Distributed Nonconvex Multiagent Optimization Over Time-Varying Networks

Ying Sun, Gesualdo Scutari, and Daniel Palomar†

Abstract—We study nonconvex distributed optimization in multiagent networks wherein the communications between nodes is modeled as a time-varying sequence of arbitrary digraphs. We introduce a novel broadcast-based algorithmic framework for the (constrained) minimization of the sum of a smooth (possibly nonconvex and nonseparable) function, i.e., the agents’ sum-utility, plus a convex (possibly nonsmooth and nonseparable) regularizer. The latter is usually employed to enforce some structure in the solution, typically sparsity. The proposed method hinges on successive convex approximation techniques and a novel broadcast protocol to disseminate information and distribute the computation over the network. Asymptotic convergence to stationary solutions is established. A key feature of the proposed algorithm is that it neither requires the double-stochasticity of the consensus matrices nor the knowledge of the graph sequence to implement. To the best of our knowledge, the proposed framework is the first broadcast-based distributed algorithm for convex and nonconvex constrained optimization over arbitrary, time-varying digraphs. Numerical results show that our algorithm outperforms current schemes on both convex and nonconvex problems.

I. INTRODUCTION

Decentralized optimization has recently received significant attention due to the emergence of large-scale distributed algorithms in machine learning, data analysis, signal processing, and control applications. A central generic problem in such applications is distributed optimization in multiagent networks wherein agents—which may be processors, computers of a cluster, nodes of a sensor network, vehicles, or UAVs—want to cooperatively minimize a global objective by means of actions taken by each agent and local coordination between neighboring nodes. Specifically, in this paper, we consider the following general class of (possibly nonconvex) multiagent problems:

$$\min_{x \in \mathcal{K}} U(x) \triangleq \sum_{i=1}^{I} f_i(x) + G(x),$$ (1)

where $f_i : \mathbb{R}^m \to \mathbb{R}$ is the cost function of agent $i$, assumed to be smooth but (possibly) nonconvex; $G : \mathbb{R}^m \to \mathbb{R}$ is a convex (possibly nonsmooth) regularizer; and $\mathcal{K}$ is a closed convex subset of $\mathbb{R}^m$. Usually the nonsmooth term is used to promote some extra structure in the solution; for instance, $G(x) = c \|x\|_1$ or $G(x) = c \sum_{i=1}^{N} \|x_i\|_2$ are widely used to impose (group) sparsity of the solution. This general formulation arises naturally from many applications, including statistical inference over (e.g., sensor and power) networks, formation control, spectrum access coordination, distributed machine learning (e.g., LASSO, logistic regression, dictionary learning, matrix completion, tensor factorization), resource allocation problems in wireless communication networks, and distributed “epidemic” message routing in networks.

Our goal is developing solution methods for the nonconvex problem (1) in the following distributed setting: i) Each agent $i$ knows only its own function $f_i$ (as well as $G$ and $\mathcal{K}$); and ii) the communication topology connecting the agents is time-varying and directed and it is not known to the agents. Time-varying communication topologies arise, for instance, in mobile wireless networks, wherein the nodes are mobile and/or communicate throughout (fast-)fading channels. Directed communication links are also a natural assumption as in many cases there is no reason to expect different nodes to transmit at the same power level or that transmitter and receivers are geographically collocated (e.g., think of ad-hoc networks).

Distributed solution methods for convex instances of Problem (1) have been widely studied in the literature, under various assumptions on network topology; some recent contributions include [1]–[8]. The majority of the aforementioned works assume either undirected digraphs or a static topology. Moreover all the algorithms developed in the aforementioned papers along with their convergence analysis are not applicable to nonconvex problems, and thus to Problem (1). We are aware of only few works dealing with distributed algorithms for some nonconvex instances of (1), namely: [9]–[12]. Among them, our previous work [11] is to date the only method applicable to the general class of nonconvex constrained problems in the form (1). However, the implementability of algorithm [11] relies on the possibility of building a sequence of double-stochastic consensus matrices that are commensurate with the sequence of underlying time-varying communication digraphs. This can limit the applicability of the method in practice, especially when the network topology is time-varying, for several reasons. First, not all digraphs are doubly-stochastic (i.e., admits a doubly stochastic adjacency matrix); some form of balancedness in the graph is needed [13], which limits the class of network topologies over which algorithm [11] can be applied. Moreover, necessary and sufficient conditions for a digraph to be doubly-stochastic are not easy to be checked in practice. Second, constructing a doubly-stochastic weight matrix matching the graph, even when possible, calls for computationally intense, generally centralized, algorithms. Third, double stochasticity prevents one from using natural broadcast schemes, in which a given agent may transmit its local estimate to all its neighbors without expecting any immediate feedback.

The analysis of the literature shows that the design of distributed algorithms for the class of problems (1) over time-varying, arbitrary digraphs is up to date a challenging and open problem, even in the case of convex cost functions $f_i$. This paper introduces the first broadcast-based distributed algorithmic framework for the aforementioned class of problems. The crux of the framework is a general convexification-decomposition technique that hinges on our recent (primal) Successive Convex Approximation (SCA) method [14, 15], coupled with a novel broadcast protocol instrumental to distribute the computation and propagate the needed information over the network; we term the new scheme “distributed Successive CONvex Approximation algorithm over Time-varying digraphs (SONATA)”. Some key desirable features of SONATA are: i) It is applicable to arbitrary (possibly) time-varying network topologies; ii) it is fully distributed, requiring neither the knowledge of the graph sequence nor the use of a double-stochastic consensus matrix; in fact, each agent just needs to broadcast its local estimates to all its neighbors without expecting any feedback; iii) it deals with nonconvex and nonsmooth objectives as well as (convex) constraints; and iv) it is...
very flexible in the choice of the approximations of $f_i$'s, which need not be necessarily its first or second order approximation (like in all current distributed gradient schemes). Asymptotic convergence to stationary solutions of Problem (1) is proved. Numerical results show that the customization of SONATA to specific convex and nonconvex applications over time-varying networks outperforms state-of-the art schemes, in terms of practical convergence while reaching the same (stationary) solutions. As a final remark, we underline that the proposed broadcast protocol is different from existing works [6], [7], [12], [15] by leveraging the idea of push-sum [17] as key step to remove the double-stochastic requirement on the consensus matrix. It significantly extends the state of the art being the first broadcast-based distributed algorithm applicable to constrained (convex and nonconvex) optimization problems (all previous schemes [6], [7], [12], [15] work only for unconstrained problems because push-sum-based updates do not preserve feasibility of the iterates).

The rest of the paper is organized as follows. In Section II we first introduce the general idea of SONATA, followed by its formal description along with its convergence properties. Some applications of SONATA are discussed in Section III along with some numerical results. Finally, Section IV draws some conclusions.

II. Algorithmic Design

We study Problem (1) under the following standard assumptions.

**Assumption A** (Problem Setup)

(A1) The set $K \neq \emptyset$ is closed and convex;

(A2) Each $f_i$ is a continuously differentiable function defined on an open set containing $K$;

(A3) Each $\nabla f_i$ is Lipschitz continuous on $K$;

(A4) $\nabla F$ is bounded on $K$, with $F(x) = \sum_i f_i(x)$;

(A5) $G$ is convex with bounded subgradients on $K$;

(A6) $U$ is coercive on $K$, i.e., $\lim_{\|x\| \to +\infty} U(x) = +\infty$.

Assumption A is standard and satisfied by many practical problems. For instance, A3-A5 hold automatically if $K$ is bounded, whereas A6 guarantees the existence of a solution. Note that each $f_i$ need not be convex and is known only by agent $i$.

**On the network connectivity.** Time is slotted and, at each time slot $n$, the network of agents is modeled as a time-varying digraph $G[n] = (\mathcal{V}, \mathcal{E}[n])$, where the set of vertices $\mathcal{V} = \{1, \ldots, I\}$ represents the $I$ agents, and the set of edges $\mathcal{E}[n]$ represents the agents’ communication links. The in-neighborhood of agent $i$ at time $n$ (including node $i$) is defined as $N_{in}^n_i = \{j(\langle j, i \rangle \in \mathcal{E}[n]) \cup \{i\}$ whereas its out-neighbor is defined as $N_{out}^n_i = \{j(\langle i, j \rangle \in \mathcal{E}[n]) \cup \{i\}$. Agent $i$ can receive information from its in-neighbors, and send information to its out neighbors. The out-degree of agent $i$ is defined as $d_i[n] = |N_{out}^n_i|$. To let information propagate over the network, we assume that the graph sequence $(G[n])_{n \in \mathbb{N}}$ possesses some “long-term” connectivity property, as formalized next.

**Assumption B** (On the graph connectivity). The graph sequence $(G[n])_{n \in \mathbb{N}}$ is $B$-strongly connected, i.e., there exists an integer $B > 0$ (possibly unknown to the agents) such that the graph with edge set $\cup_{t=kB}^{(k+1)B-1} \mathcal{E}[t]$ is strongly connected, for all $k \geq 0$.

In words, Assumption B says that the information sent by any agent $i$ at any time $n$ will reach any agent $j$ within the next $B$ time slots.

Our goal is to develop an algorithm that converges to stationary solutions of Problem (1) while being implementable in the above distributed setting (Assumptions A and B), and applicable to arbitrary network topologies without requiring any knowledge of the graph sequence $G[n]$. To shed light on the core idea of the novel framework, we first introduce an informal and constructive description of the proposed algorithm, see Sec. II-A. Sec. II-B will formally introduce SONATA along with its convergence properties.

A. SONATA at a glance

Designing distributed algorithms for Problem (1) faces two main challenges, namely: the nonconvexity of the objective function and the lack of global information on the optimization problem from the agents. To cope with these issues, SONATA combines SCA techniques (Step 1 below) with a consensus-like step implementing a novel broadcast protocol (Step 2), as described next.

**Step 1: Local SCA.** Each agent $i$ maintains a local copy of the common optimization variable $x_i$, denoted by $x_i$, which needs to be updated at each iteration; let $x_i[n]$ be the value of $x_i$ at iteration $n$. The nonconvexity of $f_i$ together with the lack of knowledge of $\sum_j f_j$, prevent agent $i$ to solve Problem (1) directly. To cope with this issues, we leverage SCA techniques: at each iteration $n$, agent $i$ solves instead a convexification of Problem (1), having the following form

$$\hat{x}_i(x_i[n]) = \underset{x_i \in K}{\arg \min} \hat{F}_i(x_i; x_i[n]) + G(x_i),$$

where the nonconvex function $F$ is replaced with the strongly convex approximation $\hat{F}_i(x_i; x_i[n])$ around $x_i[n]$, defined as

$$\hat{F}_i(x_i; x_i[n]) = f_i(x_i; x_i[n] + \pi_i[n]^T(x_i[n] - x_i[n]),$$

wherein $\hat{F}_i(x_i; x_i[n]) : K \to \mathbb{R}$ is a strongly convex surrogate of the (possibly) nonconvex $f_i$ and $\pi_i[n]$ is the linearization of the unknown term $\sum_j f_j$ around $x_i[n]$, i.e.,

$$\pi_i[n] \triangleq \sum_{j \neq i} \nabla f_j(x_i[n]).$$

Note that $\hat{x}_i(x_i[n])$ is well-defined, because (2) has a unique solution. The direct use of $\hat{x}_i$ as the new local estimate $x_i[n+1]$ may affect convergence because it might be a too “aggressive” update. To cope with this issue we introduce a step-size in the update of $x_i$:

$$v_i[n] = x_i[n] + \alpha[n] (\hat{x}_i(x_i[n]) - x_i[n]),$$

where $\alpha[n]$ is a step-size constant (to be properly chosen, see Th. 1). The idea behind the iterates (2)–(5) is to compute stationary solutions of Problem (1) as fixed-points of the mappings $\hat{x}_i(\cdot)$. A first natural question is then how to choose the surrogate functions $\hat{f}_i$’s so that the fixed points of $\hat{x}_i(\cdot)$ coincide with the stationary solutions of (1). The next proposition establishes the desired connection; the proof follows from [15] Prop. 8(b)) and thus is omitted.

**Proposition 1.** Consider Problem (1) under Assumptions A1-A6. If the surrogate function $\hat{f}_i$ is chosen to satisfy the following conditions:

(C1) $\nabla \hat{f}_i(x; x) = \nabla f_i(x)$, for all $x \in K$;

(C2) $\hat{f}_i(\cdot; y)$ is uniformly strongly convex on $K$;

(C3) $\nabla \hat{f}_i(x; \cdot)$ is uniformly Lipschitz continuous on $K$;

then the set of fixed-points of $\hat{x}_i(\cdot)$ coincides with that of stationary solutions of Problem (1).

Conditions C1-C3 are quite natural: $\hat{f}_i$ should be regarded as a (simple) convex, local, approximation of $f_i$ at the point $x$ that preserves the first order properties of $f_i$. Several feasible choices are possible for a given $f_i$; we discuss alternative options in Sec. II-C.

Here, we only remark that no extra conditions on $\hat{f}_i$ are required to guarantee convergence of the proposed algorithm.

**Step 2: Broadcasting local information.** We have now to introduce a mechanism to ensure that the local estimates $x_i$ eventually agree among all agents. To disseminate information over a time-varying digraph without requiring the knowledge of the sequence of digraphs
and a double-stochastic weight matrix, we propose the following broadcasting protocol. Given \( w_i[n] \), each agent \( i \) updates its own local estimate \( x_i \), together with one extra scalar variables \( \phi_i[n] \) (initialized to \( \phi_i[0] = 1 \)), according to:

\[
\phi_i[n+1] = \sum_{j \in \mathcal{N}_i^n} \phi_j[n] / d_j[n]; \\
x_i[n+1] = \frac{1}{\phi_i[n+1]} \sum_{j \in \mathcal{N}_i^n} \phi_j[n] v_j[n] / d_j[n].
\]

Note that steps (6)–(7) are simple to implement: All agents only need to i) broadcast their local variables \( \phi_j[n] \) and \( \phi_j[n]w_j[n] \), normalized by their current out-degree \( d_j[n] \); and ii) collect locally the information coming from their neighbors. The idea behind the use of the extra variable \( \phi_i[n] \) is to dynamically construct a row stochastic weight matrix based only on the knowledge of agents’ out-degree, so that consensus among the \( x_i \)'s can be asymptotically achieved; see Sec. II-C for more details.

**On the local update of \( \pi_i[n] \).** The algorithm developed so far is based on the computation of \( \tilde{x}_i(n) \) in (2). To do so, at each iteration, every agent \( i \) needs to evaluate \( \pi_i[n] \) and thus know locally all \( \nabla f_i(x_i[n]) \), which is not feasible in a distributed time-varying setting. To cope with this issue, we replace \( \pi_i[n] \) in (2) with an estimate \( \tilde{\pi}_i[n] \), and solve instead:

\[
\tilde{x}_i[n] = \arg \min_{x_i \in \mathbb{K}} \tilde{f}_i(x_i[n], \pi_i[n]) + \nabla \tilde{f}_i(x_i[n]) \cdot (x_i[n] - x_i[n]) + G(x_i). 
\]

The question now becomes how to update each \( \tilde{\pi}_i \), using only local information [in the form of (6)–(7)] while asymptotically converging to \( \pi_i[n] \). As in (12), rewriting first \( \pi_i[n] \) as:

\[
\pi_i[n] = I - \nabla \tilde{f}_i(x_i[n]) - \nabla f_i(x_i[n]) ,
\]

with \( \nabla \tilde{f}_i(x_i[n]) \triangleq \sum_{j=1}^{n} \nabla f_j(x_i[n]) \), we propose to update \( \tilde{\pi}_i \), mimicking (6):

\[
\tilde{\pi}_i[n] = I - y_1[n] - \nabla f_i(x_i[n]).
\]

where \( y_1[n] \) is a local variable (controlled by agent \( i \)) whose task is to asymptotically track \( \nabla \tilde{f}_i(x_i[n]) \). Similar to (6)–(7), we propose the following new gradient tracking step:

\[
y_i[n+1] = 1/n \left( \sum_{j \in \mathcal{N}_i^n} \phi_j[n] / d_j[n] \right) y_j[n] + \nabla f_i(x_i[n+1]) - \nabla f_i(x_i[n])
\]

where \( \phi_i[n+1] \) is defined in (6). Note that the update of \( y_i \) and thus of \( \pi_i[n] \) can be now performed locally by agent \( i \), with the same signaling as in (6)–(7).

**B. Successive Convex Approximation over Time-varying Digraphs**

We are now in the position to formally introduce SONATA, as stated in Algorithm 1, its convergence properties are given in Theorem 1, whose proof is omitted because of the space limitation.

**Theorem 1.** Let \( \{x_i[n]\}_{i=1}^n \) be the sequence generated by Algorithm 1, and let \( \{z[n] \} \triangleq (1/n) \sum_i \phi_i[n] \cdot x_i[n] \). Suppose that i) Assumptions A-C hold; ii) The step-size sequences \( \{\alpha_i[n]\} \), satisfying \( \alpha_i[n] \in (0,1), \sum_{n=0}^{\infty} \alpha[n] \leq +\infty \), and \( \sum_{n=0}^{\infty} \alpha[n]^2 \leq +\infty \). Then, (I) [convergence]: \( z[n] \) is bounded for all \( n \), and every limit point of \( z[n] \) is a stationary solution of Problem 1.

(2) [consensus]: \( ||x_i[n] - z[n]|| \to 0 \) as \( n \to +\infty \) for all \( i \).

**Algorithm 1: Successive Convex Approximation over Time-varying Digraphs (SONATA)**

**Data:** For all agent \( i \), \( x_i[0] \in K, \phi_i[0] = 1 \), 

\[ y_i[0] = \nabla f_i(x_i[0]), \quad \pi_i[0] = Iy_i[0] - \nabla f_i(x_i[0]) \].

Set \( n = 0 \).

[1] If \( x_i[n] \) satisfies termination criterion: STOP;

[2] Distributed Local SCA: Each agent \( i \):
(a) computes \( \tilde{x}_i[n] \) with (6).
(b) updates its local variable \( v_i \) with (6) (replace \( \tilde{x} \) with \( \tilde{x} \)).

[3] Consensus: Each agent \( i \) broadcasts its local variables and sums up the received variables:
(a) Update \( \phi_i[n+1] \) with (6).
(b) Update \( x_i[n+1] \) with (6).
(c) Update \( y_i[n+1] \) with (6).
(d) Update \( \pi_i[n+1] \) with (6).

[4] \( n \leftarrow n + 1 \), go to [1].

**C. Discussion on Algorithm 1**

**Algorithm Dynamic.** Theorem 1 states that the weighted average of the \( x_i[n] \)'s, \( z[n] \), will be driven to a stationary solution of (1) while the local agents' variables \( x_i[n] \) will asymptotically all agree to \( z[n] \). We provide next some insight on these two steps.

To see how consensus is achieved, let us combine (5)–(7). Eliminating the auxiliary variable \( v_i[n] \), one can write:

\[
x_i[n+1] = \sum_{j=1}^{n} w_{ij} [n] \cdot x_j[n] + \alpha[n] I (\bar{x}_j(x_j[n]) - x_j[n]) \]

where \( W[n] \triangleq (w_{ij}[n])_{i,j} \) is a nonnegative matrix with elements:

\[
w_{ij}[n] = \begin{cases} \sum_j \phi_j[n] / d_j[n], & \text{if } j \in \mathcal{N}_i^n[n] \\ 0, & \text{otherwise.} \end{cases} \]

It is not difficult to see that \( W[n] \) is row stochastic, \( W[n] I = 1 \). This has two implications. First, feasibility of the iterates \( x_i[n] \) is preserved at each iteration, which permits our algorithm to deal with constraints. Second, row stochasticity of \( W[n] \) is instrumental to force consensus on the \( x_i[n] \)'s, for sufficiently small \( \alpha[n] \). Since \( z[n] = \frac{1}{n} \sum_i \phi_i[n] \cdot x_i[n] \) and \( \sum_i \phi_i[n] = I \), it follows that all the \( x_i[n] \)'s, if consensual, it must be all equal to \( z[n] \).

Let us focus now on the convergence of \( z[n] \). Multiply Eq. (12) by \( \phi_i[n+1] \) and take the average among all \( i \). This leads to:

\[
z[n+1] = z[n] + \frac{\alpha[n]}{I} \sum_{i=1}^{n} \phi_i[n] (\bar{x}_i(x_i[n]) - z[n])
\]

For large \( n \), \( x_i[n] \) is close to \( z[n] \) [cf. Th. 1 (2)]. Eq. (14) then states that \( z[n] \) moves along a convex combination of the directions \( \Delta_i(z[n]) \triangleq \bar{x}_i(z[n]) - z[n] \) generated by all agents. Since each \( \Delta_i \) can be proved to be a descent direction of \( U \) at point \( z[n] \) and \( z[n] \) is a diminishing sequence, all the \( \Delta_i \)'s will force \( z[n] \) towards a stationary point of Problem 1 [cf. Th. 1 (1)].

**On the choice of the surrogate functions.** SONATA represents a gamut of algorithms, each of them corresponding to a specific choice of the surrogate function \( \tilde{f}_i \) and step-size \( \alpha[n] \). Some instances of valid \( \tilde{f}_i \)'s are given next, see (11), (15) for more examples.
–Linearization: When there is no convex structure to exploit, one can simply linearize $f_i$, which leads to
\[
\bar{f}_i (\mathbf{x}_i; \mathbf{x}_n) = f_i (\mathbf{x}_i) + \nabla f_i (\mathbf{x}_i, [n])^T (\mathbf{x}_i - \mathbf{x}_i [n]) + \frac{\tau_i}{2} \| \mathbf{x}_i - \mathbf{x}_i [n] \|^2 .
\]

In this case, SONATA becomes a distributed proximal gradient algorithm for constrained optimization.

–Partial Linearization: Consider the case that $f_i$ can be decomposed as $f_i (\mathbf{x}_i) = f_i^{(1)} (\mathbf{x}_i) + f_i^{(2)} (\mathbf{x}_i)$, where $f_i^{(1)}$ is convex and $f_i^{(2)}$ is nonconvex with Lipschitz continuous gradient. Preserving the convex part of $f_i$ while linearizing $f_i^{(2)}$ leads to the following valid surrogate
\[
\bar{f}_i (\mathbf{x}_i; \mathbf{x}_n) = f_i^{(1)} (\mathbf{x}_i) + f_i^{(2)} (\mathbf{x}_i) [n] + \frac{\tau_i}{2} \| \mathbf{x}_i - \mathbf{x}_i [n] \|^2 + \nabla f_i^{(2)} (\mathbf{x}_i [n])^T (\mathbf{x}_i - \mathbf{x}_i [n]) .
\]

–Convexification: If variable $\mathbf{x}_i$ can be partitioned as $\{\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(2)}\}$, and $f_i$ is convex with respect to $\mathbf{x}_i^{(1)}$ while nonconvex with respect to $\mathbf{x}_i^{(2)}$, then $\bar{f}_i$ can be constructed by convexifying only the nonconvex part of $f_i$, i.e.,
\[
\bar{f}_i (\mathbf{x}_i; \mathbf{x}_n) = f_i^{(1)} (\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(2)} [n]) + \frac{\tau_i}{2} \| \mathbf{x}_i^{(2)} - \mathbf{x}_i^{(2)} [n] \|^2
\]
\[
+ \nabla f_i^{(2)} (\mathbf{x}_i [n])^T (\mathbf{x}_i^{(2)} - \mathbf{x}_i^{(2)} [n]) .
\]

where $\nabla f_i^{(2)} (\mathbf{x}_i) = \nabla f_i^{(2)} (\mathbf{x}_i)$ is the gradient of $f_i$ with respect to $\mathbf{x}_i^{(2)}$.

On the choice of the step-size. Th. 1 offers some flexibility in the choice of the step-size $\alpha [n]$ sequence; the conditions therein ensure that the sequence decays to zero, but not too fast. There are many diminishing step-size rules in the literature satisfying the aforementioned conditions; see, e.g., [13]. We found the following two choices effective in our experiments:

1. $\alpha [n] = \alpha_0 / (n + 1)^3$, $\alpha_0 > 0$, $0.5 < \beta \leq 1$.
2. $\alpha [n] = \alpha [n - 1] (1 - \mu \alpha [n - 1])$, $\alpha [0] \in (0, 1)$, $\mu \in (0, 1)$.

III. APPLICATIONS

In this section, we test the performance of SONATA on both convex and nonconvex problems. For all applications, we simulate the following graph topology: at each iteration, each agent has two out-neighbors, with one belonging to a time-varying cycle and the other two randomly chosen. The step-size $\alpha [n]$ is chosen based on rule (1).

A. Robust Regression

In the first simulation, we consider a robust linear regression problem. Each agent $i$ has $n_i$ measurements of parameter $\mathbf{x}$ as $b_{ij} = a_{ij}^T \mathbf{x}$, which is corrupted by noise and outliers. To estimate $\mathbf{x}$, we solve the following problem
\[
\min \mathbf{x} \left\{ \sum_{i=1}^{I} \sum_{j=1}^{n_i} h \left( a_{ij}^T \mathbf{x} - b_{ij} \right) \right\} ,
\]
where $h$ is the Huber loss function given by
\[
h (r) = \begin{cases} r^2, & \text{if } |r| > c \\ c (2 |r| - c), & \text{if } |r| \leq c. \end{cases}
\]

The function behaves like the $\ell_1$-norm if residual $r$ is larger than the cut-off parameter $c$, and is quadratic if $|r| \leq c$.

Defining $f_i (\mathbf{x}) \triangleq \sum_{j=1}^{n_i} h (a_{ij}^T \mathbf{x} - b_{ij})$, Problem (19) is an instance of the general Problem (1) with $F = \sum_{i=1}^{I} f_i$. We provide two versions of surrogate function $\bar{f}_i$. In the first version, function $f_i$ is linearized at each iteration (cf. Eq. (15)). In the second version, we propose a SCA scheme that approximates $f_i$ at $\bar{x} [n]$ by a quadratic function $\bar{f}_i (\mathbf{x}; \bar{x} [n]) = \sum_{j=1}^{n_i} h_{ij} (\mathbf{x}; \bar{x} [n]) + \frac{\tau_i}{2} \| \mathbf{x} - \bar{x} [n] \|^2$, where $\bar{h}_{ij}$ is defined as
\[
\bar{h}_{ij} (\mathbf{x}; \bar{x} [n]) = \begin{cases} \frac{c}{\tau_{ij} [n]} (a_{ij}^T \mathbf{x} - b_{ij})^2, & \text{if } |r_{ij} [n]| > c \\ \frac{c}{\tau_{ij} [n]} (a_{ij}^T \mathbf{x} - b_{ij})^2, & \text{if } |r_{ij} [n]| \leq c, \end{cases}
\]

with $r_{ij} [n] = a_{ij}^T \mathbf{x} - b_{ij}$. Consequently, the update $\bar{x}_i [n]$ has a closed form solution given as $\bar{x}_i [n] = (2 A_i^T D_i A_i + \tau I)^{-1} \left( 2 \bar{x} [n] - \mathbf{I}_i [n] + 2 A_i^T D_i b_i \right)$, where the $j$th row of $A_i$ is $a_{ij}^T$, and the $j$th element of $b_i$ is $b_{ij}$. Matrix $D_i$ is diagonal with its $j$th diagonal being $\min \{c, c/r_{ij} [n]\}$.

We simulate $I = 30$ agents collaboratively estimate $\mathbf{x}_0$ of dimension 200 with i.i.d. uniformly distributed entries in $[-1, 1]$. Each agent only has $n_i = 20$ measures. The elements of vector $a_{ij}$ is generated following an i.i.d. Gaussian distribution, then normalized to be $\| a_{ij} \| = 1$. The measurements noise follows a Gaussian distribution with standard deviation $\sigma = 0.1$, and each agent has one measurement corrupted by an outlier following a Gaussian distribution with standard deviation $5 \sigma$. The cut-off parameter $c$ is set to be $c = 3 \sigma$.

Algorithm parameters are tuned as follows. The proximal parameter $\tau$ for our linearization scheme and SCA scheme are set to be $\tau_L = 2$ and $\tau_{SCA} = 1.5$, respectively. Step-size parameters are set to be $\alpha [0] = 0.1$ and $\mu = 0.01$ for both of them. We compare the performance with subgradient-push algorithm proposed in [15], for which the step-size parameter is set to be $\alpha [0] = 0.5$, $\mu = 0.01$. In addition, since SONATA has two consensus steps, we run subgradient-push twice in one iteration using the same graph for a fair comparison.

The performance is averaged over 100 Monte-Carlo simulations, where each time $\mathbf{x}_0$ is fixed while the noise and graph connectivity are randomly generated. Fig. [1] reports the progress of the algorithms towards optimality and consensus error, where measure $J_n ([n]$ is defined as $J_n [n] \triangleq \| \nabla F (\bar{x} [n]) \|_\infty$ and $D_n [n] \triangleq \frac{1}{I} \sum_{i=1}^{I} \| x_i [n] - \bar{x} [n] \|^2$. We can see that SONATA reaches consensus and convergence much faster than subgradient-push. In addition, SCA scheme outperforms plain linearization by exploiting the convexity of the objective function.

![Fig. 1: Optimality measurements $J_n$ and consensus error $D_n$ versus the number of iterations.](image-url)
B. Target Localization

Target Localization problem considers a number of $I$ sensors in a network collaboratively locate the position of $T$ targets. Sensor $i$ has the knowledge of the coordinate of its own location $s_i$, and the relative Euclidean distance between itself and target $t$, denoted $d_{it}$. The problem is formulated as:

$$\begin{align*}
\text{minimize} & \quad \sum_{t=1}^{T} \sum_{i=1}^{I} p_{it} \left( d_{it} - \|x_t - s_i\|^2 \right)^2 \\
\text{subject to} & \quad x_t \in K \subset \mathbb{R}^m, \forall t,
\end{align*}$$

(20)

where $K$ is a compact set and variable $x_t$ is an estimate of the location of target $t$, denoted $x_t^0$. Parameter $p_{it} \in \{0, 1\}$ takes value zero if the $i$th agent has no measurement about target $t$.

We apply SONATA to Problem (20) with $f_i(x) = \sum_{t=1}^{T} p_{it} \left( d_{it} - \|x_t - s_i\|^2 \right)^2$, where $x$ is obtained by stacking the $x_t$'s. The two SCA schemes proposed in (11) are adopted, namely, linearization (cf. Eq. (18)) and partial linearization with surrogate function

$$\tilde{f}_i(x; x[n]) = \sum_{t=1}^{T} p_{it} \left( \tilde{f}_{it}(x; x[n]) + \frac{1}{2}\|x_t - x_t[n]\|^2 \right),$$

(21)

where $\tilde{f}_{it}(x; x[n]) = x_T^T A_i x_t - b_{it}[n]^T (x_t - x_t[n])$, with $A_i = \frac{4s_i s_i^T + 2\|s_i\|^2 I_t}{\|s_i\|^2}$, and $b_{it}[n] = 4\|s_i\|^2 s_i - 4\left(\|x_t[n]\|^2 - d_{it}\right) (x_t[n] - s_i) + 8 (s_i^T x_t[n]) x_t[n]$.

In the simulation, we set the number of sensors to be $I = 30$, and the number of targets to be $T = 5$. Parameter $p_{it}$ takes value zero and one with equal probability. The locations of the sensors and targets are uniformly randomly generated in $[0, 1]^2$. We consider a noisy environment that the measured distances are corrupted by i.i.d. Gaussian noise. The noise standard deviation is set to be the minimum pairwise distance between sensors and targets.

We compare with the gradient algorithm proposed in (12) for unconstrained optimization. The radius of $K$ set to be sufficiently large so that all the points generated by the algorithm are feasible.

Algorithm parameters are tuned as follows. For our algorithm, the step-size parameters are set to be $\alpha[0] = 0.1$ and $\mu = 10^{-4}$. The proximal parameter $\tau$ of $f_i$ for the linearization scheme is selected to be $\tau_L = 7$ and that for partial linearization is selected to be $\tau_{PL} = 5$. For the benchmark algorithm, $\alpha[0] = 0.05$ and $\mu = 10^{-4}$.

A comparison of the algorithms is given in Fig. 2 which is averaged over 100 Monte-Carlo simulations. Fig. 2 shows that within 200 iteration, both consensus and convergence are achieved for all algorithms; and SONATA converges much faster than the benchmark gradient algorithm.

IV. CONCLUSION

In this paper we have proposed SONATA, a novel distributed algorithm for nonconvex constrained optimization over time-varying networks. The algorithm leverages the idea of SCA for local optimization and a new in-network broadcast protocol for distributed computing. SONATA is the first broadcast-based algorithm framework that can solve convex or nonconvex constrained optimization problems over arbitrary time-varying digraphs. Numerical result shows that our algorithm outperforms state-of-the-art schemes on considered convex and nonconvex applications.
[16] A. Nedic and A. Olshevsky, “Distributed optimization over time-varying directed graphs,” *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 601–615, 2015.

[17] D. Kempe, A. Dobra, and J. Gehrke, “Gossip-based computation of aggregate information,” in *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, 2003, Oct 2003, pp. 482-491.

[18] D. P. Bertsekas, *Nonlinear programming*. Athena Scientific, 2 ed., 1999.