

Dynamic Sharing Through the ADMM

†XuanYu Cao, and *K. J. Ray Liu, *Fellow, IEEE*

Email: †xyc@illinois.edu, *kjrlu@umd.edu

†Coordinated Science Lab, University of Illinois at Urbana-Champaign, Urbana, IL

*Department of Electrical and Computer Engineering, University of Maryland, College Park, MD

Abstract—In this paper, we study a dynamic version of the sharing problem, in which a dynamic system cost function composed of time-variant local costs of subsystems and a shared time-variant cost of the whole system is minimized. A dynamic alternating direction method of multipliers (ADMM) is proposed to track the varying optimal points of the dynamic optimization problem in an online manner. We analyze the convergence properties of the dynamic ADMM and show that, under several standard technical assumptions, the iterations of the dynamic ADMM converge linearly to some neighborhoods of the time-varying optimal points. The sizes of these neighborhoods depend on the drifts of the dynamic objective functions: the more drastically the dynamic objective function evolves across time, the larger the sizes of these neighborhoods. We also upper bound the limiting optimality gaps of the dynamic ADMM explicitly, and analyze its regret and constraint violation. Finally, two numerical examples are presented to corroborate the effectiveness of the proposed dynamic ADMM.

Index Terms—Dynamic optimization, the sharing problem, alternating direction method of multipliers

I. INTRODUCTION

Many resource allocation problems can be posed as an optimization problem which aims at minimizing a system cost consisting of local costs of subsystems and a shared cost of the whole system. This can be described as the following sharing problem [1]:

$$\text{Minimize } \sum_{i=1}^n f^{(i)}(\mathbf{x}^{(i)}) + g\left(\sum_{i=1}^n \mathbf{x}^{(i)}\right), \quad (1)$$

with variables $\mathbf{x}^{(i)} \in \mathbb{R}^p$, $i = 1, \dots, n$, where $f^{(i)} : \mathbb{R}^p \mapsto \mathbb{R}$ is the local cost function of subsystem i and $g : \mathbb{R}^p \mapsto \mathbb{R}$ is the global cost function of some commonly shared objective of all subsystems. The global cost function g takes the sum of all $\mathbf{x}^{(i)}$ as its input argument. One implicit assumption of the conventional sharing problem (1) is that both the local cost functions $f^{(i)}$ and the global cost function g are static, i.e., they do not vary with time. However, in practice, the cost or utility functions in many applications are intrinsically time-varying. For example, in power grids, the utility functions of the subsystems vary across time as the users' demands evolve, e.g., the demands climax during evening and decline between midnight and early morning. The generation cost of the power system also varies with time owing to the intermittent renewable energy sources and the fluctuation of the market prices of energy. Therefore, we are motivated to study the following dynamic version of the sharing problem in this paper:

$$\text{Minimize } \sum_{i=1}^n f_k^{(i)}(\mathbf{x}^{(i)}) + g_k\left(\sum_{i=1}^n \mathbf{x}^{(i)}\right), \quad (2)$$

where k is the time index. $f_k^{(i)} : \mathbb{R}^p \mapsto \mathbb{R}$ is the local cost function of subsystem i at time k and $g_k : \mathbb{R}^p \mapsto \mathbb{R}$ is the global cost function of the shared objective at time k . We assume that all the cost functions $f_k^{(i)}$, g_k are strongly convex and g_k has Lipschitz continuous gradient.

In the literature, dynamic optimization problems arise in various research fields and have been studied from different perspectives. In adaptive signal processing such as the recursive least squares

(RLS), the input/output data arrive sequentially, resulting in a time-varying objective function (the discounted total squared errors) to be minimized [2]. Another line of research for dynamic optimization is online convex optimization (OCO) [3]–[7]. In OCO, the time-varying cost functions are unknown a-priori and the goal is to design online algorithms with low (e.g., sublinear) regrets, i.e., the solution from the algorithms are not too worse than the optimal offline benchmarks. More broadly speaking, online learning (e.g., the weighted majority algorithm and the multiplicative weight update method) [8]–[11] and (stochastic) dynamic control/programming (e.g., Markov decision processes) [12], [13] also lie in the category of dynamic optimizations, though their problem formulations are very distinct from that of this paper.

To solve the dynamic sharing problem in an online manner, in this paper, we present a dynamic ADMM algorithm. As a primal-dual method, the ADMM is superior to its primal domain counterparts such as the gradient descent method in terms of convergence speed. In fact, recent research has shown that ADMM is among the fastest first-order methods [14], [15]. Due to its broad applicability, the ADMM has been exploited in various signal processing and control problems. A few recent studies have investigated the performance of ADMM in dynamic scenarios. In particular, an online ADMM algorithm is proposed in [16], while a distributed online ADMM algorithm is developed in [17] for decentralized optimization over networks. [16] and [17] are focused on regret analysis and the benchmark used to define the regret is the best *fixed* point in hindsight. This hinders their applications to problems in which the underlying systems are intrinsically time-varying and the best fixed point may not be very meaningful, e.g., tracking a moving object. In contrast, the tracking errors of the *dynamic* optimal points are adopted as the performance measure in this paper. For dynamic sharing problem, this is more meaningful than the best fixed point, since the underlying systems (e.g., power systems) may vary intrinsically (e.g., varying renewable generation and market prices of energy). Additionally, several stochastic ADMM algorithms [18]–[20] have been proposed to solve stochastic programs iteratively using sequential samples. Though the stochastic ADMM operates in an online manner as new samples arrive sequentially, the statistical characteristics of the stochastic program and the optimal solution do not change over time, which makes the problem setup very distinct from the dynamic sharing problem considered here. A more closely related work is [21], in which a dynamic ADMM algorithm is applied to the consensus optimization problems. However, the convergence analysis of the dynamic ADMM in [21] significantly relies on the special structure of the consensus optimization problems, in which all agents share the same decision variable. This leaves the performance of the dynamic ADMM in other optimization scenarios largely unknown. In fact, abundant existing works have dealt with various aspects of static ADMM for distributed consensus problems, e.g., distributed ADMM [22], linearized ADMM for composite consensus [23], impact of network topology [24], and weighted ADMM [25]–[27]. Nevertheless, none of them can be directly applied to the dynamic sharing problem in this paper.

Our goal in this work is to investigate the convergence behaviors of the dynamic ADMM for the dynamic sharing problem both theoretically and empirically. Specifically, A dynamic ADMM algorithm (Algorithm 1) is proposed for a more general dynamic optimization problem (Problem (6)), which encompasses the dynamic sharing problem as a special case. The dynamic ADMM can adapt to the time-varying cost functions and track the optimal points in an online manner. We analyze the convergence properties of the proposed dynamic ADMM algorithm. We show that, under standard technical assumptions, the gaps between the algorithm iterates and the time-varying optimal points converge linearly to some neighborhoods of zero. The sizes of the neighborhoods are related to the drifts of the dynamic optimization problem: the more drastically the dynamic problem evolves with time, the larger the sizes of the neighborhoods. We also upper bound the limiting optimality gaps of the dynamic ADMM explicitly. Additionally, regret and constraint violation bounds of the dynamic ADMM are established in terms of the cumulative drift of the dynamic problem.

The remaining part of this paper is organized as follows. In Section II, the dynamic ADMM algorithm is proposed. In Section III, theoretical analysis of the convergence properties of the dynamic ADMM is presented. Two numerical examples are shown in Section IV, following which we conclude this work in Section V.

II. ALGORITHM DEVELOPMENT

In this section, we first give an application of the dynamic sharing problem in power systems to justify its usefulness. Then, we develop a dynamic ADMM algorithm for a more general dynamic optimization problem, which encompasses the dynamic sharing problem as a special case.

A. Application of the Dynamic Sharing Problem

The dynamic sharing problem in (2) can be applied to many dynamic resource allocation problems. For example, consider a power grid which is divided into n power subsystems. If subsystem i receives $\mathbf{x}^{(i)}$ amount of power supplies at time k , then it gains a utility of $-f_k^{(i)}(\mathbf{x}^{(i)})$ by consuming the supplies. The utility function is time-variant because users often have different power demands at different time, e.g., 6-11pm may be the peak demand period while 2-6am may be a low demand period. On the other hand, the generation of the total power supplies of $\sum_{i=1}^n \mathbf{x}^{(i)}$ can incur a cost of $g_k(\sum_{i=1}^n \mathbf{x}^{(i)})$ for the power generator due to resource consumptions and pollution. The generation cost function g_k also varies across time owing to factors such as the intermittent renewable energy generation and the varying market prices of the traditional energy sources. Thus, the overall social cost minimization problem can be posed as a dynamic sharing problem as in (2).

Furthermore, one can consider the more general dynamic optimization problem of minimizing $\sum_{j=1}^m h_k^{(j)}(\mathbf{x}^{(S_j)})$, in which $\mathbf{x}^{(S_j)}$ is the concatenation of all $\mathbf{x}^{(i)}$ for $i \in S_j$ and $h_k^{(j)}$ is a time-varying function. The sets $\{S_j\}_{j=1, \dots, m}$ capture the general dependence structure between the terms in the objective function and the optimization variables, which encompasses the dynamic sharing problem as a special case. In principle, the analysis presented in this paper can be extended to this general problem, since it is also amenable to ADMM after some problem reformulation. Details are left as future work.

B. Development of the Dynamic ADMM

Define $\mathbf{x} = [\mathbf{x}^{(1)\top}, \dots, \mathbf{x}^{(n)\top}]^\top$, $\mathbf{A} = [\mathbf{I}_p, \dots, \mathbf{I}_p]$, and

$$f_k(\mathbf{x}) = \sum_{i=1}^n f_k^{(i)}(\mathbf{x}^{(i)}). \quad (3)$$

Then, the dynamic sharing problem (2) can be reformulated as:

$$\begin{aligned} \text{Minimize}_{\mathbf{x} \in \mathbb{R}^{np}, \mathbf{z} \in \mathbb{R}^p} \quad & f_k(\mathbf{x}) + g_k(\mathbf{z}) \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} - \mathbf{z} = \mathbf{0}. \end{aligned} \quad (4)$$

In the remaining part of this paper, we study the following more general dynamic optimization problem:

$$\begin{aligned} \text{Minimize}_{\mathbf{x} \in \mathbb{R}^N, \mathbf{z} \in \mathbb{R}^M} \quad & f_k(\mathbf{x}) + g_k(\mathbf{z}) \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c}, \end{aligned} \quad (6)$$

where $f_k : \mathbb{R}^N \mapsto \mathbb{R}$ and $g_k : \mathbb{R}^M \mapsto \mathbb{R}$ are two functions; $\mathbf{A} \in \mathbb{R}^{M \times N}$, $\mathbf{B} \in \mathbb{R}^{M \times M}$ are two matrices; $\mathbf{c} \in \mathbb{R}^M$ is a vector. The problem (4) is clearly a special case of the problem (6) by taking $N = np$, $M = p$, $\mathbf{B} = -\mathbf{I}$, $\mathbf{c} = \mathbf{0}$ and f_k decomposable as in (3). To apply the ADMM, we form the augmented Lagrangian of the problem (6):

$$\begin{aligned} \mathfrak{L}_{\rho, k}(\mathbf{x}, \mathbf{z}, \boldsymbol{\lambda}) = & f_k(\mathbf{x}) + g_k(\mathbf{z}) + \boldsymbol{\lambda}^\top (\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}) \\ & + \frac{\rho}{2} \|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|_2^2, \end{aligned} \quad (8)$$

where $\boldsymbol{\lambda} \in \mathbb{R}^M$ is the Lagrange multiplier and $\rho > 0$ is some positive constant. Applying the traditional static ADMM [1] to the dynamic augmented Lagrangian $\mathfrak{L}_{\rho, k}$, we propose a dynamic ADMM algorithm, as specified in Algorithm 1. The main difference between the dynamic ADMM in Algorithm 1 and the traditional static ADMM described in subsection II-B is that the functions f_k and g_k varies across iterations of the ADMM. The aim of this paper is to study the impact of these varying functions on the ADMM algorithm. Further, we note that one can conduct multiple ADMM iterations in each time k , i.e., multiple rounds of the updates (9)-(11) for each time k . This will potentially improve the performance of dynamic ADMM at the cost of higher computational complexity. In this paper, we focus on the case of single ADMM iteration per time slot and the analysis can be extended to the case of multiple iterations straightforwardly. In the following, we introduce two linear convergence concepts.

Definition 1. A sequence s_k is said to converge Q -linearly to s^* if there exists some constant $\theta \in (0, 1)$ such that $|s_{k+1} - s^*| \leq \theta |s_k - s^*|$ for k sufficiently large.

Definition 2. A sequence v_k is said to converge R -linearly to v^* if there exists a positive constant $\tau > 0$ and some sequence s_k Q -linearly converging to some number s^* such that $|v_k - v^*| \leq \tau |s_k - s^*|$ for k sufficiently large.

We remark that these linear convergence notions are widely adopted in the literature [25], [26], [28].

III. CONVERGENCE ANALYSIS

In this section, convergence analysis for the dynamic ADMM algorithm, i.e., Algorithm 1, is conducted. We first make several standard assumptions for algorithm analysis. Then, we show that the gaps between the algorithm iterates and the dynamic optimal points converge linearly (either Q -linearly or R -linearly) to some neighborhoods of zero (Theorem 1 and 2). The sizes of these neighborhoods depend on the *drift* (to be formally defined later) of the dynamic optimization problem (6). Further, we upper bound the limiting optimality gaps of the dynamic ADMM explicitly. Finally,

Algorithm 1 The dynamic ADMM for problem (6)

1: Initialize $\mathbf{x}_0 = \mathbf{0}, \mathbf{z}_0 = \boldsymbol{\lambda}_0 = \mathbf{0}, k = 0$

2: **Repeat:**

3: $k \leftarrow k + 1$

4: Update \mathbf{x} according to:

$$\mathbf{x}_k = \arg \min_{\mathbf{x}} f_k(\mathbf{x}) + \boldsymbol{\lambda}_{k-1}^\top \mathbf{A}\mathbf{x} + \frac{\rho}{2} \|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z}_{k-1} - \mathbf{c}\|_2^2. \quad (9)$$

5: Update \mathbf{z} according to:

$$\mathbf{z}_k = \arg \min_{\mathbf{z}} g_k(\mathbf{z}) + \boldsymbol{\lambda}_{k-1}^\top \mathbf{B}\mathbf{z} + \frac{\rho}{2} \|\mathbf{B}\mathbf{z} + \mathbf{A}\mathbf{x}_k - \mathbf{c}\|_2^2. \quad (10)$$

6: Update $\boldsymbol{\lambda}$ according to:

$$\boldsymbol{\lambda}_k = \boldsymbol{\lambda}_{k-1} + \rho(\mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{z}_k - \mathbf{c}). \quad (11)$$

regret and constraint violation bounds of the dynamic ADMM are established in terms of the cumulative drift of the dynamic problem.

A. Assumptions

Throughout the convergence analysis, we make the following standard assumptions on the functions f_k and g_k [28]–[30].

Assumption 1. For any k , g_k is strongly convex with constant $m > 0$ (m is independent of k), i.e., for any k and any $\mathbf{z}, \mathbf{z}' \in \mathbb{R}^M$:

$$(\nabla g_k(\mathbf{z}) - \nabla g_k(\mathbf{z}'))^\top (\mathbf{z} - \mathbf{z}') \geq m \|\mathbf{z} - \mathbf{z}'\|_2^2. \quad (12)$$

Assumption 2. For any k , f_k is strongly convex with constant $\tilde{m} > 0$ (\tilde{m} is independent of k), i.e., for any k and any $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^N$:

$$(\nabla f_k(\mathbf{x}) - \nabla f_k(\mathbf{x}'))^\top (\mathbf{x} - \mathbf{x}') \geq \tilde{m} \|\mathbf{x} - \mathbf{x}'\|_2^2. \quad (13)$$

Assumption 3. For any positive integer k , ∇g_k is Lipschitz continuous with constant $L > 0$ (L is independent of k), i.e., for any positive integer k and any $\mathbf{z}, \mathbf{z}' \in \mathbb{R}^M$:

$$\|\nabla g_k(\mathbf{z}) - \nabla g_k(\mathbf{z}')\|_2 \leq L \|\mathbf{z} - \mathbf{z}'\|_2. \quad (14)$$

We note that when f_k is decomposable as in (3) of the dynamic sharing problem, if for any $i = 1, \dots, n$ and positive integer k , $f_k^{(i)}$ is strongly convex with constant $\tilde{m}_i > 0$, then Assumption 2 holds with $\tilde{m} = \min_{i=1, \dots, n} \tilde{m}_i > 0$. We further assume that the matrix $\mathbf{B} \in \mathbb{R}^{M \times M}$ is nonsingular.

Assumption 4. \mathbf{B} is nonsingular.

B. Convergence Analysis

In this subsection, we study the convergence behavior of the proposed dynamic ADMM algorithm under the Assumptions 1-4. Before formal analysis, we note that if multiple ADMM iterations per time slot are allowed, existing results on static ADMM can be applied to show small optimality gaps of the dynamic ADMM in each time slot separately. One drawback of this approach is high computational burden, which renders it not suitable for many real-time applications with low computational capabilities, e.g., cheap sensors processing real-time data stream. In contrast, Algorithm 1 conducts only one single ADMM iteration per time slot and thus enjoys low computational overhead. Due to this single iteration, new analysis is needed to establish convergence in the dynamic setting in which the underlying optimization problem is varying.

Owing to the strong convexity assumption in Assumptions 1 and 2, there is a unique primal/dual optimal point pair $(\mathbf{x}_k^*, \mathbf{z}_k^*, \boldsymbol{\lambda}_k^*)$ for the dynamic optimization problem (6) at time k . Denote $\mathbf{u}_k = [\mathbf{z}_k^\top, \boldsymbol{\lambda}_k^\top]^\top$ and $\mathbf{u}_k^* = [\mathbf{z}_k^{*\top}, \boldsymbol{\lambda}_k^{*\top}]^\top$. Since \mathbf{B} is a square matrix, the eigenvalues of $\mathbf{B}\mathbf{B}^\top$ are the same as those of $\mathbf{B}^\top\mathbf{B}$. Denote the smallest eigenvalue of $\mathbf{B}\mathbf{B}^\top$, which is also the smallest eigenvalue of $\mathbf{B}^\top\mathbf{B}$, as α .

According to Assumption 4, \mathbf{B} is nonsingular, so $\mathbf{B}\mathbf{B}^\top$ and $\mathbf{B}^\top\mathbf{B}$ are positive definite and $\alpha > 0$. Define matrix $\mathbf{C} \in \mathbb{R}^{2M \times 2M}$ to be:

$$\mathbf{C} = \begin{bmatrix} \frac{\rho}{2} \mathbf{B}^\top \mathbf{B} & \\ & \frac{1}{2\rho} \mathbf{I}_M \end{bmatrix} \quad (15)$$

Since \mathbf{B} is nonsingular (Assumption 4), we know that \mathbf{C} is positive definite. Therefore, we can define a norm on \mathbb{R}^{2M} as $\|\mathbf{u}\|_{\mathbf{C}} = \sqrt{\mathbf{u}^\top \mathbf{C} \mathbf{u}}$. Define t to be any arbitrary number within the interval $(0, 1)$. A positive constant $\delta > 0$ is defined as:

$$\delta = \min \left\{ \frac{2mt}{\rho \|\mathbf{B}\|_2^2}, \frac{2\alpha\rho(1-t)}{L} \right\}, \quad (16)$$

where $\|\mathbf{B}\|_2$ is the spectral norm, i.e., the maximum singular value.

Proposition 1. For any positive integer k , we have:

$$\|\mathbf{u}_k - \mathbf{u}_k^*\|_{\mathbf{C}} \leq \frac{1}{\sqrt{1+\delta}} \|\mathbf{u}_{k-1} - \mathbf{u}_k^*\|_{\mathbf{C}}. \quad (17)$$

Proof. The updates of \mathbf{x} and \mathbf{z} can be rewritten as:

$$\nabla f_k(\mathbf{x}_k) + \mathbf{A}^\top \boldsymbol{\lambda}_{k-1} + \rho \mathbf{A}^\top (\mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{z}_{k-1} - \mathbf{c}) = \mathbf{0}, \quad (18)$$

$$\nabla g_k(\mathbf{z}_k) + \mathbf{B}^\top \boldsymbol{\lambda}_{k-1} + \rho \mathbf{B}^\top (\mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{z}_k - \mathbf{c}) = \mathbf{0}. \quad (19)$$

Combining (19) and (11) yields:

$$\nabla g_k(\mathbf{z}_k) + \mathbf{B}^\top \boldsymbol{\lambda}_k = \mathbf{0}. \quad (20)$$

Combining (18) and (11) gives:

$$\nabla f_k(\mathbf{x}_k) + \mathbf{A}^\top (\boldsymbol{\lambda}_k + \rho \mathbf{B}(\mathbf{z}_{k-1} - \mathbf{z}_k)) = \mathbf{0}. \quad (21)$$

According to Assumptions 1 and 2, the problem (6) is a convex optimization problem. Thus, Karush-Kuhn-Tucker (KKT) conditions are necessary and sufficient for optimality. Hence,

$$\nabla f_k(\mathbf{x}_k^*) + \mathbf{A}^\top \boldsymbol{\lambda}_k^* = \mathbf{0}, \quad (22)$$

$$\nabla g_k(\mathbf{z}_k^*) + \mathbf{B}^\top \boldsymbol{\lambda}_k^* = \mathbf{0}, \quad (23)$$

$$\mathbf{A}\mathbf{x}_k^* + \mathbf{B}\mathbf{z}_k^* = \mathbf{c}. \quad (24)$$

Because of the convexity of g_k (Assumption 1) and Lipschitz continuity of its gradient (Assumption 3), we have [31]:

$$\begin{aligned} & \|\nabla g_k(\mathbf{z}_k) - \nabla g_k(\mathbf{z}_k^*)\|_2^2 \\ & \leq L(\mathbf{z}_k - \mathbf{z}_k^*)^\top (\nabla g_k(\mathbf{z}_k) - \nabla g_k(\mathbf{z}_k^*)). \end{aligned} \quad (25)$$

Further using (23) and (20), we obtain:

$$(\mathbf{z}_k - \mathbf{z}_k^*)^\top \mathbf{B}^\top (\boldsymbol{\lambda}_k^* - \boldsymbol{\lambda}_k) \geq \frac{1}{L} \left\| \mathbf{B}^\top (\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*) \right\|_2^2. \quad (26)$$

According to the strong convexity of g_k (Assumption 1), we have:

$$\begin{aligned} m \|\mathbf{z}_k - \mathbf{z}_k^*\|_2^2 & \leq (\nabla g_k(\mathbf{z}_k) - \nabla g_k(\mathbf{z}_k^*))^\top (\mathbf{z}_k - \mathbf{z}_k^*) \\ & = \left(-\mathbf{B}^\top \boldsymbol{\lambda}_k + \mathbf{B}^\top \boldsymbol{\lambda}_k^* \right)^\top (\mathbf{z}_k - \mathbf{z}_k^*). \end{aligned} \quad (27)$$

Combining (26) and (27), we know that for any $t \in (0, 1)$:

$$\begin{aligned} & (\mathbf{z}_k - \mathbf{z}_k^*)^\top \mathbf{B}^\top (\boldsymbol{\lambda}_k^* - \boldsymbol{\lambda}_k) \\ & \geq tm \|\mathbf{z}_k - \mathbf{z}_k^*\|_2^2 + \frac{1-t}{L} \left\| \mathbf{B}^\top (\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*) \right\|_2^2. \end{aligned} \quad (28)$$

According to the convexity of f_k (Assumption 2), we have:

$$0 \leq (\mathbf{x}_k - \mathbf{x}_k^*)^\top (\nabla f_k(\mathbf{x}_k) - \nabla f_k(\mathbf{x}_k^*)). \quad (29)$$

Further making use of (21) and (22), we get:

$$0 \leq (\mathbf{x}_k - \mathbf{x}_k^*)^\top \mathbf{A}^\top (\boldsymbol{\lambda}_k^* - \boldsymbol{\lambda}_k + \rho \mathbf{B}(\mathbf{z}_k - \mathbf{z}_{k-1})). \quad (30)$$

Adding (28) and (30) leads to:

$$\begin{aligned} & (\mathbf{z}_k - \mathbf{z}_k^*)^\top \mathbf{B}^\top (\boldsymbol{\lambda}_k^* - \boldsymbol{\lambda}_k) \\ & + (\mathbf{x}_k - \mathbf{x}_k^*)^\top \mathbf{A}^\top (\boldsymbol{\lambda}_k^* - \boldsymbol{\lambda}_k + \rho \mathbf{B}(\mathbf{z}_k - \mathbf{z}_{k-1})) \\ & \geq tm \|\mathbf{z}_k - \mathbf{z}_k^*\|_2^2 + \frac{1-t}{L} \left\| \mathbf{B}^\top (\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*) \right\|_2^2. \end{aligned} \quad (31)$$

From (11) and (24), we get:

$$\mathbf{A}(\mathbf{x}_k - \mathbf{x}_k^*) + \mathbf{B}(\mathbf{z}_k - \mathbf{z}_k^*) = \frac{1}{\rho} (\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_{k-1}). \quad (32)$$

Making use of (32), we derive:

$$\begin{aligned} & (\mathbf{z}_k - \mathbf{z}_k^*)^\top \mathbf{B}^\top (\boldsymbol{\lambda}_k^* - \boldsymbol{\lambda}_k) \\ & + (\mathbf{x}_k - \mathbf{x}_k^*)^\top \mathbf{A}^\top (\boldsymbol{\lambda}_k^* - \boldsymbol{\lambda}_k + \rho \mathbf{B}(\mathbf{z}_k - \mathbf{z}_{k-1})) \end{aligned} \quad (33)$$

$$\begin{aligned} & = (\boldsymbol{\lambda}_k^* - \boldsymbol{\lambda}_k)^\top [\mathbf{B}(\mathbf{z}_k - \mathbf{z}_k^*) + \mathbf{A}(\mathbf{x}_k - \mathbf{x}_k^*)] \\ & + \rho (\mathbf{x}_k - \mathbf{x}_k^*)^\top \mathbf{A}^\top \mathbf{B}(\mathbf{z}_k - \mathbf{z}_{k-1}) \end{aligned} \quad (34)$$

$$\begin{aligned} & = \frac{1}{\rho} (\boldsymbol{\lambda}_k^* - \boldsymbol{\lambda}_k)^\top (\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_{k-1}) + \rho (\mathbf{A}(\mathbf{x}_k - \mathbf{x}_k^*))^\top \mathbf{B}(\mathbf{z}_k - \mathbf{z}_{k-1}) \\ & = \frac{1}{\rho} (\boldsymbol{\lambda}_{k-1} - \boldsymbol{\lambda}_k)^\top (\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*) \\ & + (\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_{k-1} - \rho \mathbf{B}(\mathbf{z}_k - \mathbf{z}_k^*))^\top \mathbf{B}(\mathbf{z}_k - \mathbf{z}_{k-1}). \end{aligned} \quad (35)$$

Rearranging (35), we obtain

$$\begin{aligned} & \frac{1}{\rho} (\boldsymbol{\lambda}_{k-1} - \boldsymbol{\lambda}_k)^\top (\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*) + \rho (\mathbf{z}_k - \mathbf{z}_k^*)^\top \mathbf{B}^\top \mathbf{B}(\mathbf{z}_{k-1} - \mathbf{z}_k) \\ & = (\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_{k-1})^\top \mathbf{B}(\mathbf{z}_{k-1} - \mathbf{z}_k) + (\mathbf{z}_k - \mathbf{z}_k^*)^\top \mathbf{B}^\top (\boldsymbol{\lambda}_k^* - \boldsymbol{\lambda}_k) \\ & + (\mathbf{x}_k - \mathbf{x}_k^*)^\top \mathbf{A}^\top (\boldsymbol{\lambda}_k^* - \boldsymbol{\lambda}_k + \rho \mathbf{B}(\mathbf{z}_k - \mathbf{z}_{k-1})). \end{aligned} \quad (36)$$

Noting that the last two product terms of the R.H.S. of (36) are the same as the L.H.S. of (31), we get

$$\begin{aligned} & \frac{1}{\rho} (\boldsymbol{\lambda}_{k-1} - \boldsymbol{\lambda}_k)^\top (\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*) + \rho (\mathbf{z}_k - \mathbf{z}_k^*)^\top \mathbf{B}^\top \mathbf{B}(\mathbf{z}_{k-1} - \mathbf{z}_k) \\ & \geq (\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_{k-1})^\top \mathbf{B}(\mathbf{z}_{k-1} - \mathbf{z}_k) + tm \|\mathbf{z}_k - \mathbf{z}_k^*\|_2^2 \\ & + \frac{1-t}{L} \left\| \mathbf{B}^\top (\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*) \right\|_2^2, \end{aligned} \quad (37)$$

which is equivalent to:

$$\begin{aligned} & \frac{1}{\rho} (\boldsymbol{\lambda}_{k-1} - \boldsymbol{\lambda}_k)^\top (\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_{k-1} + \boldsymbol{\lambda}_{k-1} - \boldsymbol{\lambda}_k^*) \\ & + \rho (\mathbf{z}_{k-1} - \mathbf{z}_k)^\top \mathbf{B}^\top \mathbf{B}(\mathbf{z}_k - \mathbf{z}_{k-1} + \mathbf{z}_{k-1} - \mathbf{z}_k^*) \end{aligned} \quad (38)$$

$$\begin{aligned} & \geq (\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_{k-1})^\top \mathbf{B}(\mathbf{z}_{k-1} - \mathbf{z}_k) + tm \|\mathbf{z}_k - \mathbf{z}_k^*\|_2^2 \\ & + \frac{1-t}{L} \left\| \mathbf{B}^\top (\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*) \right\|_2^2. \end{aligned} \quad (39)$$

This can be further rewritten as:

$$\begin{aligned} & \frac{1}{\rho} (\boldsymbol{\lambda}_{k-1} - \boldsymbol{\lambda}_k)^\top (\boldsymbol{\lambda}_{k-1} - \boldsymbol{\lambda}_k^*) + \rho (\mathbf{z}_{k-1} - \mathbf{z}_k)^\top \mathbf{B}^\top \mathbf{B}(\mathbf{z}_{k-1} - \mathbf{z}_k^*) \\ & \geq \frac{1}{\rho} \|\boldsymbol{\lambda}_{k-1} - \boldsymbol{\lambda}_k\|_2^2 + \rho \|\mathbf{B}\mathbf{z}_{k-1} - \mathbf{B}\mathbf{z}_k\|_2^2 \\ & + (\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_{k-1})^\top \mathbf{B}(\mathbf{z}_{k-1} - \mathbf{z}_k) \\ & + mt \|\mathbf{z}_k - \mathbf{z}_k^*\|_2^2 + \frac{1-t}{L} \left\| \mathbf{B}^\top (\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*) \right\|_2^2. \end{aligned} \quad (40)$$

We have:

$$\begin{aligned} & \frac{1}{\rho} (\boldsymbol{\lambda}_{k-1} - \boldsymbol{\lambda}_k)^\top (\boldsymbol{\lambda}_{k-1} - \boldsymbol{\lambda}_k^*) = -\frac{1}{2\rho} \|\boldsymbol{\lambda}_k^* - \boldsymbol{\lambda}_k\|_2^2 \\ & + \frac{1}{2\rho} \|\boldsymbol{\lambda}_{k-1} - \boldsymbol{\lambda}_k\|_2^2 + \frac{1}{2\rho} \|\boldsymbol{\lambda}_k^* - \boldsymbol{\lambda}_{k-1}\|_2^2, \end{aligned} \quad (41)$$

$$\begin{aligned} & \rho (\mathbf{z}_{k-1} - \mathbf{z}_k)^\top \mathbf{B}^\top \mathbf{B}(\mathbf{z}_{k-1} - \mathbf{z}_k^*) = -\frac{\rho}{2} \|\mathbf{B}\mathbf{z}_k^* - \mathbf{B}\mathbf{z}_k\|_2^2 \\ & + \frac{\rho}{2} \|\mathbf{B}\mathbf{z}_{k-1} - \mathbf{B}\mathbf{z}_k\|_2^2 + \frac{\rho}{2} \|\mathbf{B}\mathbf{z}_k^* - \mathbf{B}\mathbf{z}_{k-1}\|_2^2. \end{aligned} \quad (42)$$

Combining (41) and (42) and further utilizing (40) gives:

$$\begin{aligned} & \frac{1}{2\rho} \|\boldsymbol{\lambda}_{k-1} - \boldsymbol{\lambda}_k^*\|_2^2 + \frac{\rho}{2} \|\mathbf{B}\mathbf{z}_{k-1} - \mathbf{B}\mathbf{z}_k^*\|_2^2 - \frac{1}{2\rho} \|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\|_2^2 \\ & - \frac{\rho}{2} \|\mathbf{B}\mathbf{z}_k - \mathbf{B}\mathbf{z}_k^*\|_2^2 \end{aligned} \quad (43)$$

$$\begin{aligned} & = \frac{1}{\rho} (\boldsymbol{\lambda}_{k-1} - \boldsymbol{\lambda}_k)^\top (\boldsymbol{\lambda}_{k-1} - \boldsymbol{\lambda}_k^*) \\ & + \rho (\mathbf{z}_{k-1} - \mathbf{z}_k)^\top \mathbf{B}^\top \mathbf{B}(\mathbf{z}_{k-1} - \mathbf{z}_k^*) \\ & - \frac{1}{2\rho} \|\boldsymbol{\lambda}_{k-1} - \boldsymbol{\lambda}_k\|_2^2 - \frac{\rho}{2} \|\mathbf{B}\mathbf{z}_{k-1} - \mathbf{B}\mathbf{z}_k\|_2^2 \end{aligned} \quad (44)$$

$$\begin{aligned} & \geq \frac{1}{2\rho} \|\boldsymbol{\lambda}_{k-1} - \boldsymbol{\lambda}_k\|_2^2 + \frac{\rho}{2} \|\mathbf{B}\mathbf{z}_{k-1} - \mathbf{B}\mathbf{z}_k\|_2^2 \\ & + (\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_{k-1})^\top \mathbf{B}(\mathbf{z}_{k-1} - \mathbf{z}_k) \\ & + mt \|\mathbf{z}_k - \mathbf{z}_k^*\|_2^2 + \frac{1-t}{L} \left\| \mathbf{B}^\top (\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*) \right\|_2^2 \end{aligned} \quad (45)$$

$$\begin{aligned} & = \frac{1}{2\rho} \|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_{k-1} + \rho (\mathbf{B}\mathbf{z}_{k-1} - \mathbf{B}\mathbf{z}_k)\|_2^2 + mt \|\mathbf{z}_k - \mathbf{z}_k^*\|_2^2 \\ & + \frac{1-t}{L} \left\| \mathbf{B}^\top (\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*) \right\|_2^2 \end{aligned} \quad (46)$$

$$\geq mt \|\mathbf{z}_k - \mathbf{z}_k^*\|_2^2 + \frac{1-t}{L} \left\| \mathbf{B}^\top (\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*) \right\|_2^2 \quad (47)$$

$$\geq \frac{mt}{\|\mathbf{B}\|_2^2} \|\mathbf{B}\mathbf{z}_k - \mathbf{B}\mathbf{z}_k^*\|_2^2 + \frac{\alpha(1-t)}{L} \|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\|_2^2 \quad (48)$$

$$\geq \delta \left(\frac{1}{2\rho} \|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_k^*\|_2^2 + \frac{\rho}{2} \|\mathbf{B}\mathbf{z}_k - \mathbf{B}\mathbf{z}_k^*\|_2^2 \right), \quad (49)$$

where the last step is due to the definition of δ in (16). Noticing the definition of $\|\cdot\|_{\mathcal{C}}$, we get:

$$\|\mathbf{u}_{k-1} - \mathbf{u}_k^*\|_{\mathcal{C}}^2 \geq (1 + \delta) \|\mathbf{u}_k - \mathbf{u}_k^*\|_{\mathcal{C}}^2, \quad (50)$$

which is tantamount to (17). \square

Remark 1. Proposition 1 states that \mathbf{u}_k is closer to \mathbf{u}_k^* than \mathbf{u}_{k-1} with a shrinkage factor of δ . The bigger the δ , the stronger the shrinkage. Note that there is an arbitrary factor $t \in (0, 1)$ in the definition of δ in (16). By choosing $t = \frac{\alpha \rho^2 \|\mathbf{B}\|_2^2}{mL + \alpha \rho^2 \|\mathbf{B}\|_2^2}$, we get the maximum δ as $\delta_{\max} = \frac{2m\alpha\rho}{mL + \alpha\rho^2 \|\mathbf{B}\|_2^2}$. In the expression of δ_{\max} , only ρ is a tunable algorithm parameter while all other parameters are given by the optimization problem. Further, we note that δ_{\max} is maximized when $\rho = \sqrt{\frac{mL}{\alpha \|\mathbf{B}\|_2^2}}$, for which the theoretical performance bound of the dynamic ADMM is the best. This suggests that the optimal value of ρ should be neither too large nor too small. Nevertheless, in practice, this optimal ρ is hard to compute since we may not have access to m and L . Instead, we usually just use trial and error to find a reasonably ‘‘good’’ value for ρ .

Proposition 1 establishes a relation between $\|\mathbf{u}_k - \mathbf{u}_k^*\|_{\mathcal{C}}$ and $\|\mathbf{u}_{k-1} - \mathbf{u}_k^*\|_{\mathcal{C}}$. However, to describe the convergence behavior of the dynamic ADMM algorithm, what we really want is the relation between $\|\mathbf{u}_k - \mathbf{u}_k^*\|_{\mathcal{C}}$ and $\|\mathbf{u}_{k-1} - \mathbf{u}_{k-1}^*\|_{\mathcal{C}}$. This is accomplished by the following theorem.

Theorem 1. Define the drift d_k of the dynamic problem (6) to be:

$$\begin{aligned} d_k & = \sqrt{\frac{\rho}{2}} \|\mathbf{B}\|_2 \|\mathbf{z}_{k-1}^* - \mathbf{z}_k^*\|_2 \\ & + \frac{1}{\sqrt{2\rho\alpha}} \|\nabla g_{k-1}(\mathbf{z}_{k-1}^*) - \nabla g_k(\mathbf{z}_k^*)\|_2. \end{aligned} \quad (51)$$

Then, for any integer $k \geq 2$, we have:

$$\|\mathbf{u}_k - \mathbf{u}_k^*\|_{\mathcal{C}} \leq \frac{1}{\sqrt{1 + \delta}} (\|\mathbf{u}_{k-1} - \mathbf{u}_{k-1}^*\|_{\mathcal{C}} + d_k). \quad (52)$$

Proof. See Section S.1 of the supplementary material. \square

Remark 2. Theorem 1 means that the optimality gap $\|\mathbf{u}_k - \mathbf{u}_k^*\|_{\mathbf{C}}$ converges Q-linearly (with contraction factor $\sqrt{1+\delta}$) to some neighborhood of zero. The size of the neighborhood is characterized by d_k , the drift of the dynamic problem (6), which is determined by the problem formulation instead of the algorithm. The more drastically the dynamic problem (6) varies across time, the bigger the drift d_k , and the larger the size of that neighborhood. When the dynamic problem (6) degenerates to its static counterpart, i.e., f_k and g_k does not vary with k , d_k becomes zero. In such a case, Theorem 1 degenerates to the linear convergence result of static ADMM in [30]. Further, we note that the target optima \mathbf{u}_k^* is a time-varying sequence instead of a fixed point. Thus, Theorem 1 indicates that the algorithm iterate \mathbf{u}_k tracks the dynamic optima \mathbf{u}_k^* instead of converging to some fixed point. This is due to the temporal variations of the underlying optimization problems and differs from existing works on static optimization, in which the algorithm iterates usually converge to a fixed (optimal) point.

Remark 3. If higher computational overhead is affordable, one can conduct multiple ADMM iterations in each time slot to improve the performance of dynamic ADMM. As far as performance analysis is concerned, this can be regarded as repeating the same optimization problem for multiple time slots and all the analyses for the single iteration case still hold under minor modifications. In particular, when the same problem is repeated multiple times, the corresponding drift d_k is zero and the gap $\|\mathbf{u}_k - \mathbf{u}_k^*\|_{\mathbf{C}}$ decreases geometrically for multiple rounds according to (52). This partially explains why multiple ADMM iterations per time slot can improve the performance.

Remark 4. We note that in the proof of Theorem 1, the sole role of the drift d_k is to upper bound $\|\mathbf{u}_{k-1}^* - \mathbf{u}_k^*\|_{\mathbf{C}}$. Thus, one may define a new notion of drift as $\tilde{d}_k = \|\mathbf{u}_{k-1}^* - \mathbf{u}_k^*\|_{\mathbf{C}}$ and (52) (as well as other later theorems) still holds for this new drift. In this paper, we define drift d_k according to (51) because it only involves quantities in the primal optimization problem (6) such as the primal optimal point \mathbf{z}_k^* and the primal objective function g_k . It is more natural to characterize the drift of the primal problem (6) using only these primal quantities instead of the dual optimal point λ_k^* used in $\|\mathbf{u}_{k-1}^* - \mathbf{u}_k^*\|_{\mathbf{C}}$. The reason is that the temporal variations of primal quantities, e.g., $\|\mathbf{z}_{k-1}^* - \mathbf{z}_k^*\|_2$, are usually easier to estimate than those of the dual variables. For example, if \mathbf{z}_k^* represents the location of a moving target, we may estimate the variations of \mathbf{z}_k^* by the knowledge of the target's speed range. By defining the drift in terms of the primal variables, we may obtain more accurate estimates of the drifts and thus know more about the convergence performance of the dynamic ADMM. Further, we note that the drift d_k depends on the ADMM parameter ρ , which may seem unnatural at the first glance. Nevertheless, this is indeed reasonable (and inevitable) since the optimality gap $\|\mathbf{u}_k - \mathbf{u}_k^*\|_{\mathbf{C}}$ that we want to bound in (52) also depends on ρ implicitly through the norm $\|\cdot\|_{\mathbf{C}}$ (recall that the definition of \mathbf{C} in (15) depends on ρ) and \mathbf{u}_k , which is generated by dynamic ADMM with parameter ρ .

Remark 5. The notion of drift and the linear convergence result in Theorem 1 are analogous to that of [21]. Nevertheless, [21] is focused on decentralized consensus optimization problems over networks and the analysis heavily relies on the special structures of consensus optimization, in which networked agents share one common decision variable. For instance, in [21], owing to the special structures of consensus optimization, some primal/dual variables can be eliminated under appropriate initializations, which simplifies the performance analysis. Such structures no longer hold for the dynamic sharing problem in this paper. In fact, the proofs of linear convergence for the dynamic sharing problem in this paper are very different from that of [21].

The convergence property of \mathbf{u}_k has been established in Theorem

1. A more meaningful result will be about the convergence properties of $\mathbf{x}_k, \mathbf{z}_k, \lambda_k$. To this end, we want to link the quantities $\|\mathbf{x}_k - \mathbf{x}_k^*\|_2, \|\mathbf{z}_k - \mathbf{z}_k^*\|_2, \|\lambda_k - \lambda_k^*\|_2$ with $\|\mathbf{u}_k - \mathbf{u}_k^*\|_{\mathbf{C}}$. This is accomplished by the following theorem.

Theorem 2. For any integer $k \geq 2$, we have:

$$\begin{aligned} & \|\mathbf{x}_k - \mathbf{x}_k^*\|_2 \\ & \leq \frac{1}{\tilde{m}} \|\mathbf{A}\|_2 \left[\left(\sqrt{2\rho} + \|\mathbf{B}\|_2 \sqrt{\frac{2\rho}{\alpha}} \right) \|\mathbf{u}_k - \mathbf{u}_k^*\|_{\mathbf{C}} \right. \\ & \quad \left. + \|\mathbf{B}\|_2 \sqrt{\frac{2\rho}{\alpha}} \|\mathbf{u}_{k-1} - \mathbf{u}_{k-1}^*\|_{\mathbf{C}} + \sqrt{2\rho} d_k \right], \end{aligned} \quad (53)$$

where $\|\mathbf{A}\|_2$ is the spectral norm, i.e., the largest singular value, of \mathbf{A} . Furthermore, for any positive integer k , we have:

$$\|\mathbf{z}_k - \mathbf{z}_k^*\|_2 \leq \sqrt{\frac{2}{\alpha\rho}} \|\mathbf{u}_k - \mathbf{u}_k^*\|_{\mathbf{C}}, \quad (54)$$

$$\|\lambda_k - \lambda_k^*\|_2 \leq \sqrt{2\rho} \|\mathbf{u}_k - \mathbf{u}_k^*\|_{\mathbf{C}}. \quad (55)$$

Proof. See Section S.2 of the supplementary material. \square

Remark 6. Since $\|\mathbf{u}_k - \mathbf{u}_k^*\|_{\mathbf{C}}$ converges Q-linearly to some neighborhood of zero, Theorem 2 indicates that $\|\mathbf{x}_k - \mathbf{x}_k^*\|_2, \|\mathbf{z}_k - \mathbf{z}_k^*\|_2$, and $\|\lambda_k - \lambda_k^*\|_2$ converge R-linearly to neighborhoods of zero. When the dynamic optimization problem (6) degenerates to its static version, i.e., f_k and g_k does not vary with k , Theorem 2 also degenerates to its static counterpart in [28], [30].

Theorems 1 and 2 characterize the transient behaviors of the dynamic ADMM for each time k in terms of the drift d_k . Based on these results, we can bound the limiting optimality gaps of the algorithm iterates as the time k goes to infinity in the following.

Theorem 3. Suppose the drift defined in (51) satisfies $d_k \leq d, \forall k$, for some $d \in \mathbb{R}$. Then, we have:

$$\limsup_{k \rightarrow \infty} \|\mathbf{u}_k - \mathbf{u}_k^*\|_{\mathbf{C}} \leq \frac{d}{\sqrt{1+\delta}-1}, \quad (56)$$

$$\limsup_{k \rightarrow \infty} \|\mathbf{x}_k - \mathbf{x}_k^*\|_2 \leq \frac{\|\mathbf{A}\|_2}{\tilde{m}} \left[\frac{\sqrt{2\rho} + \|\mathbf{B}\|_2 \sqrt{\frac{8\rho}{\alpha}}}{\sqrt{1+\delta}-1} + \sqrt{2\rho} \right] d,$$

$$\limsup_{k \rightarrow \infty} \|\mathbf{z}_k - \mathbf{z}_k^*\|_2 \leq \sqrt{\frac{2}{\alpha\rho}} \frac{d}{\sqrt{1+\delta}-1}, \quad (57)$$

$$\limsup_{k \rightarrow \infty} \|\lambda_k - \lambda_k^*\|_2 \leq \sqrt{2\rho} \frac{d}{\sqrt{1+\delta}-1}. \quad (58)$$

Proof. See Section S.3 of the supplementary material. \square

The dynamic problem (6) falls into the general category of constrained online optimization problems [3], for which regret and constraint violation are two prevalent performance criteria. Specifically, for problem (6), the regret and the constraint violation of an algorithm at time K are defined as:

$$\text{Reg}(K) = \sum_{k=1}^K [f_k(\mathbf{x}_k) + g_k(\mathbf{z}_k) - f_k(\mathbf{x}_k^*) - g_k(\mathbf{z}_k^*)], \quad (59)$$

$$\text{Vio}(K) = \sum_{k=1}^K \|\mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{z}_k - \mathbf{c}\|_2, \quad (60)$$

where the time-varying primal optimal point $(\mathbf{x}_k^*, \mathbf{z}_k^*)$ serves as a dynamic benchmark sequence. Generally, sublinear regret and sublinear constraint violation, i.e., $\text{Reg}(K) \leq o(K)$ and $\text{Vio}(K) \leq o(K)$, are regarded as ‘‘good’’ performance. In such a case, the time-average performance of the algorithm iterates is no worse than that of the benchmark optima asymptotically since $\frac{\text{Reg}(K)}{K} \leq o(1) \rightarrow 0$ and

$\frac{\text{Vio}(K)}{K} \leq o(1) \rightarrow 0$ as $K \rightarrow \infty$. If g_k and f_k are β - and $\tilde{\beta}$ -Lipschitz continuous, respectively, it can be shown that the regret and the constraint violation can be upper bounded in terms of the tracking errors $\|\mathbf{x}_k - \mathbf{x}_k^*\|_2$ and $\|\mathbf{z}_k - \mathbf{z}_k^*\|_2$ as follows:

$$\text{Reg}(K) \leq \sum_{k=1}^K \left(\tilde{\beta} \|\mathbf{x}_k - \mathbf{x}_k^*\|_2 + \beta \|\mathbf{z}_k - \mathbf{z}_k^*\|_2 \right), \quad (61)$$

$$\text{Vio}(K) \leq \sum_{k=1}^K (\|\mathbf{A}\|_2 \|\mathbf{x}_k - \mathbf{x}_k^*\|_2 + \|\mathbf{B}\|_2 \|\mathbf{z}_k - \mathbf{z}_k^*\|_2). \quad (62)$$

These bounds manifest the connections between the regret/constraint violation and the tracking errors used in Theorems 1-3. In the following, by exploiting these connections, we establish upper bounds for the regret and the constraint violation in terms of the drift.

Theorem 4. Suppose that g_k and f_k are Lipschitz continuous with positive constants β and $\tilde{\beta}$, respectively, i.e., $\forall k, \forall \mathbf{z}, \mathbf{z}' \in \mathbb{R}^M, \forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^N$, we have

$$|g_k(\mathbf{z}) - g_k(\mathbf{z}')| \leq \beta \|\mathbf{z} - \mathbf{z}'\|_2, \quad (63)$$

$$|f_k(\mathbf{x}) - f_k(\mathbf{x}')| \leq \tilde{\beta} \|\mathbf{x} - \mathbf{x}'\|_2. \quad (64)$$

Define the **cumulative drift** of problem (6) at time K as $D_K = \sum_{k=2}^K d_k$. Then, the regret and the constraint violation of the dynamic ADMM can be bounded as:

$$\text{Reg}(K) \leq \mathcal{O}(D_K + 1), \quad \text{Vio}(K) \leq \mathcal{O}(D_K + 1). \quad (65)$$

Proof. See Section S.4 of the supplementary material. \square

From Theorem 4, we know that the dynamic ADMM achieves sublinear regret and sublinear constraint violation as long as the cumulative drift D_K is sublinear. Conversely, if the cumulative drift D_K is not sublinear, then the per time drift d_k is in constant order at least, i.e., the underlying optimization problem (6) varies in constant speed at least. In such a case, it is hard for the algorithm iterates to track the dynamic optimal points so that the regret and the constraint violation may not be sublinear. Additionally, we note that the benchmark $(\mathbf{x}_k^*, \mathbf{z}_k^*)$ that we use in (59) is a dynamic sequence instead of a fixed point. In the literature, regret analysis with respect to dynamic benchmark has been carried out for other algorithms of constrained online optimization, e.g., the saddle point method in [32]. The corresponding regret and constraint violation bounds often depend on various forms of the drifts of the underlying dynamic problems. Here, Theorem 4 gives such bounds for the proposed dynamic ADMM algorithm.

IV. NUMERICAL EXAMPLES

In this section, two numerical examples are presented to validate the efficacy of the proposed dynamic ADMM algorithm. The first example is a dynamic sharing problem and the second one is the dynamic least absolute shrinkage and selection operator (LASSO). We note that the dynamic LASSO is not a dynamic sharing problem. Recall that the dynamic ADMM (Algorithm 1) is designed for the general dynamic optimization problem (6), which includes the dynamic sharing problem as a special case. Here, we study dynamic LASSO numerically to confirm that applications of the proposed dynamic ADMM are indeed not limited to dynamic sharing problem.

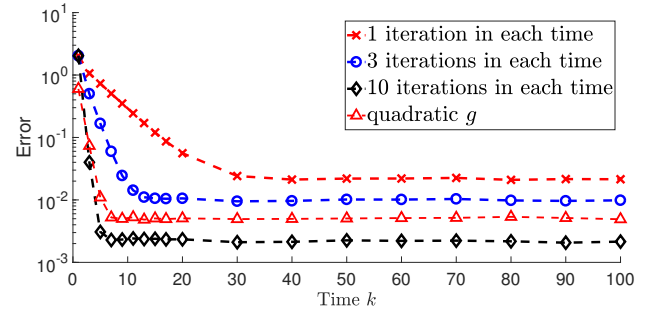


Fig. 1: The convergence curves of $\|\mathbf{x}_k - \mathbf{x}_k^*\|_2$ for different numbers of ADMM iterations per time slot.

A. The Dynamic Sharing Problem

1) *Problem Formulation:* We first consider the following dynamic sharing problem:

$$\text{Minimize}_{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)} \in \mathbb{R}^p} \sum_{i=1}^n (\mathbf{x}^{(i)} - \boldsymbol{\theta}_k^{(i)})^\top \boldsymbol{\Phi}_k^{(i)} (\mathbf{x}^{(i)} - \boldsymbol{\theta}_k^{(i)}) + \gamma \left\| \sum_{i=1}^n \mathbf{x}^{(i)} \right\|_1, \quad (66)$$

where $\boldsymbol{\theta}_k^{(i)} \in \mathbb{R}^p$, $\boldsymbol{\Phi}_k^{(i)} \in \mathbb{R}^{p \times p}$ positive definite, $\gamma > 0$ are given problem data. The problem (66) is clearly in the form of (2) with:

$$f_k^{(i)}(\mathbf{x}^{(i)}) = (\mathbf{x}^{(i)} - \boldsymbol{\theta}_k^{(i)})^\top \boldsymbol{\Phi}_k^{(i)} (\mathbf{x}^{(i)} - \boldsymbol{\theta}_k^{(i)}), \quad (67)$$

$$g_k(\mathbf{z}) = \gamma \|\mathbf{z}\|_1. \quad (68)$$

Define $\mathbf{x} = [\mathbf{x}^{(1)\top}, \dots, \mathbf{x}^{(n)\top}]^\top$, $\boldsymbol{\theta}_k = [\boldsymbol{\theta}_k^{(1)\top}, \dots, \boldsymbol{\theta}_k^{(n)\top}]^\top$ and $\boldsymbol{\Phi}_k = \text{diag} \{ \boldsymbol{\Phi}_k^{(1)}, \dots, \boldsymbol{\Phi}_k^{(n)} \}$. Thus, in terms of problem (4), we have $f_k(\mathbf{x}) = (\mathbf{x} - \boldsymbol{\theta}_k)^\top \boldsymbol{\Phi}_k (\mathbf{x} - \boldsymbol{\theta}_k)$. Applying the dynamic ADMM algorithm, i.e., Algorithm 1, to this dynamic sharing problem, we can obtain closed-form updates for (9) and (10) in each iteration by invoking the soft-threshold function since g_k is ℓ_1 norm.

2) *Generation of $\boldsymbol{\Phi}_k^{(i)}$ and $\boldsymbol{\theta}_k^{(i)}$:* We generate the problem data $\boldsymbol{\Phi}_k^{(i)}$ and $\boldsymbol{\theta}_k^{(i)}$ recursively as follows. Given $\boldsymbol{\Phi}_{k-1}^{(i)}$ ($k \geq 1$), we first generate $\tilde{\boldsymbol{\Phi}}_k^{(i)}$ according to $\tilde{\boldsymbol{\Phi}}_k^{(i)} = \boldsymbol{\Phi}_{k-1}^{(i)} + \eta_k^{(i)} \mathbf{E}_k^{(i)}$, where $\eta_k^{(i)}$ is some small positive number and $\mathbf{E}_k^{(i)}$ is a random symmetric matrix with entries uniformly distributed on $[-1, 1]$. Then, we construct the matrix $\boldsymbol{\Phi}_k^{(i)}$ as:

$$\boldsymbol{\Phi}_k^{(i)} = \begin{cases} \tilde{\boldsymbol{\Phi}}_k^{(i)}, & \text{if } \lambda_{\min}(\tilde{\boldsymbol{\Phi}}_k^{(i)}) \geq \epsilon, \text{ i.e., } \tilde{\boldsymbol{\Phi}}_k^{(i)} \succeq \epsilon \mathbf{I}, \\ \tilde{\boldsymbol{\Phi}}_k^{(i)} + [\epsilon - \lambda_{\min}(\tilde{\boldsymbol{\Phi}}_k^{(i)})] \mathbf{I}, & \text{otherwise,} \end{cases} \quad (69)$$

where $\lambda_{\min}(\cdot)$ denotes the smallest eigenvalue and $\epsilon > 0$ is some positive constant. Through this construction, we ensure that $\boldsymbol{\Phi}_k^{(i)} \succeq \epsilon \mathbf{I}$, $k = 1, 2, \dots$. In addition, $\boldsymbol{\Phi}_0$ is a random symmetric matrix whose entries are uniformly distributed on $[-1, 1]$. Given $\boldsymbol{\theta}_{k-1}^{(i)}$ ($k \geq 1$), we generate $\boldsymbol{\theta}_k^{(i)}$ according to $\boldsymbol{\theta}_k^{(i)} = \boldsymbol{\theta}_{k-1}^{(i)} + \eta_k^{(i)} \mathbf{h}_k^{(i)}$, where $\mathbf{h}_k^{(i)}$ is a random p -dimensional vector whose entries are uniformly distributed on $[-1, 1]$. $\boldsymbol{\theta}_0^{(i)}$ is also a random p -dimensional vector with entries uniformly distributed on $[-1, 1]$.

3) *Simulation Results:* In the first simulation, we set the parameters as $\eta = 0.2, \epsilon = 1, \gamma = 1, \rho = 1, p = 5, n = 20$. We use the CVX package [33] to compute the optimal point \mathbf{x}_k^* of the dynamic sharing problem (66) at time k . In each time k , we use single or multiple iterations of the ADMM updates, i.e., single or multiple rounds of the updates (9)-(11) for each time k (the original Algorithm 1 uses one single ADMM iteration per time slot). The convergence curves of $\|\mathbf{x}_k - \mathbf{x}_k^*\|_2$ (\mathbf{x}_k is the online solution given by the dynamic

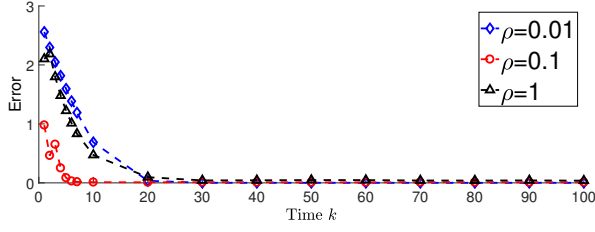


Fig. 2: The impact of the algorithm parameter ρ on the convergence behaviors ($\|\mathbf{x}_k - \mathbf{x}_k^*\|_2$) of the dynamic ADMM.

ADMM algorithm) for different numbers of ADMM iterations per time slot are shown in Fig. 1. The results are the average of 100 independent trials. We observe that $\|\mathbf{x}_k - \mathbf{x}_k^*\|_2$ converges to some neighborhoods of zero after about 7 to 30 iterations, depending on the number of ADMM iterations per time slot. This corroborates the efficacy of the proposed dynamic ADMM algorithm. Unsurprisingly, we observe that the performance of the dynamic ADMM algorithm can be enhanced by using more ADMM iterations per time slot at the expense of higher computational overhead. We note that the choice of $g_k(\mathbf{z}) = \gamma\|\mathbf{z}\|_1$ does not satisfy the strong convexity and Lipschitz continuous gradient assumptions used in the theoretical analysis. In light of this, we further run the dynamic ADMM for the quadratic function $g_k(\mathbf{z}) = \gamma\|\mathbf{z}\|_2^2$, which is strongly convex and has Lipschitz continuous gradient (all Assumptions 1-4 are satisfied). The convergence curve for this quadratic function g is also shown in Fig. 1, and one ADMM iteration is conducted in each time slot. Comparing this curve with the red cross curve, we observe that the performance of dynamic ADMM is better for the quadratic function g than for the ℓ_1 -norm function g . This suggests that the assumptions made for theoretical analysis can also be important for empirical performance.

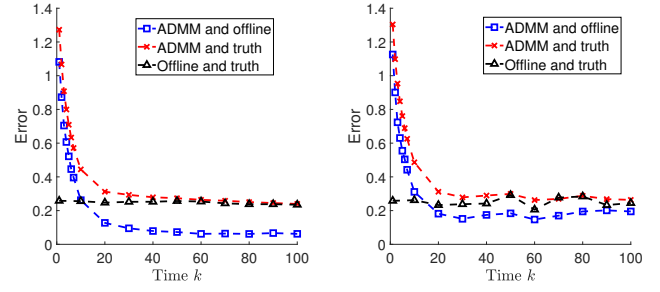
In the second simulation, we investigate the impact of the algorithm parameter ρ on the convergence performance of the dynamic ADMM. We consider three different values for ρ : 0.01, 0.1, 1. The corresponding convergence curves ($\|\mathbf{x}_k - \mathbf{x}_k^*\|_2$) are shown in Fig. 2. We find that $\rho = 0.1$ yields the best convergence performance among the three circumstances. This indicates that the importance of an appropriate value of ρ , which should be neither too large nor too small. We note that similar observations have been made in the traditional static ADMM [1].

B. Dynamic LASSO

1) *Problem Formulation:* Least absolute shrinkage and selection operator (LASSO) is an important and renowned problem in statistics and signal processing. It embodies sparsity-aware linear regression. Here, we consider a dynamic version of the LASSO since the problem data often vary with time in many real-time applications as new measurements arrive sequentially:

$$\underset{\mathbf{x} \in \mathbb{R}^p}{\text{Minimize}} \quad \frac{1}{2} \|\mathbf{F}_k \mathbf{x} - \mathbf{h}_k\|_2^2 + \gamma \|\mathbf{x}\|_1, \quad (70)$$

where $\mathbf{F}_k \in \mathbb{R}^{m \times p}$, $\mathbf{h}_k \in \mathbb{R}^m$ are time-variant problem data and $\gamma > 0$ is some positive constant controlling the sparsity of the solution. The problem (70) is clearly in the form of (6) with $f_k(\mathbf{x}) = \frac{1}{2} \|\mathbf{F}_k \mathbf{x} - \mathbf{h}_k\|_2^2$, $g_k(\mathbf{z}) = \gamma \|\mathbf{z}\|_1$, $\mathbf{A} = \mathbf{I}$, $\mathbf{B} = -\mathbf{I}$, $\mathbf{c} = \mathbf{0}$. Thus, we can apply Algorithm 1 to the problem (70), where both (9) and (10) admit closed-form solutions. Note that the problem (70) does not fall into the category of dynamic sharing problem (2) as $f_k(\mathbf{x})$ cannot be decomposed across several parts of \mathbf{x} . Our goal in this numerical example is to verify that the proposed dynamic ADMM



(a) Slowly varying case ($\eta = 0.01$). (b) Fast varying case ($\eta = 0.1$).

Fig. 3: Results for the gaps $\|\mathbf{x}_k - \mathbf{x}_k^*\|_2$, $\|\mathbf{x}_k - \tilde{\mathbf{x}}_k\|_2$, $\|\mathbf{x}_k^* - \tilde{\mathbf{x}}_k\|_2$.

works well for the general dynamic optimization problem (6), not just the dynamic sharing problem.

2) *Generation of \mathbf{F}_k and \mathbf{h}_k :* The problem data \mathbf{F}_k and \mathbf{h}_k are generated as follows. Given \mathbf{F}_{k-1} ($k \geq 1$), we generate \mathbf{F}_k according to $\mathbf{F}_k = \mathbf{F}_{k-1} + \eta_k \mathbf{W}_k$, where η_k is some small positive constant and $\mathbf{W}_k \in \mathbb{R}^{m \times p}$ is a random matrix with entries uniformly distributed on $[-1, 1]$. \mathbf{F}_0 is also a random matrix with entries uniformly distributed on $[-1, 1]$. To generate the sequence \mathbf{h}_k , we construct an auxiliary ground-truth sequence $\tilde{\mathbf{x}}_k$ as follows. We randomly select q different numbers $\{j_1, \dots, j_q\}$ from the set $\{1, \dots, p\}$, where $q \ll p$. Given $\tilde{\mathbf{x}}_{k-1}$ ($k \geq 1$), we generate $\tilde{\mathbf{x}}_k$ based on: $\tilde{\mathbf{x}}_k = \tilde{\mathbf{x}}_{k-1} + \eta_k \mathbf{u}_k$, where $\mathbf{u}_k \in \mathbb{R}^p$ is a random vector with j_l -th entry uniformly distributed on $[-1, 1]$, $l = 1, \dots, q$ and other entries equal to zero. $\tilde{\mathbf{x}}_0$ is a random vector whose j_l -th entry is uniformly distributed on $[0, 1]$, $l = 1, \dots, q$ and other entries are zero. This enforces sparsity of the ground-truth $\tilde{\mathbf{x}}_k$ to be estimated, which is the underlying hypothesis of the LASSO. With $\tilde{\mathbf{x}}_k$ and \mathbf{F}_k in hands, we generate \mathbf{h}_k according to $\mathbf{h}_k = \mathbf{F}_k \tilde{\mathbf{x}}_k + \mathbf{v}_k$, where $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$ is a m -dimensional Gaussian random vector.

3) *Simulation Results:* In the simulations, we set the parameters as: $m = 10, p = 30, q = 2, \rho = 1, \gamma = 0.2, \sigma = 0.1$. All results except Fig. 4 are the average of 100 independent trials. We consider two values, 0.01 and 0.1, for η , the parameter controlling the variation of the problem data across time. We call $\eta = 0.01$ and $\eta = 0.1$ the slowly time-variant case and the fast time-variant case, respectively. Denote the online estimate generated by applying the dynamic ADMM to the dynamic LASSO (70), the estimate given by the offline optimizer through the CVX package (i.e., the optimal point of (70)) and the ground-truth as \mathbf{x}_k , \mathbf{x}_k^* and $\tilde{\mathbf{x}}_k$, respectively. The gaps between these three quantities, i.e., $\|\mathbf{x}_k - \mathbf{x}_k^*\|_2$, $\|\mathbf{x}_k - \tilde{\mathbf{x}}_k\|_2$ and $\|\mathbf{x}_k^* - \tilde{\mathbf{x}}_k\|_2$, in the slowly time-variant case and the fast time-variant case are reported in Fig. 3-(a) and Fig. 3-(b), respectively. A few remarks are in order. First, the solution of the optimizer \mathbf{x}_k^* should be regarded as the benchmark for the dynamic ADMM as the former is the optimal point of (70), or in other words, the best that the dynamic LASSO can achieve. For both slowly and fast time-variant cases, the gaps between the dynamic ADMM and the offline optimizer, i.e., the blue line with square marker, converge to some small values after about 40 iterations. This indicates that the dynamic ADMM can track the optimal point of (70) well. Second, the gaps between the dynamic ADMM and the truth (red line with cross markers) as well as the gaps between the offline optimizer and the truth (black line with triangle markers) are similar after some 50 iterations in both slowly and fast time-variant cases. This suggests that in terms of tracking the ground-truth, the dynamic ADMM and the offline optimizer have similar performances while the former has much less computational complexity than the latter. Third, unsurprisingly, comparing 3-(a) with

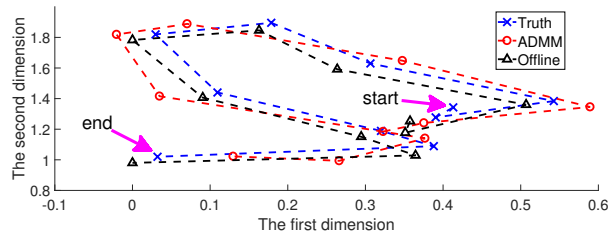


Fig. 4: The trajectories of the two nonzero dimensions in one trial

3-(b), we observe that the tracking performances of both the dynamic ADMM and the offline optimizer are related to the value of η : the larger the η , the more drastically the change of the problem data across time, the poorer the tracking performance.

Lastly, a more palpable result of the tracking performance is shown in Fig. 4, in which the trajectories of the two nonzero dimensions (i.e., i_1, i_2 , corresponding to the horizontal axis and the vertical axis, respectively) of the dynamic ADMM, the offline optimizer and the ground-truth in one trial of the fast time-variant case are shown. The starting point corresponds to $k = 10$ and the time gap between two adjacent points is 10. We observe that the dynamic ADMM can track the truth well. The tracking performance of the offline optimizer is somewhat better, but at the expense of its heavy or even intractable computational burden in many real-time applications.

V. CONCLUSION

In this paper, motivated by the dynamic sharing problem, we propose and study a dynamic ADMM algorithm, which can adapt to the time-varying optimization problems in an online manner. Theoretical analysis is presented to show that the gaps between the algorithm iterates and the dynamic optimal points converge linearly to some neighborhoods of zero. The sizes of the neighborhoods depend on the inherent evolution speed, i.e., the drift, of the dynamic optimization problem across time: the more drastically the problem evolves, the bigger the size of the neighborhood. Explicit upper bounds of the limiting optimality gaps of the dynamic ADMM are given. Moreover, regret and constraint violation bounds of the dynamic ADMM are developed in terms of the cumulative drift of the dynamic problem. Finally, numerical results are presented to validate the proposed dynamic ADMM.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their insightful comments and suggestions.

REFERENCES

- [1] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine Learning*, 2011.
- [2] S. Haykin, *Adaptive Filter Theory (3rd Ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996.
- [3] E. Hazan *et al.*, "Introduction to online convex optimization," *Foundations and Trends® in Optimization*, vol. 2, no. 3-4, pp. 157–325, 2016.
- [4] M. Mahdavi, R. Jin, and T. Yang, "Trading regret for efficiency: online convex optimization with long term constraints," *Journal of Machine Learning Research*, vol. 13, no. Sep, pp. 2503–2528, 2012.
- [5] A. Koppel, F. Y. Jakubiec, and A. Ribeiro, "A saddle point algorithm for networked online convex optimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 19, pp. 5149–5164, 2015.
- [6] A. D. Flaxman, A. T. Kalai, and H. B. McMahan, "Online convex optimization in the bandit setting: gradient descent without a gradient," in *SODA*, pp. 385–394, 2005.

- [7] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *ICML*, pp. 928–936, 2003.
- [8] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," in *FOCS*, pp. 256–261, IEEE, 1989.
- [9] S. Arora, E. Hazan, and S. Kale, "The multiplicative weights update method: a meta-algorithm and applications.," *Theory of Computing*, vol. 8, no. 1, pp. 121–164, 2012.
- [10] J. Kivinen and M. K. Warmuth, "Exponentiated gradient versus gradient descent for linear predictors," *Information and Computation*, vol. 132, no. 1, pp. 1–63, 1997.
- [11] C. Tekin, J. Yoon, and M. van der Schaar, "Adaptive ensemble learning with confidence bounds," *IEEE Transactions on Signal Processing*, vol. 65, no. 4, pp. 888–903, 2016.
- [12] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.
- [13] D. P. Bertsekas, "Dynamic programming and stochastic control," 1976.
- [14] G. França and J. Bento, "An explicit rate bound for over-relaxed ADMM," in *IEEE ISIT*, pp. 2104–2108, 2016.
- [15] P. Giselsson and S. Boyd, "Linear convergence and metric selection for douglas-rachford splitting and ADMM," *IEEE Transactions on Automatic Control*, vol. 62, no. 2, pp. 532–544, 2017.
- [16] H. Wang and A. Banerjee, "Online alternating direction method," *ICML*, 2012.
- [17] S. Hosseini, A. Chapman, and M. Mesbahi, "Online distributed admm on networks: Social regret, network effect, and condition measures," *arXiv:1412.7116*, 2014.
- [18] H. Ouyang, N. He, L. Tran, and A. G. Gray, "Stochastic alternating direction method of multipliers," *ICML*, vol. 28, pp. 80–88, 2013.
- [19] W. Zhong and J. T.-Y. Kwok, "Fast stochastic alternating direction method of multipliers," in *ICML*, pp. 46–54, 2014.
- [20] T. Suzuki, "Dual averaging and proximal gradient descent for online alternating direction multiplier method," in *ICML*, pp. 392–400, 2013.
- [21] Q. Ling and A. Ribeiro, "Decentralized dynamic optimization through the alternating direction method of multipliers," *IEEE Transactions on Signal Processing*, vol. 62, no. 5, pp. 1185–1197, 2014.
- [22] A. Makhdoumi and A. Ozdaglar, "Broadcast-based distributed alternating direction method of multipliers," in *Communication, Control, and Computing (Allerton)*, 2014 52nd Annual Allerton Conference on, pp. 270–277, IEEE, 2014.
- [23] N. S. Aybat, Z. Wang, T. Lin, and S. Ma, "Distributed linearized alternating direction method of multipliers for composite convex consensus optimization," *IEEE Transactions on Automatic Control*, vol. 63, no. 1, pp. 5–20, 2018.
- [24] G. França and J. Bento, "How is distributed admm affected by network topology?," *arXiv preprint arXiv:1710.00889*, 2017.
- [25] A. Makhdoumi and A. Ozdaglar, "Convergence rate of distributed admm over networks," *IEEE Transactions on Automatic Control*, vol. 62, no. 10, pp. 5082–5095, 2017.
- [26] Q. Ling, Y. Liu, W. Shi, and Z. Tian, "Communication-efficient weighted admm for decentralized network optimization," in *Acoustics, Speech and Signal Processing (ICASSP)*, 2016 IEEE International Conference on, pp. 4821–4825, IEEE, 2016.
- [27] A. Teixeira, E. Ghadimi, I. Shames, H. Sandberg, and M. Johansson, "Optimal scaling of the admm algorithm for distributed quadratic programming," in *IEEE CDC*, pp. 6868–6873, 2013.
- [28] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the admm in decentralized consensus optimization," *IEEE Trans. on Signal Processing*, vol. 62, no. 7, pp. 1750–1761, 2014.
- [29] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
- [30] W. Deng and W. Yin, "On the global and linear convergence of the generalized alternating direction method of multipliers," *Journal of Scientific Computing*, vol. 66, no. 3, pp. 889–916, 2016.
- [31] L. Vandenberghe, "Gradient method," *lecture notes*, UCLA, 2016.
- [32] T. Chen, Q. Ling, and G. B. Giannakis, "An online convex optimization approach to proactive network resource allocation," *IEEE Transactions on Signal Processing*, vol. 65, no. 24, pp. 6350–6364, 2017.
- [33] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1." <http://cvxr.com/cvx>, mar 2014.