A Distributed Primal-dual Interior-point Method for Loosely Coupled Problems Using ADMM

Mariette Annergren* Sina Khoshfetrat Pakazad**
Anders Hansson** Bo Wahlberg*

* Automatic Control Lab and ACCESS, School of Electrical Engineering, KTH, Stockholm, Sweden. Email:{mariette.annergren, bo.wahlberg}@ee.kth.se
** Division of Automatic Control, Department of Electrical Engineering, Linköping University, Sweden. Email: {sina.kh.pa, hansson}@isy.liu.se.

Abstract: In this paper we propose an efficient distributed algorithm for solving loosely coupled convex optimization problems. The algorithm is based on a primal-dual interior-point method in which we use the alternating direction method of multipliers (ADMM) to compute the primal-dual directions at each iteration of the method. This enables us to join the exceptional convergence properties of primal-dual interior-point methods with the remarkable parallelizability of ADMM, and hence we expect the resulting algorithm to have superior convergence properties with respect to other existing distributed optimization algorithms. Furthermore, the amount of computations that needs to be conducted by each computing agent is far less than other existing distributed algorithms for solving constrained optimization problems. In particular, the updates for all variables can be given in closed form, irrespective of the type of optimization problem.

1. INTRODUCTION

Centralized algorithms for solving optimization problems can become unviable due to lack of powerful enough centralized computational units, or because the problem cannot be formed as a centralized optimization problem due to structural constraints in the problem, such as privacy requirements. A sensible approach for solving such problems is to use distributed optimization algorithms, which rely on collaboration of multiple computing agents to solve the problem. In such a setting, each agent is assigned a local subproblem, and at every iteration it is required to solve the subproblem and communicate/collaborate with certain other agents. This is done repeatedly until the network of agents arrives/agrees on a solution. Distributed optimization methods have been studied for many years, and there are different approaches for devising such algorithms, see e.g. Bertsekas and Tsitsiklis (1997); Eckstein (1989); Boyd et al. (2011); Nedic and Ozdaglar (2009); Nedic et al. (2010).

One of the most common approaches for designing distributed algorithms is to apply first order/proximal point methods directly to the problem or some reformulation of it. In this class of distributed algorithms, the ones based on subgradient/gradient methods are perhaps among the simplest, see e.g. Nedic and Ozdaglar (2009); Nedic et al. (2010). Although the local computations that need to be performed by each agent are usually elementary, these algorithms are very sensitive to the scaling of the problem. They also generally require many iterations to converge to a solution with even medium accuracy (Bertsekas and Tsitsiklis, 1997). In order to alleviate these issues there has been a surge of interest to devise distributed algorithms based on proximal point methods, e.g. see Bertsekas and Tsitsiklis (1997); Eckstein (1989); Boyd et al. (2011); Combettes and Pesquet (2011). For certain classes of problems, for instance when the objective function of the equivalent unconstrained reformulation of the problem has two terms and/or is strongly convex, such algorithms commonly enjoy better convergence properties (Goldfarb et al., 2012; Goldstein et al., 2012) and are less sensitive to the scaling of the problem. However, they are generally more complicated in that the local computational burden is higher, and the communication protocols are more sophisticated, see e.g. Summers and Lygeros (2012); Olisson et al. (2013). Moreover, extra care must be taken if one wishes to apply proximal point methods to more general classes of problems, as these algorithms might even diverge, see e.g. Chen et al. (2013). There have been suggestions on how to modify these methods to allow application to more general problems; however, the resulting algorithms can become overly complicated to implement, particularly distributedly (Goldfarb et al., 2012; Han and Yuan, 2012; Hong and Luo, 2012).

Another approach for designing distributed optimization algorithms is to use second order methods, e.g. see Chu et al. (2011); Wei et al. (2013); Necoara and Suykens (2009). For instance, in Necoara and Suykens (2009) the authors propose a distributed optimization method based on an interior-point method. The introduced algorithm is obtained by first performing a Lagrangian decomposition of the problem and then efficiently solving the subproblems
using interior-point methods. However, in the proposed algorithm, the computational cost for solving the subproblems can still be considerable. Chu et al. (2011) propose a distributed Newton method for solving coupled unconstrained quadratic problems, which is used for anomaly detection in large populations. This distributed method is only applicable to unconstrained quadratic problems. Wei et al. (2013) propose a distributed Newton method for solving a network utility maximization problem. The cost function for such problems is given by a summation of several terms where each term depends on a single scalar variable. This structure allows the authors to employ a matrix splitting method which in turn enables them to compute the inexact Newton directions distributedly. However, this method relies on the special structure in the considered problem and hence can only be used in particular cases.

The approach presented in the latter paper falls within the class of inexact interior-point methods which have been studied thoroughly over the past two decades, e.g. see Bellavia (1998); Freund et al. (1999); Mizuno and Jarre (1999); Hansson (2000); Korzak (2000); Durazzi and Ruggiero (2003); Bellavia and Pieraccini (2004); Zhou and Toh (2004); Bonettini et al. (2005); Bonettini and Ruggiero (2007); Toh (2008); Lu et al. (2009); Al-Jeiroudi and Gondzio (2009); Curtis et al. (2010, 2012). These methods combine primal or primal-dual interior point methods with iterative algorithms for solving linear systems of equations. This is motivated by the fact that we need to solve a linear system of equations in every iteration of a primal or primal-dual interior-point method, in order to compute primal or primal-dual directions. These methods provide bounds on the required accuracy of the computed directions at each iteration in order to guarantee convergence. Freund et al. (1999) and Mizuno and Jarre (1999) consider linear programs (LPs) and focus on the design of these accuracy bounds. In particular, they provide bounds on primal and dual residuals and computed directions to assure convergence of their respective proposed inexact interior-point method. LPs are also considered in Korzak (2000) where the author proposes an inexact interior-point method with quasi-minimal residual (QMR) technique and conjugate gradient (CG) as inexact solvers of choice. Also in Al-Jeiroudi and Gondzio (2009), the authors consider LPs and they focus on devising efficient pre-conditioners for CG (PCG) algorithms for solving the underlying linear equations more efficiently. An inexact primal-dual method for solving robust optimal control problems is proposed in Hansson (2000) with QMR as the iterative solver of choice. Bellavia and Pieraccini (2004) and Zhou and Toh (2004) consider semidefinite programs and propose inexact primal-dual interior point methods for solving the problem. The inexact solvers in these papers were PCG for which they both propose efficient pre-conditioners to improve the convergence properties. They also propose similar accuracy bounds on the computed directions that depend solely on the so-called complementarity gap. In Toh (2008) a quadratic semi-definite program is considered where the author uses a pre-conditioned QMR algorithm and proposes efficient pre-conditioners for further improvement of its convergence rate. Inexact interior-point methods have also been used for solving constrained nonlinear systems of equations, which can be considered as Karush-Kuhn-Tucker (KKT) optimality conditions for general optimization problems (not necessarily convex). For instance Bellavia (1998) proposes an inexact interior-point method for solving constrained nonlinear monotone systems of equations, under the assumption that the Jacobian of the system of equations is invertible at the solution. Durazzi and Ruggiero (2003) put forth a similar framework for solving general constrained nonlinear systems of equations and they use the PCG algorithm for solving them with respect to primal-dual directions. Bonettini et al. (2005) propose an inexact interior-point method for solving constrained nonlinear system of equations which uses the so-called Hestenes’ multipliers method for solving the underlying linear systems of equations. The authors further investigate the numerical properties of the proposed method and compare with the case when they use PCG as the iterative solver of choice in Bonettini and Ruggiero (2007).

Notice that design of distributed algorithms for solving optimization problems was not the focus of any of the works discussed in the previous paragraph. In this paper, we focus on devising a distributed optimization algorithm based on a primal-dual interior-point method for solving loosely coupled optimization problems. These constitute a more general class of problems than those considered by Chu et al. (2011); Wei et al. (2013); Necoara and Suykens (2009). To this end, we first exploit the coupling in the problem using consistency constraints and use proximal splitting methods, particularly alternating direction method of multipliers (ADMM), to compute the primal-dual directions in a distributed manner.

ADMM is a method of finding saddle points of an augmented Lagrangian and, as such, a method of finding a solution of an optimization problem (Gabay and Mercier, 1976). In our approach, we use ADMM to solve the KKT conditions of a particular optimization problem that has the primal-dual directions as solution, see Section 3. The benefits of using ADMM are several. The ADMM iterations

- converge to a solution under mild assumptions (Boyd et al., 2011).
- enable the solution to be calculated in a highly distributed way, see Section 4.
- consist of subproblems that are extremely cheap to solve, see Section 4.

ADMM was first introduced by Glowinski and Marrocco (1975) for solving nonlinear Dirichlet problems. It was presented as a modified version of Uzawa's algorithm (Arrow, 1964). The method was developed further by Gabay and Mercier (1976), who stated some convergence properties. In Gabay (1983), it was shown that ADMM is equivalent to Douglas-Rachford splitting for monotone operators (Douglas and Rachford, 1956) and similar to Peaceman-Rachford splitting (Peaceman and Rachford, 1955). ADMM is related to the method of multipliers, also known as Hestenes’ multipliers method, (Hestenes, 1969; Powell, 1969), and the proximal point algorithm (Eckstein and Bertsekas, 1992). For a detailed overview of ADMM and other related methods, see Boyd et al. (2011).

We expect our proposed distributed optimization algorithm to present superior computational and convergence
properties than other distributed solvers, and we believe that the key to achieving this has been the use of ADMM for computing the primal-dual directions. We are not aware of any other iterative solvers that would present the same characteristics as listed above. We illustrate the performance of the proposed algorithm using a numerical experiment.

Outline

First we define the notation, and in Section 2 we explain the considered problem formulation and the structure of the loose coupling. In Section 3 and Section 4, we briefly describe the primal-dual interior-point method and how it can be applied in a distributed fashion for loosely coupled problems using ADMM. We illustrate the method on a numerical example in Section 5. Some conclusions and future work are stated in Section 6. In the Appendix, we have derived the explicit relations between ADMM and Uzawa’s method, and ADMM and fixed point iterations for the considered problem formulation.

Notation

The set of real numbers is denoted by \( \mathbb{R} \). The set of real \( n \)-dimensional vectors and \( n \times m \) matrices are symbolized by \( \mathbb{R}^n \) and \( \mathbb{R}^{n \times m} \), respectively, and the transpose of a matrix \( A \) is denoted by \( A^T \). Let \( \mathbb{N}_p \) represent the ordered set of positive integers \( \{1, 2, \ldots, p\} \). Given a set \( J \subseteq \{1, 2, \ldots, n\} \), the matrix \( E_J \in \mathbb{R}^{|J| \times n} \) is the matrix obtained by deleting the rows indexed by \( \mathbb{N}_n \setminus J \) from an identity matrix of order \( n \), where \( |J| \) denotes the number of elements in set \( J \). Consequently, \( E_J x \) is a \( |J| \)-dimensional vector with the components of \( x \) that correspond to the elements in \( J \), and we denote this vector \( x_J \). We denote by \( x_i^{(k)} \) the \( k \)th element of vector \( x_i \) at the \( k \)th iteration. Given vectors \( x^i \) and matrices \( A^i \) for \( i = 1, \ldots, N \), the column vector \( (x^1, \ldots, x^N) \) is all of the given vectors stacked and \( \text{blkdiag}(A^1, \ldots, A^N) \) represents a block-diagonal matrix with \( A^i \) as its diagonal blocks. Similarly, given a vector \( x \in \mathbb{R}^n \), \( \text{diag}(x_1, \ldots, x_n) \) denotes a diagonal matrix with its diagonals expressed by elements of \( x \). The minimum value of a set \( S \) or of a function is denoted by “min” and the minimizing argument of an optimization problem in denoted by “argmin”. The inequality \( x \leq y \), where \( x, y \in \mathbb{R}^n \), means \( x_i \leq y_i \) for \( i = 1, \ldots, n \).

2. LOOSELY COUPLED PROBLEMS

In this paper, we are interested in optimization problems of the form

\[
\begin{align*}
\text{minimize} & \quad f_1(x) + \cdots + f_N(x), \\
\text{subject to} & \quad G^i(x) \geq 0, \quad i = 1, \ldots, N, \\
& \quad A^i x = b^i, \quad i = 1, \ldots, N,
\end{align*}
\]

where \( f_i : \mathbb{R}^n \to \mathbb{R}, G^i : \mathbb{R}^n \to \mathbb{R}^{m_i} \) and \( A^i \in \mathbb{R}^{n \times n} \) with \( p_i < n \) and \( \text{rank}(A^i) = p_i \) for all \( i = 1, \ldots, N \). We assume that the function pairs \( f_i, G^i \) and their corresponding \( A^i \) for \( i = 1, \ldots, N \), depend only on a small subset of the elements of the variable \( x \) and we denote the ordered set of the indices of these variables by \( J_i \). We also denote the ordered set of indices of triplets \( f_i, G^i, A^i \) that depend on \( x_i \) by \( I_i \), i.e., \( I_i = \{k \mid i \in J_i\} \). We call an optimization problem loosely coupled if \( |I_i| \ll N \) for all \( i = 1, \ldots, n \).

We can explicitly express the coupling structure in (1) using the so-called consistency/consensus constraints. The problem in (1) can then be equivalently written as

\[
\begin{align*}
\text{minimize} & \quad \tilde{f}_i(s^i) + \cdots + \tilde{f}_N(s^N), \\
\text{subject to} & \quad \tilde{G}^i(s^i) \preceq 0, \quad i = 1, \ldots, N, \\
& \quad \tilde{A}^i s^i = b^i, \quad i = 1, \ldots, N, \\
& \quad \tilde{E} x = S,
\end{align*}
\]

where \( S = (s^1, \ldots, s^N) \) and \( \tilde{E} = [E_{J_1}^T, \ldots, E_{J_m}^T]^T \) with \( E_{J_i} \) as a 0–1 matrix that is obtained from an identity matrix of order \( n \) by deleting the rows indexed by \( \mathbb{N}_n \setminus J_i \). We refer to the constraints in (2d) as consistency constraints. The functions \( \tilde{f}_i : \mathbb{R}^{|I_i|} \to \mathbb{R} \) are lower dimensional descriptions of the \( f_i \) such that \( f_i(x) = \tilde{f}_i(E_{J_i} x) \) for all \( x \in \mathbb{R}^n \) and \( i = 1, \ldots, N \). In this formulation, the functions \( \tilde{G}^i : \mathbb{R}^{|I_i|} \to \mathbb{R}^{m_i} \) are defined in the same manner as the functions \( f_i \), and the matrices \( \tilde{A}^i \in \mathbb{R}^{|I_i| \times |I_i|} \) are defined by removing unnecessary columns from \( A^i \). We further assume that \( p_i < |I_i| \) and that \( \text{rank}(\tilde{A}^i) = p_i \) for all \( i = 1, \ldots, N \). In this paper, we intend to devise algorithms to solve problems of the form in (1) or (2) in a distributed manner, and we will investigate the possibility of using primal-dual interior-point methods for this purpose. The latter are briefly reviewed in the next section.

3. PRIMAL-DUAL INTERIOR-POINT METHODS

Let us consider the convex optimization problem

\[
\begin{align*}
\text{minimize} & \quad F(x), \\
\text{subject to} & \quad g_i(x) \leq 0, \quad i = 1, \ldots, m, \\
& \quad Ax = b,
\end{align*}
\]

where \( F : \mathbb{R}^n \to \mathbb{R}, g_i : \mathbb{R}^n \to \mathbb{R} \) and \( A \in \mathbb{R}^{p \times n} \) with \( p < n \) and \( \text{rank}(A) = p \). The KKT optimality conditions for this problem can be written as

\[
\begin{align*}
\nabla F(x) + \sum_{i=1}^m \lambda_i \nabla g_i(x) + A^T v &= 0, \\
\lambda_i &\geq 0, \quad i = 1, \ldots, m, \\
g_i(x) &\leq 0, \quad i = 1, \ldots, m, \\
-\lambda_i g_i(x) &= 0, \quad i = 1, \ldots, m, \\
Ax &= b.
\end{align*}
\]

Primal-dual methods solve the problem in (3) by dealing with a sequence of modified versions of the optimality conditions in (4) where we perturb (4d) as

\[\lambda_i g_i(x) = 1/t\]

with \( t > 0 \). Next we briefly review how this is done within a primal-dual framework.

3.1 A Primal-dual Interior-point Method

First notice that for any \( \lambda \) and \( x \) that satisfy

\[
\begin{align*}
\nabla F(x) + \sum_{i=1}^m \lambda_i \nabla g_i(x) + A^T v &= 0, \\
-\lambda_i g_i(x) &= 1/t, \quad i = 1, \ldots, m, \\
Ax &= b.
\end{align*}
\]

we have \( \lambda \geq 0 \) and \( g_i(x) < 0 \) for all \( i = 1, \ldots, m \). Hence, the conditions in (5) are equivalent to the KKT conditions
for the problem in (3), given in (4) with the modification to (4d). Within a primal-dual framework, given primal and dual iterates \( x^{(l)} \), \( \lambda^{(l)} \) and \( v^{(l)} \), at iteration \( l \), such that \( g_i(x^{(l)}) \leq 0 \) and \( \lambda_i^{(l)} \geq 0 \) for all \( i = 1, \ldots, m \), we first linearize (5) at these iterates which results in
\[
\left( \nabla^2 F(x^{(l)}) + \sum_{i=1}^{m} \lambda_i^{(l)} \nabla^2 g_i(x^{(l)}) \right) \Delta x + \sum_{i=1}^{m} \nabla g_i(x^{(l)}) \Delta \lambda_i + A^T \Delta v = -r^{(l)}_{\text{dual}},
\]
\[
-\lambda_i^{(l)} \nabla g_i(x^{(l)})^T \Delta x - g_i(x^{(l)}) \Delta \lambda_i = - \left( r^{(l)}_{\text{cent}} \right), \quad i = 1, \ldots, m,
\]
\[
A \Delta x = -r^{(l)}_{\text{primal}},
\]
where
\[
r^{(l)}_{\text{dual}} = \nabla F(x^{(l)}) + \sum_{i=1}^{m} \lambda_i^{(l)} \nabla g_i(x^{(l)}) + A^T v^{(l)},
\]
\[
\left( r^{(l)}_{\text{cent}} \right) = -\lambda_i^{(l)} g_i(x^{(l)})^T - 1/t, \quad i = 1, \ldots, m,
\]
\[
r^{(l)}_{\text{primal}} = A x^{(l)} - b.
\]

The linearized KKT conditions in (6) can also be written in a compact form as
\[
\begin{bmatrix}
\nabla^2 F(x^{(l)}) + \sum_{i=1}^{m} \lambda_i^{(l)} \nabla^2 g_i(x^{(l)}) - D g(x^{(l)})^T A^T \Delta x \\
- \text{diag}(\lambda^{(l)})D g(x^{(l)}) - G_d(x^{(l)}) 0
\end{bmatrix}
\begin{bmatrix}
\Delta x \\
\Delta \lambda
\end{bmatrix}
= \begin{bmatrix}
- \sum_{i=1}^{m} \lambda_i^{(l)} \nabla g_i(x^{(l)})^T \lambda_i^{(l)} g_i(x^{(l)})^T \\
0
\end{bmatrix} - \begin{bmatrix}
r^{(l)}_{\text{dual}} \\
r^{(l)}_{\text{cent}} \\
r^{(l)}_{\text{primal}}
\end{bmatrix},
\]

where \( D g(x) = \nabla g_1(x) \ldots \nabla g_m(x)^T \) and \( G_d(x) = \text{diag}(g_1(x), \ldots, g_m(x)) \). Having linearized (5), we then solve (8) for the primal-dual directions needed to update the primal and dual variables. One way to do so is by first eliminating \( \Delta \lambda \) as
\[
\Delta \lambda = -G_d(x^{(l)})^{-1} \left( \text{diag}(\lambda^{(l)})D g(x^{(l)}) \Delta x - x^{(l)}_{\text{cent}} \right).
\]

We can then rewrite (8) as
\[
\begin{bmatrix}
H^{(l)}_{\text{pd}} & A^T \\
A & 0
\end{bmatrix}
\begin{bmatrix}
\Delta x \\
\Delta v
\end{bmatrix}
= \begin{bmatrix}
r^{(l)}_{\text{dual}} \\
r^{(l)}_{\text{cent}} \\
r^{(l)}_{\text{primal}}
\end{bmatrix},
\]

where
\[
H^{(l)}_{\text{pd}} = \nabla^2 F(x^{(l)}) + \sum_{i=1}^{m} \lambda_i^{(l)} \nabla^2 g_i(x^{(l)}) - \sum_{i=1}^{m} g_i(x^{(l)}) \nabla g_i(x^{(l)})^T \lambda_i^{(l)} g_i(x^{(l)})^T,
\]
and
\[
r^{(l)} = r^{(l)}_{\text{dual}} + D g(x^{(l)})^T G_d^{-1} x^{(l)}_{\text{cent}}.
\]

Notice that the set of equations in (10) expresses the optimality conditions for the quadratic program
\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \Delta x^T H^{(l)}_{\text{pd}} \Delta x + (r^{(l)})^T \Delta x, \\
\text{subject to} & \quad A \Delta x = -r^{(l)}_{\text{primal}}.
\end{align*}
\]
Hence \( \Delta x \) and \( \Delta v \) can be computed through solving (11), and \( \Delta \lambda \) using (9). Having expressed a way to compute the primal-dual directions, we layout a primal-dual interior-point method in Algorithm 1. This algorithm is later used to solve loosely coupled problems.

**Remark 1.** Notice that we will not use (8) for computing the primal-dual directions. This is because the coefficient matrix in (8) is not symmetric and this will limit our capability to solve (8) efficiently. Instead we focus on the linear system of equations in (10), that is sometimes referred to as the augmented system. As we will see later, the structure in (10) or equivalently in (11) enables us to distribute the computations of primal-dual directions. Another approach to computing the primal-dual directions eliminates \( \Delta \lambda \) and then solves a linear set of equations (referred to as normal equations) for computing \( \Delta \lambda \). This, however, generally destroys the inherent structure in the problem and inhibits us from devising distributed solutions.

### 3.2 Step Size Computations

Here we briefly review one of the ways to compute suitable step sizes to ensure convergence of the interior-point method. At each iteration, \( l \), in order to have \( g_i(x^{(l+1)}) \leq 0 \) for \( i = 1, \ldots, m \) and \( \lambda^{(l+1)} \succeq 0 \), we first compute
\[
\alpha_{\text{max}} = \min \left\{ 1, \min \left\{ -\lambda_i^{(l)} / \Delta \lambda_{i^{(l+1)}} \mid \Delta \lambda_{i^{(l+1)}} \right\} \right\},
\]
and perform a backtracking line search as below

while \( \exists i : g_i(x^{(l)} + \alpha^{(l+1)} \Delta x^{(l+1)}) > 0 \) do
\[
\alpha^{(l+1)} = \beta \alpha^{(l+1)}
\]
end while

with \( \beta \in (0, 1) \) and \( \alpha^{(l+1)} \) initialized as 0.99\( \alpha_{\text{max}} \). In order to guarantee convergence of primal and dual residuals to zero we continue the back tracking as

while
\[
\left( \left\| x^{(l+1)} \right\|_{\text{primal}} - \left\| x^{(l+1)} \right\|_{\text{dual}} \right)^2 > (1 - \gamma \alpha^{(l+1)}) \left( \left\| x^{(l)} \right\|_{\text{primal}} - \left\| x^{(l)} \right\|_{\text{dual}} \right)^2
\]
do
\[
\alpha^{(l+1)} = \beta \alpha^{(l+1)}
\]
end while

where \( \gamma \in [0.01, 0.1] \). The resulting \( \alpha^{(l+1)} \) ensures that the iterates remain feasible and that the primal and dual residuals are decreased consistently after each iteration.

**Remark 2.** The primal-dual method presented in this section is an implementation of the so-called infeasible long-step interior-point method. There are other variants

---

**Algorithm 1 Primal-dual Interior-point Method, (Boyd and Vandenberghe, 2004)**

1. Given \( l = 0, \mu > 1, \epsilon > 0, \epsilon_{\text{feas}} > 0, \lambda^{(0)} > 0, v^{(0)}, x^{(0)} \) such that \( g_i(x^{(0)}) < 0 \) for all \( i = 1, \ldots, m \) and \( \eta^{(0)} = \sum_{i=1}^{m} -\lambda_i^{(0)} g_i(x^{(0)}) \)
2. repeat
3. \( t = \mu \eta^{(0)} / \eta^{(l)} \)
4. Given \( t, \lambda^{(l)}, v^{(l)} \) and \( x^{(l)} \) compute \( \Delta x^{(l+1)}, \Delta \lambda^{(l+1)} \), \( \Delta \lambda_{i^{(l+1)}} \) by solving (10) and (9)
5. Compute \( \alpha^{(l+1)} \) using line search
6. \( x^{(l+1)} = x^{(l)} + \alpha^{(l+1)} \Delta x^{(l+1)} \)
7. \( \lambda^{(l+1)} = \lambda^{(l)} + \alpha^{(l+1)} \Delta \lambda^{(l+1)} \)
8. \( v^{(l+1)} = v^{(l)} + \alpha^{(l+1)} \Delta v^{(l+1)} \)
9. \( l = l + 1 \)
10. \( \eta^{(l)} = \sum_{i=1}^{m} -\lambda_i^{(l)} g_i(x^{(l)}) \)
11. until \( \left\| x^{(l)} \right\|_{\text{primal}}^2, \left\| r^{(l)} \right\|_{\text{dual}}^2 \leq \epsilon_{\text{feas}} \) and \( \eta^{(l)} \leq \epsilon \)
of primal-dual methods, such as short-step, predictor-corrector and Mehrotra’s predictor-corrector, that differ in their choice of primal-dual directions. One of the major differences among these variants is in the way they perturb the KKT conditions, i.e., the choice of $t$ in (5). This means that regardless of the choice of primal-dual interior point method the structure of the coefficient matrix in the resulting linear system of equations remains the same and hence the discussions that follow can be extended to other variants of primal-dual methods.

We will next apply the primal-dual interior-point method described in this section to the problem in (2).

4. A DISTRIBUTED PRIMAL-DUAL INTERIOR-POINT METHOD FOR SOLVING LOOSELY COUPLED PROBLEMS

As was shown in Section 3, at the heart of a primal-dual interior-point method is the primal-dual directions computation. Hence, one possible distributed primal-dual interior-point method for solving (2) would be to distribute the computations of these directions. To this end, we focus on the structure of (10) for the problem in (2), which is given by

$$
\begin{bmatrix}
\Delta s^1 \\
\vdots \\
\Delta s^N \\
\Delta x \\
\Delta v^1 \\
\Delta v^2 \\
\Delta v_c \\
\Delta v_c \\
\end{bmatrix} = -
\begin{bmatrix}
r^{1,l} \\
\vdots \\
r^{N,l} \\
r^{primal} \\
r^{primal} \\
r^{primal} \\
r^{primal} \\
r^{primal} \\
\end{bmatrix},
$$

(12)

where $\Delta v$ and $\Delta v_c$ are the dual variable directions for constraints in (2c) and (2d), respectively; $H_{pd}^{(l)} = \text{blkdiag}(H_{pd}^{1,l}, \ldots, H_{pd}^{N,l})$ with

$$H_{pd}^{i,l} = \nabla^2 f_i(s^{i,l}) + \sum_{j=1}^{m_i} \lambda_j^{i,l} \nabla^2 G^{i,l}_j(s^{i,l})$$

and

$$\lambda_j^{i,l} \geq 0, \quad j = 1, \ldots, m_i$$

The system of equations in (12) describes the optimality conditions for the optimization problem

Algorithm 2 ADMM

1. Given $k = 0$, $\rho > 0$, $\epsilon_{pri}, \epsilon_{dual} > 0$, $x^{(0)}$ and $\psi^{(0)}$
2. repeat
3. $S^{(k+1)} = \arg\min_S \left\{ F_1(S) + \frac{\rho}{2} \|AS - Bx^{(k)} - c + \psi^{(k)}\|^2 \right\}$
4. $x^{(k+1)} = \arg\min_x \left\{ F_2(x) + \frac{\rho}{2} \|AS^{(k+1)} - Bx - c + \psi^{(k)}\|^2 \right\}$
5. $\psi^{(k+1)} = \psi^{(k)} + (AS^{(k+1)} + Bx^{(k+1)} - c)\epsilon$
6. if $\|AS^{(k+1)} + Bx^{(k+1)} - c\|^2 < \epsilon_{dual}$ then
7. Terminate the algorithm
8. end if
9. $k = k + 1$
10. until algorithm is terminated

Remark 3. $\psi = (1/\rho)v$ is the scaled dual variable, see Boyd et al. (2011).

minimize $\Delta S, \Delta x$

$$\sum_{i=1}^{N} \frac{1}{2} (\Delta s^i)^T H_{pd}^{(l)} \Delta s^i + (r^{i,l})^T (\Delta s^i + \psi_i^{(l)}) + (r^{i,l})^T E \Delta x,$$

subject to $\bar{A}(\Delta s^i + t^{i,l}) = b$, $i = 1, \ldots, N$, (13b) $\Delta S - E \Delta x = E x^{(l)} - S^{(l)}$, (13c)

Notice that (13) has the same coupling structure as in (1) and can be solved distributedly. This will then enable us to compute the primal-dual directions in a distributed manner. To this end, we use ADMM, which is described in detail in the following sections.

4.1 Distributed Primal-dual Direction Computations for Loosely Coupled Problems

The problem in (13) is of the form

minimize $\Delta S, \Delta x$

$$\sum_{i=1}^{N} \frac{1}{2} (\Delta s^i)^T H_{pd}^{(l)} \Delta s^i + (r^{i,l})^T \Delta s^i + \Delta S - B \Delta x = c,$$

subject to $A \Delta S + B \Delta x = c$, (14)

which can be solved distributedly using proximal splitting methods, for instance using ADMM as described in Algorithm 2. In particular, problem (13) can be written as

$$\min_{\Delta S, \Delta x} \sum_{i=1}^{N} \frac{1}{2} (\Delta s^i)^T H_{pd}^{(l)} \Delta s^i + (r^{i,l})^T \Delta s^i$$

subject to

$$\begin{bmatrix}
\Delta s^1 \\
\vdots \\
\Delta s^N \\
\end{bmatrix} = -
\begin{bmatrix}
r^{1,l} \\
\vdots \\
r^{N,l} \\
\end{bmatrix},
$$

with

$$\Delta s^i = \Delta s^{i,l} - \frac{1}{t} \frac{1}{t} 1.$$
can now compute $\Delta \lambda$.

Algorithm 2. Notice that the termination condition for the primal-dual directions is expressed in Algorithm 3. 

The updates for dual variable directions are then given by 

\[
\Delta \tilde{x}^{i,(k+1)} = \Delta \tilde{x}^{i,(k)} + \left( A \Delta s^{i,(k+1)} + r^i_{\text{cent}} \right),
\]

for $i = 1, \ldots, N$. Notice that the dual variable directions (19) are scaled and they have to be rescaled to give the actual dual variable directions; that is 

\[
\Delta \tilde{v} = \left( 1/\rho \right) \Delta v, \quad \Delta \tilde{v}_c = \left( 1/\rho \right) \Delta v_c.
\]

Having computed the directions $\Delta s$, $\Delta x$, $\Delta v$ and $\Delta v_c$, we can now compute $\Delta \lambda$ as 

\[
\Delta \lambda = - \text{diag}(\tilde{G}^i(s^{i,(l)}))^{-1} \times \left( \text{diag}(\lambda^{i,(l)} \tilde{D} \tilde{G}^i(s^{i,(l)})) \Delta s - r^i_{\text{cent}} \right),
\]

for $i = 1, \ldots, N$. The distributed algorithm for computing the primal-dual directions is expressed in Algorithm 3. 

Algorithm 3 ADMM-Based Primal-dual Direction Computation

1: Given $k = 0$, $\rho > 0$, $\epsilon_{\text{pri}}, \epsilon_{\text{dual}} > 0$, $\Delta S^{(0)}$, $\Delta \psi^{(0)}$ and $\Delta \psi^{(0)}$
2: for $i = 1, 2, \ldots, N$ do
3: Communicate with all agents $r$ belonging to $\text{Ne}(i)$
4: for all $j \in J_i$ do
5: $\Delta x_j^{(k)} = \frac{1}{|J_i|} \sum_{q \in \mathcal{J}_j} \left( E^T \Delta s_q^{(k)} \right)_j$
6: end for
7: end for
8: repeat
9: for $i = 1, 2, \ldots, N$ do
10: $\Delta \tilde{x}^{i,(k+1)} = - \left[ H^i_{\text{pd}}(k) + \rho (I + (A^i)^T A^i) \right]^{-1} \times \left[ r^i_{\text{pd}}(k) + \rho \left( \Delta \tilde{x}^{i,(k)} - \Delta x^k_{J_i} \right) \right]$
11: $\Delta v^{i,(k+1)} = \Delta v^{i,(k)} + \left( A \Delta s^{i,(k+1)} + r^i_{\text{cent}} \right)$
12: for all $j \in J_i$ do
13: $\Delta x_j^{(k+1)} = \frac{1}{|J_i|} \sum_{q \in \mathcal{J}_j} \left[ E^T \Delta s_q^{(k+1)} + r^i_{\text{cent}} \right]$
14: end for
15: Check whether $\|\Delta \tilde{x}^{i,(k+1)} - \Delta x^k_{J_i}\|^2 \leq \epsilon_{\text{dual}}/N$, $\|\Delta s^{i,(k+1)} - \Delta s^k_{J_i}\|^2 \leq \epsilon_{\text{pri}}/2N$ and $\|\tilde{A} \Delta s^{i,(k+1)} + r^i_{\text{cent}}\|^2 \leq \epsilon_{\text{pri}}/2N$
16: end for
17: if condition in step (11) satisfied for all $i = 1, \ldots, N$ then
18: Terminate the algorithm
19: end if
20: $k = k + 1$
21: until algorithm is terminated
22: for $i = 1, 2, \ldots, N$ do
23: $\Delta \lambda^i = - \text{diag}(\tilde{G}^i(s^{i,(l)}))^{-1} \times \left( \text{diag}(\lambda^{i,(l)} \tilde{D} \tilde{G}^i(s^{i,(l)})) \Delta s - r^i_{\text{cent}} \right)$
24: end for
25: end for

Remark 5. The computational effort for each iteration of Algorithm 3 is dominated by the cost of updating $\Delta s^{i,(k+1)}$, which requires factorizing the matrices $H^i_{\text{pd}}(k) + \rho (I + (A^i)^T A^i)$ for $i = 1, \ldots, N$. Notice that in case $\rho$ is chosen to be a constant, these matrices remain the same within each iteration of this algorithm, and hence the computational burden of each instance of Algorithm 3 can be significantly reduced by pre-caching these factorizations and reusing them in the subsequent iterations. In fact even if $\rho$ is nonconstant we can adopt a procedure that would allow us to update the factorizations of these matrices without having to recompute them entirely (Liu et al., 2013, sec. 4.2).

Remark 6. As was mentioned we can use other proximal splitting methods than ADMM, for solving (13) distributedly, possibly with better convergence properties. However, unlike ADMM, this generally requires a reformulation of (13), which can in turn complicate the recovery of the dual variable directions, $\Delta v$ and $\Delta v_c$. We have
therefore restricted ourselves in this paper to using ADMM for this purpose, in order to keep the presentation simple.

4.2 Improving convergence rate

We can improve the convergence rate of ADMM by choosing the penalty parameter $\rho$ carefully, using over-relaxation and warm starting the ADMM iterations. In general it is an open problem how to choose $\rho$ optimally. There are some results available for example by Ghadimi et al. (2013), but they are not readily applicable to our problem formulation. However, certain heuristics suggest that $\rho$ should be chosen such that the primal and dual residuals converge at the same rate (Boyd et al., 2011). In over-relaxation, we replace the primal quantity $\Delta^k$ with
\[
\alpha_{OR} A \Delta^k + \Delta^k - (1 - \alpha_{OR}) (B \Delta^k - c), \quad \alpha_{OR} \in (1, 2),
\]
in the update of $\Delta^k$ and scaled dual variables $\Delta \bar{v}$ and $\Delta \bar{u}$ (Boyd et al., 2011). We can improve the convergence rate by warm starting ADMM; that is, using the solution of the previous ADMM iteration as initial condition in the current iteration (Boyd et al., 2011). The improvement in convergence rate is due to that the primal-dual directions of the interior point method do not change much as the iterates are approaching a solution.

4.3 Iterative solvers for saddle point systems

The optimality conditions (12) is a saddle point system where the solution, since strong duality holds, is a saddle point of the Lagrangian function of optimization problem (14), by Boyd and Vandenberghe (2004). In addition, a saddle point of the Lagrangian function is a saddle point of the augmented Lagrangian, and vice versa (Gabay and Mercier, 1976). Consequently, to find a solution of (12), we can instead consider the saddle point system corresponding to the augmented Lagrangian. The benefit of using the augmented Lagrangian is improved convergence properties when using dual methods for solving the saddle point system (Arrow, 1964; Hestenes, 1969; Fortin and Glowinski, 1983).

Uzawa’s method and fixed point iterations A well known algorithm for solving saddle point systems such as (12) is Uzawa’s method (Arrow, 1964). We solve the system of equations (12) using ADMM (Algorithm 2), which was originally derived as a modified version of Uzawa’s method (Glowinski and Marroco, 1975; Gabay and Mercier, 1976). ADMM applied to problem (14) is equivalent to Uzawa’s method applied to the problem corresponding to the augmented Lagrangian of (14) (Arrow, 1964; Hestenes, 1969; Powell, 1969) with one Gauss-Seidel iteration (Saad, 2003) in the update of the primal variables (Glowinski and Marroco, 1975; Gabay and Mercier, 1976). In addition, ADMM can be viewed as fixed point iterations of a pre-conditioned version of (12), see e.g. Parikh and Boyd (2014). The similarities between ADMM, Uzawa’s method and fixed point iterations are explored explicitly for our problem in Appendix (A.1.2).

Remark 7. Uzawa’s method is also equivalent to the method of multipliers when applied to the augmented Lagrangian and the relaxation parameter in Uzawa’s method is set to be equal to the penalty parameter in the method of multipliers, see Benzi et al. (2005).

Other iterative methods For a rigorous overview of the iterative methods available, we refer to Saad (2003); Benzi et al. (2005). Problem (12) has an indefinite system matrix with an upper left block matrix that is singular, which limits the number of applicable methods or at least requires some pre-conditioning beforehand. For example, we could use the conjugate gradient method applied to the normal equations of (12), see Saad (2003).

4.4 Distributed Primal-dual Interior-point Method

Using Algorithm 3 we can compute the primal-dual directions $\Delta x$, $\Delta \bar{v}$, $\Delta \bar{u}$, $\Delta \lambda$ and $\Delta \lambda$ distributedly. This algorithm can then be used in step 4 of Algorithm 1 which distributes the major computations within the primal-dual interior-point method. Combining Algorithms 3 and 1 results in Algorithm 4, which is a distributed primal-dual interior-point method for solving (2). Notice that the required minimum value computations in steps 9 and 14 of Algorithm 4 can be performed distributedly using algorithms such as min-consensus, Iutzeler et al. (2012). We note that Remark 4, concerning the termination condition in Algorithm 3, applies to the termination condition of Algorithm 4.

**Algorithm 4 Distributed Primal-dual Interior-point Method**

1. Given $t = 0$, $\mu > 0$, $\epsilon_0 > 0$, $\epsilon_{feas} > 0$, $\lambda(0) > 0$, $\psi(0)$, $S(0)$ such that $G^*(\psi(0)) < 0$ and $\nu(0) = \sum_{j=1}^{m} - \lambda_j^0 \tilde{G}_j^*(\psi(0))$ for all $i = 1, \ldots, N$, and $\Delta S(0)$, $\Delta \psi(0)$ and $\Delta \nu(0)$
2. for $i = 1, 2, \ldots, N$
3. Communicate with all agents $r$ belonging to $Ne(i)$
4. for all $j \in J_i$
5. $x_j^{(0)} = \left( \frac{\sum_{l=1}^{N} c_{ij} \nu_l^{(0)}}{\max_{l=1}^{N} \{\nu_l^{(0)}\}} \right)
6. $\bar{v}_i^{(0)} = \frac{1}{S(0)} \sum_{q \in I_j} (E^T J_q s_\theta(q))$
7. $\lambda_l^{(0)} = \max \{\nu_l^{(0)}\}$
8. repeat
9. $t = \mu \left( \sum_{j=1}^{N} m_j \right) / \min_{l=1}^{N} \{\nu_l^{(0)}\}$
10. Given $t$, $\lambda^{(l)}$, $\nu^{(l)}$ and $x^{(l)}$ compute $\Delta \psi^{(l+1)}$, $\Delta \lambda^{(l+1)}$, $\Delta \nu^{(l+1)}$ using Alg. 3 with the initial iterates $\Delta \psi^{(l)}$, $\Delta \lambda^{(l)}$ and $\Delta \nu^{(l)}$
11. for $i = 1, 2, \ldots, N$
12. Compute local step size $\alpha_i^{(l+1)}$ using the approach presented in Section 3 and based on local problem information and local iterates
13. end for
14. $\alpha^{(l+1)} = \min \{\alpha_i^{(l+1)}\}$
15. for $i = 1, 2, \ldots, N$
16. $x_j^{(l+1)} = x_j^{(l)} + \alpha^{(l+1)} \Delta x_j^{(l)}$
17. $s_j^{(l+1)} = s_j^{(l)} + \alpha^{(l+1)} \Delta s_j^{(l+1)}$
18. $\nu_j^{(l+1)} = \nu_j^{(l)} + \alpha^{(l+1)} \Delta \nu_j^{(l+1)}$
19. $\lambda_l^{(l+1)} = \lambda_l^{(l)} + \alpha^{(l+1)} \Delta \lambda_l^{(l+1)}$
20. end for
21. $l = l + 1$
22. for $i = 1, 2, \ldots, N$
23. $\psi_i^{(l)} = \sum_{j=1}^{m} x_j^{(l+1)} G_j^*(\psi_i^{(l)})$
24. $\|x_i^{(l)} - \psi_i^{(l)}\|_2 \leq \epsilon_{feas} / N$ and $\|\nu_i^{(l)} - \lambda_i^{(l)} \|_2 \leq \epsilon_{feas} / N$
25. end for
26. until $\|\Delta \psi^{(l)}\|, \|\Delta \lambda^{(l)}\| \leq \epsilon_{feas} / N$ and $\|\Delta \nu^{(l)}\| \leq \epsilon_{feas} / N$ for all $i = 1, \ldots, N$
5. NUMERICAL EXPERIMENT

To illustrate the proposed method, we apply it to a randomly generated optimization problem of the form (2). We consider five subproblems ($N = 5$). The total number of variables ($n$), equality constraints ($p$) and inequality constraints ($m$) are 304, 460, and 161, respectively. The indices defining the consistency constraints ($J_i$) are drawn from a standard uniform distribution over the interval $[0,900]$, and there are 261 such constraints. The optimization problem is solved using Algorithm 4 with $\epsilon_{pri}$, $\epsilon_{pri}$, $\epsilon_{feas}$ and $\epsilon_{dual}$ set to $10^{-6}$. The tuning parameters for the primal dual method are $\alpha = 0.01$, $\beta = 0.5$ and $\mu = 15$. The penalty parameter of ADMM is set to one ($\rho = 1$). We do not use over-relaxation; in other words, we set $\alpha_{OR} = 1$. However, we do take advantage of warm starting.

Figure 1 shows the number of iterations performed in ADMM for each instance of the primal dual method. The number of iterations required in ADMM typically drops dramatically during the first couple of iterations of the primal dual method. We believe that this is due to the warm starting effect. As stated above, we used a fixed accuracy in ADMM over all iterations in the interior-point method. It is believed that the total number of ADMM iterations can be decreased significantly by using an adaptive accuracy instead without affecting the number of iterations of the interior-point method, see for example Bonettini and Ruggiero (2007). In Figure 2, we see how the relative error in the optimal value decreases with the number of iterations in the interior-point method. The true optimal value is obtained by solving the optimization problem using cvx (Grant and Boyd, 2014).

![ADMM iterations](image1.png)

Fig. 1. ADMM iterations. Number of ADMM iterations in each instance of the primal-dual method. The accuracies are set to $\epsilon_{feas} = \epsilon_{dual} = 10^{-6}$.

![Relative error in optimal value](image2.png)

Fig. 2. Relative error in optimal value. The relative error of the optimal value, $p$, for each iteration of the primal dual method. We consider the true optimal value, $p^*$, to be the value obtained using cvx. The accuracies are set to $\epsilon_{feas} = \epsilon_{dual} = 10^{-6}$.

6. CONCLUSION

We have proposed an efficient distributed primal-dual interior-point method for loosely coupled problems using ADMM (Algorithm 4). Due to the nature of the interior-point method, the loosely coupled structure of the problem is preserved in the linear system of equations that provides the primal-dual directions. ADMM takes advantage of this structure and makes the direction calculations highly parallelizable. Consequently, we expect the proposed method to have superior convergence properties with respect to other distributed algorithms. Of course, we can use Algorithm 4 on problems with completely coupled structure as well, but we can not expect the same superior convergence as for the loosely coupled structure. As future work, we will establish a way of choosing the required accuracy in the termination condition of the inner iterations (ADMM iterations) with respect to the accuracy obtained in the outer iterations (interior-point method iterations). This is to avoid unnecessary inner iterations when the accuracy of the current outer iteration is low, as elaborated in Bonettini et al. (2005). In addition, we will look at different tuning possibilities of ADMM to decrease the number of necessary inner iterations. Also, we will provide performance comparisons between Algorithm 4 and other distributed algorithms.

REFERENCES

G. Al-Jeiroudi and J. Gondzio. Convergence analysis of the inexact infeasible interior-point method for linear optimization. *Journal of Optimization Theory and Applications*, 141(2):231–247, 2009.

K. J. Arrow. *Studies in linear and non-linear programming*. Stanford University Press Stanford, Calif., 1964.

S. Bellavia. Inexact interior-point method. *Journal of Optimization Theory and Applications*, 96(1):109–121, 1998.

S. Bellavia and S. Pieraccini. Convergence analysis of an inexact infeasible interior point method for semidefinite programming. *Computational Optimization and Applications*, 29(3):289–313, 2004.

M. Benzi, Gene H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14 (1):1–137, April 2005.

D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

S. Bonettini and V. Ruggiero. Some iterative methods for the solution of a symmetric indefinite KKT system. *Computational Optimization and Applications*, 38(1):3–25, 2007.

S. Bonettini, E. Galligani, and V. Ruggiero. An inexact Newton method combined with Hestenes multipliers scheme for the solution of Karush-Kuhn-Tucker systems.
S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

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.

C. Chen, B. He, Y. Ye, and X. Yuan. The Direct Extension of ADMM for Multi-block Convex Minimization Problems is Not Necessarily Convergent. *Optimization Online e-prints*, September 2013.

E. Chu, D. Gorinevsky, and S. Boyd. Scalable statistical monitoring of fleet data. In *Proceedings of the 18th IFAC World Congress*, pages 13227–13232, Milan, Italy, August 2011.

P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, volume 49 of *Springer Optimization and Its Applications*, pages 185–212. Springer New York, 2011.

F. Curtis, O. Schenk, and A. Wächter. An interior-point algorithm for large-scale nonlinear optimization with inexact step computations. *SIAM Journal on Scientific Computing*, 32(6):3447–3475, 2010.

F. E. Curtis, J. Huber, O. Schenk, and A. Wächter. A note on the implementation of an interior-point algorithm for nonlinear optimization with inexact step computations. *Mathematical Programming*, 136(1):209–227, 2012.

J. Douglas and H. H. Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Transaction of the American Mathematical Society*, 82:421–489, 1956.

C. Durazzi and V. Ruggiero. A Newton inexact interior-point method for large scale nonlinear optimization problems. *Annali dell’Università di Ferrara*, 49(1):333–357, 2003.

J. Eckstein. *Splitting methods for monotone operators with application to parallel optimization*. PhD dissertation, Massachusetts Institute of Technology, 1989.

J. Eckstein and D. P. Bertsekas. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55:293–318, 1992.

M. Fortin and R. Glowinski. Chapter 1 augmented lagrangian methods in quadratic programming. In Michel Fortin and Roland Glowinski, editors, *Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems*, volume 15 of *Studies in Mathematics and Its Applications*, pages 1 – 46. Elsevier, 1983.

R. W. Freund, F. Jarre, and S. Mizuno. Convergence of a class of inexact interior-point algorithms for linear programs. *Mathematics of Operations Research*, 24(1):50–71, 1999.

D. Gabay. Applications of the method of multipliers to variational inequalities. In M. Fortin and R. Glowinski, editors, *Augmented Lagrangian Methods: Applications to the Solution of Boundary-Value Problems*. North-Holland, 1983.

D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17–40, 1976.

E. Ghaedim, A. Teixeira, I. Shames, and M. Johansson. Optimal parameter selection for the alternating direction method of multipliers (ADMM): Quadratic problems. *ArXiv e-prints*, Jun. 2013.

R. Glowinski and A. Marroco. Sur l’approximation, par éléments finis d’ordre un, et la résolution, par penalisation-dualité, d’une classe de problèmes de Dirichlet non linéaires. *Rev. Franc. Automat. Inform. Rech. Operat.*, R, 9(2):41–76, 1975.

D. Goldfarb, S. Ma, and K. Scheinberg. Fast alternating linearization methods for minimizing the sum of two convex functions. *Mathematical Programming*, pages 1–34, 2012.

T. Goldstein, B. ODonoghue, and S. Setzer. Fast alternating direction optimization methods. Technical Report CAM report 12-35, UCLA, 2012.

M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. *http://cvxr.com/cvx*, March 2014.

D. Han and X. Yuan. A note on the alternating direction method of multipliers. *Journal of Optimization Theory and Applications*, 155(1):227–238, 2012.

A. Hansson. A primal-dual interior-point method for robust optimal control of linear discrete-time systems. *IEEE Transactions on Automatic Control*, 45(9):1639–1655, September 2000.

M. R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4(5):303–320, 1969. ISSN 0022-3239. doi: 10.1007/BF00276773.

M. Hong and Z.-Q. Luo. On the Linear Convergence of the Alternating Direction Method of Multipliers. *ArXiv e-prints*, August 2012.

F. Iutzeler, P. Ciblat, and J. Jakubowicz. Analysis of Max-Consensus algorithms in wireless channels. *IEEE Transactions on Signal Processing*, 60(11):6103–6107, 2012.

J. Korzak. Convergence analysis of inexact infeasible-interior-point algorithms for solving linear programming problems. *SIAM Journal on Optimization*, 11(1):133–148, 2000.

Z. Liu, A. Hansson, and L. Vandenberghe. Nuclear norm system identification with missing inputs and outputs. *Systems & Control Letters*, 62(8):605–612, 2013.

Z. Lu, R. D. C. Monteiro, and J. W. O’Neal. An iterative solver-based long-step infeasible primal-dual path-following algorithm for convex qp based on a class of preconditioners. *Optimization Methods Software*, 24(1):123–143, February 2009.

S. Mizuno and F. Jarre. Global and polynomial-time convergence of an infeasible-interior-point algorithm using inexact computation. *Mathematical Programming*, 84(1):105–122, 1999.

I. Necoara and J. A. K. Suykens. Interior-point lagrangian decomposition method for separable convex optimization. *Journal of Optimization Theory and Applications*, 143(3):567–588, 2009.

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

A. Nedic, A. Ozdaglar, and P.A. Parrilo. Constrained consensus and optimization in multi-agent networks. *IEEE Transactions on Automatic Control*, 55(4):922–
Thus, we can write the optimality conditions (12) as

$$\begin{aligned}
\Delta S, \Delta x, \Delta \bar{w} &= \text{saddle point of the augmented Lagrangian as discussed in Section 4.3.1.}
\end{aligned}$$

E. Wei, A. Ozdaglar, and A. Jadbabaie. A distributed Newton method for network utility maximization–I: Algorithm. IEEE Transactions on Automatic Control, 58(9):2162–2175, 2013.

K. Zhou and K. Toh. Polynomiality of an inexact infeasible interior point algorithm for semidefinite programming. Mathematical Programming, 99(2):261–282, 2004.

Appendix A. ADMM, FIXED POINT ITERATIONS AND UZAWA’S METHOD

As stated in Section 4.3.1, ADMM can be viewed as fixed point iterations and as a modified version of Uzawa’s method. In this appendix, we explore these relations in detail for our specific problem formulation.

A.1 Problem formulation

The functions $F_1(\Delta S)$ and $F_2(\Delta x)$ in problem (14) are of the form

$$F_1(\Delta S) = \frac{1}{2} \Delta S^T \tilde{F}_1 \Delta S + f_1^T \Delta S$$

and

$$F_2(\Delta x) = \tilde{f}_2^T \Delta x.$$ 

Thus, we can write the optimality conditions (12) as

$$
\begin{bmatrix}
\tilde{F}_1 & 0 & \rho A^T \\
0 & \rho B^T & 0 \\
\rho A & \rho B & 0
\end{bmatrix}
\begin{bmatrix}
\Delta S \\
\Delta x \\
\Delta \bar{w}
\end{bmatrix}
=
\begin{bmatrix}
-\tilde{f}_1 \\
-\tilde{f}_2 \\
\rho c
\end{bmatrix},
$$

where $\Delta \bar{w} = (\Delta \bar{w}_1, \Delta \bar{w}_2, \Delta \bar{w}_3)$. Solving (A.1) is equivalent to finding a saddle point of the Lagrangian function

$$L(\Delta S, \Delta x, \Delta \bar{w}) = F_1(\Delta S) + F_2(\Delta x) + \Delta \bar{w}^T (AS + B\Delta x - c),$$

which is a saddle point of the augmented Lagrangian as well,

$$\begin{aligned}
L_{\rho}(\Delta S, \Delta x, \Delta \bar{w}) &= F_1(\Delta S) + F_2(\Delta x) + \\
\frac{\rho}{2} \| \Delta S + B\Delta x - c + \Delta \bar{w} \|^2_2.
\end{aligned}$$

The augmented Lagrangian function is, in turn, the Lagrangian function of the optimization problem

$$\begin{aligned}
\text{minimize} & \quad F_1(\Delta S) + F_2(\Delta x) + \frac{\rho}{2} \| \Delta S + B\Delta x - c \|^2_2, \\
\text{subject to} & \quad A\Delta S + B\Delta x = c,
\end{aligned}$$

which is equivalent to (14).

Fixed point iterations \quad The ADMM iterations of problem (14) can be written in closed form as

$$\begin{aligned}
\Delta S^{(k+1)} &= M_1(\tilde{f}_1 + \rho A^T c - \rho A^T B\Delta x^{(k)} - \rho A^T \Delta \bar{w}^{(k)}), \\
\Delta x^{(k+1)} &= M_2(\tilde{f}_2 + \rho B^T c - \rho B^T A\Delta S^{(k+1)} - \rho B^T \Delta \bar{w}^{(k)}), \\
\Delta \bar{w}^{(k+1)} &= \Delta \bar{w}^{(k)} + A\Delta S^{(k+1)} + B\Delta x^{(k+1)} - c,
\end{aligned}$$

with

$$M_1 = (\tilde{F}_1 + \rho A^T A)^{-1}$$

and

$$M_2 = (\rho B^T B)^{-1}.$$ 

Equations (A.5) correspond to the iterations

$$
\begin{bmatrix}
\Delta S^{(k+1)} \\
\Delta x^{(k+1)} \\
\Delta \bar{w}^{(k+1)}
\end{bmatrix} = G
\begin{bmatrix}
\Delta S^{(k)} \\
\Delta x^{(k)} \\
\Delta \bar{w}^{(k)}
\end{bmatrix} + f,
$$

with

$$G =
\begin{bmatrix}
0 & -\rho M_1 A^T B & 0 \\
0 & \rho^2 M_2 B^T A M_1 A^T B & 0 \\
-\rho M_1 A^T B & 0 & \rho M_2 B^T (\rho M_1 A^T - I) A M_1 A^T - \rho BM_2 B^T + I
\end{bmatrix},$$

and

$$f =
\begin{bmatrix}
M_1 m_1 \\
M_2 m_2 - \rho M_2 B^T A M_1 m_1 \\
-c + AM_1 m_1 + B (M_2 m_2 - \rho M_2 B^T A M_1 m_1)
\end{bmatrix},$$

where

$$m_1 = -\tilde{f}_1 + \rho A^T c$$

and

$$m_2 = -\tilde{f}_2 + \rho B^T c.$$ 

The iterations (A.6) clearly show that $\Delta x$ and $\Delta \bar{w}$ make up the state of the algorithm, whereas $\Delta S$ only is an intermediate result, see Boyd et al. (2011). We can view (A.6) as an iterative solver for a pre-conditioned version of the system of equations (A.1). That is, the iteration matrix $G$ and the vector $f$ can be expressed as

$$G = I - M_{\text{PRE,1}}^{-1} A_{\text{KKT}}$$

and

$$f = M_{\text{PRE,1}}^{-1} b_{\text{KKT}},$$

respectively, where $M_{\text{PRE,1}}$ is a pre-conditioner defined as

$$M_{\text{PRE,1}} =
\begin{bmatrix}
\tilde{F}_1 & -\rho A^T B & \rho A^T \\
0 & 0 & \rho B^T \\
\rho A & \rho B & -\rho I
\end{bmatrix}. $$

For details regarding iterative solvers and pre-conditioners, see Saad (2003). When ADMM converges, we get the fixed point iterations

$$
\begin{bmatrix}
\Delta S \\
\Delta x \\
\Delta \bar{w}
\end{bmatrix} = G
\begin{bmatrix}
\Delta S \\
\Delta x \\
\Delta \bar{w}
\end{bmatrix} + f \Leftrightarrow A_{\text{KKT}}
\begin{bmatrix}
\Delta S \\
\Delta x \\
\Delta \bar{w}
\end{bmatrix} = b_{\text{KKT}},
$$

which is equal to the system of equations that we would like to solve, namely (A.1).
Alternating direction method of multipliers ADMM applied to problem (14) is equivalent to Uzawa’s method applied to problem (A.4) with one Gauss-Seidel iteration (Saad, 2003) in the update of the primal variables (Glowinski and Marrocco, 1975; Gabay and Mercier, 1976) and the relaxation parameter equal to the penalty parameter $\rho$.

First note that the optimality conditions of problem (A.4),

$$
\begin{bmatrix}
M_1^{-1} & \rho A & \rho B \\
\rho B^T A & M_2^{-1} & \rho B^T \\
\rho A & \rho B & 0
\end{bmatrix}
\begin{bmatrix}
\Delta S \\
\Delta x \\
\Delta w
\end{bmatrix} =
\begin{bmatrix}
m_1 \\
m_2 \\
\rho c
\end{bmatrix},
$$

(A.11)

are equivalent to those of problem (14). In fact, the system of equations (A.11) is a preconditioned version of (A.1), that is, $A_{\text{PRE},2}^{\text{KKT}} = M_{\text{PRE},2}^{\text{KKT}} A_{\text{KKT}}$ and $b_{\text{PRE},2}^{\text{KKT}} = M_{\text{PRE},2}^{\text{KKT}} b_{\text{KKT}}$, with

$$
M_{\text{PRE},2} = \begin{bmatrix} I & 0 & -A^T \\ 0 & I & -B^T \\ 0 & 0 & I \end{bmatrix}.
$$

(A.12)

Second, in Uzawa’s method the Lagrangian function is minimized jointly over the primal variables while in ADMM, it is minimized approximately by sequential minimization over $\Delta S$ and $\Delta x$, that is, by performing one Gauss-Seidel iteration with respect to $\Delta S$ and $\Delta x$ (Saad, 2003). The first step of Uzawa’s method applied to our problem is to solve

$$
\begin{bmatrix}
M_1^{-1} \rho A^T B & \rho A^T \\
\rho B^T A & M_2^{-1} \rho B^T \\
\rho A & \rho B & 0
\end{bmatrix}
\begin{bmatrix}
\Delta S \\
\Delta x \\
\Delta w
\end{bmatrix} =
\begin{bmatrix}
m_1 - \rho A^T \Delta \bar{w}^k \\
m_2 - \rho B^T \Delta \bar{w}^k \\
\rho c
\end{bmatrix},
$$

(A.13)

which is the closed form solution of

$$(\Delta S^{(k+1)}, \Delta x^{(k+1)}) = \arg \min_{\Delta S, \Delta x} \left\{ L_{\rho}(\Delta S, \Delta x, \Delta \bar{w}^k) \right\} = \arg \min_{\Delta S, \Delta x} \left\{ F_1(\Delta S) + F_2(\Delta x) + \frac{\rho}{2} \| A\Delta S + B\Delta x - c + \Delta \bar{w}^k \|_2^2 \right\}.
$$

The system of equations (A.13) can be solved approximately using Gauss-Seidel; that is,

$$
\begin{bmatrix}
\Delta S^{k+1} \\
\Delta x^{k+1}
\end{bmatrix} =
\begin{bmatrix}
M_1^{-1} & 0 \\
\rho B^T A & M_2^{-1}
\end{bmatrix}
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & -A^T B \\
M_1 & 0 & \rho B^T A \rho B^T \\
0 & M_2 & 0 \\
0 & 0 & \rho B^T A M_2^{-1} \rho B^T \\
0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\Delta S^k \\
\Delta x^k
\end{bmatrix} +
\begin{bmatrix}
M_1^{-1} \rho A^T B \rho B^T A M_2^{-1} \rho B^T W^k \\
M_2^{-1} \rho B^T A \rho B^T \Delta \bar{w}^k
\end{bmatrix}.
$$

(A.14)

Iteration (A.14) is equivalent to the primal updates (A.6) in ADMM. Third, the dual update in Uzawa’s method is equivalent to that of ADMM.

Remark 8. ADMM can be viewed as an iterative solver of the format (A.6) where

$$
G = I - M_{\text{PRE},3}^{-1} M_{\text{PRE},2}^{\text{KKT}},
$$

$$
f = M_{\text{PRE},3}^{-1} M_{\text{PRE},2}^{\text{KKT}} b_{\text{KKT}},
$$

$$
M_{\text{PRE},3} = \begin{bmatrix}
M_1^{-1} & 0 & 0 \\
\rho B^T A M_2^{-1} & 0 & 0 \\
\rho A & \rho B & -\rho I
\end{bmatrix}.
$$