Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Distributed Interior-point Method for Loosely Coupled Problems ⋆

Sina Khoshfetrat Pakazad * **Anders Hansson** *
Martin S. Andersen **

* *Division of Automatic Control, Department of Electrical Engineering, Linköping University, Sweden. Email: {sina.kh.pa, hansson}@isy.liu.se.*
** *Department of Applied Mathematics and Computer Science, Technical University of Denmark, Denmark. Email: mskan@dtu.dk*

Abstract: In this paper, we put forth distributed algorithms for solving loosely coupled unconstrained and constrained optimization problems. Such problems are usually solved using algorithms that are based on a combination of decomposition and first order methods. These algorithms are commonly very slow and require many iterations to converge. In order to alleviate this issue, we propose algorithms that combine the Newton and interior-point methods with proximal splitting methods for solving such problems. Particularly, the algorithm for solving unconstrained loosely coupled problems, is based on Newton's method and utilizes proximal splitting to distribute the computations for calculating the Newton step at each iteration. A combination of this algorithm and the interior-point method is then used to introduce a distributed algorithm for solving constrained loosely coupled problems. We also provide guidelines on how to implement the proposed methods efficiently, and briefly discuss the properties of the resulting solutions.

1. INTRODUCTION

Distributed optimization methods are specially crucial in the absence of a centralized computational unit or when this unit lacks the necessary computational power for solving the problem at all or in a timely manner. In either case, distributed algorithms enable us to solve the problem without the need for a central computational unit and through collaboration of a number of computational agents. Such methods also allow us to solve the problem by solving subproblems that are easier and possibly faster to solve. Distributed methods for solving optimization problems 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 commonly used approaches for devising distributed optimization algorithms is based on applying gradient/subgradient methods directly to the problem, see e.g., Nedic and Ozdaglar (2009); Nedic et al. (2010). The resulting algorithms from such approaches are usually simple and easy to implement, however, they are very sensitive to the scaling of the problem and suffer from very slow convergence, Bertsekas and Tsitsiklis (1997). Another approach for constructing distributed algorithms is based on decomposition techniques and proximal splitting methods. To this end, the optimization problem is first decomposed and then a splitting method of choice, e.g., the alternating direction method of multipliers (ADMM), is applied to the decomposed problem. This will define the computational routines that each agent should perform locally and also will set the communication/collaboration protocol among the agents, e.g., see Bertsekas and Tsitsiklis (1997); Eckstein (1989); Boyd et al. (2011); Combettes and Pesquet (2011). Even though the resulting algorithms using this approach tend to be more complicated than the previous type of algorithms, they enjoy a faster rate of convergence for certain classes of optimization problems, e.g., when the objective function of the equivalent unconstrained reformulation of the problem, has two terms and/or is strongly convex, Goldfarb et al. (2012); Goldstein et al. (2012). For more general classes of problems, however, for instance when the objective function has more than two terms of which several are non-smooth, e.g., indicator functions, they can perform very poorly or may even fail to converge, e.g., see Chen et al. (2013). Although there exist modifications to splitting methods that allow us to apply them to more general classes of optimization problems, the resulting algorithms can become overly complicated to implement (particularly distributedly), Goldfarb et al. (2012); Han and Yuan (2012); Hong and Luo (2012), or the local computations can still be considerable, e.g., they may require each agent to solve an equality/inequality constrained problem at each iteration, Summers and Lygeros (2012); Ohlsson et al. (2013).

In order to allay the aforementioned issues, there has been a recent interest in studying the possibility of using second order methods for designing distributed optimization algorithms, e.g., see Chu et al. (2011); Wei et al. (2013); Necoara and Suykens (2009). For instance, in Wei et al. (2013), the authors propose a distributed Newton method for solving a network utility maximization problem. The cost function for such problems comprise of a summation of several terms where each term depends on a single scalar

⋆ This work has been supported by the Swedish Department of Education within the ELLIIT project.

variable. This structure allows the authors to employ a matrix splitting method which in turn enables them to compute the Newton directions distributedly. However, this method relies on the special structure in the considered problem and hence can only be used for problems with such structure. 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 solving the subproblems using interior-point methods, efficiently. However, in the proposed algorithm, the computational cost for solving the subproblems can still be considerable. The authors in 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.

In this paper, we investigate the possibility of utilizing Newton's and interior-point methods for designing distributed algorithms for solving loosely coupled optimization problems. Notice that this type of problems constitute a more general class of problems than the ones in the above mentioned papers. For this purpose, we first exploit the coupling in the problem using consistency constraints and use proximal splitting methods to compute the Newton directions in a distributed manner. This then enables us to propose distributed implementations of the Newton method, which can be used for solving unconstrained loosely coupled problems. In order to solve constrained loosely coupled problems, we also put forth a distributed interior-point method which relies on the proposed distributed Newton method. Furthermore, in contrast to the methods proposed in Nedic and Ozdaglar (2009), in all the proposed algorithms each of the agents is only required to solve an unconstrained or equality constrained quadratic program at each iteration.

Notation

We denote by $\mathbb{R}$ the set of real scalars and by $\mathbb{R}^{n\times m}$ the set of real $n \times m$ matrices. The transpose of a matrix A is denoted by A^T and the column and null space of this matrix is denoted by $\mathcal{C}(A)$ and $\mathcal{N}(A)$, respectively. We denote the set of positive integers $\{1, 2, \dots, p\}$ with $\mathbb{N}_p$. Given a set $J \subset \{1, 2, \dots, n\}$, the matrix $E_J \in \mathbb{R}^{|J|\times n}$ is the 0-1 matrix that is 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. This means that $E_J x$ is a $|J|$- dimensional vector with the components of x that correspond to the elements in J, and we denote this vector with x_J. With $x_l^{i,(k)}$ we denote the lth element of vector x^i at the kth iteration. Also given vectors x^i for $i = 1, \dots, N$, the column vector $(x^1, \dots, x^N)$ is all of the given vectors stacked.

2. NEWTON'S METHOD FOR SOLVING EQUALITY CONSTRAINT PROBLEMS

Consider the following equality-constrained optimization problem

$$\underset{x}{\text{minimize}} \quad F(x) \tag{1a}$$
$$\text{subj. to} \quad Ax = b, \tag{1b}$$

Algorithm 1 Newton's Method

Given $l = 0$, $\epsilon_{\text{nt}} > 0$ and $x^{(0)}$ such that $Ax^{(0)} = b$
repeat
  Compute $\Delta x_{\text{nt}}^{(l)}$ by solving (2)
  Compute $\lambda^{(l)}$
  if $(\lambda^{(l)})^2/2 \leq \epsilon_{\text{nt}}$ **then**
    $x^* = x^{(l)}$
    Terminate the iterations
  end if
  Compute the step size $\alpha^{(l)}$ using line search
  $x^{(l+1)} = x^{(l)} + \alpha^{(l)} \Delta x_{\text{nt}}^{(l)}$
  $l = l + 1$
until iterations are terminated

where $F : \mathbb{R}^n \to \mathbb{R}$ is convex and twice differentiable, and $A \in \mathbb{R}^{p\times n}$ with $\text{rank}(A) = p < n$. This problem can be solved iteratively using Newton's method, (Boyd and Vandenberghe, 2004, Ch. 10), where at each iteration the variables are updated as $x^{(l+1)} = x^{(l)} + \alpha^{(l)} \Delta x_{\text{nt}}^{(l)}$, with $\Delta x_{\text{nt}}^{(l)}$ and $\alpha^{(l)}$ denoting the so-called Newton direction and its corresponding step size, respectively. The Newton direction $\Delta x_{\text{nt}}^{(l)}$ is defined as the solution of

$$\underset{\Delta x}{\text{minimize}} \quad F(x^{(l)}) + \nabla F(x^{(l)})^T \Delta x + \frac{1}{2} \Delta x^T \nabla^2 F(x^{(l)}) \Delta x \tag{2a}$$
$$\text{subj. to} \quad A(x^{(l)} + \Delta x) = b, \tag{2b}$$

which constitutes a quadratic approximation of (1) at $x^{(l)}$, (Boyd and Vandenberghe, 2004, Ch. 10). Notice that if we assume that $Ax^{(l)} = b$ for all $l \geq 0$, then the constraint in (2b) would be $A\Delta x = 0$. The corresponding step size $\alpha^{(l)}$ is commonly computed using either exact or backtracking line search methods. The stopping/termination criterion for this iterative algorithm is based on the so-called Newton decrement which is defined as

$$\lambda^{(l)} = \sqrt{(\Delta x_{\text{nt}}^{(l)})^T \nabla^2 F(x^{(l)}) \Delta x_{\text{nt}}^{(l)}}, \tag{3}$$

and it provides an estimate of the sub-optimality of the iterates. Hence, Newton's method terminates when the Newton decrement falls below a given threshold ϵ. This iterative scheme is summarized in Algorithm 1.

3. INTERIOR-POINT METHOD

The optimization problem in (1) can be extended to include inequality constraints as

$$\underset{x}{\text{minimize}} \quad F(x) \tag{4a}$$
$$\text{subj. to} \quad Ax = b \tag{4b}$$
$$g_i(x) \leq 0 \quad i = 1, \dots, m, \tag{4c}$$

where $g_i : \mathbb{R}^n \to \mathbb{R}$ are all convex and twice differentiable. Also assume that the problem admits a strictly feasible solution, i.e., there exists x such that $Ax = b$ and $g_i(x) < 0$ for all $i = 1, \dots, m$. This allows us to solve (4) using an interior-point method which requires solving a sequence of equality-constraint problems, defined as

$$\underset{x}{\text{minimize}} \quad tF(x) + \sum_{i=1}^{m} \phi_i(x) \tag{5a}$$
$$\text{subj. to} \quad Ax = b, \tag{5b}$$

for increasing values of $t > 0$. The functions $\phi_i(x) := -\log(-g_i(x))$ are so-called logarithmic barrier functions

Algorithm 2 Interior-point Method

1: Given $q = 0$, $t^{(0)} > 0$, $\mu > 1$, $\epsilon_p > 0$ and feasible $x^{(0)}$
2: **repeat**
3: Compute $x^*(t^q)$ by solving (5) using Alg. 1 starting at $x^{(q)}$
4: $x^{(q+1)} = x^*(t^q)$
5: **if** $m/t^{(q)} < \epsilon_p$ **then**
6: $x^* = x^{(q+1)}$
7: Terminate the iterations
8: **end if**
9: $t^{(q+1)} = \mu t^{(q)}$
10: $q = q + 1$
11: **until** iterations are terminated

which (since $\phi(x) \to \infty$ as $g_i(x) \to 0$) virtually create barriers that prevent x from violating any of the inequality constraints in (4c). Notice that for every fixed value of t the problem in (5) is precisely in the form of (1), and it can be solved using Newton's method, as is described in Algorithm 1. After the Newton method has converged to a solution for a given t, $x^*(t)$, then within the interior-point method procedure, we gradually increase t and perform the same procedure again. This is done until m/t is below a certain threshold. This procedure is summarized in Algorithm 2. Next, we define loosely coupled problems and describe how we can take advantage of distributed computations for solving such problems using algorithms 1 and 2.

4. UNCONSTRAINED LOOSELY COUPLED OPTIMIZATION PROBLEMS

4.1 A Definition

Consider the following unconstrained optimization problem

$$\text{minimize} \quad F_1(x) + \cdots + F_N(x), \tag{6}$$

where the convex and twice differentiable functions F_i for $i = 1, \ldots, N$, only depend on a small subset of the elements of the variable x. Particularly, let us denote the ordered set of indices of variables that appear in F_i by J_i. We also denote the ordered set of indices of terms in the cost function that depend on x_i by $\mathcal{I}_i$, i.e., $\mathcal{I}_i = \{k \mid i \in J_k\}$. We call an optimization problem loosely coupled if $|\mathcal{I}_i| \ll N$ for all $i = 1, \ldots, n$. This problem can be rewritten as the following equality-constrained optimization problem

$$\underset{S,x}{\text{minimize}} \quad f_1(s^1) + \cdots + f_N(s^N) \tag{7a}$$

$$\text{subj. to} \quad \bar{E}x = S, \tag{7b}$$

where $S = (s^1, \ldots, s^N)$ and $\bar{E} = \begin{bmatrix} E_{J_1}^T & \cdots & E_{J_N}^T \end{bmatrix}^T$, with E_{J_i} a 0-1 matrix that is obtained from an identity matrix of order n by deleting the rows indexed by $\mathbb{N}_n \setminus J$. We refer to the constraints in (7b) as consistency constraints. Also f_is are lower dimensional descriptions of F_is such that $F_i(x) = f_i(E_{J_i}x)$ for all $x \in \mathbb{R}^n$ and $i = 1, \ldots, N$. Notice that if we define $F(S, x) = f_1(s^1) + \cdots + f_N(s^N)$ and $A = \begin{bmatrix} -I & \bar{E} \end{bmatrix}$, the problem in (7) is in the same form as the problem in (1). We can now form the quadratic approximation in (2) for this problem, as below

$$\underset{\Delta s^1, \ldots \Delta s^1, \Delta x}{\text{minimize}} \quad \sum_{i=1}^{N} f_i(s^{i,(l)}) + \nabla f_i(s^{i,(l)})^T \Delta s^i + \frac{1}{2}(\Delta s^i)^T \nabla^2 f_i(s^{i,(l)}) \Delta s^i \tag{8a}$$

$$\text{subj. to} \quad \bar{E}(x^{(l)} + \Delta x) = S^{(l)} + \Delta S, \tag{8b}$$

Algorithm 3 ADMM

1: Given $k = 0$, $\rho > 0$, $\epsilon_{\text{pri}}, \epsilon_{\text{dual}} > 0$, $y^{(0)}$ and $v^{(0)} = 0$
2: **repeat**
3: $x^{(k+1)} = \text{prox}_{\frac{1}{\rho}T_1}(y^{(k)} + v^{(k)})$
4: $y^{(k+1)} = \text{prox}_{\frac{1}{\rho}T_2}(x^{(k+1)} - v^{(k)})$
5: $v^{(k+1)} = v^{(k)} + (y^{(k+1)} - x^{(k+1)})$
6: **if** $\|y^{(k+1)} - x^{(k+1)}\|^2 < \epsilon_{\text{pri}}$ && $\|y^{(k+1)} - y^{(k)}\|^2 < \epsilon_{\text{dual}}$ **then**
7: Terminate the algorithm
8: **end if**
9: $k = k + 1$
10: **until** algorithm is terminated

where $\Delta S = (\Delta s^1, \Delta s^2, \ldots, \Delta s^N)$. Assuming that $S^{(l)} \in \mathcal{C}(\bar{E})$, (8b) only requires $\Delta S \in \mathcal{C}(\bar{E})$. This problem can then be rewritten as the following unconstrained non-smooth optimization problem

$$\underset{\Delta s^1, \ldots \Delta s^1}{\text{minimize}} \quad \sum_{i=1}^{N} f_i(s^{i,(l)}) + \nabla f_i(s^{i,(l)})^T \Delta s^i + \frac{1}{2}(\Delta s^i)^T \nabla^2 f_i(s^{i,(l)}) \Delta s^i + \mathcal{I}_{\mathcal{C}}(\Delta S), \tag{9}$$

where $\mathcal{I}_{\mathcal{C}}$ is the indicator function for the column space of $\bar{E}$. Considering the imposed structure in the cost function of the problem in (9), it can be solved distributedly using so-called proximal splitting, Combettes and Pesquet (2011); Boyd et al. (2011); Eckstein (1989), which is the subject of the next section.

4.2 Proximal Splitting Methods

In many applications we are faced with convex problems of the form

$$\text{minimize} \quad T_1(x) + T_2(x) \tag{10}$$

where minimizing the joint problem is much harder than solving problems including only individual terms of the cost function, Combettes and Pesquet (2011); Boyd et al. (2011); Khoshfetrat Pakazad et al. (2013). Proximal splitting algorithms enable us to solve the problem in (10) by solving minimization problems that are based on individual terms in the cost function. This is done through the use of the so-called proximity operators of these terms which are defined as follows. Given a closed convex function T, the proximity operator for this function, $\text{prox}_T : \mathbb{R}^n \to \mathbb{R}$, is defined as the unique minimizer of

$$\underset{y}{\text{minimize}} \quad T(y) + \frac{1}{2}\|x - y\|^2.$$

There are different classes of proximal splitting methods, Eckstein (1989). In this paper we, however, only consider the Alternating Direction Method of Multipliers (ADMM) which has been extensively used recently for design of distributed algorithms in many applications, e.g., see Boyd et al. (2011); Bertsekas and Tsitsiklis (1997). A general description of ADMM for the optimization problem in (10) is given in Algorithm 3. This algorithm can also accommodate a varying penalty parameter, ρ. This has shown to improve the convergence properties of the algorithm, and depending on the specifications of the problem there are different approaches for updating this parameter at each iteration, Boyd et al. (2011); Ghadimi et al. (2013). Also there exist several alternatives for the termination of the algorithm (the 6th step of Algorithm 3), see e.g.,

Boyd et al. (2011); Bertsekas and Tsitsiklis (1997). Proximal splitting algorithms and particularly ADMM can perform very well when applied to quadratic problems, Boley (2012), and as we will see later they can be implemented very efficiently. Next, we describe how ADMM enables us to distribute the computations for solving (7).

4.3 Distributed Computation of the Newton Direction

As was discussed in Section 4.1, the problems in (1) and (7) have the same structure, and hence we can solve (7) using the Newton method as described in Algorithm 1. Recall that in the Newton method, in order to compute the Newton direction at each iteration, we need to solve the corresponding quadratic approximation of the problem. As was also shown in Section 4.1, the quadratic approximation for the problem in (7), can be equivalently rewritten as in (9). Notice that (9) is in the same format as (10), where T_1 involves a summation of N decoupled terms and T_2 corresponds to $\mathcal{I}_{\mathcal{C}}$. With these definitions for T_1 and T_2 we now apply ADMM to (9). The 3rd and 4th steps of Algorithm 3 can then be written as

$$\begin{aligned}\Delta S^{(k+1)} &= \operatorname{prox}_{\frac{1}{\rho}T_1}(\Delta Y^{(k)} + v^{(k)}) \\ &= \underset{\Delta s^1,\ldots,\Delta s^N}{\arg\min} \quad \sum_{i=1}^{N} \bar{f}_i(\Delta s^i) + \frac{\rho}{2}\|\Delta y^{i,(k)} + v^{i,(k)} - \Delta s^i\|^2\end{aligned} \tag{11a}$$

$$\begin{aligned}\Delta Y^{(k+1)} &= \operatorname{prox}_{\frac{1}{\rho}\mathcal{I}_{\mathcal{C}}}(\Delta Y^{(k)} + v^{(k)}) = P_{\mathcal{C}}(\Delta S^{(k+1)} - v^{(k)}) \\ &= \bar{E}(\bar{E}^T\bar{E})^{-1}\bar{E}^T(\Delta S^{(k+1)} - v^{(k)}) = \bar{E}(\bar{E}^T\bar{E})^{-1}\bar{E}^T\Delta S^{(k+1)}\end{aligned} \tag{11b}$$

where

$$\bar{f}_i(\Delta s^i) = f_i(s^{i,(l)}) + \nabla f_i(s^{i,(l)})^T\Delta s^i + \frac{1}{2}(\Delta s^i)^T\nabla^2 f_i(s^{i,(l)})\Delta s^i$$

$$\Delta Y^k = (\Delta y^{1,(k)},\ldots,\Delta y^{N,(k)})$$

$$v^k = (v^{1,(k)},\ldots,v^{N,(k)})$$

and $v^{i,(k)} = v^{i,(k-1)} + (\Delta y^{i,(k)} - \Delta x^{i,(k)})$, which is given by the 5th step of Algorithm 3. Notice that the last equality in (11b) holds since if $v^{(0)} = 0$, then $v^{(k)} \in \mathcal{N}(\bar{E}(\bar{E}^T\bar{E})^{-1}\bar{E}^T)$ for all $k \geq 1$. As we see from (11a), the update for $\Delta S^{(k+1)}$ can be computed in a decentralized manner using N computing agents, where at each iteration every agent i calculates

$$\begin{aligned}\Delta s^{i,(k+1)} &= \underset{\Delta s^i}{\arg\min} \quad \bar{f}_i(\Delta s^i) + \frac{\rho}{2}\|\Delta y^{i,(k)} + v^{i,(k)} - \Delta s^i\|^2 \\ &= \left(\nabla^2 f_i(s^{i,(l)}) + \rho I\right)^{-1}\left(\rho(\Delta y^{(k)} + v^{i,(k)}) - \nabla f_i(s^{i,(l)})\right)\end{aligned} \tag{12}$$

where $\Delta y^{i,(k)}$ and $v^{i,(k)}$ are locally available to each agent. The update in (11b) can also be performed distributedly, which is explained as follows. Let $\Delta z^{(k+1)} = (\bar{E}^T\bar{E})^{-1}\bar{E}\Delta S^{(k+1)}$. The update rule in (11b) can then be rewritten as

$$\begin{aligned}\Delta Y^{(k+1)} &= (\Delta y^{1,(k+1)},\ldots,\Delta y^{N,(k+1)}) \\ &= \bar{E}\Delta z^{(k+1)} = (\Delta z_{J_1}^{(k+1)},\ldots,\Delta z_{J_N}^{(k+1)}).\end{aligned} \tag{13}$$

Notice that $\bar{E}^T\bar{E} = \operatorname{diag}(|\mathcal{I}_1|,\ldots,|\mathcal{I}_N|)$, and hence the update for each component, j, of Δx can be expressed as

$$\Delta z_j^{(k+1)} = \frac{1}{|\mathcal{I}_j|}\sum_{q\in\mathcal{I}_j}\left(E_{J_q}^T\Delta s^{q,(k+1)}\right)_j. \tag{14}$$

As a result, for each agent i to compute $\Delta y^{i,(k+1)} = \Delta z_{J_i}^{(k+1)}$, it needs to communicate to all agents in $\mathrm{Ne}(i) = \{j \mid J_i \cap J_j \neq \emptyset\}$ which are referred to as the neighbors of agent i. The ADMM-based Newton direction computation can then be summarized as in Algorithm 4.

Algorithm 4 ADMM-Based Newton Direction Computation

1: Given $k = 0$, $\rho > 0$, $\epsilon_{\text{pri}}, \epsilon_{\text{dual}} > 0$, $\Delta Y^0 \in \mathcal{C}(\bar{E})$ and $v^{(0)} = 0$
2: **repeat**
3: **for** $i = 1, 2, \ldots, N$ **do**
4: $\Delta s^{i,(k+1)} = \left(\nabla^2 f_i(s^{i,(l)}) + \rho I\right)^{-1}\left(\rho(\Delta y^{i,(k)} + v^{i,(k)}) - \nabla f_i(s^{i,(l)})\right)$
5: Communicate with all agents r belonging to $\mathrm{Ne}(i)$
6: **for** all $j \in J_i$ **do**
7: $\Delta z_j^{(k+1)} = \frac{1}{|\mathcal{I}_j|}\sum_{q\in\mathcal{I}_j}\left(E_{J_q}^T\Delta s^{q,(k+1)}\right)_j$
8: **end for**
9: $\Delta y^{i,(k+1)} = \Delta z_{J_i}^{(k+1)}$
10: $v^{i,(k+1)} = v^{i,(k)} + \left(\Delta y^{i,(k+1)} - \Delta s^{i,(k+1)}\right)$
11: Check whether $\|\Delta y^{i,(k+1)} - \Delta y^{i,(k)}\|^2 \leq \epsilon_{\text{dual}}/N$ and $\|\Delta s^{i,(k+1)} - \Delta y^{i,(k+1)}\|^2 \leq \epsilon_{\text{pri}}/N$
12: **end for**
13: **if** condition in step (10) satisfied for all $i = 1, \ldots, N$ **then**
14: $\Delta x_{\text{nt}} = \Delta z^{(k+1)}$
15: $\Delta S_{\text{nt}} = \Delta Y^{(k+1)}$
16: Terminate the algorithm
17: **end if**
18: $k = k + 1$
19: **until** algorithm is terminated

Notice that the termination condition of Algorithm 4 (based on the 11th and 13th steps of the algorithm) can be established distributedly, provided that all agents declare their status of convergence (step 11). Also observe that the satisfaction of this termination condition implies the satisfaction of the termination condition in Algorithm 3. There are other ways of establishing convergence of Algorithm 4 to a solution (and possibly more efficient, e.g., based on Iutzeler et al. (2012)). However, for the sake of brevity we abstain from discussing such methods in this paper.

Remark 1. In Algorithm 4, the computational effort at each iteration for each agent is dominated by the update of $\Delta s^{i,(k+1)}$ which requires factorizing the matrix $\nabla^2 f_i(s^{i,(l)}) + \rho I$. Notice that in case ρ is chosen to be constant for all k, this matrix will also be constant for all k. This means that each agent would only need to compute this factorization once at the first iteration of the algorithm and use the precomputed factorization in the remaining iterations. This pre-caching of the factorization significantly reduces the overall computational cost of computing the Newton direction using Algorithm 4. Notice that even if the penalty parameter is not chosen to be constant, it is still possible to utilize the method presented in (Liu et al., 2013, sec. 4.2), to use pre-caching of factorizations to achieve similar computational efficiency.

5. NEWTON'S METHOD WITH DISTRIBUTED STEP COMPUTATION

We can now combine Algorithm 4 with the Newton method expressed in Algorithm 1, to provide a distributed computational scheme for the problem in (6). This is presented in Algorithm 5. Similar to Algorithm 4, the termination condition for the Newton iterations in Algorithm 5 can also be checked in a distributed manner with limited communications among the agents. To this end, each agent needs to compute its local Newton decrement, $\lambda^{i,(l)}$, and declare its local convergence status based on this

Algorithm 5 ADMM-based Newton Method

Given $l=0$, $\epsilon_{\rm nt}>0$, $S^{(0)}, \Delta S_{\rm nt}^{(-1)} \in \mathcal{C}(\bar{E})$
repeat
  Compute $\Delta S_{\rm nt}^{(l)}$ using Algorithm 4 with starting point $\Delta Y^{(0)} = \Delta S_{\rm nt}^{(l-1)}$
  for $i=1,2,\dots,N$ **do**
    Compute $(\lambda^{i,(l)})^2 = (\Delta s_{\rm nt}^{i,(l)})^T \nabla^2 f_i(s^{i,(l)}) \Delta s_{\rm nt}^{i,(l)}$
    Check whether $(\lambda^{i,(l)})^2/2 \leq \epsilon_{\rm nt}/N$
  end for
  if condition in step (6) satisfied for all $i=1,\dots,N$ **then**
    $S^* = S^{(l)}$
    Terminate the algorithm
  end if
  Compute the step size $\alpha^{(l)}$ using line search
  for $i=1,2,\dots,N$ **do**
    $s^{i,(l+1)} = s^{i,(l)} + \alpha^{(l)} \Delta s_{\rm nt}^{i,(l)}$
  end for
  $l = l+1$
until algorithm is terminated

quantity (condition in step (6)). Although not explained in Algorithm 5, the step size computation in step (11) of the algorithm can also be performed distributedly. For this purpose, each agent would firstly need to perform a back tracking line search based on its local objective function, $\bar{f}_i(\Delta s^i)$, and compute a suitable local step size. The step size $\alpha^{(l)}$ will then be chosen as the smallest of the local step sizes, which can be derived using max/min consensus algorithms, Iutzeler et al. (2012). A similar approach has also been used in Wei et al. (2013).

Remark 2. Having defined Algorithm 5, some comments are in order. Firstly, notice that in the 3rd step of Algorithm 5, the ADMM iterations for computing the Newton direction, are warm-started using the computed Newton direction from the previous Newton iteration. In case the line search is not done too aggressively this can potentially reduce the number of required ADMM iterations for finding the next Newton direction. Secondly, observe that the computed Newton directions $\Delta S_{\rm nt}^{(l)} \in \mathcal{C}(\bar{E})$ for all $l \geq 0$ and hence, $S^{(l)} \in \mathcal{C}(\bar{E})$ for all $l \geq 0$. This means that the consistency constraints in (7b) are always satisfied.

So far we have proposed a distributed scheme for solving unconstrained loosely coupled problems. In the upcoming section, we extend the definition of loosely coupled problems to that of constrained ones and show how we can derive similar algorithms to solve such problems.

6. CONSTRAINED LOOSELY COUPLED OPTIMIZATION PROBLEMS

We now extend the definition of loosely coupled problems (provided in Section 4.1) by first adding convex inequality constraints as

$$\begin{aligned} \text{minimize} \quad & F_1(x) + \cdots + F_N(x) \\ \text{subj. to} \quad & G_i(x) \leq 0 \quad i=1,\dots,N \end{aligned} \tag{15}$$

where we assume that the function pairs F_i, G_i for $i=1,\dots,N$, are only dependent on a small subset of the elements of the variable x. Also let the description of the coupling among the function pairs be described as was for the problem in (6). Then this problem can be written as

$$\underset{S,x}{\text{minimize}} \quad f_1(s^1) + \cdots + f_N(s^N) \tag{16a}$$
$$\text{subj. to} \quad g_i(s^i) \leq 0, \quad i=1,\dots,N \tag{16b}$$
$$\bar{E}x = S \tag{16c}$$

Algorithm 6 ADMM-based Interior-point Method

Given $q=0$, $t^{(0)}>0$, $\mu>1$, $\epsilon_p>0$ and feasible $S^{(0)}$
repeat
  Compute $S^*(t^{(q)})$ by solving (5) using Alg. 5 starting at $S^{(q)}$
  $S^{(q+1)} = S^*(t^{(q)})$
  if $m/t^{(q)} < \epsilon_p$ **then**
    $S^* = S^{(q+1)}$
    Terminate the iterations
  end if
  $t^{(q+1)} = \mu t^{(q)}$
  $q = q+1$
until iterations are terminated

where the functions f_i and g_i are defined similarly as in Section 4.1 for the problem in (7). Notice that this problem is in the same format as (4) and can be solved using the interior-point method, which accordingly requires solving

$$\begin{aligned} \text{minimize} \quad & \underbrace{tf_1(s^1) - \log(-g_1(s^1))}_{h_1(s^1)} + \cdots + \underbrace{tf_N(s^N) - \log(-g_N(s^N))}_{h_N(s^N)} \\ \text{subj. to} \quad & \bar{E}x = S \end{aligned} \tag{17}$$

for a sequence of increasing t. For every given t then the problem in (17) can be solved using Algorithm 5, with a modification to step 3 of the algorithm. Particularly, the 4th step of Algorithm 4 (that is used in step 3 of Algorithm 5) needs to be modified as

$$\Delta s^{i,(k+1)} = \left(\nabla^2 h_i(s^{i,(l)}) + \rho I\right)^{-1} \left(\rho(\Delta y^{i,(k)} + v^{i,(k)}) - \nabla h_i(s^{i,(l)})\right)$$

where

$$\nabla^2 h_i(s^{i,(l)}) = \nabla^2 f_i(s^{i,(l)}) + \frac{1}{g_i(s^{i,(l)})^2} \nabla g_i(s^{i,(l)}) \nabla g_i(s^{i,(l)})^T - \frac{1}{g_i(s^{i,(l)})} \nabla^2 g_i(s^{i,(l)})$$

and $\nabla h_i(s^{i,(l)}) = \nabla f_i(s^{i,(l)}) - \frac{1}{g_i(s^{i,(l)})} \nabla g_i(s^{i,(l)})$. Notice that after this modification remarks 1 and 2 still apply. The distributed scheme for solving (15) can then be obtained by combining the modified Algorithm 5 with Algorithm 2. This is summarized in Algorithm 6. The problem in (15) can be further extended by adding equality constraints. Assuming that the equality constraints also enjoy similar type of coupling as above, the problem can be written as

$$\underset{S,x}{\text{minimize}} \quad f_1(s^1) + \cdots + f_N(s^N) \tag{18a}$$
$$\text{subj. to} \quad g_i(s^i) \leq 0, \quad i=1,\dots,N \tag{18b}$$
$$A^i s^i = b^i \quad i=1,\dots,N \tag{18c}$$
$$\bar{E}x = S \tag{18d}$$

where $A^i \in \mathbb{R}^{p_i \times |J_i|}$ and $\text{rank}(A^i) = p_i < |J_i|$. This problem can also be solved using Algorithm 6 where in its 3rd step, Algorithm 5 is applied to

$$\begin{aligned} \text{minimize} \quad & \underbrace{tf_1(s^1) - \log(-g_1(s^1))}_{h_1(s^1)} + \cdots + \underbrace{tf_N(s^N) - \log(-g_N(s^N))}_{h_N(s^N)} \\ \text{subj. to} \quad & A^i s^i = b^i \quad i=1,\dots,N \\ & \bar{E}x = S \end{aligned}$$

This in turn requires another modification to Algorithm 4, that is used in the 3rd step of Algorithm 5. Specifically, the 4th step of Algorithm 4 should be changed to

$$\Delta s^{i,(k+1)} = \underset{A^i \Delta s^i = 0}{\arg\min} \left\{ \nabla h_i(s^{i,(l)})^T \Delta s^i + \frac{1}{2} (\Delta s^i)^T \nabla^2 h_i(s^{i,(l)}) \Delta s^i + \frac{\rho}{2} \left\| \Delta y^{i,(k)} + v^{i,(k)} - \Delta s^i \right\|^2 \right\} \tag{19}$$

Solving this problem is equivalent to solving the following linear system of equations

$$\begin{bmatrix} \nabla^2 h_i(s^{i,(l)}) + \rho I & (A^i)^T \\ A^i & 0 \end{bmatrix} \begin{bmatrix} \Delta s^i \\ u \end{bmatrix} = \begin{bmatrix} \rho(\Delta y^{i,(k)} + v^{i,(k)}) - \nabla h_i(s^{i,(l)}) \\ 0 \end{bmatrix}. \quad (20)$$

The computational cost for this is dominated by the cost for calculating the factorization of the coefficient matrix. Notice that, similar to (12), this coefficient matrix is also constant within the ADMM iterations and hence, the comments made in Remark 1 still apply. Even if ρ is varying similar techniques as in Liu et al. (2013) can be applied.

Remark 3. Notice that the resulting solution from Algorithm 6, S^*, satisfies the local constraints, i.e., $g_i(s^{i,*}) \leq 0$ and $A^i s^{i,*} = b^i$ for all $i = 1, \ldots, N$. However, S^* does not necessarily satisfy the consistency constraints. In fact it is possible to upper-bound the consistency error in S^* as follows. Assume that $S^{(0)}$ is both locally feasible and consistent. Then the consistency error of $S^{(1)}$ can be computed as

$$\begin{aligned} e_c(1) &= \left\| S^{(1)} - \bar{E}(\bar{E}^T\bar{E})^{-1}\bar{E}^T S^{(1)} \right\|^2 \\ &= \sum_{j=0}^{l_{\max}(1)} (\alpha^{(j,1)})^2 \left\| \Delta S^{j,1} - \bar{E}(\bar{E}^T\bar{E})^{-1}\bar{E}^T \Delta S^{(j,1)} \right\|^2 \\ &\leq \sum_{j=0}^{l_{\max}(1)} (\alpha^{(j,1)})^2 \epsilon_{\text{pri}}, \end{aligned} \quad (21)$$

where $\Delta S^{(j,1)}$ and $\alpha^{(j,1)}$ denote the Newton direction the step size used in the jth Newton iteration in the 1st iteration of the interior-point method, respectively. Also $l_{\max}(1)$ denotes the number of required Newton iterations to converge to $S^{(1)}$. Now assuming that Algorithm 6 requires $q_{\max}$ iterations to converge, we can bound the consistency error of S^* as $e_c(q_{\max}) \leq \sum_{q=1}^{q_{\max}} \sum_{j=0}^{l_{\max}(q)} (\alpha^{(j,q)})^2 \epsilon_{\text{pri}}$.

Remark 4. The proposed methods in this paper are closely related to inexact Newton methods, where the Newton direction is computed in a distributed manner. This indicates that the methods described in this paper, can potentially suffer from slow convergence when the iterates are very close to the optimal solution. The similarities between the two classes of methods can also give us insight on the convergence properties of the proposed methods and how to avoid slow convergence near optimal solution.

7. CONCLUSIONS

In this paper, we proposed distributed optimization algorithms for solving unconstrained and constrained loosely coupled optimization problems. These algorithms are based on the Newton and interior-point methods where we used the inherent structure in the problem to distribute the required computations within these methods. Particularly, by exploiting the coupling in the problem and using proximal splitting methods we showed how the Newton direction can be computed in a distributed manner.

Notice that the proposed methods in this paper, all require a feasible starting point. As a future research directions, we intend to extend the proposed algorithms to also accommodate infeasible starting points. Also we intend to investigate primal-dual interior-point methods.

REFERENCES

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

D. Boley. Local linear convergence of ADMM on quadratic or linear programs. Technical report, Department of Computer Science and Engineering, University of Minnesota, 2012.

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.

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

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

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.

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.

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.

S. Khoshfetrat Pakazad, M. S. Andersen, and A. Hansson. Distributed Solutions for Loosely Coupled Feasibility Problems Using Proximal Splitting Methods. *ArXiv e-prints*, 2013.

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

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):922–938, April 2010.

H. Ohlsson, T. Chen, S. Khoshfetrat Pakazad, L. Ljung, and S. Shankar Sastry. Scalable Anomaly Detection in Large Homogenous Populations. *ArXiv e-prints*, September 2013.

T.H. Summers and J. Lygeros. Distributed model predictive consensus via the alternating direction method of multipliers. In *50th Annual Allerton Conference on Communication, Control, and Computing*, pages 79–84, 2012.

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.