F(\bar{x}^t) - F^* + \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}_i} p_i (\Omega_{ij} \bar{x}_j - \bar{y}_{ij}) \\
\leq \frac{1}{2t} \sum_{i \in \mathcal{N}} \left( \frac{d_i + 1}{\gamma} ||p_i||^2 + \frac{1}{c_i} ||x^* - x_i^0||^2 \right),
\]

(5.36)

where the inequality follows from the facts that \( \sum_{j \in \mathcal{N}_i} \bar{y}_{ij} = 0 \) and \( c_i = (L_i + \gamma d_i(d_i+1))^{-1} \).

Hence, setting \( p = 0 \) in (5.36) leads to

\[
F(\bar{x}^t) - F^* \leq \frac{1}{t} \sum_{i \in \mathcal{N}} \frac{1}{2c_i} ||x^* - x_i^0||^2.
\]

(5.37)

From the convexity of \( \Phi_i \), (5.35) and the fact \((\Omega \otimes I_n)x^* = 0\), it follows that

\[
0 \leq F(\bar{x}^t) - F^* + p^T(\Omega \otimes I_n)(\bar{x}^t - x^*) = F(\bar{x}^t) - F^* + p^T(\Omega \otimes I_n)\bar{x}^t.
\]

(5.38)

Adding the last term to both sides, and invoking (5.36) for \( p \) such that \( p_i = 2p_i^* \) for \( i \in \mathcal{N} \), we get

\[
p^T(\Omega \otimes I_n)\bar{x}^t \leq F(\bar{x}^t) - F^* + (2p^*)^T(\Omega \otimes I_n)\bar{x}^t
\]
where the last inequality follows from bounding $\parallel \mathbf{x}^* - \mathbf{x}_i^0 \parallel^2$. Using the local bounds $\kappa_i$, we can choose $\min \{ \kappa_i \}$ and $\max \{ \lambda_i \}$ to denote the eigenvalues of the graph Laplacian $\mathbf{L}$ and choosing a dual optimal to (5.34), we bound (5.36)

Invoking (5.36) once again for $\mathbf{p} = \mathbf{p}^* + (\Omega \otimes I_n) \mathbf{x}^t / \parallel (\Omega \otimes I_n) \mathbf{x}^t \parallel$ and using (5.38), we get

$$
\parallel (\Omega \otimes I_n) \mathbf{x}^t \parallel \leq F(\tilde{\mathbf{x}}^t) - F^* + \mathbf{p}^T (\Omega \otimes I_n) \mathbf{x}^t + \parallel (\Omega \otimes I_n) \mathbf{x}^t \parallel
$$

$$
\leq \frac{1}{2t} \sum_{i \in \mathcal{N}} \left( d_i + \frac{1}{\gamma} \parallel p_i^* \parallel + \sum_{j \in \mathcal{N}_i} \Omega_{ij} \parallel \mathbf{x}^t \parallel^2 + \frac{1}{c_i} \parallel x_i^* - x_i^0 \parallel^2 \right)
$$

$$
\leq \frac{1}{t} \left( \frac{\max d + 1}{\gamma} + (1 + \parallel \mathbf{p}^* \parallel^2) + \sum_{i \in \mathcal{N}} \frac{1}{2c_i} \parallel x_i^* - x_i^0 \parallel^2 \right),
$$

(5.40)

where the last inequality follows from $\parallel a + b \parallel^2 \leq 2 \parallel a \parallel^2 + 2 \parallel b \parallel^2$. Next, appropriately choosing a dual optimal to (5.34), we bound $\parallel \mathbf{p}^* \parallel$, which appears in (5.39) and (5.40).

Since $\mathcal{G}$ is connected, rank$(\Omega) = N - 1$ and $\psi_1 \geq \ldots \geq \psi_{N-1} > \psi_N = 0$, where $\{ \psi_i \}_{i=1}^N$ denote the eigenvalues of the graph Laplacian $\Omega \in \mathbb{R}^{\mathcal{N} \times \mathcal{N}}$. From the definitions given at the beginning of Section 5.2, given the SVD of $(M \otimes I_n)$, the eigenvalue decomposition of $(\Omega \otimes I_n)$ can be written as $V \Sigma^2 V^T$, where $V = [v_1 \ldots v_R] \in \mathbb{R}^{n \times \mathcal{N}}$, $\Sigma = \text{diag}(\sigma)$, and $R := \text{rank}(\Omega \otimes I_n) = n(N - 1)$. Note that $\max \{ \sigma_r^2 : r = 1, \ldots, R \} = \psi_1$, and $\min \{ \sigma_r^2 : r = 1, \ldots, R \} = \psi_{N-1} > 0$. Since $(\Omega \otimes I_n)^T \mathbf{p}^*$ is in the row space of $(\Omega \otimes I_n)$, there exists $\{ \lambda_r \}_{r=1}^R$ such that $-(\Omega \otimes I_n)^T \mathbf{p}^* = \sum_{r=1}^R \lambda_r v_r$. Following [70], choose $\mathbf{p} := \sum_{r=1}^R \lambda_r v_r / \sigma_r^2 = V \text{diag}(\lambda / \sigma^2)$, where diag$(\lambda / \sigma^2) \in \mathbb{R}^{n \times R}$ is a diagonal matrix with its $r$-th diagonal entry is $\lambda_r / \sigma_r^2$ for $r = 1, \ldots, R$. Hence, $\mathbf{p} \in \mathbb{R}^{n \times \mathcal{N}}$ satisfies $-(\Omega \otimes I_n)^T \mathbf{p} = V \Sigma^2 V^T \text{diag}(\lambda / \sigma^2) = -(\Omega \otimes I_n)^T \mathbf{p}^*$; therefore, $-(\Omega \otimes I_n)^T \mathbf{p} \in \partial F(\mathbf{x}^*)$ as well. Using the local bounds $\kappa_i$ on the subdifferential of each $\Phi_i$ at $\mathbf{x}^* \in \mathbb{R}^n$, we can bound $\parallel \mathbf{p} \parallel$ from above as follows:

$$
\parallel \mathbf{p} \parallel^2 = \sum_{r=1}^R \left( \frac{\lambda_r}{\sigma_r^2} \right)^2 \leq \frac{1}{\psi_{N-1}^2} \sum_{r=1}^R \lambda_r^2 = \frac{1}{\sigma_{\min}^2(\Omega)} \parallel (\Omega \otimes I_n)^T \mathbf{p}^* \parallel^2.
$$

Hence, we can conclude that

$$
\parallel \mathbf{p} \parallel^2 \leq \frac{1}{\sigma_{\min}^2(\Omega)} \sum_{i \in \mathcal{N}} \kappa_i^2.
$$

(5.41)
Note $\sigma_{\min}(\Omega) = \psi^2_{N-1}$. Using (5.41) within (5.39) and (5.40), and combining the resulting inequalities together with (5.38) and (5.37) implies the desired bounds.

When $x^0$ is chosen such that $x^0_i = 0$ for $i \in N$, the bounds in Theorem 5.9 can be simplified further. Indeed, observe that $\sum_{i \in N} \frac{1}{\epsilon_i} \|x^* - x^0_i\|^2 = (\sum_{i \in N} L_i + \gamma d_i(d_i + 1)) \|x^*\|^2 = (\gamma \|\Omega\|^2_F + \sum_{i \in N} L_i) \|x^*\|^2$. Therefore, for $t \geq 1$,

$$\max \{ |F(\bar{x}^t) - F^*|, \|\Omega \otimes I_n\| \bar{x}^t\| \} \leq \frac{1}{2t} \left[ \frac{4(d_{\max} + 1)}{\gamma} \left( \sum_{i \in N} \kappa_i^2 \sigma_{\min}(\Omega) + 1 \right) + \left( \gamma \|\Omega\|^2_F + \sum_{i \in N} L_i \right) \|x^*\|^2 \right].$$

### 5.2.3 DPGA-II Algorithm

Let $M \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{N}|}$ be the oriented edge-node incidence matrix of $\mathcal{G}$, defined at the beginning of this section. Then (5.4) can be written as

$$\min_{x \in \mathbb{R}^{|N|}} \left\{ F(x) := \sum_{i \in N} \Phi_i(x_i) : (M \otimes I_n) x = 0 \right\}, \quad (5.42)$$

where $x^T = [x_1^T, \ldots, x_N^T]^T$ and $\Phi_i$ is defined in (5.1). In the rest of this section, assume that (5.3) has a solution, and (5.42) satisfies Assumption 5.5.

For each $(i, j) \in \mathcal{E}$, define new set of primal variables $y_{ij} \in \mathbb{R}^n$, and let $y = [y_{ij}]_{(i,j) \in \mathcal{E}} \in \mathbb{R}^{|\mathcal{E}|}$. Now consider the following equivalent formulation:

$$\min_{y, \{x_i\}_{i \in N}} \sum_{i \in N} \Phi_i(x_i)$$

$$x_i - y_{ij} = 0 : \alpha_{ij}, \forall (i, j) \in \mathcal{E}$$

$$x_j - y_{ij} = 0 : \beta_{ij}, \forall (i, j) \in \mathcal{E}, \quad (5.43)$$

where $\alpha_{ij} \in \mathbb{R}^n$ and $\beta_{ij} \in \mathbb{R}^n$ denote the Lagrange multiplier vectors corresponding to the primal constraints $x_i - y_{ij} = 0$, and $x_j - y_{ij} = 0$ in (5.43). Similarly, define $\alpha = [\alpha_{ij}]_{(i,j) \in \mathcal{E}} \in \mathbb{R}^{|\mathcal{E}|}$, and $\beta = [\beta_{ij}]_{(i,j) \in \mathcal{E}} \in \mathbb{R}^{|\mathcal{E}|}$.

Note that the formulation (5.43) is also a special case of (5.5) with $g(y) = 0$, and one can employ PG-ADMM or SPG-ADMM to solve (5.43). In the rest, we focus on the implementation details of PG-ADMM. Indeed, it can be easily observed that $A_i$ of (5.5) takes the following form for (5.43): $A_i = (I \otimes I_n) \in \mathbb{R}^{nd_i \times n}$, where $1 \in \mathbb{R}^{d_i}$ is a vector of all ones. Furthermore, it can also be easily observed that $\sum_{i=1}^N B_i^T B_i$ in Lemma 5.7 is equal to $2I_{n|\mathcal{E}|}$ for (5.43). For all $i \in N$, we set the stepsize $c_i$ to its upper bound given in Lemma 5.7, i.e., $c_i = 1/(L_i + \gamma \sigma_{\max}(A_i))$. Hence, for the formulation (5.43), this corresponds to setting $c_i = 1/(L_i + \gamma d_i)$ since $\sigma_{\max}(A_i) = \sqrt{d_i}$ for $i \in N$. 

102
The smooth part of the augmented Lagrangian $\phi_\gamma$ corresponding to the formulation (5.43) can be written as

$$\phi_\gamma(x, y, \alpha, \beta) = \sum_{i \in \mathcal{N}} f_i(x_i) + \sum_{(i, j) \in \mathcal{E}} \left( \alpha_{ij}^T (x_i - y_{ij}) + \beta_{ij}^T (x_j - y_{ij}) \right) + \frac{\gamma}{2} \sum_{(i, j) \in \mathcal{E}} \left( \|x_i - y_{ij}\|^2 + \|x_j - y_{ij}\|^2 \right)$$

for a fixed penalty parameter $\gamma > 0$; hence, $\nabla_x \phi_\gamma$ can be computed as

$$\nabla_x \phi_\gamma(x^k, y^k, \alpha^k, \beta^k) = \nabla f(x^k) + \sum_{j: (i, j) \in \mathcal{E}} \left( \alpha_{ij}^k + \gamma (x_i^k - y_{ij}^k) \right) + \sum_{j: (j, i) \in \mathcal{E}} \left( \beta_{ji}^k + \gamma (x_j^k - y_{ij}^k) \right)$$

and the steps of PG-ADMM in (5.6) take the following form:

$$x_{i}^{k+1} = \text{prox}_{\alpha_{ij}, \beta_{ij}} \left( x_i^k - c_j \nabla_x \phi_\gamma(x^k, y^k, \alpha^k, \beta^k) \right), \quad j \in \mathcal{N}, \quad (5.45)$$

$$y_{ij}^{k+1} = \arg\min_{y_{ij}} \left\{ - \left( \alpha_{ij}^k + \beta_{ij}^k \right)^T y_{ij} + \frac{\gamma}{2} \left( \|x_i^{k+1} - y_{ij}\|^2 + \| x_j^{k+1} - y_{ij}\|^2 \right) \right\}, \quad (i, j) \in \mathcal{E}, \quad (5.46)$$

$$\alpha_{ij}^{k+1} = \alpha_{ij}^k + \gamma (x_i^{k+1} - y_{ij}^{k+1}), \quad (i, j) \in \mathcal{E}, \quad (5.47)$$

$$\beta_{ij}^{k+1} = \beta_{ij}^k + \gamma (x_j^{k+1} - y_{ij}^{k+1}), \quad (i, j) \in \mathcal{E}. \quad (5.48)$$

Let $\{x^0_i\}_{i \in \mathcal{N}}$ denote the set of initial primal iterates. For $k \geq 0$, (5.46) can be solved in closed form:

$$y_{ij}^{k+1} = \frac{\alpha_{ij}^k + \beta_{ij}^k}{2\gamma} + \frac{x_i^{k+1} + x_j^{k+1}}{2}.$$  

(5.49)

On the other hand, from (5.47) and (5.48), it follows that for $k \geq 0$

$$\frac{\alpha_{ij}^{k+1} + \beta_{ij}^{k+1}}{2\gamma} = \frac{\alpha_{ij}^k + \beta_{ij}^k}{2\gamma} + \frac{x_i^{k+1} + x_j^{k+1}}{2} - y_{ij}^{k+1} = 0.$$

Hence, for each $(i, j) \in \mathcal{E}$, we have $\alpha_{ij}^k + \beta_{ij}^k = 0$ for $k \geq 1$. Suppose we initialize $\alpha^0 = \beta^0 = 0$, i.e., $\alpha_{ij}^0 = \beta_{ij}^0 = 0$ for all $(i, j) \in \mathcal{E}$. Therefore, using (5.49), we can conclude that for each $(i, j) \in \mathcal{E}$, $y_{ij}^k = (x_i^k + x_j^k)/2$ for all $k \geq 1$. Hence, $\alpha_{ij}^k = \frac{\gamma}{2} \sum_{\ell=1}^k (x_i^\ell - x_j^\ell)$, and $\beta_{ij}^k = \frac{\gamma}{2} \sum_{\ell=1}^k (x_j^\ell - x_i^\ell)$ for all $k \geq 1$. Therefore, combining these recursions, (5.44)
can be computed for all \( k \geq 1 \) as

\[
\nabla_{x_i} \phi_{\gamma}(x^k, y^k, \alpha^k, \beta^k) = \nabla f(x^k_i) + \frac{\gamma}{2} \sum_{j \in O_i} (x^k_i - x^k_j) + \sum_{j \neq i} \alpha^k_{ij} + \sum_{j \neq i} \beta^k_{ij},
\]

\[
= \nabla f(x^k_i) + \frac{\gamma}{2} \left[ \sum_{j \in O_i} (x^k_i - x^k_j) + \sum_{\ell=1}^k \sum_{j \in O_i} (x^\ell_i - x^\ell_j) \right],
\]

\[
= \nabla f(x^k_i) + \frac{\gamma}{2} \left[ \sum_{j \in N_i} \Omega_{ij} x^k_j + \sum_{\ell=1}^k \sum_{j \in N_i} \Omega_{ji} x^\ell_j \right]. \tag{5.50}
\]

Finally, by initializing \( y^0_{ij} = (x^0_i + x^0_j)/2 \) for all \((i, j) \in E\), the computation of \( \nabla_{x_j} \phi_{\gamma} \) in step (5.45) can be simplified. Indeed, define \( s^k_i := \sum_{j \in N_i} \Omega_{ij} x^k_j \) for \( k \geq 0 \); and let \( p^k_i := \gamma \sum_{\ell=1}^k s^k_i \) for \( k \geq 1 \) and \( p^0_i = 0 \). This initialization and (5.50) together imply that for \( k \geq 0 \)

\[
\nabla_{x_i} \phi_{\gamma}(x^k, z^k, \lambda^k) = \nabla f_i(x^k_i) + \frac{1}{2} \left( p^k_i + \gamma s^k_i \right).
\]

Therefore, the steps in (5.45)-(5.48) can be simplified as shown in Figure 5.2 below.

![Algorithm DPGA-II](image)

**Figure 5.2: Distributed Proximal Gradient Algorithm II (DPGA-II)**

### 5.2.4 Error Bounds for DPGA-II & Effect of Topology

In this section, we examine the effect of network topology on the convergence rate of DPGA-II, which is nothing but PG-ADMM customized to the decentralized formulation in (5.43) as discussed in Section 5.2.3. To obtain simple \( \mathcal{O}(1) \) constants in the error bounds, we set \( \alpha^0_{ij} = \beta^0_{ij} = 0 \) for all \((i, j) \in E\).

**Theorem 5.10.** Given arbitrary \( x^0 \) and \( \gamma > 0 \), let \( \{x^k\}_{k \geq 1} \) be the DPGA-II iterate sequence, generated as shown in Figure 5.2. The average sequence \( \{\bar{x}^t\}_{t \geq 1} \), defined as

\[104\]
\[ \bar{x}_i^t = (\sum_{k=1}^{t_i} x_i^k)/t \] for \( i \in \mathcal{N} \) and \( t \geq 1 \), converges to an optimal solution to (5.4) with the following suboptimality and infeasibility bounds satisfied for all \( t \geq 1 \),

\[
- \frac{1}{t} \left( \frac{4}{\gamma \sigma_{\min}(\Omega)} \sum_{i \in \mathcal{N}} \kappa_i^2 + \sum_{i \in \mathcal{N}} \frac{1}{2c_i} \|x^* - x_i^0\|^2 \right) \leq F(\bar{x}^t) - F^* \leq \frac{1}{t} \left( \sum_{i \in \mathcal{N}} \frac{1}{2c_i} \|x^* - x_i^0\|^2 \right),
\]

\[
\left( \sum_{(i,j) \in \mathcal{E}} \|\bar{x}_i^t - \bar{x}_j^t\|^2 \right)^{1/2} \leq \frac{1}{t} \left( \frac{2}{\gamma \sigma_{\min}(\Omega)} \left( \sum_{i \in \mathcal{N}} \kappa_i^2 + \sigma_{\min}(\Omega) \right) + \sum_{i \in \mathcal{N}} \frac{1}{2c_i} \|x^* - x_i^0\|^2 \right),
\]

where \( \kappa_i > 0 \) denotes an upper bound on the elements of \( \partial \Phi_i(x^*) \), i.e., if \( q \in \partial \Phi_i(x^*) \), then \( \|q\| \leq \kappa_i \), for each \( i \in \mathcal{N} \).

**Proof:** Suppose a solution to (5.3) exists; then \( x^* \in \mathbb{R}^n \) is an optimal solution to (5.3) if and only if \( x_i^* = x^* \) for \( i \in \mathcal{N} \) is optimal to (5.42), and \( (x^*, y^*) \) is optimal to (5.43) for \( y^* \in \mathbb{R}^{n|\mathcal{E}|} \) such that \( y_{ij}^* = x^* \) for all \( (i,j) \in \mathcal{E} \). The formulation (5.42), equivalent to (5.43), can be written as

\[
\min_{x} \{ F(x) := \sum_{i \in \mathcal{N}} \Phi_i(x_i) : x_i - x_j = 0 : \theta_{ij}, (i,j) \in \mathcal{E} \},
\]

(5.51)

where \( \theta_{ij} \in \mathbb{R}^n \) denotes the dual variable corresponding to the primal constraint \( x_i - x_j = 0 \).

Recall that we assume (5.42) satisfies Assumption 5.5, hence, there exists a primal-dual optimal pair \( (x^*, \theta^*) \) for (5.42), where \( x^* \in \mathbb{R}^{n|\mathcal{N}|} \) such that \( x_i^* = x^* \) for \( i \in \mathcal{N} \) and \( \theta^* := [\theta_{ij}^*]_{(i,j) \in \mathcal{E}} \). Note that Assumption 5.5 is satisfied if Slater’s condition holds, i.e., \( \bigcap_{i \in \mathcal{N}} r(\text{dom} \, \xi_i) \neq \emptyset \).

According to optimality conditions for (5.51), \( \theta^* := [\theta_{ij}^*]_{(i,j) \in \mathcal{E}} \) is an optimal dual solution to (5.51) if and only if

\[
0 \in \partial \Phi_i(x^*) + \sum_{j: (i,j) \in \mathcal{E}} \theta_{ij}^* - \sum_{j: (j,i) \in \mathcal{E}} \theta_{ji}^*, \quad \forall \ i \in \mathcal{N},
\]

(5.52)

i.e., \( -(M \otimes I_n)^\top \theta^* \in F(x^*) \) and \( x_i^* = x^* \) for \( i \in \mathcal{N} \), where \( x^* \in \mathbb{R}^n \) is an optimal solution to (5.3). Similarly, according to first order optimality conditions for (5.43), \( \alpha^* \in \mathbb{R}^{n|\mathcal{E}|} \) and \( \beta^* \in \mathbb{R}^{n|\mathcal{E}|} \) are dual optimals to (5.43) if and only if

\[
0 \in \partial \Phi_i(x^*) + \sum_{j: (i,j) \in \mathcal{E}} \alpha_{ij}^* + \sum_{j: (j,i) \in \mathcal{E}} \beta_{ji}^*, \quad \forall \ i \in \mathcal{N},
\]

and \( \alpha_{ij}^* + \beta_{ij}^* = 0, \quad \forall \ (i,j) \in \mathcal{E} \).
Therefore, given $\theta^* := [\theta^*_{ij}]_{(i,j) \in E}$ an optimal dual solution to (5.51), one can construct a dual optimal solution $(\alpha^*, \beta^*)$ to (5.43) by simply setting $\alpha^* = \theta^*$ and $\beta^* = -\theta^*$. In the rest of the proof, fix $x^* \in \mathbb{R}^n$ and $\theta^* = [\theta^*_{ij}]_{(i,j) \in E}$ such that $x^*$ is an optimal solution to (5.3), and $\theta^*$ is a dual solution to (5.51); thus, $(x^*, y^*, \alpha^*, \beta^*)$ is a primal-dual optimal solution to (5.43), where $x_i^* = x^*$ for $i \in \mathcal{N}$, $y_{ij}^* = x^*$, $\alpha^*_{ij} = \theta^*_{ij}$ and $\beta^*_{ij} = -\theta^*_{ij}$ for all $(i, j) \in \mathcal{E}$.

For $i \in \mathcal{N}$, since $A_i = (1 \otimes I_n) \in \mathbb{R}^{nd_i \times n}$, where $1 \in \mathbb{R}^{d_i}$ is a vector of all ones, $A_i^T A_i = d_i I_n$, hence, $c_i = 1/(L_i + \gamma d_i)$ implies that $I_n - \gamma c_i A_i^T A_i = L_i/(L_i + \gamma d_i) I_n$, which leads to $1/c_i \|x_i^0 - x^*\|_i - \gamma c_i A_i^T A_i = L_i \|x_i^0 - x^*\|^2_i$. Also $\sum_{i=1}^{N} d_i B_i = 2I_{n|\mathcal{E}|}$; and since we initialize $y^0$ such that $y^0_{ij} = (x_i^0 + x_j^0)/2$ and $y_{ij}^* = x^*$ for all $(i, j) \in \mathcal{E}$, it trivially follows that

$$
\|y^* - y^0\|^2_{\sum_{i=1}^{N} B_i^T B_i} = 2 \sum_{(i,j) \in \mathcal{E}} \|x^* - (x_i^0 + x_j^0)/2\|^2 \leq \sum_{(i,j) \in \mathcal{E}} \left( \|x^* - x_i^0\|^2 + \|x^* - x_j^0\|^2 \right) = \sum_{i \in \mathcal{N}} d_i \|x^* - x_i^0\|^2.
$$

Recall that we initialize $\alpha^0 = \beta^0 = 0$. Furthermore, Lemma 5.7 holds for all $\alpha = [\alpha_{ij}]_{(i,j) \in \mathcal{E}} \in \mathbb{R}^{n|\mathcal{E}|}$ and $\beta = [\beta_{ij}]_{(i,j) \in \mathcal{E}} \in \mathbb{R}^{n|\mathcal{E}|}$, where $\alpha_{ij}, \beta_{ij} \in \mathbb{R}^n$ for $(i, j) \in \mathcal{E}$. Hence, given $\theta = [\theta_{ij}]_{(i,j) \in \mathcal{E}}$ for some $\theta_{ij} \in \mathbb{R}^n$ for all $(i, j) \in \mathcal{E}$, we set $\alpha_{ij} = \theta_{ij}$ and $\beta_{ij} = -\theta_{ij}$ for all $(i, j) \in \mathcal{E}$, and invoke Lemma 5.7 for this specific choice of $\alpha = \theta$ and $\beta = -\theta$ to obtain the following inequality, customized for (5.30): for all $\theta$ and $t \geq 1$,

$$
F(\bar{x}^t) - F^* + \theta^T (M \otimes I_n) \bar{x}^t = F(\bar{x}^t) - F^* + \sum_{(i,j) \in \mathcal{E}} \theta^T_{ij} (\bar{x}_i^t - \bar{x}_j^t)
$$

$$
= F(\bar{x}^t) - F^* + \sum_{(i,j) \in \mathcal{E}} \alpha^T_{ij} (\bar{x}_i^t - \bar{y}_i^j) + \beta^T_{ij} (\bar{x}_j^t - \bar{y}_i^j)
$$

$$
\leq \frac{1}{\gamma t} \sum_{(i,j) \in \mathcal{E}} \|\theta_{ij}\|^2 + \frac{1}{2t} \sum_{i \in \mathcal{N}} L_i \|x^* - x_i^0\|^2 + \frac{\gamma}{2t} \|y^* - y^0\|^2_{\sum_{i=1}^{N} B_i^T B_i}
$$

$$
\leq \frac{1}{t} \left( \frac{1}{\gamma} \|\theta\|^2 + \sum_{i \in \mathcal{N}} \frac{1}{2c_i} \|x^* - x_i^0\|^2 \right). \quad (5.53)
$$

Hence, setting $\theta = 0$ in (5.53) leads to

$$
F(\bar{x}^t) - F^* \leq \frac{1}{t} \sum_{i \in \mathcal{N}} \frac{1}{2c_i} \|x^* - x_i^0\|^2. \quad (5.54)
$$
From convexity of $\Phi_i$, (5.52) and the fact that $(M \otimes I_n)x^* = 0$, it follows that
\[
0 \leq F(\bar{x}^t) - F^* + \theta^*^T (M \otimes I_n)(\bar{x}^t - x^*) = F(\bar{x}^t) - F^* + \theta^*^T (M \otimes I_n)\bar{x}^t. \tag{5.55}
\]

Adding the last term to both sides, and invoking (5.53) for $\theta$ such that $\theta_{ij} = 2\theta_{ij}^*$ for $(i, j) \in \mathcal{E}$, we get
\[
\theta^*^T (M \otimes I_n)\bar{x}^t \leq F(\bar{x}^t) - F^* + (2\theta^*)^T (M \otimes I_n)\bar{x}^t \leq \frac{1}{t} \left( \frac{4}{\gamma} \|\theta^*\|^2 + \sum_{i \in \mathcal{N}} \frac{1}{2c_i} \|x^* - x_i^0\|^2 \right). \tag{5.56}
\]

Invoking (5.53) once again for $\theta = \theta^* + (M \otimes I_n)\bar{x}^t/\|(M \otimes I_n)\bar{x}^t\|$ and using (5.55), we get
\[
\left( \sum_{(i,j) \in \mathcal{E}} \|\bar{x}^t_i - \bar{x}^t_j\|^2 \right)^{1/2} = \|(M \otimes I_n)\bar{x}^t\| \leq F(\bar{x}^t) - F^* + \theta^*^T (M \otimes I_n)\bar{x}^t + \|(M \otimes I_n)\bar{x}^t\|
\leq \frac{1}{t} \left( \sum_{(i,j) \in \mathcal{E}} \frac{1}{\gamma} \|\theta^*\|_ij + \frac{\bar{x}^t_i - \bar{x}^t_j}{\|(M \otimes I_n)\bar{x}^t\|} \right)^2 + \sum_{i \in \mathcal{N}} \frac{1}{2c_i} \|x^* - x_i^0\|^2
\leq \frac{1}{t} \left( \frac{2}{\gamma} (1 + \|\theta^*\|^2) + \sum_{i \in \mathcal{N}} \frac{1}{2c_i} \|x^* - x_i^0\|^2 \right) \tag{5.57}
\]
where the last inequality follows from $\|a + b\|^2 \leq 2\|a\|^2 + 2\|b\|^2$. Next, by appropriately choosing a dual optimal (5.51), we bound $\|\theta^*\|$, which appears in (5.56) and (5.57).

Recall that $M^T M = \Omega \in \mathbb{R}^{|\mathcal{N}| \times |\mathcal{N}|}$, i.e., the Laplacian of $\mathcal{G}$, and since $\mathcal{G}$ is connected, we have $\text{rank}(M) = \text{rank}(\Omega) = N-1$ and $\psi_1 \geq \ldots \geq \psi_{N-1} > \psi_N = 0$, where $\{\psi_i\}_{i=1}^N$ denote the eigenvalues of $\Omega$. Since $(M \otimes I_n)^T \theta^*$ lies in the row space, there exists $\{\lambda_r\}_{r=1}^R$ such that $-(M \otimes I_n)^T \theta^* = \sum_{r=1}^R \lambda_r v_r$. Following [70], choose $\tilde{\theta} := -\sum_{r=1}^R u_r \lambda_r / \sigma_r = U \text{diag}(\lambda / \sigma)$, where $\text{diag}(\lambda / \sigma) \in \mathbb{R}^{R \times R}$ is a diagonal matrix with its $r$-th diagonal entry is $\lambda_r / \sigma_r$ for $r = 1, \ldots, R$. Hence, $\tilde{\theta} \in \mathbb{R}^{n|\mathcal{E}|}$ satisfies $-(M \otimes I_n)^T \tilde{\theta} = V \Sigma U^T U \text{diag}(\lambda / \sigma) = -(M \otimes I_n)^T \theta^*$; therefore, $-(M \otimes I_n)^T \tilde{\theta} \in \partial F(x^*)$ as well. Recall that the optimality conditions stated in (5.52) for the formulation (5.51) can be written more compactly as $0 \in \partial F(x^*) + (M \otimes I_n)^T \theta^*$; hence, $\tilde{\theta}$ is an optimal dual solution to (5.51) as well. Finally, using the local bounds $\kappa_i$ on the subdifferential of $\Phi_i$ at $x^* \in \mathbb{R}^n$, we can bound $\|\tilde{\theta}\|$ from above as follows:
\[
\|\tilde{\theta}\|^2 = \sum_{r=1}^R \frac{\lambda_r^2}{\sigma_r^2} \leq \frac{1}{\psi_{N-1}} \sum_{r=1}^R \frac{\lambda_r^2}{\sigma_{\min}(\Omega)} \|(M \otimes I_n)^T \theta^*\|^2.
\]
Hence, \( \|\bar{\theta}\|^2 \leq \frac{1}{\sigma_{\min}(\Omega)} \sum_{i \in \mathcal{N}} \kappa_i^2 \). Thus, using this bound within (5.56) and (5.57), and combining the resulting inequalities together with (5.54) and (5.55) implies the desired bounds.

Note when \( x^0 \) is chosen such that \( x^0_i = 0 \) for \( i \in \mathcal{N} \), the bounds in Theorem 5.10 can be simplified further. Indeed, observe that

\[
\sum_{i \in \mathcal{N}} \frac{1}{c_i} \|x^* - x^0_i\|^2 = \left( \sum_{i \in \mathcal{N}} L_i + \gamma d_i \right) \|x^*\|^2 = (2\gamma |\mathcal{E}| + \sum_{i \in \mathcal{N}} L_i) \|x^*\|^2.
\]

Therefore, for all \( t \geq 1 \),

\[
\max \left\{ |F(\bar{x}^t) - F^*|, \left( \sum_{(i,j) \in \mathcal{E}} \|x^i_t - x^j_t\|^2 \right)^{1/2} \right\} \leq \frac{4}{\gamma} \left( \sum_{i \in \mathcal{N}} \frac{\kappa_i^2}{\sigma_{\min}(\Omega)} + 1 \right) + \left( \gamma |\mathcal{E}| + \sum_{i \in \mathcal{N}} L_i \right) \|x^*\|^2.
\]

### 5.2.5 Adaptive step-size strategy

One important property of DPGA methods is their ability to adopt an adaptive step-size sequence for each node. Note \( L_i \), Lipschitz constants of \( \nabla f_i \), may not be known in advance or may be too large for certain nodes – leading to very small steps since \( c_i = O(L_i^{-1}) \), i.e., \( c_i = (L_i + \gamma d_i(d_i + 1))^{-1} \) for DPGA-I, and \( c_i = (L_i + \gamma d_i)^{-1} \) for DPGA-II – see Fig. 5.1 and Fig. 5.2. On the other hand, it is elementary to check that all the proofs given above still go through if node \( i \in \mathcal{N} \) uses the step size \( c^k_i = (s^k_i + \gamma d_i(d_i + 1))^{-1} \) for DPGA-I and \( c^k_i = (s^k_i + \gamma d_i)^{-1} \) for DPGA-II at the \( k \)-th iteration such that

\[
f_i(x^i_{k+1}) \leq f_i(x^i_k) + \left\langle \nabla f_i(x^i_k), \Delta^k_i \right\rangle + \frac{s^k_i}{2} \|\Delta^k_i\|^2
\]

holds, where \( \Delta^k_i := x^i_{k+1} - x^i_k \) and \( x^i_{k+1} \) is computed using \( c^k_i \) instead of \( c_i \). Clearly, \( s^k_i \leq L_i \). Since this condition can be checked locally, one can possibly take longer steps compared to \( c_i \) and still has a convergence guarantee. In contrast, distributed algorithms that use constant step size \( c > 0 \) for all nodes, e.g., PG-EXTRA [74], cannot take advantage of this trick. We adopted the following rule in our numerical tests: let \( v > 1 \), for \( k \geq 1 \) we set \( s^k_i = s^{k-1}_i \ell_{k-1} \) where \( \ell_{k} \geq 0 \) is the smallest integer such that (5.58) holds, and \( s^0_i = L_i \) for \( i \in \mathcal{N} \).

### 5.2.6 Stochastic gradient variants of DPGA-I and DPGA-II

As corollaries of Theorem 5.9 and Theorem 5.10 we provide the error bounds for the stochastic gradient variants of DPGA-I and DPGA-II, displayed in Figure 5.3 and Figure 5.4, respectively. Both SDPGA-I and SDPGA-II employ SFO \( G_i \) defined in Definition 5.2 for \( i \in \mathcal{N} \) instead of accessing to \( \nabla f_i \). Suppose that \( \cap_{i \in \mathcal{N}} \text{ri(dom} \xi_i) \neq \emptyset \)
and $D := \max_{i \in \mathcal{N}} \sup_{x,x' \in \text{dom}(\xi_i)} ||x - x'|| < +\infty$. Then under Assumption 5.3, slightly modifying the proofs of Theorem 5.9 and Theorem 5.10 by invoking the result of Lemma 5.7 for the case $\sigma > 0$, we immediately obtain the bounds for SDPGA-I in Corollary 5.11 and SDPGA-II in Corollary 5.12.

**Corollary 5.11.** Given arbitrary $x^0$ and $\gamma > 0$, let $p_i^0 = 0$ for all $i \in \mathcal{N}$, and $\{x^k\}_{k \geq 1}$ be the SDPGA-I iterate sequence, generated as shown in Figure 5.3. The average sequence $\{\bar{x}^t\}_{t \geq 1}$, defined as $\bar{x}^t_i = (\sum_{k=1}^t x^k_i)/t$ for $i \in \mathcal{N}$ and $t \geq 1$, w.p.1 converges to an optimal solution to (5.4) with the following suboptimality and infeasibility bounds satisfied for all $t \geq 1$,

$$
E[\|F(\bar{x}^t) - F^*\|] \leq \frac{1}{t} \left( \frac{2(d_{\max} + 1)}{\gamma \sigma_2^2(\Omega)} \sum_{i \in \mathcal{N}} \kappa_i^2 + \sum_{i \in \mathcal{N}} \frac{1}{2} \|x^* - x_0^0\|^2 \right) + \frac{N}{\sqrt{t}} (D^2 + 2\sigma^2)
$$

$$
E[\|(\Omega \otimes I_n)\bar{x}^t\|] \leq \frac{1}{t} \left( \frac{d_{\max} + 1}{\gamma \sigma_2^2(\Omega)} \left( \sum_{i \in \mathcal{N}} \kappa_i^2 + \sigma_2^2(\Omega) \right) + \sum_{i \in \mathcal{N}} \frac{1}{2} \|x^* - x_0^0\|^2 \right) + \frac{N}{\sqrt{t}} (D^2 + 2\sigma^2)
$$

where $\kappa_i > 0$ denotes an upper bound on the elements of $\partial \Phi_i(x^*)$, i.e., if $q \in \partial \Phi_i(x^*)$, then $\|q\| \leq \kappa_i$, for each $i \in \mathcal{N}$.

**Corollary 5.12.** Given arbitrary $x^0$ and $\gamma > 0$, let $p_i^0 = 0$ for all $i \in \mathcal{N}$, and $\{x^k\}_{k \geq 1}$ be the SDPGA-II iterate sequence, generated as shown in Figure 5.4. The average sequence $\{\bar{x}^t\}_{t \geq 1}$, defined as $\bar{x}^t_i = (\sum_{k=1}^t x^k_i)/t$ for $i \in \mathcal{N}$ and $t \geq 1$, w.p.1 converges to an optimal solution to (5.4) with the following suboptimality and infeasibility bounds satisfied for all $t \geq 1$,

$$
E[\|F(\bar{x}^t) - F^*\|] \leq \frac{1}{t} \left( \frac{4}{\gamma \sigma_2(\Omega)} \sum_{i \in \mathcal{N}} \kappa_i^2 + \sum_{i \in \mathcal{N}} \frac{1}{2} \|x^* - x_0^0\|^2 \right) + \frac{N}{\sqrt{t}} (D^2 + 2\sigma^2)
$$

$$
E \left[ \sum_{(i,j) \in \mathcal{E}} \|\bar{x}^t_i - \bar{x}^t_j\|^2 \right] \leq \frac{1}{t} \left( \frac{2}{\gamma \sigma_2(\Omega)} \left( \sum_{i \in \mathcal{N}} \kappa_i^2 + \sigma_2^2(\Omega) \right) + \sum_{i \in \mathcal{N}} \frac{1}{2} \|x^* - x_0^0\|^2 \right) + \frac{N}{\sqrt{t}} (D^2 + 2\sigma^2)
$$

where $\kappa_i > 0$ denotes an upper bound on the elements of $\partial \Phi_i(x^*)$, i.e., if $q \in \partial \Phi_i(x^*)$, then $\|q\| \leq \kappa_i$, for each $i \in \mathcal{N}$.

### 5.3 Numerical results

In this section, we compared DPGA-I and DPGA-II with PG-EXTRA, distributed ADMM and its variant proposed in [74], [70] and [87], respectively, on the sparse group...
LASSO problem with Huber loss (the same problem we used in Chapter 4):

\[
\text{min}_{x \in \mathbb{R}^n} \sum_{i \in \mathcal{N}} [\beta_1 \|x\|_1 + \beta_2 \|x\|_{G_i} + h_\delta(A_i x - b_i)],
\]

where \(\beta_1, \beta_2 > 0\), for each \(i \in \mathcal{N}\), \(A_i \in \mathbb{R}^{m_i \times n}\), \(b_i \in \mathbb{R}^{m_i}\), and \(\|x\|_{G_i} := \sum_{k=1}^K \|x_{g_i(k)}\|_2\) denotes the group norm with respect to the partition \(G_i\) of \([1, n]\) := \{1, \ldots, n\}, i.e., \(G_i = \{g_i(k)\}_{k=1}^K\) such that \(\bigcup_{k=1}^K g_i(k) = [1, n]\), and \(g_i(j) \cap g_i(k) = \emptyset\) for all \(j \neq k\); and \(h_\delta\) denotes the Huber loss function, i.e., for any \(m \geq 1\) let \(h_\delta : \mathbb{R}^m \rightarrow \mathbb{R}\) such that

\[
h_\delta(y) := \max\{z^T y - \frac{1}{2\delta} \|z\|^2 : \|z\|_\infty \leq \delta\}.
\]

In this case, \(f_i(x) := h_\delta(A_i x - b_i)\) and \(\xi_i(x) := \beta_1 \|x\|_1 + \beta_2 \|x\|_{G_i} \).

Next, we briefly describe the competitive algorithms: PG-EXTRA [74], the distributed ADMM algorithm in [70]; and a more efficient variant to the ADMM that exploits the problem structure in (5.59). Recall that \(\Omega \in \mathbb{R}^{N \times N}\) denotes the Laplacian of the graph \(\mathcal{G} = (\mathcal{N}, \mathcal{E})\), \(\mathcal{N}_i := \mathcal{O}_i \cup \{i\}\), where \(\mathcal{O}_i\) denotes the set of neighboring nodes of \(i \in \mathcal{N}\).
5.3.1 Distributed ADMM Algorithm

As discussed in Section 5.2.1, (5.4) can be equivalently written as in (5.30). Makhdoumi & Ozdaglar [70] establish that when an ADMM algorithm with penalty parameter $\gamma > 0$ is implemented on (5.30) alternatingly minimizing in $x$ and $y$, the subproblems can be simplified as shown in Figure 5.5; hence, the primal iterates $y$ and the dual iterates $\lambda$ are never computed in the implementation. It is shown in [70] that suboptimality and consensus violation converge to 0 with a rate $\mathcal{O}(1/k)$, and in each iteration every node communicates $2n$ scalars, i.e., $x_i \in \mathbb{R}^n$ and $p_i + \gamma s_i \in \mathbb{R}^n$. Moreover, each node stores $3n$ scalars at each iteration, i.e., $x_i, s_i, p_i \in \mathbb{R}^n$.

Algorithm ADMM ($\gamma, x^0$)

Initialization: $c_i = (\gamma d_i (d_i + 1))^{-1}$, $p_i^0 = 0$, $i \in \mathcal{N}$

Step $k$: ($k \geq 0$) For $i \in \mathcal{N}$ compute

1. $x_i^{k+1} = \text{prox}_{c_i (\xi_i + f_i)} \left( x_i^k - c_i \sum_{j \in \mathcal{N}_i} \Omega_{ij} (p_j^k + \gamma s_j^k) \right)$.
2. $s_i^{k+1} = \sum_{j \in \mathcal{N}_i} \Omega_{ij} x_j^{k+1} / (d_i + 1)$
3. $p_i^{k+1} = p_i^k + \gamma s_i^{k+1}$

Figure 5.5: ADMM algorithm

From now on, we refer to this algorithm that directly works with $\Phi_i = \xi_i + f_i$ as ADMM—see Fig. 5.5. Computing $\text{prox}_{\Phi_i}$ for each $i \in \mathcal{N}$ is the computational bottleneck in each iteration of ADMM. Note that computing $\text{prox}_{\Phi_i}$ for (5.59) is almost as hard as solving the problem. To deal with this issue, Aybat et al. [87] considered the following reformulation:

$$\min_{x_i, y_i \in \mathbb{R}^n, z_i, \tilde{z}_i \in \mathbb{Z}_i} \sum_{i \in \mathcal{N}} \xi_i (x_i) + f_i (y_i)$$

s.t. $\Omega_{ij} x_j = z_{ij}$, $\Omega_{ij} y_j = \tilde{z}_{ij}$, $i \in \mathcal{N}$, $j \in \mathcal{N}_i$

$x_i = q_i$, $y_i = q_i$, $i \in \mathcal{N}$,

and proposed a split ADMM algorithm (SADMM), displayed in Fig. 5.6 where $c > 0$ denotes the penalty parameter. Steps of SADMM can be derived by minimizing the augmented Lagrangian alternatingly in $(x, y)$, and in $(z, \tilde{z}, q)$ while fixing the other. As in [70], computing $(z, \tilde{z}, q)$ can be avoided by exploiting the structure of optimality conditions. Convergence of SADMM with $\mathcal{O}(1/k)$ rate follows immediately from the results on the convergence of ADMM [92]. In each iteration of SADMM, every node
communicates \(4n\) scalars, i.e., \(x_i \in \mathbb{R}^n\), \(y_i \in \mathbb{R}^n\), \(p_i + \gamma s_i \in \mathbb{R}^n\) and \(\bar{p}_i + \gamma \bar{s}_i \in \mathbb{R}^n\). Moreover, each node stores \(7n\) scalars, i.e., \(x_i, y_i, s_i, \bar{s}_i, p_i, \bar{p}_i, r_i \in \mathbb{R}^n\).

**Algorithm SADMM (\(\gamma, x^0\))**

Initialization: \(y^0 = x^0, c_i = (\gamma d_i(d_i + 1) + \gamma)^{-1}, p_i^0 = \bar{p}_i^0 = 0, i \in \mathcal{N}\)

Step \(k\): (\(k \geq 0\)) For \(i \in \mathcal{N}\) compute

1. \(x_i^{k+1} = \text{prox}_{c_i \xi_i}\left(x_i^k - c_i[x_i^k + \gamma(x_i^k - y_i^k)/2 + \sum_{j \in \mathcal{N}_i} \Omega_{ij}(p_j^k + \gamma s_j^k)]\right)\)
2. \(y_i^{k+1} = \text{prox}_{c_i f_i}\left(y_i^k + c_i[x_i^k + \gamma(x_i^k - y_i^k)/2 - \sum_{j \in \mathcal{N}_i} \Omega_{ij}(\bar{p}_j^k + \gamma \bar{s}_j^k)]\right)\)
3. \(s_i^{k+1} = \sum_{j \in \mathcal{N}_i} \Omega_{ij}x_j^{k+1}/(d_i + 1), \quad p_i^{k+1} = p_i^k + \gamma s_i^{k+1}\)
4. \(\bar{s}_i^{k+1} = \sum_{j \in \mathcal{N}_i} \Omega_{ij}y_j^{k+1}/(d_i + 1), \quad \bar{p}_i^{k+1} = \bar{p}_i^k + \gamma \bar{s}_i^{k+1}\)
5. \(r_i^{k+1} = \gamma_i^k + \gamma(x_i^{k+1} - y_i^{k+1})/2\)

Figure 5.6: Split ADMM algorithm

### 5.3.2 PG-EXTRA

Given two mixing matrices, \(W = [W_{ij}], \bar{W} = [\bar{W}_{ij}] \in \mathbb{R}^{|\mathcal{N}| \times |\mathcal{N}|}\), Shi et al. \[74\] show that PG-EXTRA, displayed in Fig. 5.7, can solve (5.3) with \(\Phi_i = \xi_i + f_i\) as in (5.1) (\(\xi_i\) and \(f_i\) convex and \(\nabla f_i\) is Lipchitz continuous with constant \(L_i\)) in a distributed fashion with \(\mathcal{O}(1/k)\) rates on sub-optimality and consensus violation in terms of squared residuals of KKT violation and consensus violation, respectively. It is assumed that \(W\) and \(\bar{W}\) satisfy: i) \(W = W^T\) and \(\bar{W} = \bar{W}^T\), ii) for each \(i \in \mathcal{N}\), \(W_{ij} = \bar{W}_{ij} = 0\) for all \(j \notin \mathcal{N}_i\), iii) \(\{x \in \mathbb{R}^{|\mathcal{N}|} : (W - \bar{W})x = 0\} = \{t1 : t \in \mathbb{R}\}\) and \(\{x \in \mathbb{R}^{|\mathcal{N}|} : (I - \bar{W})x = 0\} \supseteq \{t1 : t \in \mathbb{R}\}\), iv) \(\bar{W} \succ 0\) and \(\frac{L + W}{2} \succeq \bar{W} \succeq W\). For instance, when \(W = W^T\) such that \(W_{ij} \geq 0\) for all \(i, j\), and \(W1 = 1\), setting \(\bar{W} := \frac{L + W}{2}\) guarantees that \((W, \bar{W})\) satisfy these assumptions. In our experiments, the mixing matrices are chosen as \(W := I - \frac{\Omega}{d_{\max} + 1}\) and \(\bar{W} := \frac{L + W}{2} = I - \frac{\Omega}{2(d_{\max} + 1)}\) (see Section 2.3 in \[75\]).

According to \[74\], \(x^* = [x^*_i]_{i \in \mathcal{N}}\) is an optimal solution to (5.4) with \(\Phi_i = \xi_i + f_i\) as in (5.1) if and only if there exist \(q_i^* \in \mathbb{R}^{n|\mathcal{N}|}\) and \(g_i^* \in \mathbb{R}^{n|\mathcal{N}|}\) such that \(q^* = (U \otimes I_n)p\) for some \(p \in \mathbb{R}^{n|\mathcal{N}|}\) and \(g_i^* \in \partial \xi_i(x_i^*)\) satisfying \((U \otimes I_n)x^* = 0\) and \((U \otimes I_n)q^* + c(g^* + \nabla f(x^*)) = 0\), where \(c > 0\) is the given step size parameter, \(U := (W - \bar{W})^{1/2}\), and \(\nabla f(x) := \nabla f_i(x_i)_{i \in \mathcal{N}} \in \mathbb{R}^{n|\mathcal{N}|}\) for any \(x = [x_i]_{i \in \mathcal{N}}\). Note that null space of \(U\) only contains \(1 \in \mathbb{R}^{n|\mathcal{N}|}\); therefore, \((U \otimes I_n)x = 0\) implies \(x_1 = \ldots = x_N\).
Let \( \{x^k\} \) be the PG-EXTRA iterate sequence generated as in Fig. 5.7 and sequence \( \{q^k\} \) be defined as

\[
q^k = \sum_{t=0}^{k}(U \otimes I_n)x^t.
\]

According to Theorem 1 and Theorem 2 in [74], when the step size \( c \in \left(0, \frac{2\lambda_{\text{max}}(W)}{L_{\text{max}}}\right) \), where \( L_{\text{max}} = \max_{i \in \mathcal{N}} L_i \), then \( \{x^k\} \) satisfies

\[
\frac{1}{t} \sum_{k=0}^{t} \| (U \otimes I_n)q^k + c(\nabla f(x^k) + g^{k+1}) \|^2_W = \mathcal{O}\left(\frac{1}{t}\right),
\]

\[
\frac{1}{t} \sum_{k=0}^{t} \| (U \otimes I_n)x^k \|^2 = \mathcal{O}\left(\frac{1}{t}\right),
\]

where \( g_i^{k+1} \in \partial \xi_i(x_i^{k+1}) \). As also pointed out in the introduction, we consider this rate result as \( \mathcal{O}(1/\sqrt{t}) \) because (5.60) can only guarantee \( \| (U \otimes I_n)\bar{x}^t \| = \mathcal{O}(1/\sqrt{t}) \), where \( \bar{x}^t := \sum_{k=1}^{t}x^k/t \). On the other hand, according to Theorems 5.9 and 5.10, DPGA-I and DPGA-II iterate sequences satisfy \( \| (\Omega \otimes I_n)\bar{x}^t \| = \mathcal{O}(1/t) \) and \( (\sum_{(ij) \in \mathcal{E}} \| \bar{x}_i^t - \bar{x}_j^t \|^2)^{1/2} = \mathcal{O}(1/t) \), respectively.

Adopting mixing matrices \( (W, \bar{W}) \) to be able to set the stepsize \( c > 0 \) independent of the global topology of \( \mathcal{G} = (\mathcal{N}, \mathcal{E}) \) may still require certain parameters, determined by the global topology of \( \mathcal{G} \), to be in the common knowledge of all nodes \( \mathcal{N} \). In particular, \( W = I - \frac{\Omega}{2d_{\text{max}}} \succeq 0 \) and \( \bar{W} = \frac{L+W}{2} \succeq \frac{1}{2}I \); hence, \( c \) can be chosen \( c \in (0, \frac{1}{L_{\text{max}}} ) \), which is independent of the global topology, and only depends on \( L_{\text{max}} \); however, all the nodes need to know \( d_{\text{max}} \), and \( L_{\text{max}} \) which can be computed using some max-consensus algorithm. This assumption may not be attainable for very large scale fully distributed networks, and computing parameters such as \( L_{\text{max}} \) and \( d_{\text{max}} \) may violate the privacy requirements of the nodes. Also note that since the stepsize \( c > 0 \) is the same for all nodes, PG-EXTRA cannot take advantage of the adaptive step size strategy discussed in Section 5.2.5.

**Algorithm PG-EXTRA \( (c, x^0, W, \bar{W}) \)**

**Step 0:** all nodes \( i \in \mathcal{N} \) do
1. \( x_i^{1/2} = \sum_{j \in \mathcal{N}_i} W_{ij}x_j^0 - c\nabla f_i(x_i^0) \)
2. \( x_i^1 = \text{prox}_{\xi_i}(x_i^{1/2}) \)

**Step k:** \( (k \geq 1) \) all nodes \( i \in \mathcal{N} \) do
1. \( x_i^{k+1+1/2} = \sum_{j \in \mathcal{N}_i} W_{ij}x_j^k - \sum_{j \in \mathcal{N}_i} \bar{W}_{ij}x_j^k + x_i^{k+1/2} - c[\nabla f_i(x_i^{k+1}) - \nabla f_i(x_i^k)] \)
2. \( x_i^{k+2} = \text{prox}_{\xi_i}(x_i^{k+1+1/2}) \)

Figure 5.7: Proximal Gradient Exact First-order Algorithm (PG-EXTRA)
### Table 5.1: Comparison of DPGA-I, DPGA-II, PG-EXTRA, ADMM, and SADMM (Termination time T=1800 sec)

<table>
<thead>
<tr>
<th>Size</th>
<th>Alg.</th>
<th>Rel. Suboptimality</th>
<th>Consensus Violation (V)</th>
<th>Walltime (sec.)</th>
<th># of communication rounds</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Case 1</td>
</tr>
</tbody>
</table>
| 5.3.3 Implementation details and numerical results

In Lemma 4.15, we show that \( \text{prox}_{\xi} \) can be computed in closed form. On the other hand, when ADMM, and SADMM are implemented on (5.59), one needs to compute \( \text{prox}_{\phi} \) and \( \text{prox}_{f_i} \), respectively; however, these proximal operations do not assume closed form solutions. Therefore, in order to be fair, we computed them using an efficient interior point solver MOSEK (ver. 7.1.0.12).

In our experiments, the network was either a star tree or a clique with either 5 or 10 nodes. The remaining problem parameters defining \( \{\xi_i; f_i\}_{i \in N} \) were set as follows. We set \( \beta_1 = \beta_2 = \frac{1}{N}, \delta = 1 \), and \( K = 10 \). Let \( n = Kn_g \) for \( n_g \in \{100, 300\} \), i.e., \( n \in \{1000, 3000\} \). We generated partitions \( \{G_i\}_{i \in N} \) in two different ways. For test problems in CASE 1,
we created a single partition \( G = \{g(k)\}_{k=1}^{K} \) by generating \( K \) groups uniformly at random such that \( |g(k)| = n_g \) for all \( k \); and set \( G_i = G \) for all \( i \in \mathcal{N} \), i.e., \( \xi_i(x) := \beta_1 \|x\|_1 + \beta_2 \|x\|_G \) for all \( i \in \mathcal{N} \). For the test problems in Case 2, we created a different partition \( G_i \) for each node \( i \), in the same manner as in Case 1. For all \( i \in \mathcal{N} \), \( m_i = \frac{n}{2N} \), \( \mathbb{R}^{m_i} \ni b_i = A_i \bar{x} \) for \( \bar{x}_j = (-1)^j e^{-(j-1)/m_{\pi}} \) for \( j \in [1, n] \), and \( A_i \in \mathbb{R}^{m_i \times n} \) is set as \( A_i = 0.5 \pi_i \bar{A}_i \), where the elements of \( \bar{A}_i \in \mathbb{R}^{m_i \times n} \) are i.i.d. Bernoulli random variables with success probability \( \frac{1}{2} \). Our choice of \( \{A_i\}_{i \in \mathcal{N}} \) will make sure that there is a significant deviation among \( \{L_i\} \), i.e., \( \max_{i \in \mathcal{N}} L_i / \min_{i \in \mathcal{N}} L_i \approx 4 \) since \( L_i = \sigma_{\text{max}}^2(A_i) \). This type of setting is expected to adversely affect constant step algorithms, e.g., \( c = \mathcal{O}(1/L_{\text{max}}) \) for PG-EXTRA. For all the algorithms, we initialize the iterate sequence from the origin. And for ADMM methods, the penalty parameter was chosen specifically for each problem setup by searching for the best penalty over a line segment where the total number of ADMM iterations to terminate exhibits a convex behavior – similar to Section 4.2.1 in [93].

We solved the distributed optimization problem (5.4) using PG-EXTRA, DPGA-I, DPGA-II, ADMM, and SADMM for both cases, on both star trees, and cliques, and for \( N = \{5, 10\} \) and \( n_g \in \{100, 300\} \). For each problem setting, we randomly generated 5 instances. For benchmarking, we solved the central problem (5.59) using SDPT3 for both cases. Note that for Case 1, \( \sum_{i \in \mathcal{N}} \xi_i(x) = \|x\|_1 + \|x\|_G \) and its prox map can be computed efficiently, while for Case 2, \( \sum_{i \in \mathcal{N}} \xi_i(x) \) does not assume a simple prox map. Therefore, for Case 1 we were also able to use FISTA [29, 30, 94] to solve the central problem (5.59) by exploiting the result of Lemma 4.13. All the algorithms are terminated when the relative suboptimality, \( |F^k - F^*|/|F^*| \), is less than \( 10^{-3} \), and consensus violation, \( V^k \), is less than \( 10^{-4} \), where \( F^k \) equals \( \sum_{i \in \mathcal{N}} \Phi_i(x^k_i) \) for PG-EXTRA, DPGA-I, DPGA-II, DFAL and ADMM, and to \( \sum_{i \in \mathcal{N}} \Phi_i \left( \frac{x_i^k + y_i^k}{2} \right) \) for SADMM; \( V^k \) equals to \( \max_{(ij) \in \mathcal{E}} \|x_i^k - x_j^k\|_2/\sqrt{n} \) for PG-EXTRA, DPGA-I, DPGA-II, DFAL, and ADMM, and to \( \max_{(ij) \in \mathcal{E}} \|x_i^k - x_j^k\|_2, \max_{i \in \mathcal{N}} \|x_i^k - y_i^k\|_2 \)/\sqrt{n} \) for SADMM. If the stopping criteria are not satisfied in \( T = 1800 \) seconds (30 min.), we terminated the algorithm and report the statistics corresponding to the iterate at the termination.

We solved the central problem (5.59) with SDPT3 and FISTA for benchmarking. We run DPGA algorithms on the decentralized problem both with constant step and adaptive step rules - see Section 5.2.5. In Table 5.1, 'xxx (CS)' and 'xxx (AS)' stand for "algorithm xxx is used with constant step and adaptive step rules, respectively." We used PG-EXTRA, ADMM, and SADMM with suggested parameters. For the results separated
by comma, the left and right ones are for the star tree and clique, respectively. Table 5.1 displays the means over 5 replications for each case. Table 5.1 shows that DPGA-I, DPGA-II and PG-EXTRA finish the jobs much faster than ADMM and SADMM – as expected due to not so simple $\text{prox}_{\Phi_i}$ and $\text{prox}_{f_i}$ operations required for ADMM and SADMM, respectively. PG-EXTRA runs slower than DPGA, mainly because it uses a stepsize that is the same for all the nodes. Moreover, adaptive step-size strategy worked very well in our tests, and it lead to speedup for both DPGA-I and DPGA-II by a factor of at least 2 when compared to constant step-size strategy. It is worth mentioning that run-times reported do not include the effect of communication. However, in real life, transmitting information also takes time. The number of communication rounds per iteration are 2 for DPGA-I, 1 for DPGA-II, 2 for PG-EXTRA, 2 for ADMM, and 4 for SADMM - see Table 3.2. Thus, we expect the result to be more in favor of DPGA-II as the communication time is also taken into consideration when implemented in real networks.

5.4 Conclusion

In this chapter, we studied distributed proximal gradient ADMM and its stochastic counterpart for distributed minimization of composite convex functions over connected networks. The convergence rates of these methods were analyzed. Comparing with existing works, the advantages of our methods are as follows: DPGA-I, DPGA-II, SDPGA-I and SDPGA-II are fully distributed, i.e., the agents are not required to know any global parameters depending on the entire network topology, e.g., the second smallest eigenvalue of the Laplacian; instead, we only assume that agents know who their neighbors are. Using only local communication, our node-based distributed algorithms require less communication burden and memory storage compared to edge-based distributed algorithms. The proposed algorithms consist of a single loop, i.e., there are no outer and inner iteration loops; therefore, they are easy and practical to be implemented over distributed networks. To sum up, there are many practical problems where one can compute the prox map for $\xi_i$ efficiently; however, computing the prox map for $\Phi_i = \xi_i + f_i$ is not easy. The methods proposed in this chapter can compute an $\epsilon$-optimal $\epsilon$-feasible solution in $O(\epsilon^{-1})$ iterations without assuming bounded $\nabla f_i$ for any $i \in \mathcal{N}$, where each iteration requires computing $\text{prox}_{\xi_i}$ and $\nabla f_i$ for $i \in \mathcal{N}$, and one or two communication rounds among the neighbors – hence, $O(\epsilon^{-1})$ communications per node in total.
Bibliography


120


EDUCATION

Pennsylvania State University, State College, Pennsylvania  
PhD Dual Degree in Industrial Engineering & Operations Research 06/2011 - 05/2016

Stevens Institute of Technology, Hoboken, New Jersey  
Master of Engineering in Nanotechnology 08/2008 - 06/2010

Beijing Institute of Technology, Beijing, China  
Bachelor of Science in Materials Science and Engineering 09/2004 - 06/2008

WORK EXPERIENCE

Data Scientist, Cognitive System  
Monsanto Company, St. Louis, MO 04/2016 - Present
✓ Worked with business owners to map business requirements into technical solutions
✓ Conceived and designed seamless intelligence & end-to-end automated solutions to support business units and initiatives by working closely with data architects and software engineers
✓ Developed and implemented optimization & machine learning algorithms in <IBM DOC> and <Python>, including applying algorithms from out-of-box tool and custom algorithm development
✓ Performed prod deployment & analyses, provided training to end users and saved 25% of operational cost

Data Scientist (Intern), Global Breeding Analytics  
Monsanto Company, St. Louis, MO 05/2015 - 08/2015
✓ Worked collaboratively with interdisciplinary scientists to optimize breeding selection process
✓ Analyzed large complex data, and developed, prototyped & validated an automated algorithm using machine learning, optimization and data mining techniques in <R>
✓ Built the web application in <Shiny>, and deployed the end tool to the stakeholders

Research Assistant  
Industrial Engineering Department, Penn State University 08/2013 - 03/2016

SKILLS

Programming: Python, R, Shiny, Matlab, IBM DOC, IBM Cplex, SQL, Gurobi, Mosek and GAMS
Software: MS Office Suite, Simio, Arena, @Risk, Stat Tools and Adobe Photoshop
Course: Machine Learning, Introduction to Data Science, Regression, Clustering, Recommender system
Operating Systems: Power user of Mac OS X and Windows 8/7/XP, familiar with Linux

ACTIVITIES

Speaker, 2014 IEEE Control & Decision Conference
Invited speaker, 2015 & 2014 INFORMS and MOPTA Conference
Volunteer, 2008 Beijing Olympic Games

Address: 16400 BALSAM RIDGE DR #B, WILDWOOD, MO 63011