Distributed Dantzig-Wolfe Decomposition

Mohamed El Tonbari* and Shabbir Ahmed†
School of Industrial & Systems Engineering,
Georgia Institute of Technology

Abstract

Dantzig-Wolfe decomposition (DWD) is a classical algorithm for solving large-scale linear programs whose constraint matrix involves a set of independent blocks coupled with a set of linking rows. The algorithm decomposes such a problem into a master problem and a set of independent subproblems that can be solved in a distributed manner. In a typical implementation, the master problem is solved centrally. In certain settings, solving the master problem centrally is undesirable or infeasible. For example, in the case of decentralized storage of data, or when independent agents who are responsible for the subproblems desire privacy of information. In this paper, we propose a fully distributed DWD algorithm that relies on solving the master problem using a distributed consensus-based Alternating Direction Method of Multipliers (ADMM) algorithm. We derive error bounds on the optimality gap and feasibility violation of the proposed approach. We provide preliminary computational results for our algorithm using a Message Passing Interface (MPI) implementation on randomly generated instances where we obtain high quality solutions.

1 Introduction

1.1 Dantzig-Wolfe Decomposition

Dantzig-Wolfe decomposition (DWD) [10] is a classical algorithm for solving a large-scale linear program whose constraint matrix involves a set of independent blocks coupled with a set of linking rows. This class of problems are of the form

$$\begin{aligned} \min \quad & \sum_{n=1}^{N} c_n^\top x_n \\ \text{s.t.} \quad & \sum_{n=1}^{N} A_n x_n = t \\ & x_n \in X_n, \ \forall n = 1, ..., N \end{aligned} \tag{P}$$

where N is the number of blocks, the set X_n denotes the feasible region of the n-th block (or subproblem) with the decision vector x_n, and the constraint system $\sum_{n=1}^{N} A_n x_n = t$ denotes the system of linking constraints. The DWD method decomposes problem (P) into a master problem and a set of N independent subproblems. Throughout this paper we assume that the sets X_n for $n = 1, \ldots, N$ are polytopes (i.e. bounded polyhedra). In this case the master problem is a reformulation where the variables are replaced by a convex combination of the extreme points of

*Email: mtonbari@gatech.edu
†Email: sahmed@isye.gatech.edu

X_n. From Minkowski's theorem, the master problem can be written as

$$\begin{aligned}
\min \quad & \sum_{n=1}^{N} \sum_{i \in I_n} c_n^\top x_n^i \lambda_{ni} \\
\text{s.t.} \quad & \sum_{n=1}^{N} \sum_{i \in I_n} A_n x_n^i \lambda_{ni} = t \\
& \sum_{i \in I_n} \lambda_{ni} = 1, \ \forall n = 1, ..., N \\
& \lambda_{ni} \geq 0, \ \forall i \in I_n, \ \forall n = 1, ..., N
\end{aligned}$$

where $\{x_n^i\}_{i \in I_n}$ and I_n correspond to the extreme points of X_n and their index set, respectively; the variable λ_{ni} is the convex multiplier associated with extreme point i of subproblem n.

Since the number of extreme points of the subproblem polytopes can be exponentially large, a restricted master problem (RMP) is solved instead, using only a subset of the extreme points (or columns). Using an optimal dual solution of the RMP, each subproblem is solved independently to find a new extreme point whose reduced cost is negative. This process (called column generation) is repeated until no more extreme points or columns with negative reduced costs can be found. A solution to RMP then provides an optimal solution to (P). See [2, 11, 17, 23] for details and applications of DWD and column generation.

1.2 Contributions

In each iteration of DWD, the subproblems can be solved independently in a distributed manner but the RMP is solved centrally. In certain settings, solving the master problem centrally is undesirable or infeasible. For example if we interpret the variables of each suproblem as the decision variables of independent agents, then solving the RMP centrally requires the agents to share data of their constraints and objective, potentially violating privacy. Alternatively, due to memory limitations, it may be infeasible to include columns from all subproblems in the RMP. In this paper, we propose a fully distributed DWD algorithm that relies on solving the master problem using a consensus-based Alternating Direction Method of Multipliers (ADMM) algorithm. The ADMM approach provides ϵ-optimal and δ-feasible dual solutions to the RMP in a distributed fashion. When solving the RMP, each subproblem need only share their dual variables with a central coordinator, thus preserving privacy. Each subproblem then attempts to generate a new column by solving the pricing subproblem using the collected approximate dual solutions. Generated columns are added to the RMP and the process is repeated until no columns with negative reduced cost can be added, within a specified tolerance. We dynamically adjust the tolerances, where we first aim for inaccuracte dual solutions to speed up ADMM convergence. Once there are no more columns to be added, we decrease the tolerances by some positive factor, and repeat the whole process. This yields computational benefits, decreasing the number of times we need to solve ADMM to high accuracy. We show that we can recover a primal solution to the original problem that is close to feasible and close to optimal. We prove bounds on the feasibility violation and optimality gap for the recovered primal solution. Finally, we provide preliminary computational results for the proposed algorithm using a Message Passing Interface (MPI) implementation on randomly generated instances where we obtain high quality solutions.

1.3 Prior Work

Solving the dual of RMP in a distributed manner leads to approximate dual solutions used in the pricing subproblems, as opposed to standard DWD where exact optimal dual solutions are readily available. Several alternatives to standard DWD proposed in the literature use approximate dual solutions to solve the pricing subproblems to circumvent unstable behavior resulting from using exact dual solutions [13]. One method is to add a penalty term to reduce the variation of obtained dual solutions [25, 14]; another method involves using primal-dual interior point method to solve for approximate dual solutions to the RMP which are well-centered, where the optimality tolerance of the interior point method is dynamically adjusted to reduce computation time needed to solve the RMP [13, 14, 15]. In these methods, approximate dual solutions are used to solve the pricing subproblems, where tolerances are adjusted to satisfy a specified duality gap. However, these methods require solving the RMP centrally.

There has been a lot of work done on consensus problems and developing distributed optimization algorithms. A classic method is dual decomposition where the linking constraints are relaxed, and the problem becomes separable. The algorithm alternates between solving local subproblems independently and a central step which updates the dual variables as in dual ascent [4]. Many variants of dual decomposition have been proposed, such as using subgradients to update the dual iterates when optimizing nonsmooth functions [20]. Many of the distributed methods are optimized over a network, where agents, treated as nodes, only share limited information with neighboring nodes according to a transition matrix [22, 18, 27]. A survey on dual decomposition techniques for distributed optimization and consensus problems is given in [21]. Dual decomposition is known to suffer from weak convergence properties, which led to the augmented Lagrangian method. It provides stronger convergence properties and requires fewer assumptions than dual ascent (the method behind dual decomposition), but leads to an optimization problem that cannot be solved in a distributed way. ADMM has been developed to leverage the decomposability of dual decomposition, and the nice convergence properties of the augmented Lagrangian method [4]. Theoretical results and convergence properties of ADMM have been thoroughly studied in [4, 16, 24, 5, 1]. Studies on parameter tuning have also been done, notably the penalty parameter [12, 26].

The remainder of the paper is organized as follows. Section 2 formally defines the problem structure we are interested in and establishes the notation used throughout. In section 3, we give a brief overview of consensus-based ADMM and discuss traditionally used stopping criteria. In section 4, we describe our algorithm and prove bounds on the optimality gap and feasibility violation. We include numerical results to illustrate our method in section 5.

2 Preliminaries

We are interested in problems of the form of (P) where c_n and X_n are the cost vector and local constraints of block $n = 1, ..., N$. A_n is the constraint matrix of block n in the linking constraints.

To simplify notation, we rewrite the master problem as

$$\begin{aligned} \min \quad & \sum_{n=1}^{N} \sum_{i \in I_n} c_n^i \lambda_{ni} \\ \text{s.t.} \quad & \sum_{n=1}^{N} \sum_{i \in I_n} A_n^i \lambda_{ni} = t \\ & \sum_{i \in I_n} \lambda_{ni} = 1, \ \forall n = 1, ..., N \\ & \lambda_{ni} \geq 0, \ \forall i \in I_n, \ \forall n = 1, ..., N \end{aligned} \tag{MP}$$

where $c_n^i = c_n^\top x_n^i$ and $A_n^i = A x_n^i$ for all $i \in I_n$ and for all n.

We assume each X_n to be a non-empty polytope, so that there exists $L_n > 0$ such that $\left\|x_n^i\right\|_2 \leq L_n$ for all extreme points x_n^i of X_n. We further assume problem (MP) to be feasible and to have an optimal solution. The dual of (MP) is

$$\begin{aligned} \max \quad & t^\top \pi + \sum_{n=1}^{N} u_n \\ \text{s.t} \quad & A_n^{i\top} \pi + u_n \leq c_n^i \quad \forall i \in I_n, \ \forall n = 1, ..., N \end{aligned}$$

where π are dual variables associated with the linking constraints and u_n are dual variables associated with the convexity constraints $\sum_{i \in I_n} \lambda_{ni} = 1$ for all n.

Let $M > 0$ be an upper bound on the absolute values of the components of π. We show in Lemma 1 that we can pick M to be finite and polynomial in the entries of the data. For notational convenience in the proof of Lemma 1, let A be the horizontal concatenation of the N matrices A_n and let c represent the concatenation of the cost vectors $\{c_n\}_{n=1}^N$. For an index set J, let A_J be a matrix formed by columns $j \in J$ of A and c_J be a vector formed by components $j \in J$ of c. Let $B \in \mathcal{B}$ be an index set of the columns of A representing a basis, so that A_B is an invertible submatrix of A, where $\mathcal{B}$ is the set of all possible bases. Let $\mathcal{N}_B$ be the remaining indices not in B. Finally, let e be the vector of all ones of appropriate size and e_j be the vector of all zeros with a one in the j^{th} component.

Lemma 1. *Enforcing lower and upper bounds $-M$ and M on the components of π where $M > \max_{B \in \mathcal{B}, j \in \mathcal{N}_B} \{\sqrt{\sum_{n=1}^N \|c_n\|_2^2} \left\|A_B^{-1} e_j\right\|_2\}$ leads to an equivalent dual problem.*

Proof. An equivalent big-M formulation of (P) is

$$\begin{aligned} \min \quad & \sum_{n=1}^{N} c_n^\top x_n + M e^\top y^+ + M e^\top y^- \\ \text{s.t.} \quad & \sum_{n=1}^{N} A_n x_n + y^+ - y^- = t \\ & x_n \in X_n \\ & y^+, y^- \geq 0 \end{aligned} \tag{P_M}$$

Since (P) is assumed to have an optimal solution, (P_M) is equivalent to (P) if M is large enough and at any optimal solution, $y^+ = y^- = 0$. Note that the dual of the master problem reformulated from (P_M) is the same as the dual of (MP) with added bounds $-M \leq \pi \leq M$.

If we assume to solve problem (P_M) using the simplex method where at any point in the algorithm, only columns with a negative reduced cost can enter the basis, we need M to be big enough so that reduced costs of y^+ and y^- remain positive for all bases $B \in \mathcal{B}$. At a basis B which does not contain a component from either y^+ or y^-, the reduced costs of components y_j^+ and y_j^- are $M - c_B^\top A_B^{-1} e_j$ and $M + c_B^\top A_B^{-1} e_j$, respectively.

We need $M - c_B^\top A_B^{-1} e_j > 0$ and $M + c_B^\top A_B^{-1} e_j > 0$ which implies $M > \left| c_B^\top A_B^{-1} e_j \right|$. Moreover

$$\left| c_B^\top A_B^{-1} e_j \right| \leq \sqrt{\sum_{n=1}^{N} \|c_n\|_2^2} \left\| A_B^{-1} e_j \right\|_2$$

Thus, $M > \sqrt{\sum_{n=1}^{N} \|c_n\|_2^2} \left\| A_B^{-1} e_j \right\|_2 \; \forall B \in \mathcal{B}$ and $\forall j \in \mathcal{N}_B$ would be sufficient to obtain an equivalent problem and the result follows. □

We refer to the optimal objective values of the master problem and its dual by z^*_{MP} and z^*_{DM}. When considering a subset of the extreme points, we refer to the restricted primal and dual problems as (RMP) and (RDM), and their optimal values by z^*_{RMP} and z^*_{RDM}, respectively. Approximate solutions and their objective values are denoted by a hat. The notations $\|\cdot\|$ and $\|\cdot\|_F$ refer to the ℓ_2-norm and Frobenius norm, respectively. Finally, the terms agent and block will be used interchangeably.

3 ADMM Overview

We give a brief overview of ADMM used as a consensus method. We present the algorithm and discuss typical convergence conditions. Detailed discussion of ADMM method and its convergence properties can be found in [4].

3.1 Consensus-Based ADMM

The ADMM method can be used to solve problems of the following type

$$\begin{aligned} \max_{x} \quad & \sum_{n=1}^{N} f_n(x) \\ \text{s.t} \quad & A_n x \leq b_n, \; \forall n = 1, ..., N \end{aligned} \tag{A1}$$

where $f_n : \mathbb{R}^n \to \mathbb{R}$ are convex functions, and $A \in \mathbb{R}^{m \times n}$. The objective function and constraints are linked solely by the variable x. We can equivalently rewrite (A1) as

$$\begin{aligned} \max_{x_n, x} \quad & \sum_{n=1}^{N} f_n(x_n) \\ \text{s.t} \quad & A_n x_n \leq b_n, \; \forall n = 1, ..., N && \text{(3.1a)} \\ & x_n = x, \; \forall n = 1, ..., N && \text{(3.1b)} \end{aligned}$$

Let $\lambda_n \in \mathbb{R}^m$ and $\alpha_n \in \mathbb{R}^n$ be the Lagrangian multipliers of (3.1a) and (3.1b), respectively, for $n = 1, ..., N$. Taking the augmented Lagrangian of (3.1) gives

$$\begin{aligned} \max_{x_n, x} \quad & \sum_{n=1}^{N} \left[f_n(x_n) + \alpha_n^\top (x - x_n) - \frac{\rho}{2} \|x - x_n\|^2 \right] \\ \text{s.t} \quad & A_n x_n \leq b_n, \; \forall n = 1, ..., N \end{aligned} \tag{AL}$$

where $\rho > 0$ is a predetermined penalty parameter. Define the objective of (AL) as

$$\mathcal{L}_\rho(x, x_1, ..., x_N, \alpha_1, ..., \alpha_N) = \sum_{n=1}^{N} \left[f_n(x_n) + \alpha_n^\top (x - x_n) - \frac{\rho}{2} \|x - x_n\|^2 \right]$$

The ADMM method consists of alternating between maximizing the function $\mathcal{L}_\rho$ over $(x, x_1, ..., x_N)$ and minimizing over $(\alpha_1, ..., \alpha_N)$, where the maximization step is done sequentially, so that we first maximize over $(x_1, ...x_N)$ before maximizing over x. This allows to solve the former in a distributed fashion. The ADMM steps at an iteration k can be summarized as follows:

$$x_n^{k+1} \leftarrow \underset{x_n}{\text{argmax}}\ \mathcal{L}_\rho(x^k, x_1, ..., x_N, \alpha_1^k, ..., \alpha_N^k),\ \forall n = 1, ..., N \tag{3.2a}$$

$$x^{k+1} \leftarrow \underset{x}{\text{argmax}}\ \mathcal{L}_\rho(x, x_1^{k+1}, ..., x_N^{k+1}, \alpha_1^k, ..., \alpha_N^k) \tag{3.2b}$$

$$\alpha_n^{k+1} \leftarrow \alpha_n^k - \rho(x^{k+1} - x_n^{k+1}),\ \forall n = 1, ..., N \tag{3.2c}$$

Note that (3.2b) is an unconstrained maximization problem for which there exists a closed form solution. We have:

$$\begin{aligned} &\nabla_x \mathcal{L}_\rho(x, x_1^{k+1}, ..., x_N^{k+1}, \alpha_1^k, ..., \alpha_N^k) = 0 \\ \Rightarrow &\sum_{n=1}^{N} \left[\alpha_n^k - \rho(x^{k+1} - x_n^{k+1}) \right] = 0 \\ \Rightarrow &x^{k+1} = \frac{1}{N} \sum_{n=1}^{N} x_n^{k+1} + \frac{1}{N\rho} \sum_{n=1}^{N} \alpha_n^k \end{aligned}$$

We also note that (3.2c) is a gradient step where the step size is the penalty parameter ρ.

3.2 Convergence and Stopping Criteria

Let $\boldsymbol{\alpha}$ be the vertical concatenation of vectors $\{\alpha_n\}_{n=1}^N$. As proven in [4], under the assumption that the functions f_n in (A1) are convex, proper and closed, and assuming that $\mathcal{L}_0(x, x_1, ..., x_N, \boldsymbol{\alpha})$, where $\rho = 0$, has a saddle point, then as $k \to \infty$, we have the following:

(i) Primal feasibility violation vanishes: $\sqrt{\sum_{n=1}^N \left\| x^{k+1} - x_n^{k+1} \right\|^2} \to 0,\ n = 1, ..., N$

(ii) Dual feasibility violation vanishes: $\rho \left\| x^{k+1} - x^k \right\| \to 0$

(iii) Optimality gap vanishes: $\left\| f(x^*) - f(x^k) \right\| \to 0$

(iv) Dual vector $\boldsymbol{\alpha}$ converges to an optimal dual solution: $\left\| \boldsymbol{\alpha^k} - \boldsymbol{\alpha}^* \right\| \to 0$

In (i), we define primal feasibility violation to be

$$\left\| \begin{matrix} x^{k+1} - x_1^{k+1} \\ \vdots \\ x^{k+1} - x_N^{k+1} \end{matrix} \right\| = \sqrt{\sum_{n=1}^{N} \left\| x^{k+1} - x_n^{k+1} \right\|^2}$$

This implies that we reach consensus as $k \to \infty$, i.e $\left\| x^{k+1} - x_n^{k+1} \right\| \to 0$ for all n.

For our purposes, we also assume the functions f_n to be differentiable. This assumption is satisfied in our case since we are dealing with linear cost functions. Note that dual feasibility in

(3.1) is equivalent to $\nabla_{x_n} f_n(x_n) - A_n^\top \lambda_n - \alpha_n = 0$. From the optimality conditions of (AL), we have at iteration k:

$$\begin{aligned}
&\nabla_{x_n} f_n(x_n^{k+1}) - A_n^\top \lambda_n^{k+1} - \alpha_n^k + \rho(x^k - x_n^{k+1}) = 0 \\
\Rightarrow &\nabla_{x_n} f_n(x_n^{k+1}) - A_n^\top \lambda_n^{k+1} - \alpha_n^k + \rho(x^k - x_n^{k+1} + x^{k+1} - x^{k+1}) = 0 \\
\Rightarrow &\nabla_{x_n} f_n(x_n^{k+1}) - A_n^\top \lambda_n^{k+1} - \alpha_n^k + \rho(x^{k+1} - x_n^{k+1}) + \rho(x^k - x^{k+1}) = 0 \\
\Rightarrow &\nabla_{x_n} f_n(x_n^{k+1}) - A_n^\top \lambda_n^{k+1} - \alpha_n^{k+1} = \rho(x^{k+1} - x^k)
\end{aligned}$$

Thus, dual feasibility in (3.1) amounts to having $\rho(x^{k+1} - x^k) = 0$. As suggested in [4], it is reasonable to terminate ADMM once we reach primal and dual feasibility within some tolerance. Given specified tolerances ϵ_p and ϵ_d, we terminate ADMM once $\sqrt{\sum_{n=1}^N \left\| x^{k+1} - x_n^{k+1} \right\|^2} \leq \epsilon_p$ and $\rho \left\| x^{k+1} - x^k \right\| \leq \epsilon_d$.

4 Distributed Dantzig-Wolfe Algorithm

We first present the distributed Dantzig-Wolfe (DDW) algorithm before deriving error bounds on the optimality gap and feasibility violation.

4.1 DDW Algorithm

We hereafter define k to be the ADMM iteration coun-ter and ℓ to be the Dantzig-Wolfe outer iteration counter.

To solve the restricted master problem in a distributed fashion, we solve a reformulation of the dual of the (RMP). The reformulation permits us to perform consensus-based ADMM. We split the dual vector π associated with the linking constraints into N copies as in (3.1), to get the following equivalent formulation:

$$\begin{aligned}
\max \quad & \sum_{n=1}^N \left[\frac{1}{N} t^\top \pi_n + u_n \right] \\
\text{s.t} \quad & A_n^{i\top} \pi_n + u_n \leq c_n^i, \ \forall i \in I_n, \ n = 1, ..., N \\
& \pi_n = \pi, \ n = 1, ..., N
\end{aligned} \tag{DM}$$

Note that in (MP), the problem is linked by the rows. Performing ADMM directly on (MP) would lead to N blocks where we would need to optimize with respect to each x_n sequentially. Not only is the problem no longer decomposable, but ADMM is not guaranteed to converge when dealing with more than two blocks [6]. The dual of the master problem, however, is linked by the decision variable π. Performing ADMM on (DM) leads to a more natural consensus-based algorithm with guaranteed convergence, where the first ADMM block corresponds to solving for each π_n independently, and the second block corresponds to optimizing with respect to π.

We denote the restricted problem of (DM), i.e. one involving constraints corresponding to only a subset of the columns, by (RDM) and its optimal value by z^*_{RDM}. We take the augmented Lagrangian of the restricted dual problem by relaxing the copy constraints as in (AL) and get a

separable problem with respect to variables (π_n, u_n):

$$\begin{aligned} \max_{\pi_n, u_n} \quad & \sum_{n=1}^{N} \left[\frac{1}{N} t^\top \pi_n + u_n + \alpha_n^\top (\pi - \pi_n) - \frac{\rho}{2} \|\pi - \pi_n\|^2 \right] \\ \text{s.t} \quad & A_n^{i^\top} \pi_n + u_n \le c_n^i, \ \forall i \in I_n, \forall n = 1, ..., N \end{aligned}$$

At iteration k of ADMM and using current iterates π^k and α_n^k, each agent n solves

$$\begin{aligned} \max_{\pi_n, u_n} \quad & \frac{1}{N} t^\top \pi_n + u_n + \alpha_n^{k^\top} (\pi^k - \pi_n) - \frac{\rho}{2} \left\| \pi^k - \pi_n \right\|^2 \\ \text{s.t} \quad & A_n^{i^\top} \pi_n + u_n \le c_n^i, \ \forall i \in I_n^\ell \end{aligned} \qquad (ARDM_n)$$

where $I_n^\ell \subseteq I_n$ is the index set of extreme points of block n at outer iteration ℓ. From (3.2), the steps to solving (RDM) can be summarized as follows:

1. Each agent solves ($ARDM_n$) and collects optimal solutions (π_n^{k+1}, u_n^{k+1})
2. $\pi^{k+1} \leftarrow \frac{1}{N} \sum_{n=1}^{N} (\pi_n^{k+1}) + \frac{1}{N\rho} \sum_{n=1}^{N} \alpha_n^k$
3. $\alpha_n^{k+1} = \alpha_n^k - \rho(\pi^{k+1} - \pi_n^{k+1})$

First note that $A_n^{i^\top} \pi_n^{k+1} + u_n^{k+1} \le c_n^i$ is satisfied for all $i \in I_n^\ell$ and for all n, since (π_n^{k+1}, u_n^{k+1}) is a solution of ($ARDM_n$). Thus, $\pi_n^{k+1} = \pi^{k+1}$ are the only violated constraints. To avoid confusion, we refer to $\sqrt{\sum_{n=1}^{N} \left\| \pi^{k+1} - \pi_n^{k+1} \right\|^2}$ as the dual feasibility violation, and $\rho \left\| \pi^{k+1} - \pi^k \right\|$ as the primal feasibility violation. Note that this is the opposite of what is defined in Section 3 because we are performing ADMM on the dual problem here. We then perform steps 1-3 until $\sqrt{\sum_{n=1}^{N} \left\| \pi^{k+1} - \pi_n^{k+1} \right\|^2} \le \epsilon_d$ and $\rho \left\| \pi^{k+1} - \pi^k \right\| \le \epsilon_p$, where ϵ_d and ϵ_p are dual and primal feasibility tolerances, respectively.

Each agent n then solves a pricing subproblem to look for an extreme point with negative reduced cost:

$$z_{SEP}^n = \min_x \{ c_n^\top x_n - \pi^{k+1^\top} A_n x_n - u_n^{k+1} : x_n \in X_n \}$$

Let x_n^* be an optimal solution. In standard DWD, we would add x_n^* as a new column if $z_{SEP}^n < 0$. However, the dual solution $(\pi^{k+1}, \{u_n^{k+1}\}_{n=1}^N)$ is ϵ-optimal and only close to feasible for the current (RMP). It is possible that we find a column whose reduced cost is negative and close to 0 when evaluated at the approximate dual solutions, but is in fact already in the current (RMP). It is also possible that at the (unavailable) optimal dual solution, the reduced cost is actually positive and the extreme point should not be added. To ensure a finite algorithm, agent n only adds x_n^* as a new extreme point if $z_{SEP}^n < -\max_{i \in I_n^\ell} \{ \left\| A_n^i \right\| \} \epsilon_d$. This is justified by Lemma 2. After ADMM terminates, if $c_n^i - A_n^{i^\top} \pi^{k+1} - u_n^{k+1} \ge -\left\| A_n^i \right\| \epsilon_d$ for all i and n, and $-\max_{i \in I_n^\ell} \{ \left\| A_n^i \right\| \} \epsilon_d \le z_{SEP}^n = c_n^\top x_n^* - \pi^{k+1^\top} A_n x_n^* - u_n^{k+1} < 0$, then we cannot guarantee that x_n^* is a necessary extreme point. In other words, we can only trust z_{SEP}^n within $\max_{i \in I_n^\ell} \{ \left\| A_n^i \right\| \} \epsilon_d$.

Lemma 2. *If ADMM terminates with $\left\| \pi^{k+1} - \pi_n^{k+1} \right\| \le \epsilon_d$ for all n, we have*

$$c_n^i - A_n^{i^\top} \pi^{k+1} - u_n^{k+1} \ge -\left\| A_n^i \right\| \epsilon_d$$

for all $i \in I_n^\ell, n = 1, ..., N$.

Proof. We have

$$\begin{aligned}
&\left\| \begin{matrix} \pi^{k+1} - \pi_1^{k+1} \\ \vdots \\ \pi^{k+1} - \pi_N^{k+1} \end{matrix} \right\| \leq \epsilon_d \\
\Rightarrow &\sum_{n=1}^{N} \left\| \pi^{k+1} - \pi_n^{k+1} \right\|^2 \leq \epsilon_d^2 \\
\Rightarrow &\left\| \pi^{k+1} - \pi_n^{k+1} \right\|^2 \leq \epsilon_d^2 \\
\Rightarrow &\left\| \pi^{k+1} - \pi_n^{k+1} \right\| \leq \epsilon_d, \ \forall n = 1, ..., N
\end{aligned}$$

For any n, computing the distance between $c_n^i - A_n^{i^\top} \pi^{k+1} - u_n^{k+1}$ and $c_n^i - A_n^{i^\top} \pi_n^{k+1} - u_n^{k+1}$ gives us

$$\begin{aligned}
\left\| c_n^i - A_n^{i^\top} \pi^{k+1} - u_n^{k+1} - c_n^i + A_n^{i^\top} \pi_n^{k+1} + u_n^{k+1} \right\| &= \left\| A_n^{i^\top} (\pi^{k+1} - \pi_n^{k+1}) \right\| \\
&\leq \left\| A_n^i \right\| \epsilon_d, \ \forall i \in I_n^\ell
\end{aligned} \tag{4.1}$$

Since $c_n^i - A_n^{i^\top} \pi_n^{k+1} - u_n^{k+1} \geq 0$ for all $i \in I_n^\ell$, (4.1) implies $c_n^i - A_n^{i^\top} \pi^{k+1} - u_n^{k+1} \geq -\left\| A_n^i \right\| \epsilon_d$ for all $i \in I_n^\ell$ and n. □

Once the columns are added to the (RMP), we use the solutions of the last iterates π^{k+1} and α_n^{k+1} as warm starts for π^1 and α_n^1 for all n in the next outer iteration $\ell + 1$. If $z_{SEP}^n \geq -\max_{i \in I_n^{\ell+1}} \{ \left\| A_n^i \right\| \} \epsilon_d$ for all n, we terminate the algorithm and retrieve approximate primal solutions $\hat{x}_n \leftarrow \sum_{i \in I_n^{\ell+1}} \lambda_{ni}^{k+1} x_n^i$ for all $n = 1, ..., N$, where λ_{ni}^{k+1} are the Lagrangian multipliers associated with the constraints in ($ARDM_n$), $i \in I_n^{\ell+1}, n = 1, ..., N$. The overall DWD algorithm is summarized in Algorithm 1. We describe the algorithm in a master-worker framework where agents are referred to as processors. At each ADMM iteration, the master node calls the `BROADCAST()` function to send the current estimate of π to each processor, and the `RECEIVE()` function to collect each processor's dual solution π_n obtained from solving ($ARDM_n$).

4.2 Convergence

We now prove the convergence of DDW and provide bounds on the optimality gap and feasibility violation. The quality of the dual solutions obtained by the consensus ADMM algorithm directly affects the quality of the recovered primal solution. We are able to reduce the optimality gap and feasibility violation by tweaking the primal and dual infeasibility tolerances ϵ_p and ϵ_d. Recall since we are solving the dual of (MP) using ADMM, we refer to the Lagrangian multipliers α_n in the objective of ($ARDM_n$) and the multipliers λ_{ni} associated with the constraints as primal variables, and π, π_n and u_n as dual variables; we refer to $\sqrt{\sum_{n=1}^{N} \left\| \pi^{k+1} - \pi_n^{k+1} \right\|^2}$ as the dual feasibility violation and $\rho \left\| \pi^{k+1} - \pi^k \right\|$ as the primal feasibility violation.

Recall that z_{MP}^* and z_{DM}^* refer to the optimal objective values of the master problem (MP) and its dual, respectively; z_{RMP}^* and z_{RDM}^* refer to the optimal values of their restrictive counterparts; objective values and solutions resulting from the DDW algorithm are denoted by a hat such as $\hat{z}_{RDM}$. As shown in [4] and other sources in the literature, given tolerances $\epsilon, \epsilon_p, \epsilon_d > 0$, we can assume that ADMM terminates with $z_{RDM}^* - \hat{z}_{RDM} \leq \epsilon$, $\rho \left\| \pi^{k+1} - \pi^k \right\| \leq \epsilon_p$ and $\sqrt{\sum_{n=1}^{N} \left\| \pi^{k+1} - \pi_n^{k+1} \right\|^2} \leq \epsilon_d$. The following lemmas will be helpful in proving the error bounds.

Algorithm 1 DDW Algorithm

Input: tolerances ϵ_p and ϵ_d, penalty parameter ρ
Let I_n^1 be the initial set of columns for each block n
Initialize π^1, α_n^1 for all n and $\ell = 0$
while columns added **do**
    $\ell \leftarrow \ell + 1$
    Initialize primal and dual residuals $r_p = \infty$ and $r_d = \infty$

    /*Solve RDM using consensus-based ADMM*/
    Initialize $k = 0$
    while $r_d > \epsilon_d$ and $r_p > \epsilon_p$ **do**
        $k \leftarrow k + 1$
        `BROADCAST(`π^k`)`
        for each processor $n = 1, ..., N$ **do**
            Solve ($ARDM_n$)
            Collect optimal solutions (π_n^{k+1}, u_n^{k+1})
            Collect Lagrangian multipliers λ_n^{k+1}
        end for
        `RECEIVE(`$\{\pi_n^{k+1}\}_{n=1}^N$`)`
        $\pi^{k+1} \leftarrow \frac{1}{N}\sum_{n=1}^N(\pi_n^{k+1}) + \frac{1}{N\rho}\sum_{n=1}^N \alpha_n^k$
        $\alpha_n^{k+1} = \alpha_n^k - \rho(\pi^{k+1} - \pi_n^{k+1})$
        $r_d \leftarrow \sqrt{\sum_{n=1}^N \left\|\pi^{k+1} - \pi_n^{k+1}\right\|^2}$
        $r_p \leftarrow \rho\left\|\pi^{k+1} - \pi^k\right\|$
    end while
    `BROADCAST(`π^{k+1}`)`

    /*Solve pricing subproblems*/
    for each processor $n = 1, ..., N$ **do**
        $z_{SEP}^n \leftarrow \min_{x_n}\{c_n^\top x_n - {\pi^{k+1}}^\top A_n x_n - u_n^{k+1} : x_n \in X_n\}$
        if $z_{SEP}^n < -\max_{i \in I_n^\ell}\{\left\|A_n^i\right\|\}\epsilon_d$ **then**
            $x_n^i \leftarrow \arg\min\{c_n^\top x_n - {\pi^{k+1}}^\top A_n x_n - u_n^{k+1} : x_n \in X_n\}$
            Add extreme point x_n^i: $I_n^{\ell+1} \leftarrow I_n^\ell \cup \{i\}$
        else
            $I_n^{\ell+1} \leftarrow I_n^\ell$
        end if
    end for
    $\pi^1 \leftarrow \pi^{k+1}$
    $\alpha_n^1 \leftarrow \alpha_n^{k+1},\ \forall n = 1, ..., N$
end while

/* Each processor n retrieves primal solution*/
$\hat{x}_n \leftarrow \sum_{i \in I_n^{\ell+1}} \lambda_{ni}^{k+1} x_n^i,\ \forall n = 1, ..., N$

Lemma 3. *After the first iteration of DDW, the Lagrangian multipliers α_n^k associated with the copy constraints are primal feasible for all n, i.e for $k \geq 0$, we have $\sum_{n=1}^N \alpha_n^{k+1} = 0$.*

Proof. From the updates, we have with $k \geq 0$:

$$\begin{aligned}
\pi^{k+1} &= \frac{1}{N}\sum_{n=1}^{N}\pi_n^{k+1} + \frac{1}{N\rho}\sum_{n=1}^{N}\alpha_n^k \\
\Rightarrow \sum_{n=1}^{N}\alpha_n^k &= N\rho\pi^{k+1} - \rho\sum_{n=1}^{N}\pi_n^{k+1} \\
\Rightarrow \sum_{n=1}^{N}\alpha_n^k &= \rho\left(N\pi^{k+1} - \sum_{n=1}^{N}\pi_n^{k+1}\right)
\end{aligned}$$

and

$$\alpha_n^{k+1} = \alpha_n^k - \rho(\pi^{k+1} - \pi_n^{k+1}),\ \forall n = 1, ..., N$$

Summing over n, we get

$$\begin{aligned}
\sum_{n=1}^{N}\alpha_n^{k+1} &= \sum_{n=1}^{N}\alpha_n^k - \rho\left(N\pi^{k+1} - \sum_{n=1}^{N}\pi_n^{k+1}\right) \\
&= \rho\left(N\pi^{k+1} - \sum_{n=1}^{N}\pi_n^{k+1}\right) - \rho\left(N\pi^{k+1} - \sum_{n=1}^{N}\pi_n^{k+1}\right) \\
&= 0
\end{aligned}$$

□

Theorem 1 establishes the feasibility violation at the recovered primal solution.

Theorem 1 (Feasibility Violation)**.** *Given a primal feasibility tolerance $\epsilon_p > 0$, DDW terminates with a solution $\hat{x}_n = \sum_{i \in I_n^{\ell+1}} \lambda_{ni}^{k+1} x_n^i$ such that:*

$$\begin{aligned}
&\left\| \sum_{n=1}^{N} A_n \hat{x}_n - t \right\| \leq N\epsilon_p \\
&\sum_{i \in I_n^{\ell}} \lambda_{ni}^{k+1} = 1,\ \forall n
\end{aligned}$$

Proof. At outer iteration ℓ and iteration k of ADMM, let the Lagrangian functions of ($ARDM_n$) for each n be:

$$\begin{aligned}
Q_n(\pi_n, \alpha_n^k, \pi^k, \lambda) = \frac{1}{N}t^\top\pi_n + u_n + \alpha_n^{k\top}(\pi^k - \pi_n) - \frac{\rho}{2}\left\|\pi^k - \pi_n\right\|_2^2 \\
+ \sum_{i \in I_n^{\ell}} \lambda_{ni}(c_n^i - A_n^{i\top}\pi_n - u_n)
\end{aligned}$$

where $\{\lambda_{ni}\}_{i \in I_n^{\ell}}$ are the multipliers of the constraints in ($ARDM_n$).

We have the following optimality conditions in ($ARDM_n$):

$$\begin{aligned}
&\lambda_{ni}^{k+1}(c_n^i - A_n^{i^\top}\pi_n^{k+1} - u_n^{k+1}) = 0,\ \forall i \in I_n^\ell && \text{(Complementary Slackness)}\\
&\lambda_{ni}^{k+1} \geq 0,\ \forall i \in I_n^\ell && \text{(Dual Feasibility)}\\
&\nabla_{\pi_n} Q_n = \frac{1}{N}t - \alpha_n^k + \rho(\pi^k - \pi_n^{k+1}) - \sum_{i\in I_n^\ell} A_n^i \lambda_{ni}^{k+1} = 0 && \\
&\nabla_{u_n} Q_n = 1 - \sum_{i\in I_n^\ell} \lambda_{ni}^{k+1} = 0 && \text{(Stationarity)}
\end{aligned}$$

Thus, the convexity constraints in (RMP) $\sum_{i\in I_n^\ell} \lambda_{ni}^{k+1} = 1$ are satisfied for all n and $\lambda_{ni}^{k+1} \geq 0$ for all n and i.

We rewrite the stationarity condition with respect to π_n as

$$\begin{aligned}
\nabla_{\pi_n} Q_n &= \frac{1}{N}t - \alpha_n^k + \rho(\pi^k - \pi_n^{k+1}) - \sum_{i\in I_n^\ell} A_n^i \lambda_{ni}^{k+1}\\
&= \frac{1}{N}t - \alpha_n^k + \rho(\pi^k - \pi^{k+1} + \pi^{k+1} - \pi_n^{k+1}) - \sum_{i\in I_n^\ell} A_n^i \lambda_{ni}^{k+1}\\
&= \frac{1}{N}t - \alpha_n^k + \rho(\pi^k - \pi^{k+1}) + \rho(\pi^{k+1} - \pi_n^{k+1}) - \sum_{i\in I_n^\ell} A_n^i \lambda_{ni}^{k+1}\\
&= \frac{1}{N}t - \rho(\pi^{k+1} - \pi^k) - \sum_{i\in I_n^\ell} A_n^i \lambda_{ni}^{k+1} - \alpha_n^{k+1} \qquad (4.2)
\end{aligned}$$

where the last equality holds from $\alpha_n^{k+1} = \alpha_n^k - \rho(\pi^{k+1} - \pi_n^{k+1})$.

Summing $\nabla_{\pi_n} Q_n$ over $n = 1, ..., N$, we get

$$\begin{aligned}
\sum_{n=1}^N \nabla_{\pi_n} Q_n &= t - \rho\sum_{n=1}^N(\pi^{k+1} - \pi^k) - \sum_{n=1}^N\sum_{i\in I_n^\ell} A_n^i \lambda_{ni}^{k+1} - \sum_{n=1}^N \alpha_n^{k+1}\\
&= t - \rho N(\pi^{k+1} - \pi^k) - \sum_{n=1}^N\sum_{i\in I_n^\ell} A_n^i \lambda_{ni}^{k+1}\\
&= 0
\end{aligned}$$

where the second equality follows because $\sum_{n=1}^N \alpha_n^{k+1} = 0$ from Lemma 3.

Then

$$\begin{aligned}
&\left\|\sum_{n=1}^N \nabla_{\pi_n} Q_n\right\| = 0 \geq \left\|t - \sum_{n=1}^N\sum_{i\in I_n^\ell} A_n^i \lambda_{ni}^{k+1}\right\| - N\rho\left\|\pi^{k+1} - \pi^k\right\|\\
\Rightarrow &\left\|t - \sum_{n=1}^N\sum_{i\in I_n^\ell} A_n^i \lambda_{ni}^{k+1}\right\| \leq N\rho\left\|\pi^{k+1} - \pi^k\right\|\\
\Rightarrow &\left\|t - \sum_{n=1}^N A_n \hat{x}_n\right\| \leq N\epsilon_p
\end{aligned}$$

where $\hat{x}_n = \sum_{i\in I_n^\ell} \lambda_{ni}^{k+1} x_n^i = \sum_{i\in I_n^{\ell+1}} \lambda_{ni}^{k+1} x_n^i$, since $I_n^{\ell+1} = I_n^\ell$ for all n when DDW terminates. □

Before deriving error bounds on the optimality gap, we first introduce bounds on $z^*_{DM} - \hat{z}_{RDM}$ and $\hat{z}_{RMP} - \hat{z}_{RDM}$. Adding these two will then give us bounds on $\hat{z}_{RMP} - z^*_{DM}$, or equivalently $\hat{z}_{RMP} - z^*_{MP}$. The following is a known relationship between $\hat{z}_{RDM}$, z^*_{DM} and z^*_{RDM} (cf. [17]).

Lemma 4. *After terminating ADMM, we have* $\hat{z}_{RDM} + \sum_{n=1}^{N} \min\{0, z^n_{SEP}\} \leq z^*_{DM} \leq z^*_{RDM}$.

Proof. If $z^n_{SEP} < 0$ for some n, then we can set $\hat{u}'_n = \hat{u}_n + z^n_{SEP}$. Doing so for each n, we get a feasible solution $(\hat{\pi}, \{\hat{u}'_n\}_{n=1}^N)$ to (DM) with objective value $\hat{z}_{RDM} + \sum_{n=1}^{N} \min\{0, z^n_{SEP}\} \leq z^*_{DM}$. Moreover, $z^*_{DM} \leq z^*_{RDM}$ since (RDM) is a relaxation of (DM). □

Proposition 1. *Given that ADMM terminates with* $z^*_{RDM} - \hat{z}_{RDM} \leq \epsilon$ *and we terminate DDW when* $z^n_{SEP} \geq -\max_{i \in I^\ell_n}\{\|A^i_n\|\}\epsilon_d$ *for all* n*, we have*

$$-\sum_{n=1}^{N} \|A_n\|_F L_n \leq z^*_{DM} - \hat{z}_{RDM} \leq \epsilon$$

where L_n *is a bound on all extreme points of the set* X_n*, defining the local constraints of agent* n.

Proof. By Lemma 4, $\hat{z}_{RDM} + \sum_{n=1}^{N} \min\{0, z^n_{SEP}\} \leq z^*_{DM} \leq z^*_{RDM}$. Since $z^*_{RDM} - \hat{z}_{RDM} \leq \epsilon$ and $z^n_{SEP} \geq -\max_{i \in I^\ell_n}\{\|A^i_n\|\}\epsilon_d$ for all n after terminating ADMM, we have:

$$\begin{aligned}
&\hat{z}_{RDM} + \sum_{n=1}^{N} \min\{0, z^n_{SEP}\} \leq z^*_{DM} \leq z^*_{RDM} \\
\Rightarrow &\hat{z}_{RDM} - \hat{z}_{RDM} + \sum_{n=1}^{N} \min\{0, z^n_{SEP}\} \leq z^*_{DM} - \hat{z}_{RDM} \leq z^*_{RDM} - \hat{z}_{RDM} \\
\Rightarrow &-\sum_{n=1}^{N} \max_{i \in I^\ell_n}\{\|A^i_n\|\}\epsilon_d \leq z^*_{DM} - \hat{z}_{RDM} \leq \epsilon \\
\Rightarrow &-\sum_{n=1}^{N} \|A_n\|_F L_n \leq z^*_{DM} - \hat{z}_{RDM} \leq \epsilon
\end{aligned}$$

where the last inequality holds from our assumption that $\|x^i_n\| \leq L_n$ for all extreme points of block n: $\|A^i_n\| = \|A_n x^i_n\| \leq \|A_n\|_F L_n$. □

Proposition 2. *Terminating ADMM with primal and dual feasibility tolerances* ϵ_p *and* ϵ_d*, respectively, we have at any outer iteration* ℓ

$$|\hat{z}_{RMP} - \hat{z}_{RDM}| \leq \epsilon_d \sum_{n=1}^{N} \|A_n\|_F L_n + mMN\epsilon_p$$

where m *is the number of linking constraints,* M *is an upperbound on the absolute values of the components of* π *as in Lemma 1, and* N *is the number of agents.*

Proof. The complementary slackness conditions for (RDM) are

$$\lambda_{ni}(c^i_n - A^{i^\top}_n \pi - u) = 0, \ \forall i \in I^\ell_n, \ n = 1, ..., N \tag{4.3}$$

$$\pi^\top (\sum_{n=1}^{N} \sum_{i \in I^\ell_n} A^i_n \lambda_{ni} - t) = 0, \ n = 1, ..., N \tag{4.4}$$

$$u_n (\sum_{i \in I^\ell_n} \lambda_{ni} - 1) = 0, \ n = 1, ..., N \tag{4.5}$$

Note that $\lambda_{ni}^{k+1}(c_n^i - A_n^{i^\top}\pi_n^{k+1} - u_n^{k+1}) = 0$ from the optimality conditions in ($ARDM_n$). Thus, plugging $\pi^{k+1}, u_n^{k+1}, \lambda_{ni}^{k+1}$ into (4.3), we get for each n:

$$\begin{aligned}\left|\lambda_{ni}^{k+1}(c_n^i - A_n^{i^\top}\pi^{k+1} - u_n^{k+1})\right| &= \Big|\lambda_{ni}^{k+1}(c_n^i - A_n^{i^\top}\pi^{k+1} - u_n^{k+1}) \\ &\qquad - \lambda_{ni}^{k+1}(c_n^i - A_n^{i^\top}\pi_n^{k+1} - u_n^{k+1})\Big| \\ &= \left|\lambda_{ni}^{k+1}A_n^{i^\top}(\pi_n^{k+1} - \pi^{k+1})\right| \\ &\le \left|\lambda_{ni}^{k+1}\right| \left\|A_n^i\right\|\epsilon_d, \ \forall i \in I_n^\ell\end{aligned}$$

Summing over $i \in I_n^\ell$, we get

$$\begin{aligned}\sum_{i\in I_n^\ell}\left|\lambda_{ni}^{k+1}(c_n^i - A_n^{i^\top}\pi^{k+1} - u_n^{k+1})\right| &\le \sum_{i\in I_n^\ell}\lambda_{ni}^{k+1}\left\|A_n^i\right\|\epsilon_d \\ &\le \max_{i\in I_n^\ell}\{\left\|A_n^i\right\|\}\epsilon_d \\ &\le \|A_n\|_F L_n\epsilon_d\end{aligned}$$

The first inequality follows because $\left|\lambda_{ni}^{k+1}\right| = \lambda_{ni}^{k+1}$ and the second inequality holds because $\sum_{i\in I_n^\ell}\lambda_{ni}^{k+1} = 1$. Summing over n gives us

$$\begin{aligned}&\left|\sum_{n=1}^N\sum_{i\in I_n^\ell}c_n^i\lambda_{ni}^{k+1} - \sum_{n=1}^N\sum_{i\in I_n^\ell}\lambda_{ni}^{k+1}(A_n^{i^\top}\pi^{k+1} + u_n^{k+1})\right| \le \sum_{n=1}^N\|A_n\|_F L_n\epsilon_d \\ \Rightarrow &\left|\hat{z}_{RMP} - \left[\sum_{n=1}^N\sum_{i\in I_n^\ell}\lambda_{ni}^{k+1}(A_n^{i^\top}\pi^{k+1} + u_n^{k+1})\right]\right| \le \sum_{n=1}^N\|A_n\|_F L_n\epsilon_d \qquad (4.6)\end{aligned}$$

Moreover, using the feasibility violation bound, (4.4) becomes

$$\left|\pi^{k+1^\top}(\sum_{i=1}^N\sum_{i\in I_n^\ell}A_n^i\lambda_{ni}^{k+1} - t)\right| \le \left\|\pi^{k+1}\right\|N\epsilon_p \le mMN\epsilon_p$$

Since $u_n^{k+1}(\sum_{i\in I_n^\ell}\lambda_{ni}^{k+1} - 1) = 0$ is satisfied for all n, we have

$$\begin{aligned}&\left|\pi^{k+1^\top}(\sum_{n=1}^N\sum_{i\in I_n^\ell}A_n^i\lambda_{ni}^{k+1} - t) + \sum_{n=1}^N u_n^{k+1}(\sum_{i\in I_n^\ell}\lambda_{ni}^{k+1} - 1)\right| \le mMN\epsilon_p \\ \Rightarrow &\left|\left[\sum_{n=1}^N\sum_{i\in I_n^\ell}\lambda_{ni}^{k+1}(A_n^{i^\top}\pi^{k+1} + u_n^{k+1})\right] - \left[\pi^{k+1^\top}t + \sum_{n=1}^N u_n^{k+1}\right]\right| \le mMN\epsilon_p \\ \Rightarrow &\left|\left[\sum_{n=1}^N\sum_{i\in I_n^\ell}\lambda_{ni}^{k+1}(A_n^{i^\top}\pi^{k+1} + u_n^{k+1})\right] - \hat{z}_{RDM}\right| \le mMN\epsilon_p \qquad (4.7)\end{aligned}$$

Adding (4.6) and (4.7) gives us

$$|\hat{z}_{RMP} - \hat{z}_{RDM}| \le \epsilon_d\sum_{n=1}^N\|A_n\|_F L_n + mMN\epsilon_p$$

□

Theorem 2 (Optimality Gap)**.** *DDW terminates with a solution $\hat{x}$ such that:*

$$-\epsilon - \gamma\epsilon_d - mMN\epsilon_p + \leq \hat{z}_{RMP} - z^*_{MP} \leq \gamma(\epsilon_d + 1) + mMN\epsilon_p$$

where $\gamma = \sum_{n=1}^{N} \|A_n\|_F L_n$.

Proof. By Proposition 1,

$$-\epsilon \leq \hat{z}_{RDM} - z^*_{DM} \leq \sum_{n=1}^{N} \|A_n\|_F L_n \tag{4.8}$$

and by Proposition 2:

$$-\epsilon_d \sum_{n=1}^{N} \|A_n\|_F L_n - mMN\epsilon_p \leq \hat{z}_{RMP} - \hat{z}_{RDM} \leq \epsilon_d \sum_{n=1}^{N} \|A_n\|_F L_n + mMN\epsilon_p \tag{4.9}$$

Letting $\gamma = \sum_{n=1}^{N} \|A_n\|_F L_n$ and adding (4.8) and (4.9), we get

$$\begin{aligned} &-\epsilon - \gamma\epsilon_d - mMN\epsilon_p \leq \hat{z}_{RMP} - z^*_{DM} \leq \gamma(\epsilon_d + 1) + mMN\epsilon_p \\ \Rightarrow &-\epsilon - \gamma\epsilon_d - mMN\epsilon_p \leq \hat{z}_{RMP} - z^*_{MP} \leq \gamma(\epsilon_d + 1) + mMN\epsilon_p \end{aligned}$$

where the second inequalities follow from strong duality, i.e $z^*_{DM} = z^*_{MP}$. □

5 Computational Experiments

In this section, we present preliminary computational experiments where we solve randomly generated instances using the proposed DDW algorithm. The algorithm is implemented in Python using a Message Passing Interface package, mpi4py [9, 8, 7]. Gurobi is used to solve all optimization problems, including the quadratic programs resulting from the augmented Lagrangian ($ARDM_n$), where a barrier method is used. Experiments were run on a 3.00 GHz Amazon server running on Linux, with 36 cores and 2 threads per core, capable of running up to 72 processes in parallel.

5.1 ADMM Parameters

In Section 3, we described the ADMM method and reasonable stopping conditions, but have not discussed how to choose the tolerances and the penalty parameter ρ. In our experiments, we follow the guidelines provided in [4], where we dynamically adjust ρ according to the primal and dual residuals, so that they are a factor of μ away from each other. At the end of iteration k, we update ρ as follows:

$$\rho^{k+1} \leftarrow \begin{cases} \tau^{inc}\rho^k & \text{if } \|r_d\| > \mu\|r_p\| \\ \frac{\rho^k}{\tau^{dec}} & \text{if } \|r_p\| > \mu\|r_d\| \\ \rho^k & \text{otherwise} \end{cases}$$

Intuitively, increasing ρ would put more weight on the terms $\|\pi - \pi_n\|^2$, thus reducing dual feasibility violation, and alternatively reducing primal feasibility when decreasing ρ. In our experiments, we pick $\mu = 100$, $\tau^{inc} = \tau^{dec} = 2$ and $\rho^0 = 100$

We also dynamically adjust the tolerances, as in [13], where we solve the pricing subproblems with inaccurate dual solutions. We first solve DDW with $\epsilon_p = \epsilon_d = 5 \times 10^{-1}$, then divide the

tolerances by 10. We repeat the process until we reach target tolerances $\epsilon'_p = 5 \times 10^{-2}$ and $\epsilon'_d = 5 \times 10^{-4}$, where we solve ADMM and the pricing subproblems one last time. With ADMM being the bottleneck of the algorithm, we reduce the number of times needed to solve ADMM to high accuracy, thus significantly reducing computation time. We note that the threshold used to add a new column depends on the dual tolerance (Lemma 2). This means that when the tolerances are high at the beginning of the algorithm, the requirement to add a new column is harsher. We found it computationally beneficial to be more aggressive by always setting the threshold according to the target tolerance ϵ'_d, which in our case would be $-\max_{i \in I_n^\ell}\{\|A_n^i\|\}10^{-4}$ at outer iteration ℓ, i.e. we do not adjust the threshold according to the current value of ϵ_d. However, doing so can lead to an inefficient algorithm where we keep adding unnecessary columns because the column's true reduced cost might not actually be negative. To this end, we terminate DDW (or divide the tolerances by 10 and repeat) if ADMM terminates after 1 iteration, in which case the dual solutions converged to the same values as in the previous outer iteration.

Finally, at the early stages of the algorithm, the RMP might be infeasible if the starting set of extreme points is too small. We circumvent this by adding upper and lower bounds $-M_n \leq \pi_n \leq M_n$ for each block n to ensure a bounded dual problem. This is equivalent to solving the RMP using a big-M method as discussed in Lemma 1. Computing M exactly however is prohibitive. We circumvent this by having each block n set bounds $-M_n \leq \pi_n \leq M_n$ where $M_n = 10\|c_n\|$.

5.2 Instance Generation

We generate random instances of the form

$$\begin{aligned} \min \quad & \sum_{n=1}^{N} c_n^\top x_n \\ \text{s.t.} \quad & \sum_{n=1}^{N} A_n x_n \geq t \\ & B_n x_n \leq b_n, \ \forall n = 1, ..., N \\ & 0 \leq x_n \leq u_n, \ \forall n = 1, ..., N \end{aligned}$$

where the coefficients of the matrices A_n and B_n are from the discrete uniform distribution $\mathcal{U}\{-10, 20\}$, and the components of the cost vector are from $\mathcal{U}\{-10, 30\}$. Let ℓ_i be the sum of the entries in row i of the linking constraints, i.e $\ell_i = \sum_{n,j}(A_n)_{ij}$, where $(A_n)_{ij}$ is component (i, j) of A_n; similarly let β_i^n be the sum of the entries of row i of B_n. The vectors t and b_n were generated according to the sum of each row of the constraint matrix. We construct component i of t as follows:

$$\begin{cases} t_i \sim \mathcal{U}\{2\ell_i, 3\ell_i\}, & \text{if } \ell_i > 0 \\ t_i \sim \mathcal{U}\{3\ell_i, 2\ell_i\}, & \text{if } \ell_i < 0 \\ t_i = 0, & \text{if } \ell_i = 0 \end{cases}, i = 1, ..., m$$

where m is the number of linking constraints. Similarly, component i of b_n is constructed as

$$\begin{cases} (b_n)_i \sim \mathcal{U}\{2\beta_i^n, 3\beta_i^n\} & \text{if } \beta_i^n > 0 \\ (b_n)_i \sim \mathcal{U}\{3\beta_i^n, 2\beta_i^n\}, & \text{if } \beta_i^n < 0 \\ (b_n)_i = 0, & \text{if } \beta_i^n = 0 \end{cases}, i = 1, ..., m_n$$

where m_n is the number of constraints in block n. Moreover, to ensure a bounded region, we add upper and lower bounds to the variables, where $u_n = 30$ for all n.

5.3 Performance

We perform five sets of experiments, each set involving 100, 1000, 5000, 10000 and 20000 total variables across all blocks. The results are reported in Tables 1-5. For simplicity, each block has the same number of variables. For example, in an experiment with a 1000 variables and 10 blocks, each block has 100 variables. Moreover, the block constraint matrices B_n are square matrices.

For each set of experiment, we vary the number of blocks and the number of linking constraints. We report the optimality gap, computed as $\frac{|\hat{z}_{RMP} - z^*_{MP}|}{|z^*_{MP}|}$, where $\hat{z}_{RMP}$ is the objective value of the RMP evaluated at the recovered primal solution and z^*_{MP} is the optimal objective value of the instance solved using the concurrent algorithm of the Gurobi LP solver where dual simplex, primal simplex and barrier method are run in parallel. The first to finish returns the optimal solution. To compute the feasibility violation, let $\hat{i}$ be the most violated constraint, i.e $\hat{i}$ is the index of the maximum of the vector $t - \sum_{n=1}^{N} A_n \hat{x}_n$, where $\hat{x}_n$ is the recovered approximate solution. The relative feasibility violation is then

$$\frac{\max\{t_{\hat{i}} - \sum_{n,j} (A_n)_{\hat{i}j} (\hat{x}_n)_j, 0\}}{|t_{\hat{i}}|}$$

We report runtimes using our algorithm, Gurobi, and a classical Dantzig-Wolfe (DW) implementation where the master is solved centrally and the subproblems are solved in parallel.

We first note that by solving the RMP in a distributed fashion, we must sacrifice some accuracy, both in terms of optimality and feasibility. Despite this, we obtain high quality solutions across all our experiments, with optimality gaps ranging from the order of $10^{-4}\%$ to $10^{-1}\%$, except for the experiment with 100 variables, 2 blocks and 10 linking constraint, where the optimality gap is slightly higher at about 2 %. Slightly reducing the tolerance led to a gap similar to the other experiments, however. The relative feasibility violation is always very close to 0, ranging from the order of 10^{-7} to 0.

As for runtimes, we note that DDW is slower but close to vanilla DW. The difference in runtimes grows as we increase the number of blocks and linking constraints. By solving the RMP in a distributed fashion, convergence time of ADMM to an approximate solution outweighs the benefits of distributing the computational efforts, even for very large problems. There are two main components slowing down the convergence of ADMM: the number of linking constraints and the number of blocks. The former increases the dimension of the dual vectors which need to reach consensus, whereas the latter increases the number of vectors which need to reach consensus. When the number of blocks and variables are small, the problem is not hard enough for Gurobi. As we increase the number of blocks and variables, Dantzig-Wolfe starts to leverage the structure and outperform Gurobi, and only then can we hope for DDW to outperform Gurobi. However, increasing the number of blocks further starts to hurt DDW as we increase convergence time of ADMM. There is a clear trade-off between leveraging the structure arising from a large number of blocks, and the additional time needed for ADMM to converge.

From our experiments, DDW becomes competitive with Gurobi as we move to problems with 10000 and 20000 variables, and even runs faster in some cases. More notably, in the case of 20000 variables, DDW is up to 2 or 4 times faster for 10 and 20 blocks. However, as we move to 100 blocks, DDW is only faster when there is only 1 linking constraint, and falls behind as we increase the latter.

Although we are more focused on showing the validity of the algorithm in this work, we note that significantly increasing the difficulty of the master problem would show benefits of using DDW over Gurobi, as already noted in some of our instances, and potentially even over a typical DW implementation.

Table 1: 100 Variables

N	m	Optimality Gap	Relative Feasibility Violation	DDW Time (sec)	Gurobi Time (sec)	DWD Time (sec)
2	1	3.227×10^{-5}	0	0.14	0.002	0.17
	2	4.025×10^{-4}	4.403×10^{-5}	0.09	0.003	0.05
	5	3.743×10^{-4}	2.782×10^{-5}	0.24	0.003	0.05
	10	2.131×10^{-2}	2.571×10^{-5}	0.85	0.003	0.09
4	1	9.056×10^{-6}	8.161×10^{-5}	0.10	0.002	0.04
	2	2.091×10^{-4}	7.483×10^{-5}	0.11	0.002	0.04
	5	6.692×10^{-4}	7.839×10^{-5}	0.29	0.002	0.05
	10	5.794×10^{-3}	2.609×10^{-5}	0.88	0.003	0.05
5	1	2.516×10^{-4}	1.288×10^{-4}	0.12	0.002	0.04
	2	2.039×10^{-4}	7.817×10^{-5}	0.16	0.002	0.04
	5	9.216×10^{-5}	2.083×10^{-4}	0.26	0.002	0.04
	10	1.600×10^{-4}	8.923×10^{-5}	0.75	0.003	0.05
10	1	1.443×10^{-4}	3.296×10^{-4}	0.09	0.001	0.04
	2	2.626×10^{-3}	2.563×10^{-4}	2.00	0.001	0.04
	5	6.801×10^{-4}	5.104×10^{-4}	0.58	0.002	0.04
	10	1.546×10^{-3}	1.801×10^{-4}	0.58	0.002	0.04

Table 2: 1000 Variables

N	m	Optimality Gap	Relative Feasibility Violation	DDW Time (sec)	Gurobi Time (sec)	DWD Time (sec)
5	1	4.292×10^{-4}	4.965×10^{-6}	0.28	0.09	0.13
	2	2.778×10^{-4}	6.966×10^{-6}	0.50	0.09	0.17
	5	5.123×10^{-4}	6.482×10^{-6}	1.70	0.17	0.32
	10	1.912×10^{-3}	4.391×10^{-6}	2.50	0.24	0.48
10	1	1.982×10^{-4}	0	0.21	0.05	0.06
	2	1.576×10^{-4}	1.314×10^{-5}	0.43	0.09	0.11
	5	2.741×10^{-4}	3.613×10^{-6}	1.10	0.09	0.12
	10	1.132×10^{-3}	7.307×10^{-6}	2.20	0.14	0.16
20	1	1.492×10^{-5}	0	0.18	0.03	0.05
	2	2.321×10^{-4}	6.546×10^{-6}	0.60	0.04	0.06
	5	4.598×10^{-4}	4.879×10^{-5}	1.00	0.06	0.07
	10	2.016×10^{-3}	1.626×10^{-5}	1.70	0.09	0.08

Table 3: 5000 Variables

N	m	Optimality Gap	Relative Feasibility Violation	DDW Time (sec)	Gurobi Time (sec)	DWD Time (sec)
5	1	1.679×10^{-4}	9.966×10^{-7}	5.4	4.4	4.6
	2	2.169×10^{-4}	1.711×10^{-6}	7.6	5.5	6.5
	5	9.624×10^{-4}	2.687×10^{-6}	14.7	9.9	13.3
	10	1.327×10^{-3}	2.007×10^{-7}	36.3	11.3	26.5
10	1	1.394×10^{-4}	5.154×10^{-6}	1.4	1.5	0.9
	2	3.840×10^{-4}	2.198×10^{-6}	2.1	3.1	1.3
	5	4.448×10^{-4}	7.318×10^{-7}	5.3	4.0	2.2
	10	8.568×10^{-4}	1.463×10^{-6}	8.3	4.8	3.1
20	1	2.362×10^{-4}	0	0.8	1.1	0.3
	2	3.444×10^{-4}	0	1.2	1.2	0.4
	5	4.717×10^{-4}	1.815×10^{-6}	1.2	1.7	0.6
	10	8.526×10^{-4}	1.824×10^{-6}	5.4	1.7	0.8
50	1	2.984×10^{-5}	0	1.5	0.4	0.1
	2	1.237×10^{-4}	3.818×10^{-6}	1.9	0.5	0.2
	5	4.968×10^{-4}	1.951×10^{-5}	4.8	0.6	0.3
	10	3.904×10^{-4}	2.535×10^{-6}	9.5	0.6	0.3
100	1	2.232×10^{-4}	0	4.4	0.3	0.3
	2	8.245×10^{-5}	0	4.1	0.3	0.3
	5	3.008×10^{-4}	8.019×10^{-6}	8.4	0.3	0.3
	10	9.736×10^{-4}	3.971×10^{-6}	13.4	0.4	0.5

Table 4: 10000 Variables

N	m	Optimality Gap	Relative Feasibility Violation	DDW Time (sec)	Gurobi Time (sec)	DWD Time (sec)
5	1	2.748×10^{-4}	1.816×10^{-6}	28.8	16.7	26.2
	2	2.625×10^{-4}	5.841×10^{-7}	41.9	26.0	33.5
	5	7.315×10^{-4}	3.561×10^{-7}	80.2	33.1	79.9
	10	2.337×10^{-3}	1.329×10^{-6}	179.0	73.2	158.0
10	1	1.071×10^{-4}	4.131×10^{-6}	6.4	10.0	5.8
	2	3.119×10^{-4}	4.173×10^{-6}	7.5	10.6	6.6
	5	1.004×10^{-3}	4.080×10^{-7}	16.2	20.8	12.3
	10	1.177×10^{-3}	6.497×10^{-7}	35.2	29.3	21.4
20	1	1.966×10^{-4}	9.892×10^{-7}	1.8	6.8	1.3
	2	2.888×10^{-4}	0	3.0	8.4	1.8
	5	5.952×10^{-4}	1.850×10^{-6}	7.2	9.2	2.7
	10	7.540×10^{-4}	2.432×10^{-6}	11.7	9.3	3.2
100	1	1.895×10^{-5}	0	2.5	1.1	0.4
	2	2.495×10^{-4}	0	8.1	1.2	0.8
	5	1.964×10^{-4}	1.960×10^{-6}	15.1	1.2	0.9
	10	8.860×10^{-4}	8.547×10^{-7}	31.9	1.4	1.1

Table 5: 20000 Variables

N	m	Optimality Gap	Relative Feasibility Violation	DDW Time (sec)	Gurobi Time (sec)	DWD Time (sec)
5	1	5.551×10^{-5}	0	188.0	104.5	157.0
	2	2.177×10^{-4}	1.194×10^{-7}	231.0	201.1	206.0
	5	6.603×10^{-4}	6.062×10^{-7}	518.6	535.1	482.5
	10	1.485×10^{-3}	4.097×10^{-7}	1092.0	528.1	1030.0
10	1	1.220×10^{-4}	1.136×10^{-6}	35.9	78.1	30.0
	2	3.956×10^{-4}	5.940×10^{-7}	43.4	56.7	39.1
	5	5.093×10^{-4}	1.428×10^{-7}	95.5	206.4	75.5
	10	1.828×10^{-3}	9.904×10^{-8}	176.5	207.5	145.0
20	1	1.452×10^{-4}	0	9.4	42.4	6.6
	2	7.122×10^{-4}	0	11.6	40.3	8.7
	5	5.102×10^{-4}	1.662×10^{-6}	23.0	57.8	14.7
	10	1.187×10^{-3}	1.492×10^{-7}	38.3	60.1	23.9
100	1	1.320×10^{-4}	6.012×10^{-6}	3.6	5.0	0.9
	2	1.015×10^{-4}	3.268×10^{-6}	9.8	4.9	1.6
	5	4.490×10^{-4}	2.281×10^{-6}	18.5	5.4	1.8
	10	7.416×10^{-4}	1.044×10^{-6}	32.6	6.0	2.4

5.4 Parallel Efficiency and Scalability

To measure how well the algorithm and our implementation scale as we increase the number of blocks and available cores, we use two common metrics as in [19]. The first one measures the speedup gained by using the available cores. The second metric measures core utilization and time lost in communication and synchronization. We compute the two metrics for instances with 9000, 18000 and 36000 total variables. For each set of instances, we experiment with 5, 10, 20, 36 and 72 blocks. As before, each block contains the same number of variables.

Parallel Speedup Let t_p be the time it takes for DDW to terminate using p cores. We compute the ratio $\frac{t_1}{t_p}$ for each experiment and report results in figure 1. We observe similar trends for different number of total variables. The computational gain from parallelizing decreases as we increase the number of blocks. This is mainly due to cores sitting idle, waiting on other processes to finish, as well as communication overhead increasing with the number of cores used. This is confirmed by our analysis on core utilization.

Core Utilization To estimate core utilization, we measure total time spent doing useful computations, communication time, and synchronization time where a core is sitting idle waiting on others to finish their computations. For each core, if we define these three values as T_u, T_c and T_s, respectively, then core utilization can be estimated as $\frac{T_u}{T_u+T_c+T_s}$ [19]. Figure 2 reports average core utilization for each instance. We again see diminishing returns where average utilization decreases as the number of blocks and cores used increases. However, it seems that the average utilization is slightly better as we increase the number of total variables. Figures 1 and 2 have the main objective of showcasing the slowdowns that can happen by idle cores, indicating potential benefits in an asynchronous implementation of our algorithm.

Figure 1: Ratio of runtimes between serial and parallel implementations

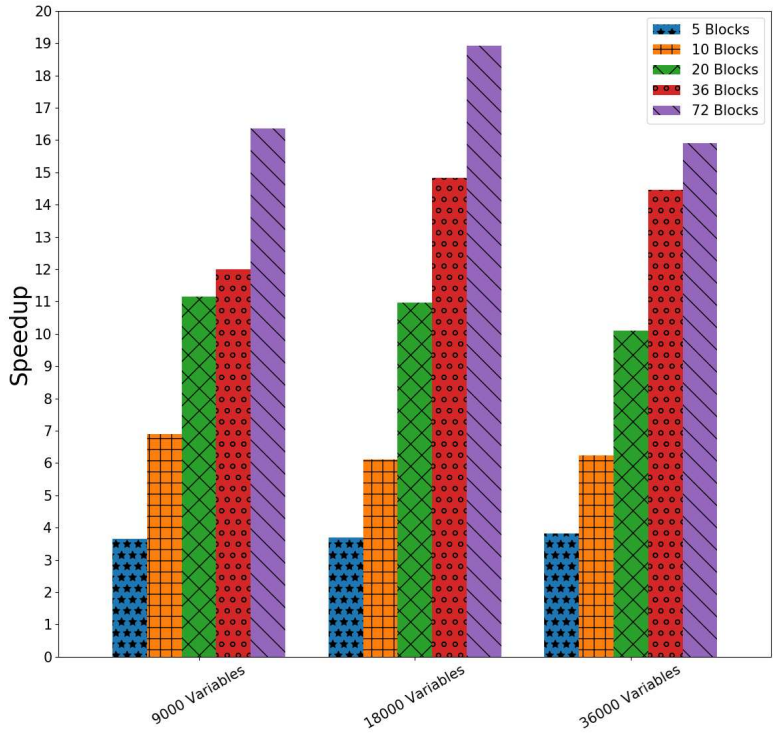
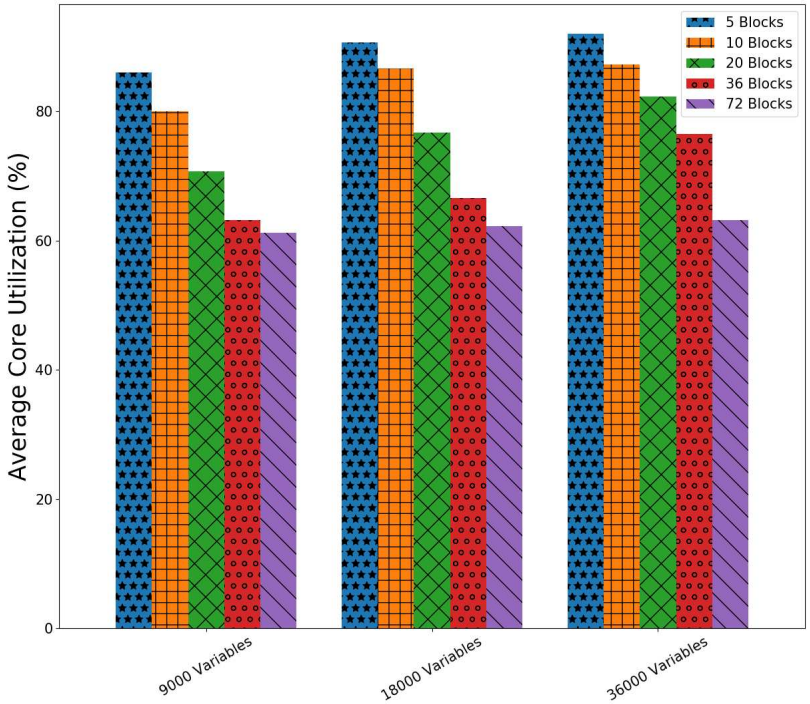


Figure 2: Average core utilization



6 Conclusion

In this paper, we proposed a fully distributed Dantzig-Wolfe decomposition algorithm for loosely coupled large-scale linear programs. As opposed to the standard Dantzig-Wolfe algorith, we solve the master problem using consensus-based ADMM, thus preserving privacy of information of the agents. We proved convergence of the algorithm and provided error bounds on the feasibility and optimality gaps. We illustrated our method using an MPI implementation on randomly generated problems and showed that we are able to achieve high accuracy in reasonable time. We note that it is possible to use other algorithms or a more sophisticated versions of ADMM to solve the consensus problem. As the difficulty and size of the problems for each block increases, the cost per iteration of ADMM can become prohibitive. Certain workarounds involve linearizing the objective of the augmented Lagrangian, yielding computational benefits [4]. Other interesting consensus algorithms include a distributed interior point method which might converge faster than first-order distributed methods [3]. Finally, as suggested by our experiments, an asynchronous implementation of DDW has the potential to improve computation times.

Acknowledgment

This work has been supported in part by the Office of Naval Research Award N000141812075.

References

[1] A. Beck. *First-Order Methods in Optimization*. SIAM, Philadelphia, PA, 2017.

[2] D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.

[3] A. Bitlislioğlu, I. Pejcic, and C. Jones. Interior point decomposition for multi-agent optimization. *IFAC-PapersOnLine*, 50(1):233–238, 2017.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[5] X. Cai, D. Han, and X. Yuan. On the convergence of the direct extension of ADMM for three-block separable convex minimization models with one strongly convex function. *Computational Optimization and Applications*, 66(1):39–73, 2017.

[6] C. Chen, B. He, Y. Ye, and X. Yuan. The direct extension of admm for multi-block convex minimization problems is not necessarily convergent. *Mathematical Programming*, 155(1-2):57–79, 2016.

[7] L. Dalcín, R. Paz, and M. Storti. MPI for python. *Journal of Parallel and Distributed Computing*, 65(9):1108–1115, 2005.

[8] L. Dalcın, R. Paz, M. Storti, and J. D'Elıa. MPI for python: Performance improvements and MPI-2 extensions. *Journal of Parallel and Distributed Computing*, 68(5):655–662, 2008.

[9] L. D. Dalcin, R. R. Paz, P. A. Kler, and A. Cosimo. Parallel distributed computing using python. *Advances in Water Resources*, 34(9):1124–1139, 2011.

[10] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960.

[11] G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors. *Column generation*. Springer, 2005.

[12] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson. Optimal parameter selection for the alternating direction method of multipliers (admm): quadratic problems. *IEEE Transactions on Automatic Control*, 60(3):644–658, 2015.

[13] J. Gondzio, P. Gonzalez-Brevis, and P. Munari. New developments in the primal-dual column generation technique. *European Journal of Operational Research*, 224(1):41–51, 2012.

[14] J. Gondzio, P. Gonzalez-Brevis, and P. Munari. Large-scale optimization with the primal-dual column generation method. *Mathematical Programming Computation*, 8(1):47–82, 2016.

[15] J. Gondzio and R. Sarkissian. Column generation with a primal-dual method. *Logilab Technical Report 96.6, University of Geneva*, 1996.

[16] M. Hong and Z.-Q. Luo. On the linear convergence of the alternating direction method of multipliers. *Mathematical Programming*, 162(1-2):165–199, 2017.

[17] M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

[18] J. F. Mota, J. M. Xavier, P. M. Aguiar, and M. Püschel. Distributed optimization with local domains: Applications in mpc and network flows. *IEEE Transactions on Automatic Control*, 60(7):2004–2009, 2015.

[19] L.-M. Munguía, S. Ahmed, D. A. Bader, G. L. Nemhauser, and Y. Shao. Alternating criteria search: a parallel large neighborhood search algorithm for mixed integer programs. *Computational Optimization and Applications*, 69(1):1–24, 2018.

[20] A. Nedic and A. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

[21] A. Nedic and A. Ozdaglar. Cooperative distributed multi-agent optimization. In D. P. Palomar and Y. C. Eldar, editors, *Convex optimization in signal processing and communications*. Cambridge university press, 2010.

[22] A. Nedic, A. Ozdaglar, and P. A. Parrilo. Constrained consensus and optimization in multi-agent networks. *IEEE Transactions on Automatic Control*, 55(4):922–938, 2010.

[23] G. L. Nemhauser and L. Wolsey. *Integer Programming and Combinatorial Optimization*. Wiley, 1988.

[24] R. Nishihara, L. Lessard, B. Recht, A. Packard, and M. I. Jordan. A general analysis of the convergence of ADMM. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 343–352, 2015.

[25] O. Briant, C. Lemaréchal, P. Meurdesoif, S. Michel, N. Perrot, and F. Vanderbeck. Comparison of bundle and classical column generation. *Mathematical Programming*, 113(2):299–344, 2006.

[26] B. Wohlberg. Admm penalty parameter selection by residual balancing. *arXiv preprint arXiv:1704.06209*, 2017.

[27] C. Xi, Q. Wu, and U. A. Khan. On the distributed optimization over directed networks. *Neurocomputing*, 267:508–515, 2017.