Distributed Solutions of Convex Feasibility Problems with Sparsely Coupled Constraints

Yingying Xiao, Jianghai Hu

Abstract— **In this paper, distributed algorithms for solving the convex feasibility problem with sparse constraints are proposed. The proposed algorithms exploit the sparsity of the constraints to reduce the storage and communication requirements for individual agents: each agent only maintains its own variable together with its desired values for the variables of those neighboring agents whose valuations help determining its feasibility; at each iteration, each agent carries out projection and consensus operations based on information received from only the relevant neighboring agents. We show that the proposed algorithms converge asymptotically to a feasible solution starting from any initial guess and, under some further assumptions, the convergence speed is exponential. The algorithms' effectiveness is demonstrated through the simulation results on several application examples.**

I. INTRODUCTION

The convex feasibility problem (CFP), also called the convex intersection problem, is the problem of finding a common point that belongs to a family of nonempty closed convex sets. A well known problem in applied mathematics, CFP has found a wide range of practical applications, e.g., image recovery [1], model predictive control [2], network localization (with or without measurement errors) [3], [4], to name a few.

There has been a tremendous amount of existing literature on the solution of CFP. A majority of existing approaches, especially the earlier ones, are centralized in that a central solver updates a guess of the solution iteratively to satisfy all the convex constraints simultaneously. A particular popular class of such approaches is the alternative projection method and its variants (e.g. [5], [6]). Centralized solution algorithms of CFP have the advantage of easy implementation and guaranteed convergence. On the other hand, they often scale poorly as the number of constraints increases.

Recently, due to the advent of distributed optimization and consensus algorithms, distributed solution of CFP has increasingly attracted the attention of control community. A noteworthy effort toward this direction is the distributed algorithms for solving linear equations as proposed in [7], [8], [9] and their extensions to solving nonlinear equations [10], [11] and using approximate projections [12]. Some of the earlier work on general CFP along this direction can be found in [13], [14], [15]. CFP can also be formulated as distributed optimization problems; in this context, relevant approaches can be found in, e.g., [16]. In these distributed algorithms, the convex constraints are partitioned and assigned to a group of agents, each of which maintains a local guess of the solution. These local guesses are updated by the agents individually to satisfy their own constraints and to reach consensus via inter-agent information exchanges.

A drawback of the aforementioned algorithms is that they require each agent to keep and broadcast to its neighbors a vector whose dimension is the same as that of the solution to the problem, which can result in excessive storage and communication requirements on individual agents. For example, the network could consist of hundreds or even thousands of agents with a comparable number of variables to be solved. One way to mitigate this issue is to partition not only the constraints, but also the solution vector, into different parts and assign them to individual agents. Two approaches along this direction are proposed in [17] and [18] for solving linear algebraic equations. In [17], each agent partitions its copy of the solution into multiple blocks and broadcasts periodically or randomly only one of them to its neighbors. In [18], the sparsity of the matrix A in the equation $Ax = b$ is exploited and each agent keeps and broadcasts only the part of the solution x relevant to its own constraints, which often has a much reduced dimension than x.

In this paper, we adopt a similar approach to [18] to study the distributed solution of the more general CFP whose constraints are sparse: each constraint involves only a relatively few number of the variables in the solution vector. In many application examples, e.g., the network localization problem (see Section III-B), the sparsity comes naturally (inter-agent measurements are available only for agents sufficiently close to each other).

This material is based upon work supported by the National Science Foundation under Grant No. 1329875.

School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA {xiao106,jianghai}@purdue.edu

In our proposed algorithms, each agent maintains a local variable for itself as well as a set of variables representing its desired values for some other relevant agents' local variables. In each iteration, an agent sends its local variable only to those agents whose constraints depend on it; and send its desired values only to those agents whose variables help determining its local feasibility. This significantly reduces the amount of storage and communication required for individual agents. To update its local variable, each agent alternatively performs a projection and a consensus operation. We show that the algorithms converge to a feasible solution asymptotically and, under further assumptions, exponentially fast.

The remainder of this paper is organized as follows. The problem to be studied is formulated in Section II. Several potential applications are given in Section III. Section IV presents the proposed distributed algorithms; their convergence analysis is given in Section V. Section VI shows the simulation results of some application examples. Finally, Section VII concludes the paper.

II. PROBLEM FORMULATION

Consider a set of m agents indexed by $\mathcal{I} = \{1, \dots, m\}$. Assume each agent $i \in \mathcal{I}$ maintains a (local) variable $x_i \in \mathbb{R}^{n_i}$ of its own, which needs to satisfy a constraint of the following form:

$$x_i \in \mathcal{D}_i \left((x_j)_{j \in \mathcal{N}_i^+} \right). \tag{1}$$

Here, $\mathcal{N}_i^+ \subset \mathcal{I} \backslash \{i\}$ is a set of agents whose variables are needed to determine the feasible set of x_i; $(x_j)_{j \in \mathcal{N}_i^+}$ is the stacked vector of all the variables of agents in $\mathcal{N}_i^+$; and $\mathcal{D}_i \left((x_j)_{j \in \mathcal{N}_i^+} \right)$ is a closed subset of $\mathbb{R}^{n_i}$ which may vary with $(x_j)_{j \in \mathcal{N}_i^+}$. Equivalently, the constraint (1) can be written as

$$\left(x_i, (x_j)_{j \in \mathcal{N}_i^+} \right) \in \mathcal{F}_i, \tag{2}$$

where $\mathcal{F}_i$ is a suitably chosen closed subset of the product space of x_i and $(x_j)_{j \in \mathcal{N}_i^+}$.

Remark 1: The constraint (1) on the variable of agent i is in general non-local as it depends on the variables of other agents in $\mathcal{N}_i^+$. In the case $\mathcal{N}_i^+ = \emptyset$, the feasible set $\mathcal{D}_i$ of x_i becomes a (fixed) subset of $\mathbb{R}^{n_i}$ and the constraint on x_i becomes a local constraint.

A directed graph $\mathcal{G}_d$, called the *dependency graph*, can be constructed to represent the interdependency of the feasibility of the agents' variables: $\mathcal{G}_d$ has the vertex set $\mathcal{I}$, and an edge from j to i whenever the feasible set of x_i depends on x_j. Note that there is no self loops in $\mathcal{G}_d$. The aforementioned set $\mathcal{N}_i^+$ is exactly the in-neighborhood of vertex i in $\mathcal{G}_d$; thus we call agents indexed by $\mathcal{N}_i^+$ the *in-neighbors* of agent i. Similarly, the out-neighborhood of vertex i in $\mathcal{G}_d$, denoted by $\mathcal{N}_i^- \subset \mathcal{I} \setminus \{i\}$, indexes the *out-neighbors* of agent i, namely, agents whose variables' feasibility depends (at least partially) on the value of x_i. The two neighborhoods $\mathcal{N}_i^+$ and $\mathcal{N}_i^-$ may intersect or even be identical (see Example 1 below). Denote $\mathcal{N}_i := \mathcal{N}_i^+ \cup \mathcal{N}_i^-$.

Example 1: The linear equation $Ax = b$ with

$$A = \left[\begin{array}{cc|c} 1 & 0 & -1 \\ \hline 1 & 1 & 1 \\ 0 & 1 & 1 \end{array}\right] \text{ and } b = \left[\begin{array}{c} 0 \\ \hline 0 \\ -1 \end{array}\right] \tag{3}$$

has a unique solution $x^* = A^{-1}b = (1, -2, 1)$. Partition $x \in \mathbb{R}^3$ into $x = (x_1, x_2)$ where $x_1 = (z_1, z_2) \in \mathbb{R}^2$ and $x_2 \in \mathbb{R}$ are the variables of agents 1 and 2, respectively. With the row partitions of A and b in (3), the private constraint of agent 1 is $\begin{bmatrix} 1 & 0 \end{bmatrix} x_1 - x_2 = 0$, i.e., $z_1 = x_2$, which is under constrained for determining x_1. The private constraint of agent 2 is

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_1 + \begin{bmatrix} 1 \\ 1 \end{bmatrix} x_2 = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \quad \Leftrightarrow \quad \begin{cases} x_2 = -z_1 - z_2 \\ x_2 = -z_2 - 1, \end{cases}$$

which is overly constrained for determining x_2. The neighbor sets of the two agents are given by $\mathcal{N}_1^+ = \mathcal{N}_1^- = \{2\}$ and $\mathcal{N}_2^+ = \mathcal{N}_2^- = \{1\}$.

The agent *communication graph* $\mathcal{G}_c$ is the directed graph with the vertex set $\mathcal{I}$ and an edge set so that an edge from j to i exists whenever agent i can receive information from agent j via direct communication.

Assumption 1 (Communicability): $\mathcal{G}_d$ and its transpose $\mathcal{G}_d^T$ are both subgraphs of $\mathcal{G}_c$. Here, $\mathcal{G}_d^T$ is the graph obtained by reversing the direction of every edge of $\mathcal{G}_d$.

Assumption 1 implies that each agent can have two-way communications (i.e. send information to and receive information from) with any of its in-neighbors and out-neighbors. In other words, two agents can communicate directly between each other whenever the feasibility of one's variable depends on the other. Through communications, agents can share their variables, while keeping their respective constraints private.

In this paper we study the following problem.

Problem 1 (Distributed Feasibility Problem): Design distributed algorithms consistent with the communication graph and maintaining the privacy of individual agents' constraints so that a valuation of $(x_i)_{i \in \mathcal{I}}$ can be (asymptotically) obtained that satisfies the private constraints of all agents.

Denote $x := (x_i)_{i \in \mathcal{I}} \in \mathbb{R}^n$ where $n = \sum_{i \in \mathcal{I}} n_i$. The following assumptions are made throughout this paper.

Assumption 2 (Feasibility): There exists at least one valuation of x, which we denote by $x^* := (x_i^*)_{i \in \mathcal{I}}$, that satisfies all the constraints (2).

Assumption 3 (Convexity): In constraint (2) the set $\mathcal{F}_i$ is convex for each $i \in \mathcal{I}$.

As a result of Assumption 3, the feasible set $\mathcal{D}_i\left((x_j)_{j\in\mathcal{N}_i^+}\right)$ in constraint (1) is also convex.

III. APPLICATION EXAMPLES

Two instances of Problem 1 are presented below.

A. Distributed Solution of Linear Programs/Equations

Let $A \in \mathbb{R}^{\ell\times n}$, $b \in \mathbb{R}^\ell$ be such that the linear program $Ax \le b$ has at least one feasible solution x^*. Suppose that different portions of the variable x and the inequalities are held separately by a group of agents indexed by $\mathcal{I}$, i.e., there exist the block partitions $x = (x_1, \cdots, x_m)$,

$$\left[\begin{array}{c|c} A & B \end{array}\right] = \left[\begin{array}{ccc|c} A_{11} & \cdots & A_{1m} & b_1 \\ \vdots & \ddots & \vdots & \vdots \\ A_{m1} & \cdots & A_{mm} & b_m \end{array}\right]$$

so that agent $i \in \mathcal{I}$ has n_i variables, $x_i \in \mathbb{R}^{n_i}$, and ℓ_i private linear inequality constraints,

$$A_{i1}x_1 + \cdots + A_{im}x_m \le b_i \in \mathbb{R}^{\ell_i}. \tag{4}$$

Here, we assume $n_i, \ell_i \ge 1$ with $\sum_i n_i = n$ and $\sum_i \ell_i = \ell$; and "$\le$" denotes entry-wise comparison. The sets of in-neighbors and out-neighbors of agent i are $\mathcal{N}_i^+ = \{j \in \mathcal{I} \,|\, A_{ij} \ne 0\}$ and $\mathcal{N}_i^- = \{j \in \mathcal{I} \,|\, A_{ji} \ne 0\}$, respectively, and the constraint (4) can be recast as

$$A_{ii}x_i + \textstyle\sum_{j\in\mathcal{N}_i^+} A_{ij}x_j \le b_i. \tag{5}$$

Distributed solution of the above linear program (and as a special case, the linear equation $Ax = b$) is an instance of Problem 1 (see Example 1).

Example 2: Consider the linear program $-\varepsilon\mathbf{1} \le Ax - b \le \varepsilon\mathbf{1}$ with $x = (x_1, x_2, x_3) \in \mathbb{R}^3$,

$$A = \left[\begin{array}{c|c|c} 1 & 0 & -1 \\ \hline 0 & 0 & 1 \\ \hline 0 & 1 & 1 \end{array}\right], \quad b = \left[\begin{array}{c} 1 \\ \hline -1 \\ \hline 1 \end{array}\right], \quad \mathbf{1} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}. \tag{6}$$

There are three agents with the variables $x_1, x_2, x_3 \in \mathbb{R}$ and the following private constraints, respectively: $|x_1 - x_3 - 1| \le \varepsilon$ for agent 1; $|x_3 + 1| \le \varepsilon$ for agent 2; and $|x_2 + x_3 + 1| \le \varepsilon$ for agent 3. Their neighbor sets are $\mathcal{N}_1^+ = \{3\}$, $\mathcal{N}_1^- = \emptyset$; $\mathcal{N}_2^+ = \mathcal{N}_2^- = \{3\}$; $\mathcal{N}_3^+ = \{2\}$ and $\mathcal{N}_3^- = \{1, 2\}$. Note that the constraint of agent 2 does not involve its own variable x_2, which is allowed in our problem formulation. Further, $\bar{x} = A^{-1}b = (0, 2, -1)$ is a feasible solution for any $\varepsilon \ge 0$.

B. Network Localization

Consider a group of agents (sensors, robots, vehicles) deployed on $\mathbb{R}^2$ with unknown locations $x_i \in \mathbb{R}^2$, $i \in \mathcal{I}$. Suppose each agent $i \in \mathcal{I}$ is equipped with sensors that can measure the relative distance and/or orientation of some other agents $j \in \mathcal{N}_i^+$ within its sensing range:
(i) *Relative orientation (Angle-of-Arrival) measurement*: the direction of the vector $x_j - x_i$ is measured against a compass onboard agent i. This imposes a constraint as $\angle(x_j - x_i) \in \Theta_{ij}$, where $\angle$ denotes the phase angle and Θ_{ij} is a singleton $\{\theta_{ij}\}$ if the measurement is precise and an interval $[\theta_{ij} - \delta, \theta_{ij} + \delta]$ if the measurement is imprecise. Equivalently, we have $x_i \in \mathcal{D}_{ij}(x_j)$ where $\mathcal{D}_{ij}(x_j)$ is a ray or a cone pivoted at x_j.
(ii) *Relative distance measurement*: the distance $\|x_j - x_i\|$ is measured using, e.g., the strength of signal received by agent i from agent j. This incurs a constrain as $r_1 \le \|x_i - x_j\| \le r_2$, i.e., $x_i \in \mathcal{D}_{ij}(x_j)$ where $\mathcal{D}_{ij}(x_j)$ is a ring-shaped region centered at x_j.

The private constraint of agent i consists of all the above constraints, $x_i \in \mathcal{D}_i := \cap_{j\in\mathcal{N}_i^+}\mathcal{D}_{ij}(x_j)$. The network localization problem is to find the locations of all agents consistent with measurement data. This is an instance of Problem 1 if $r_1 = 0$.

IV. PROPOSED ALGORITHMS

We now describe the algorithms that can iteratively solve the CFP. In these algorithms, in addition to its own variable x_i, each agent i will maintain an additional set of variables, one for each of its in-neighbors: x_j^i, $\forall j \in \mathcal{N}_i^+$. The variable x_j^i is the value of the agent j's variable as *desired* by agent i, which could differ from the actual value of x_j. See Fig. 1 for an example. Define

$$\mathbf{x}_i := \left(x_i, (x_j^i)_{j\in\mathcal{N}_i^+}\right),$$

which consists of all the variables maintained by agent i. We call $\mathbf{x}_i$ the *augmented variable* of agent i. The totality of all $\mathbf{x}_i$'s is denoted by $\mathbf{x} := (\mathbf{x}_i)_{i\in\mathcal{I}}$. For a sparse graph $\mathcal{G}_d$, we have $\dim(\mathbf{x}) \ll mn$.

Instead of Problem 1, the proposed algorithms will try to solve the following problem:

Problem 2: Design distributed algorithms consistent with the communication graph $\mathcal{G}_c$ so that a valuation of $\mathbf{x}$ is asymptotically obtained that satisfies, for all $i \in \mathcal{I}$,

$$\mathbf{x}_i \in \mathcal{F}_i, \tag{7}$$

$$x_i = x_i^j, \quad \forall j \in \mathcal{N}_i^-. \tag{8}$$

The constraint (7) is simply (2) with x_j replaced by x_j^i and is localized as it involves agent i's augmented variable only. The constraint (8) is the consensus constraint that specifies each agent's variable to be exactly the same as desired by its out-neighbors.

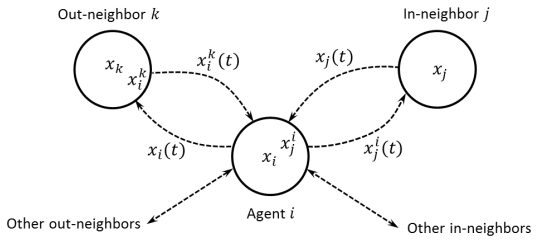


Fig. 1: Information exchanges at round t of the algorithms. For the variable x_i kept by agent i, agent i sends its actual value $x_i(t)$ to and receives its desired values $x_i^j(t)$ from its out-neighboring agents k. Agent i also sends to its in-neighboring agents j its desired values $x_j^i(t)$ of their variables x_j and receives from them their actual values $x_j(t)$.

It is easy to see that any $\mathbf{x}$ satisfying both (7) and (8) for all i corresponds to a solution x to Problem 1, and vice versa. Hence, Problem 2 is equivalent to Problem 1. By Assumption 2, the existence of a feasible solution x^* for Problem 1 implies the corresponding feasible solution $\mathbf{x}^* = (\mathbf{x}_i^*)_{i\in\mathcal{I}}$ for Problem 2.

The proposed algorithms are carried out in rounds. At rounds $t = 0, 1, \ldots$, each agent i in a certain set $\mathcal{I}_t \subset \mathcal{I}$ collects from its in-neighbors the current values of their variables, $(x_j(t))_{j\in\mathcal{N}_i^+}$, to compute a local feasible solution $\mathbf{z}_i(t) = T_i(\mathbf{x}_i(t), (x_j(t))_{j\in\mathcal{N}_i^+})$ and then from its out-neighbors their desired values of x_i, $(z_i^k(t))_{k\in\mathcal{N}_i^-}$, to update $\mathbf{x}_i$ according to $\mathbf{x}_i(t+1) = Q_i(z_i(t), (z_i^k(t))_{k\in\mathcal{N}_i^-})$. Here, T_i and Q_i are operators to be defined later on so that compared to $\mathbf{x}_i(t)$, $\mathbf{x}_i(t+1)$ is closer to satisfying the constraints (7) and (8).

A. Synchronous Algorithm

In the synchronous algorithm, all the agents update their augmented variables simultaneously and in parallel at each iteration according to (**??**). The update operator T_i for agent i at iteration t consists of three steps:

(i) First, agent i receives $(x_j(t))_{j\in\mathcal{N}_i^+}$ from its in-neighbors and update $\mathbf{x}_i(t)$ to $\mathbf{y}_i(t) = \left(y_i(t), (y_j^i(t))_{j\in\mathcal{N}_i^+}\right)$ where

$$y_i(t) = x_i(t), \;\; y_j^i(t) = x_j(t), \; \forall j \in \mathcal{N}_i^+. \quad (9)$$

(ii) Then, based on $\mathbf{y}_i(t)$ and the feasible set $\mathcal{F}_i$, agent i computes $\mathbf{z}_i(t) = S_1(\mathbf{y}_i(t))$ according to

$$\mathbf{z}_i(t) = (1-\alpha_i)\mathbf{y}_i(t) + \alpha_i \cdot \mathcal{P}_{\mathcal{F}_i}(\mathbf{y}_i(t)). \quad (10)$$

Here, $\mathcal{P}_{\mathcal{F}_i}$ denotes the orthogonal projection operator onto $\mathcal{F}_i$ and $\alpha_i \in (0, 2)$ is a constant.

(iii) Finally, agent i receives $(z_i^k(t))_{k\in\mathcal{N}_i^-}$ from its out-neighbors and updates its augmented variable to

$$x_i(t+1) = \tfrac{1}{|\mathcal{N}_i^-|+1}\left(z_i(t) + \textstyle\sum_{k\in\mathcal{N}_i^-} z_i^k(t)\right),$$
$$x_j^i(t+1) = z_j^i(t), \quad \forall j \in \mathcal{N}_i^+. \quad (11)$$

Intuitively (10) and (11) help to improve the $\mathbf{x}_i$'s satisfaction of the constraints (7) and (8), respectively. The synchronous algorithm is summarized in Algorithm 1.

Algorithm 1 Synchronous Algorithm

Initialize $\mathbf{x}_i(0)$, $\forall i \in \mathcal{I}$, and let $t \leftarrow 0$;
repeat
  for all $i \in \mathcal{I}$ **do**
    Agent i receives $x_j(t)$ from all in-neighbors j;
    Agent i computes $\mathbf{y}_i(t)$ according to (9);
    Agent i computes $\mathbf{z}_i(t)$ according to (10);
  end for
  for all $i \in \mathcal{I}$ **do**
    Agent i receives $z_i^k(t)$ from all out-neighbors k;
    Agent i computes $\mathbf{x}_i(t+1)$ according to (11);
  end for
  $t \leftarrow t+1$;
until certain convergence criteria are met
Return $\mathbf{x}(t)$.

Remark 2: If agent i has no out-neighbors, i.e., $\mathcal{N}_i^- = \emptyset$, then (11) is trivial: $\mathbf{x}_i(t+1) = \mathbf{z}_i(t)$.

The convergence property of the above algorithm will be established in Section V.

B. Asynchronous Algorithm

Suppose the set $\mathcal{I}_t$ of agents updating their augmented variables at each round t has the property that it does not contain any pair i, j neighboring to each other in the dependency graph $\mathcal{G}_d$. For instance, $\mathcal{I}_t$ may consist of only one agent for each t. Under the above assumption, an asynchronous algorithm is described in Algorithm 2.

V. CONVERGENCE ANALYSIS

We first introduce some useful notions.

Definition 1 ([19]): A continuous map $T : \mathbb{R}^n \to \mathbb{R}^n$ is called a paracontraction w.r.t. a norm $\|\cdot\|$ on $\mathbb{R}^n$ if $\|T(x)-y\| < \|x-y\|$ for any $x \notin \mathcal{A}$ and $y \in \mathcal{A}$, where $\mathcal{A} := \{y \,|\, T(y) = y\}$ is the set of fixed points of T.

The projection operator $\mathcal{P}_{\mathcal{F}}$ onto a nonempty closed convex set $\mathcal{F}$ is a paracontraction w.r.t. the Euclidean norm. Indeed, $(1-\alpha)\cdot\mathrm{Id} + \alpha\cdot\mathcal{P}_{\mathcal{F}}$ where Id is the identity map is also a paracontraction for any $\alpha \in (0, 2)$ [19].

The following fact will be useful later on. Its proof is straightforward hence omitted.

Algorithm 2 Asynchronous Algorithm

Initialize $\mathbf{x}_i(0)$, $\forall i \in \mathcal{I}$, and set $t \leftarrow 0$;
repeat
  for all $i \in \mathcal{I}_t$ **do**
    Agent i computes $\mathbf{z}_i(t)$ with $\mathbf{y}_i(t)$ replaced by $\mathbf{x}_i(t)$ in (10);
    Agent i receives $x_i^k(t)$ from all out-neighbors k;
    $x_i(t+1) \leftarrow \frac{1}{|\mathcal{N}_i^-|+1}\left(z_i(t) + \sum_{k\in\mathcal{N}_i^-} x_i^k(t)\right)$;
    for all $k \in \mathcal{N}_i^-$ **do**
      $x_i^k(t+1) \leftarrow x_i(t+1)$;
    end for
  end for
  $t \leftarrow t+1$;
until certain convergence criteria are met
Return $\mathbf{x}(t)$.

Lemma 1: Suppose $T_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$ is a paracontractions w.r.t. the norm $\|\cdot\|_i$ for $i = 1, \ldots, m$. Then $T = T_1 \times \cdots \times T_m : \mathbb{R}^n \to \mathbb{R}^n$ where $n = \sum_{i=1}^m n_i$ is a paracontraction w.r.t. the norm $\|x\| := (\sum_{i=1}^m \|x_i\|_i^p)^{1/p}$ for $x = (x_1, \ldots, x_m) \in \mathbb{R}^n$ and $p \geq 1$.

As a consequence of Lemma 1, the operator $S_1 : \mathbf{y}(t) \mapsto \mathbf{z}(t)$ defined by (10) is a paracontraction w.r.t the Euclidean norm with the fixed point set being $\hat{\mathcal{F}}_1 \times \cdots \times \hat{\mathcal{F}}_m$ where $\hat{\mathcal{F}}_i := \{\mathbf{x}_i \,|\, \mathbf{x}_i \in \mathcal{F}_i\}$. Now consider the operator $S_2 : \mathbf{z}(t) \mapsto \mathbf{y}(t+1)$, which is composed of the operator (11) at round t followed by the operator (9) at round $t+1$. It is easily seen that, for any $i \in \mathcal{I}$, $(y_i(t+1), (y_i^k(t+1))_{k\in\mathcal{N}_i^-})$ is exactly the projection of $(z_i(t), (z_i^k(t))_{k\in\mathcal{N}_i^-})$ onto the subspace $\{(x_i, (x_i^k)_{k\in\mathcal{N}_i^-}) \,|\, x_i = x_i^k,\ \forall k \in \mathcal{N}_i^-\}$. The operator S_2, being the product of all such projections for $i \in \mathcal{I}$, is a paracontraction w.r.t. the Euclidean norm by Lemma 1, whose fixed point set is the subspace specified by the constraints (8) for all $i \in \mathcal{I}$.

We now cite a key result proved in [19].

Theorem 1: Suppose $S_i : \mathbb{R}^n \to \mathbb{R}^n$, $i = 1, \ldots, \ell$, are paracontractions w.r.t. the norm $\|\cdot\|$ and suppose their fixed point sets $\mathcal{A}_i$'s have nonempty intersection. Starting from any $x(0) \in \mathbb{R}^n$ and for any sequence $\sigma(0), \sigma(1), \ldots \in \{1, \ldots, \ell\}$ so that each index i appears infinitely often, the iteration

$$x(t+1) = S_{\sigma(t)}(x(t)), \quad \forall t = 0, 1, \ldots,$$

will converge to a point $x_\infty = \lim_{t\to\infty} x(t) \in \cap_{i=1}^\ell \mathcal{A}_i$.

The following is a direct consequence of Theorem 1.

Theorem 2: Starting from any initial guess $\mathbf{x}(0)$, the result $\mathbf{x}(t)$ returned by Algorithm 1 will converge asymptotically to a feasible solution to Problem 2.

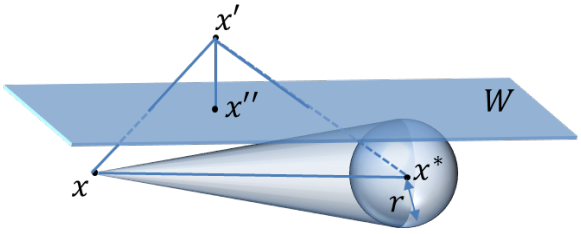


Fig. 2: Proof of Lemma 2.

Proof: It suffices to prove that the sequence $\mathbf{z}(t)$ will converge asymptotically to a solution to Problem 2. As discussed in the paragraph after Lemma 1, the sequence $\mathbf{z}(t)$ is obtained from the iteration

$$\mathbf{z}(t+1) = S_1 \circ S_2(\mathbf{z}(t)), \quad t = 0, 1, \ldots,$$

where S_1 and S_2 are two paracontractions w.r.t. the Euclidean norm whose sets of fixed points are specified by the constraints in (7) and (8), respectively. By Theorem 1, $\mathbf{z}(t)$ will converge to some $\mathbf{z}_\infty$ satisfying both (7) and (8) for all i, namely, a solution to Problem 2. ■

To study the convergence rate of Algorithm 1, we need the following result.

Lemma 2: Suppose that $\mathcal{A}_1$ and $\mathcal{A}_2$ are two closed convex subsets of $\mathbb{R}^d$ whose intersection $\mathcal{A}_1 \cap \mathcal{A}_2$ contains an interior point x^* of $\mathcal{A}_1$, i.e., there exists $r > 0$ such that the closed ball $\mathcal{B}(x^*, r)$ centered at x^* with the radius r is contained in $\mathcal{A}_1$. Let $\mathcal{P}_{\mathcal{A}_1}$ and $\mathcal{P}_{\mathcal{A}_2}$ be the projection operators onto these two sets, respectively. Then, for any $x \in \mathcal{A}_1 \setminus \mathcal{A}_2$,

$$d_{\mathcal{A}_2}\left(\mathcal{P}_{\mathcal{A}_1}(\mathcal{P}_{\mathcal{A}_2}(x))\right) \leq \beta \cdot d_{\mathcal{A}_2}(x).$$

Here, $\beta := \max\left\{\frac{\sqrt{\|x-x^*\|^2 - r^2}}{\|x-x^*\|}, 0\right\} \in [0, 1)$ and $d_{\mathcal{A}_2}(x)$ denotes the distance of $x \in \mathbb{R}^d$ to the set $\mathcal{A}_2$.

Proof: Let $x \in \mathcal{A}_1 \setminus \mathcal{A}_2$ be arbitrary and denote $x' = \mathcal{P}_{\mathcal{A}_2}(x)$ and $x'' = \mathcal{P}_{\mathcal{A}_1}(x')$ (see Fig. 2). Assume without loss that $x' \notin \mathcal{A}_1$ (otherwise $x' = x''$ and the conclusion is trivial), which implies that $x \notin \mathcal{B}(x^*, r)$. Hence $x \neq x'$ and $x' \neq x''$. Since $x', x^* \in \mathcal{A}_2$, the line segment $\overline{x'x^*}$ between x' and x^* is contained entirely in $\mathcal{A}_2$. The fact that $x' = \mathcal{P}_{\mathcal{A}_2}(x)$ implies that the angle that $\overline{x'x^*}$ and $\overline{x'x}$ make at x' cannot be acute, for otherwise an interior point on $\overline{x'x^*}$ would be closer to x than x' is. In other words, x' is inside the closed ball $\mathcal{B}_1$ centered at $(x + x^*)/2$ with the radius $\|x - x^*\|/2$.

Since $x'' = \mathcal{P}_{\mathcal{A}_1}(x')$, a supporting hyperplane W of $\mathcal{A}_1$ is given by the one that passes through x'' and is orthogonal to $\overline{x'x''}$. Note that x' needs to be on one side of W while the convex hull $\mathcal{C}$ of the point x and the ball $\mathcal{B}(x^*, r)$ needs to be on its other side. Due to this

constraint, the ratio $\|x'-x''\|/\|x'-x\|$ for all $x' \in \mathcal{B}_1 \setminus \{x\}$ is uniformly bounded from above by $\frac{\sqrt{\|x-x^*\|^2-r^2}}{\|x-x^*\|}$, where the upper bound is achieved when W is tangential to $\mathcal{C}$ along its conic part of the surface and $x' \in \mathcal{B}_1 \setminus \{x\}$ approaches x asymptotically. Combined with the fact that $x' = x''$ when $x \in \mathcal{B}(x^*, r)$, we have $\|x' - x''\|/\|x' - x\| \leq \beta$ where β is defined in the statement of the lemma. As a result, $d_{\mathcal{A}_2}(x'') \leq \|x'' - x'\| \leq \beta\|x' - x\| = \beta \cdot d_{\mathcal{A}_2}(x)$, $\forall x \in \mathcal{A}_1 \setminus \mathcal{A}_2$. ■

Using Lemma 2, we next characterize the convergence rate of Algorithm 1.

Theorem 3: Suppose the feasible solution x^* in Assumption 2 is an interior point of each constraint set $\mathcal{F}_i$ in (2). Then Algorithm 1 with $\alpha_i = 1, \forall i \in \mathcal{I}$, converges exponentially fast to a feasible solution of the Problem 2.

Proof: As shown in the proof of Theorem 2, the sequence $\mathbf{z}(t)$ generated by Algorithm 1 satisfies the condition that $\mathbf{z}(t) \in \mathcal{A}_1 := \hat{\mathcal{F}}_1 \times \cdots \times \hat{\mathcal{F}}_m$ and $\mathbf{z}(t+1) = \mathcal{P}_{\mathcal{A}_1}(\mathcal{P}_{\mathcal{A}_2}(\mathbf{z}(t+1)))$, $\forall t = 0, 1, \ldots$, where $\mathcal{A}_2$ is the subspace defined by the constraints (8). By assumption, $\mathbf{x}^* \in \mathcal{A}_2$ is an interior point of $\mathcal{A}_1$. Thus, by Lemma 2, we have $d_{\mathcal{A}_2}(\mathbf{z}(t+1)) \leq \beta(t) \cdot d_{\mathcal{A}_2}(\mathbf{z}(t))$ for some $\beta(t) = \max\left\{\frac{\sqrt{\|\mathbf{z}(t)-\mathbf{x}^*\|^2-r^2}}{\|\mathbf{z}(t)-\mathbf{x}^*\|}, 0\right\} \in [0, 1)$. As $\|\mathbf{z}(t) - \mathbf{x}^*\|$ is monotonically nonincreasing ($\mathcal{P}_{\mathcal{A}_1}$ and $\mathcal{P}_{\mathcal{A}_1}$ are paracontractions), so is $\beta(t)$. This implies that $\beta(t) \leq \rho$, $\forall t$, by some $\rho < 1$, and that $\mathbf{z}(t)$ is a sequence in $\mathcal{A}_1$ that converges to $\mathcal{A}_2$ exponentially fast. This proves the desired conclusion. ■

Similarly we can show that the asynchronous algorithm (Algorithm 2) will also converge to a solution to Problem 2, provided that each agent $i \in \mathcal{I}$ is included in infinitely many sets $\mathcal{I}_t$ for $t = 0, 1, \ldots$. The convergence rate of Algorithm 2, however, depends on the schedule sequence $\mathcal{I}_t$ and is much more difficult to characterize.

VI. SIMULATION RESULTS

In this section, the simulation results of several numerical examples are presented. In all these examples, the initial value $x_i^j(0)$ are set to be $x_i(0)$.

We firstly apply Algorithm 1 to the linear program in Example 2 with $\varepsilon = 0.01, 0.1, 0.5$, respectively. The results are shown in Fig. 3, where x^* is the converged feasible solution of the algorithm in each case.

We next consider the network localization problem in section III-B. Thirty agents are randomly placed inside a planar region. Among them, two are anchors that know their exact locations (at least two anchors are needed to localize the network [4]). The other free agents need to estimate their positions based on the relative orientation measurements from their neighbors that are within a certain range. Fig. 4 (i) shows one instance of the true agent

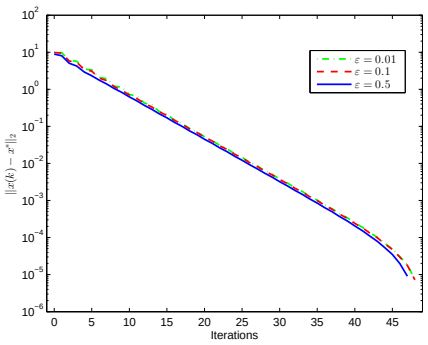
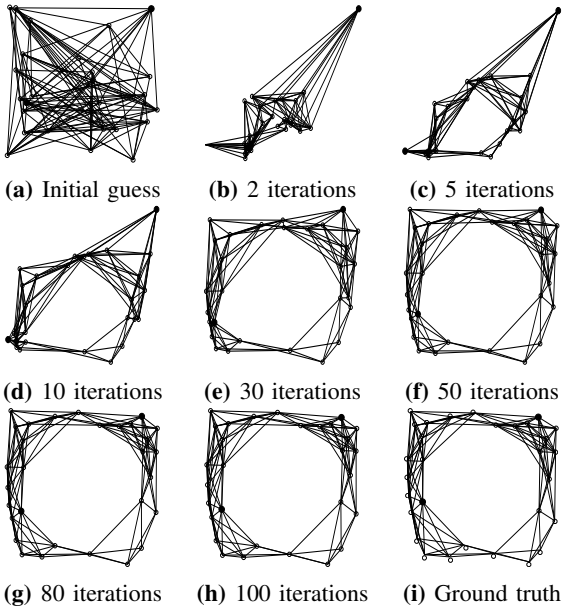


Fig. 3: Plot of $\|x(k) - x^*\|$ vs k when applying Algorithm 1 with $\alpha_1 = \alpha_2 = \alpha_3 = 1.9$ to Example 2 with different ε.



(a) Initial guess **(b)** 2 iterations **(c)** 5 iterations

(d) 10 iterations **(e)** 30 iterations **(f)** 50 iterations

(g) 80 iterations **(h)** 100 iterations **(i)** Ground truth

Fig. 4: Results of applying Algorithm 1 to the network localization problem with 2 anchors among 30 agents.

locations and the relative orientation measurements (one for each edge). Starting from random initial guesses for the free agent locations and assuming no measurement errors, the iterative results of applying Algorithm 1 are plotted in Fig. 4. The algorithm converges to the ground truth in about 50 iterations. Fig. 5 plots the convergence rates of the algorithm for three different sets of α_i: $\alpha_i \equiv 0.5$, $\alpha_i \equiv 1$, $\alpha_i \equiv 1.9$. For this example $\alpha_i \in (1, 2)$ induces faster convergence than $\alpha_i \in (0, 1)$.

Suppose the relative orientation measurements are inaccurate ($\delta \neq 0$). Setting $\delta = 0°$, $2.5°$, $4°$, $8°$, respectively, the results of applying Algorithm 1 are shown in Fig. 6, which plots the sum of constraint violations

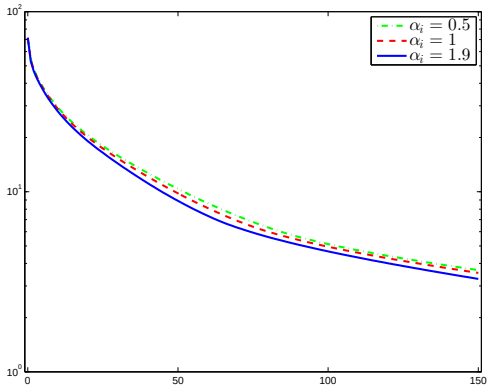


Fig. 5: Comparison of the convergence rates of Algorithm 1 with different α_i for the network localization problem. The value $\sum_{i\in\mathcal{I}_f} \|x_i(k) - x_i^*\|$ versus iteration number k is plotted.

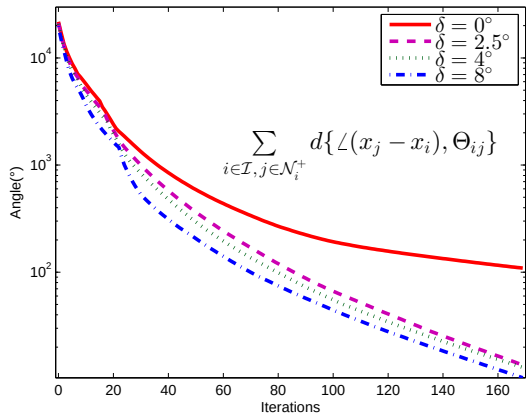


Fig. 6: Comparison of the convergence rates of Algorithm 1 for the network localization problem with measurement errors.

$d\{\angle(x_j(k) - x_i(k)), \Theta_{ij}\}$ vs. iteration number k. As can be seen, with a larger error range δ and hence a larger feasible set, the algorithm converges faster to a feasible solution, which is not the ground truth in general.

VII. CONCLUSIONS AND FUTURE WORKS

In this paper, distributed algorithms for solving the convex feasibility problem with sparse constraints are proposed. The convergence properties of these algorithms are established and their effectiveness are demonstrated through numerical examples. Future research directions include further reducing the storage and communication requirements on the agents, considering time-varying constraints and dependency graph, and improved convergence performance through adaptive adjustment of α_i and weights in the consensus operation.

REFERENCES

[1] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-point algorithms for inverse problems in science and engineering*. Springer, 2011, pp. 185–212.

[2] A. Bemporad, D. Bernardini, and P. Patrinos, "A convex feasibility approach to anytime model predictive control," *arXiv preprint arXiv:1502.07974*, 2015.

[3] J. Ash and L. Potter, "Sensor network localization via received signal strength measurements with directional antennas," in *Proceedings of the 2004 Allerton Conference on Communication, Control, and Computing*, 2004, pp. 1861–1870.

[4] G. Zhu and J. Hu, "A distributed continuous-time algorithm for network localization using angle-of-arrival information," *Automatica*, vol. 50, no. 1, pp. 53–63, 2014.

[5] H. H. Bauschke and J. M. Borwein, "On projection algorithms for solving convex feasibility problems," *SIAM review*, vol. 38, no. 3, pp. 367–426, 1996.

[6] Y. Censor, W. Chen, P. L. Combettes, R. Davidi, and G. T. Herman, "On the effectiveness of projection methods for convex feasibility problems with linear inequality constraints," *Computational Optimization and Applications*, vol. 51, no. 3, pp. 1065–1088, 2012.

[7] S. Mou, J. Liu, and A. S. Morse, "A distributed algorithm for solving a linear algebraic equation," *IEEE Transactions on Automatic Control*, vol. 60, no. 11, pp. 2863–2878, 2015.

[8] G. Shi and B. D. Anderson, "Distributed network flows solving linear algebraic equations," in *American Control Conference (ACC), 2016*. IEEE, 2016, pp. 2864–2869.

[9] B. Anderson, S. Mou, A. S. Morse, and U. Helmke, "Decentralized gradient algorithm for solution of a linear equation," *arXiv preprint arXiv:1509.04538*, 2015.

[10] D. Fullmer, L. Wang, and A. S. Morse, "A distributed algorithm for computing a common fixed point of a family of paracontractions," *IFAC-PapersOnLine*, vol. 49, no. 18, pp. 552–557, 2016.

[11] D. Fullmer, J. Liu, and A. S. Morse, "An asynchronous distributed algorithm for computing a common fixed point of a family of paracontractions," in *Decision and Control (CDC), 2016 IEEE 55th Conference on*. IEEE, 2016, pp. 2620–2625.

[12] Y. Lou, G. Shi, K. H. Johansson, and Y. Hong, "Approximate projected consensus for convex intersection computation: Convergence analysis and critical error angle," *IEEE Transactions on Automatic Control*, vol. 59, no. 7, pp. 1722–1736, 2014.

[13] R. Aharoni and Y. Censor, "Block-iterative projection methods for parallel computation of solutions to convex feasibility problems," *Linear Algebra Appl.*, vol. 120, pp. 165–175, 1989.

[14] S.-S. Chang, J. Kim, and X. Wang, "Modified block iterative algorithm for solving convex feasibility problems in banach spaces," *Journal of Inequalities and Applications*, vol. 2010, no. 1, p. 869684, 2010.

[15] A. Nedic, A. Ozdaglar, and P. A. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, 2010.

[16] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[17] X. Gao, J. Liu, and T. Başar, "Stochastic communication-efficient distributed algorithms for solving linear algebraic equations," in *Control Applications (CCA), 2016 IEEE Conference on*. IEEE, 2016, pp. 380–385.

[18] S. Mou, Z. Lin, L. Wang, D. Fullmer, and A. S. Morse, "A distributed algorithm for efficiently solving linear equations and its applications (special issue jcw)," *Systems & Control Letters*, vol. 91, pp. 21–27, 2016.

[19] L. Elsner, I. Koltracht, and M. Neumann, "Convergence of sequential and asynchronous nonlinear paracontractions," *Numerische Mathematik*, vol. 62, no. 1, pp. 305–319, 1992.