Distributed quadratic programming under Asynchronous and Lossy Communications via Newton-Raphson Consensus

Ruggero Carli   Giuseppe Notarstefano   Luca Schenato   Damiano Varagnolo

Abstract

Quadratic optimization problems appear in several interesting estimation, learning and control tasks. To solve these problems in peer-to-peer networks it is necessary to design distributed optimization algorithms supporting directed, asynchronous and unreliable communication. This paper addresses this requirement by extending a promising distributed convex optimization algorithm, known as Newton-Raphson consensus, and originally designed for static and undirected communication. Specifically, we modify this algorithm so that it can cope with asynchronous, broadcast and unreliable lossy links, and prove that the optimization strategy correctly converge to the global optimum when the local cost functions are quadratic. We then support the intuition that this robustified algorithm converges to the true optimum also for general convex problems with dedicated numerical simulations.

I. Introduction

Distributed quadratic optimization problems arise in several estimation and control problems in peer-to-peer networks. To cope with real-world requirements, algorithms need to be designed to work under asynchronous, directed and faulty communications. Despite being the distributed optimization literature already rich, most of the existing contributions have been proved to work in networks whose communication schemes follow synchronous, undirected, and often time-invariant information exchange mechanisms.

The first completely distributed optimization algorithms appearing in the literature rely on primal sub-gradient iterations [4], [5] and on dual ones [7], [8]. To induce robustness in the computation and improve convergence in the case of non-strictly convex functions Alternating Direction Method of Multipliers (ADMM) schemes have been proposed [9], [10]. Recently contributions have been given to increase the convergence speed of this technique by means of accelerated consensus schemes [11], [12]. All these algorithms have been proved to converge to the global optimum under the assumption of fixed and undirected topology. Recently sub-gradient based algorithms for switching topologies have been proposed in [13] and [1], while gradient-based strategies supporting time-varying, directed and event-triggered communication schemes are presented in [14] and [15].

Another class of algorithms exploits the exchange of active constraints among the network nodes. A constraints consensus algorithm has been proposed in [16] to solve linear, convex and general abstract programs, see also [17]. These were the first distributed optimization algorithms working under asynchronous and direct communication. Recently the constraint exchange idea has been combined with dual decomposition and cutting-plane methods to solve distributed robust convex optimization problems via polyhedral approximations [18]. Although well-suited for asynchronous and directed communications, these algorithms mainly solve constrained optimization problems in which the number of constraints is much smaller than the number of decision variables (or vice-versa). Another technique suitable for asynchronous communication that exploits contraction maps is the one proposed in [19], but it requires strong assumptions on the structure of the cost functions. Other methods with some degree of connections related to this manuscript are [1], where the push-sum method is used to establish distributed optimization on general time-varying directed graphs, and [2], [3], where authors propose an Newton-like approach alternative to the one proposed here.

An approach for unconstrained optimization is to exploit the Newton-Raphson consensus strategy in [20]. This algorithm shows very interesting convergence properties, is proved to work under synchronous and asynchronous symmetric gossip communications [21], but requires undirected and perfectly reliable communications.

In this paper we propose for quadratic optimization problems an extension of the Newton-Raphson consensus approach, so that it handle asynchronous broadcast (rather than gossip) communications over a directed graph and lossy communications. Specifically, we combine a push-sum consensus method proposed in [22], to achieve average consensus in time-varying directed networks, with the original Newton-Raphson consensus idea introduced in [20]. We moreover add the technique proposed in [23] to handle packets losses. In particular, we show that when applied to quadratic programs the extended Newton-Raphson consensus algorithm turns out to be a forward product of column stochastic matrices which, under the broadcast communication, is shown to be a stationary and ergodic process.

This work is supported by the Framework Programme for Research and Innovation Horizon 2020 under grant agreement n. 636834 (DISIRE), and the Swedish research council Norrbottens Forskningsråd.

R. Carli and L. Schenato are with the Department of Information Engineering, University of Padova, Via Gradenigo 6/a, 35131 Padova, Italy { carlirug | schenato }@dei.unipd.it.

Giuseppe Notarstefano is with the Department of Engineering, Università del Salento, Via per Monteroni, 73100 Lecce, Italy giuseppe.notarstefano@unisalento.it.

D. Varagnolo is with the Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology, Forskargatan 1, 97187 Luleå, Sweden damiano.varagnolo@ltu.se.

Section II formulates our problem and working assumptions. Section III introduces the proposed algorithm and its proof of convergence, while Section IV adds to it the robustness to packet losses. Finally, Section V collects some numerical experiments corroborating our results.

II. Problem Formulation and Assumptions

We consider the separable optimization problem

$$x^* := \min_x \sum_{i=1}^{N} f_i(x) \tag{1}$$

under the assumptions that each f_i is a quadratic function known only to node i.

We aim at designing an algorithm solving (1) with the following features, i.e., being distributed, asynchronous and robust w.r.t. packet losses. Formally, we consider a network with set of nodes $V = \{1, \ldots, N\}$ and an asynchronous broadcast communication protocol with packet losses. That is, at a given time-instant there is just one node $i \in V$ transmitting information to all its neighbors in a given fixed, directed and strongly connected graph $\mathcal{G} = (V, \mathcal{E})$, while the others either merely receive the information or do nothing. Here $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, i.e., $(i, j) \in \mathcal{E}$ if there is an edge from node i to node j. The edge (i, j) models the fact that node j can receive directly information from node i. By $\mathcal{N}_i^{\text{out}}$ we denote the set of *out-neighbors* of node i, i.e., $\mathcal{N}_i^{\text{out}} := \{j \in \mathcal{V} \mid (i, j) \in \mathcal{E}\}$. Similarly, $\mathcal{N}_i^{\text{in}}$ denotes the set of *in-neighbors* of node i, i.e., $\mathcal{N}_i^{\text{in}} := \{j \in \mathcal{V} \mid (j, i) \in \mathcal{E}\}$.

As for the concept of time, each node has its local, individual timer that randomly, and independently from other nodes, triggers when to transmit. We assume the following:

Assumption II.1 *The waiting times between local triggering events are exponential i.i.d. random variables for all the nodes in* $\{1, \ldots, N\}$.

Let then $\sigma(t) \in \{1, \ldots, N\}$, $t = 1, 2, \ldots$ be the sequence identifying the generic t-th triggered node. Assumption II.1 implies that $\sigma(t)$ is an i.i.d. uniform process on the alphabet $\{1, \ldots, N\}$. Each triggering will induce an iteration of the distributed optimization algorithm, so that t will indicate the t-th iteration of the algorithm itself.

To solve (1), each node i stores in its memory a local copy of the decision variable x, say x_i, and referred to as the local decision variable. With the new notation (1) reads as

$$\min_{x_1, \ldots, x_N} \sum_{i=1}^{N} f_i(x_i) \quad \text{s.t. } x_i = x_j \text{ for all } (i, j) \in \mathcal{E}. \tag{2}$$

The strong connectivity of graph $\mathcal{G}$ ensures then that the optimal solution of (2) is given by $x_1 = \ldots = x_N = x^*$, i.e., that problems (1) and (2) are equivalent.

III. The asynchronous Newton-Raphson Consensus algorithm

In this section we depart from the symmetric gossip asynchronous Newton-Raphson Consensus (a-NRC) proposed in [21] and derive a broadcast version of the algorithm that supports directed communication networks. For readability reason, we start with an a-NRC algorithm that does not handle packet losses, and describe the complete version in Section IV. We then assume that each node i is endowed of an embedded memory, stores in it the variables x_i, g_i, g_i^{old}, h_i, h_i^{old}, z_i and y_i, and initializes them as

$$\begin{aligned} y_i &= g_i^{\text{old}} = g_i = -f_i'(0) \\ z_i &= h_i^{\text{old}} = h_i = f_i''(0) \\ x_i &= 0. \end{aligned}$$

Let $\varepsilon \in (0, 1]$ be a real parameter and let, w.l.o.g., $\sigma(t) = i$, so that node i is the one broadcasting its information during the t-th iteration of the algorithm. We also define the following thresholding operator: if $c > 0$ is the scalar bounding the second derivatives of the local costs to be known, then

$$[z]_c := \begin{cases} z & \text{if } z \geq c \\ c & \text{otherwise.} \end{cases}$$

A pseudo-code implementation of the a-NRC algorithm is given in Algorithm 1.

Algorithm 1 asynchronous Newton-Raphson Consensus (a-NRC)

```
1: before transmission, node $i$ updates its local variables as
       $y_i \leftarrow \frac{1}{|\mathcal{N}_i^{\text{out}}|+1}\left[y_i + g_i - g_i^{\text{old}}\right]$
       $z_i \leftarrow \frac{1}{|\mathcal{N}_i^{\text{out}}|+1}\left[z_i + h_i - h_i^{\text{old}}\right]$
       $g_i^{\text{old}} \leftarrow g_i$
       $h_i^{\text{old}} \leftarrow h_i$
       $x_i \leftarrow (1-\varepsilon)x_i + \varepsilon\frac{y_i}{[z_i]_c}$
       $g_i \leftarrow f_i''(x_i)x_i - f_i'(x_i)$
       $h_i \leftarrow f_i''(x_i)$
2: node $i$ then broadcasts $y_i$ and $z_i$ to its neighbors;
3: each neighbor $j \in \mathcal{N}_i^{\text{out}}$ updates its local variables as
       $y_j \leftarrow y_i + y_j + g(x_j) - g(x_j^{\text{old}})$
       $z_j \leftarrow z_i + z_j + h(x_j) - h(x_j^{\text{old}})$
       $g_j^{\text{old}} \leftarrow g_j$
       $h_j^{\text{old}} \leftarrow h_j$
       $x_j \leftarrow (1-\varepsilon)x_j + \varepsilon\frac{y_j}{[z_j]_c}$
       $g_j \leftarrow f_j''(x_j)x_j - f_j'(x_j)$
       $h_j \leftarrow f_j''(x_j)$
```

A. Convergence of the a-NRC algorithm for quadratic costs

Assume the local costs to be

$$f_i(x) = \frac{1}{2}(a_i x - b_i)^2, \qquad a_i \neq 0. \tag{3}$$

so that the optimal solution of (1) becomes

$$x^* = \frac{\sum_{i=1}^N a_i b_i}{\sum_{i=1}^N a_i^2}.$$

In the next proposition we show that for the quadratic case the global convergence of the algorithm is ensured for every $\varepsilon \in (0,1]$.

Proposition III.1 *Let the local costs f_i be as in* (3)*, Assumption II.1 hold true, and $\varepsilon \in (0,1]$. Then the trajectory $t \to \boldsymbol{x}(t)$ almost surely and asymptotically reaches consensus on the optimal solution x^*, i.e.,*

$$\mathbb{P}\left[\lim_{t\to\infty} \boldsymbol{x}(t) = x^*\mathbf{1}\right] = 1.$$

Proof: To prove the result, we start describing the a-NRC algorithm in a compact form. Let $\boldsymbol{x}$, $\boldsymbol{g}^{\text{old}}$, $\boldsymbol{h}^{\text{old}}$, $\boldsymbol{g}$, $\boldsymbol{h}$, $\boldsymbol{y}$, $\boldsymbol{z}$, $\boldsymbol{f}'(x)$, and $\boldsymbol{f}''(x)$ denote the vectors of the stacked corresponding local quantities, e.g.,

$$\begin{aligned}
\boldsymbol{x} &:= [x_1, \ldots, x_N]^T \\
\boldsymbol{f}''(x) &:= [f_1''(x_1), \ldots, f_N''(x_N)]^T .
\end{aligned}$$

Let also $\boldsymbol{f}''(\boldsymbol{x}(t))\boldsymbol{x}(t)$ and $\dfrac{\boldsymbol{y}(t-1)}{[\boldsymbol{z}(t-1)]_c}$ indicate element-wise operations, i.e.,

$$\begin{aligned}
\boldsymbol{f}''(\boldsymbol{x}(t))\boldsymbol{x}(t) &:= \left[f_1''(x_i(t))x_1(t), \ldots, f_N''(x_i(t))x_N(t)\right]^T \\
\frac{\boldsymbol{y}(t-1)}{[\boldsymbol{z}(t-1)]_c} &:= \left[\frac{y_1(t-1)}{[z_1(t-1)]_c}, \ldots, \frac{y_N(t-1)}{[z_N(t-1)]_c}\right]^T .
\end{aligned}$$

Let moreover $P_i \in \mathbb{R}^{N \times N}$, $i \in \{1, \dots, N\}$, be

$$P_i := I - e_i e_i^T + \frac{1}{|\mathcal{N}_i^{\text{out}}| + 1} \sum_{j \in \mathcal{N}_i \cup \{i\}} e_j e_i^T$$

with e_i the i-th vector of the standard orthonormal basis. Notice moreover that every P_i has nonnegative elements and is column stochastic, i.e., is s.t. $\mathbf{1}^T P_i = \mathbf{1}^T$, with $\mathbf{1}$ the N-dimensional vector with all the components equal to one. With this notation, the a-NRC algorithm can be written as

$$\begin{aligned}
\boldsymbol{y}(t) &= P_{\sigma(t)} \big(\boldsymbol{y}(t-1) + \boldsymbol{g}(t-1) - \boldsymbol{g}^{\text{old}}(t-1)\big) \\
\boldsymbol{z}(t) &= P_{\sigma(t)} \big(\boldsymbol{z}(t-1) + \boldsymbol{h}(t-1) - \boldsymbol{h}^{\text{old}}(t-1)\big) \\
\boldsymbol{g}^{\text{old}}(t) &= \boldsymbol{g}(t-1) \\
\boldsymbol{h}^{\text{old}}(t) &= \boldsymbol{h}(t-1) \\
\boldsymbol{x}(t) &= (1-\varepsilon)\boldsymbol{x}(t-1) + \varepsilon \frac{\boldsymbol{y}(t-1)}{[\boldsymbol{z}(t-1)]_c} \\
\boldsymbol{g}(t) &= \boldsymbol{f}''\big(\boldsymbol{x}(t)\big)\boldsymbol{x}(t) - \boldsymbol{f}'\big(x(t)\big) \\
\boldsymbol{h}(t) &= \boldsymbol{f}''\big(\boldsymbol{x}(t)\big)
\end{aligned}$$

Thus, the evolutions of $\boldsymbol{y}$ and $\boldsymbol{z}$ are described by the time-varying linear dynamics with column-stochastic state matrices

$$\boldsymbol{y}(t) = P_{\sigma(t)} \boldsymbol{y}(t-1), \qquad \boldsymbol{y}(0) = a_i b_i,$$

and

$$\boldsymbol{z}(t) = P_{\sigma(t)} \boldsymbol{z}(t-1), \qquad \boldsymbol{z}(0) = a_i^2.$$

Notice that, since the column-stochastic state matrices in the updates above guarantee that $\boldsymbol{z}(t)$ remains positive along the entire evolution, the $[\cdot]_c$ operator is never active.

Write then

$$\frac{\boldsymbol{y}(t)}{\boldsymbol{z}(t)} = \frac{\boldsymbol{y}(t)}{\boldsymbol{v}(t)} \frac{\boldsymbol{v}(t)}{\boldsymbol{z}(t)}$$

with the new variable $\boldsymbol{v}(t)$ evolving as

$$\boldsymbol{v}(t) = P_{\sigma(t)} \boldsymbol{v}(t-1), \qquad \boldsymbol{v}(0) = \mathbf{1},$$

and let

$$\boldsymbol{\omega}_y(t) = \frac{\boldsymbol{y}(t)}{\boldsymbol{v}(t)}, \qquad \boldsymbol{\omega}_z(t) = \frac{\boldsymbol{z}(t)}{\boldsymbol{v}(t)}.$$

Following [22], we then consider the algorithm

$$\boldsymbol{\xi}(t) = \frac{\boldsymbol{s}(t)}{\boldsymbol{\omega}(t)}$$

where $\boldsymbol{\xi}, \boldsymbol{s}, \boldsymbol{\omega} \in \mathbb{R}^N$ and where the dynamics of $\boldsymbol{s}$ and $\boldsymbol{\omega}$ are ruled by

$$\boldsymbol{s}(t) = D(t)\boldsymbol{s}(t-1), \qquad \boldsymbol{s}(0) = \boldsymbol{\xi}(0)$$

and

$$\boldsymbol{\omega}(t) = D(t)\boldsymbol{\omega}(t-1), \qquad \boldsymbol{\omega}(0) = \mathbf{1},$$

with $D(t)$ a column-stochastic matrix. Under the assumptions that

- $\{D(t)\}_{t \geq 0}$ is a stationary and ergodic sequence of column-stochastic matrices with positive diagonals;
- $\mathbb{E}[D]$ is irreducible;

from [22, Thm IV.1] it follows that

$$\mathbb{P}\left[\lim_{t \to \infty} \boldsymbol{\xi}(t) = \left(\frac{1}{N}\sum_{i=1}^{N} \xi_i(0)\right)\mathbf{1}\right] = 1.$$

Now notice that $\{P_{\sigma(t)}\}$ is a stationary and ergodic sequence defined on the alphabet $\{P_1, \dots, P_N\}$, that all the matrices P_i have positive diagonals, and that the matrix

$$\overline{P} := \mathbb{E}\left[P_{\sigma(t)}\right] = \frac{1}{N}\sum_{i=1}^{N} P_i$$

is s.t. $\overline{P}_{ij} \neq 0$ if $(i,j) \in \mathcal{E}$. Since the graph $\mathcal{G}$ is strongly connected and the matrix $\overline{P}$ has positive diagonal elements, it follows that $\overline{P}$ is irreducible. Hence we can conclude that, almost surely,

$$\lim_{t\to\infty} \boldsymbol{\omega}_y(t) = \left(\frac{1}{N}\sum_{i=1}^{N} \widetilde{\boldsymbol{y}}(0)\right)\mathbf{1} = \left(\frac{1}{N}\sum_{i=1}^{N} a_i b_i\right)\mathbf{1}$$

and

$$\lim_{t\to\infty} \boldsymbol{\omega}_z(t) = \left(\frac{1}{N}\sum_{i=1}^{N} \widetilde{\boldsymbol{z}}(0)\right)\mathbf{1} = \left(\frac{1}{N}\sum_{i=1}^{N} a_i^2\right)\mathbf{1}.$$

Therefore, again almost surely,

$$\lim_{t\to\infty} \boldsymbol{x}(t) = \frac{(1/N\sum_{i=1}^{N} a_i b_i)\mathbf{1}}{(1/N\sum_{i=1}^{N} a_i^2)\mathbf{1}} = x^*\mathbf{1}.$$

■

IV. Robustification of the a-NRC algorithm to packet losses

Consider the realistic situation where some communication links might fail, i.e., where node i performs a broadcast communication but not every out-neighbor might receive the transmitted information due to, e.g., wireless packets corruption phenomena.

To cope with this possibility of communication losses we robustify the a-NRC algorithm 1 against this type of communication failures. To this aim, we take advantage of the technique proposed in [23], to obtain average consensus algorithms converging to the right average over general directed graphs and in presence of stochastic packet losses.

We thus assume that every node i stores in its memory, in addition to the variables x_i, x_i^{old}, z_i, y_i, also the variables $b_{i,y}$, $b_{i,z}$, $r_{i,y}^{(j)}$ and $r_{i,z}^{(j)}$ for every $j \in \mathcal{N}_i^{\text{in}}$. The meanings of these variables are the following:

- $b_{i,y}$ and $b_{i,z}$ are quantities used by node i to locally keep track of the total mass of (respectively) states y_i and z_i. In this robustified version of the algorithm the quantities $b_{i,y}$ and $b_{i,z}$ are broadcast by node i to its out-neighbors;
- $r_{j,y}^{(i)}$ and $r_{j,z}^{(i)}$ are instead quantities used by node j to locally keep track of the total mass of (respectively) states y_i and z_i of the neighbor i of j. In other words, with $r_{j,y}^{(i)}$ and $r_{j,z}^{(i)}$ node j tracks the status of node i: when the communication link from i to j is available, node j updates $r_{j,y}^{(i)}$ and $r_{j,z}^{(i)}$ with the received $b_{i,y}$ and $b_{i,z}$; otherwise, in case of communication failure, $r_{j,y}^{(i)}$ and $r_{j,z}^{(i)}$ remain equal to the previous total masses received.

A pseudo-code implementation of the robust asynchronous Newton-Raphson Consensus (ra-NRC) algorithm in presented in Algorithm 2, where w.l.o.g. $\sigma(t) = i$, i.e. node i is the node triggering iteration t.

A. Convergence of Algorithm 2 for the quadratic costs case

Let us model the communication failures process as follows:

Assumption IV.1 *Let w.l.o.g. node i be the transmitting node during the generic t-th iteration. Then the generic neighbor $j \in \mathcal{N}$ receives information from i with probability q_{ij}, $0 < q_{ij} \leq 1$, i.e.,*

$$\mathbb{P}\big[(i,j) \textit{ is reliable } \big] = q_{ij}, \quad 0 < q_{ij} \leq 1.$$

Additionally, we assume that the successes of transmissions are independent among different links and in time.

We can then prove the global convergence properties of the ra-NRC Algorithm 2 applied to quadratic local costs as in (3) and for every $\varepsilon \in (0,1]$:

Proposition IV.2 *Let the local costs f_i be as in* (3)*, Assumptions II.1 and IV.1 hold true, and $\varepsilon \in (0,1]$. Then the trajectory $t \to \boldsymbol{x}(t)$ reaches almost surely and asymptotically consensus on the optimal solution x^*, i.e.,*

$$\mathbb{P}\left[\lim_{t\to\infty} \boldsymbol{x}(t) = x^*\mathbf{1}\right] = 1.$$

Proof: assume that, for $i \in V$, node i stores in memory also the fictitious variables v_i, $b_{i,v}$ and $r_{i,v}^{(j)}$ for $j \in \mathcal{N}_i^{\text{in}}$, where $v_i(0) = 1$, $b_{i,v}(0) = 0$ and $r_{i,v}^{(j)}(0) = 0$ for $j \in \mathcal{N}_i^{\text{in}}$. Assume that these fictitious variables are updated in parallel to the other variables stored memory in by the nodes as follows: if $\sigma(t) = i$, then

Algorithm 2 robust asynchronous Newton-Raphson Consensus (ra-NRC)

```
1: before transmission, node $i$ updates its local variables as
$$\begin{aligned} y_i &\leftarrow \frac{1}{|\mathcal{N}_i^{\text{out}}|+1}\left[y_i + g_i - g_i^{\text{old}}\right] \\ z_i &\leftarrow \frac{1}{|\mathcal{N}_i^{\text{out}}|+1}\left[z_i + h_i - h_i^{\text{old}}\right] \\ g_i^{\text{old}} &\leftarrow g_i \\ h_i^{\text{old}} &\leftarrow h_i \\ x_i &\leftarrow (1-\varepsilon)x_i + \varepsilon\frac{y_i}{[z_i]_c} \\ g_i &\leftarrow f_i''(x_i)x_i - f_i'(x_i) \\ h_i &\leftarrow f_i''(x_i) \\ b_{i,y} &\leftarrow b_{i,y} + y_i \\ b_{i,z} &\leftarrow b_{i,z} + z_i \end{aligned}$$
2: node $i$ then broadcasts to its neighbors $b_{i,y}$ and $b_{i,z}$ to its neighbors;
3: each neighbor $j \in \mathcal{N}_i^{\text{out}}$ updates (if receiving the packet, otherwise it does nothing) its local variables as
$$\begin{aligned} y_j &\leftarrow b_{i,y} - r_{j,y}^{(i)} + y_j + g_j - g_j^{\text{old}} \\ z_j &\leftarrow b_{i,z} - r_{j,z}^{(i)} + z_j + h(x_j) - h(x_j^{\text{old}}) \\ g_j^{\text{old}} &\leftarrow g_j \\ h_j^{\text{old}} &\leftarrow h_j \\ x_j &\leftarrow (1-\varepsilon)x_j + \varepsilon\frac{y_j}{[z_j]_c} \\ g_j &\leftarrow f_i''(x_j)x_i - f_i'(x_j) \\ h_j &\leftarrow f_i''(x_j) \\ r_{j,y}^{(i)} &\leftarrow b_{i,y} \\ r_{j,z}^{(i)} &\leftarrow b_{i,z} \end{aligned}$$
```

(i) node i updates v_i and $b_{i,v}$ as

$$\begin{aligned} v_i &\leftarrow \frac{1}{|\mathcal{N}_i^{\text{out}}|+1} v_i \\ b_{i,v} &\leftarrow b_{i,v} + v_i \end{aligned}$$

(ii) node i then broadcasts to its neighbors the quantity $b_{i,v}$;

(iii) each neighbor $j \in \mathcal{N}_i^{\text{out}}$ updates (if receiving the packet, otherwise it does nothing) its local variables v_j and $r_{j,v}^{(i)}$ as

$$\begin{aligned} v_j &\leftarrow b_{i,v} - r_{j,v}^{(i)} \\ r_{j,v}^{(i)} &\leftarrow b_{i,v} \end{aligned}$$

Now let $\boldsymbol{v}(t) = [v_1(t), \ldots, v_N(t)]^T$. Since the $[\cdot]_c$ operator is never active in this quadratic case, we can write that

$$\frac{\boldsymbol{y}(t)}{\boldsymbol{z}(t)} = \frac{\boldsymbol{y}(t)}{\boldsymbol{v}(t)}\,\frac{\boldsymbol{v}(t)}{\boldsymbol{z}(t)}.$$

Observe that, as before, if t_i is the first t for which $\sigma(t) = i$ or $\sigma(t) = j$ with $i \in \mathcal{N}_j^{\text{out}}$, i.e., the first t for which i either broadcasts or receives information for the first time, then for $t < t_i$ it holds that

$$g_i = a_i b_i, \quad g_i^{\text{old}} = 0, \quad h_i = a_i^2, \quad h_i^{\text{old}} = 0,$$

while for $t \geq t_i$ it holds that

$$g_i(t) = g_i^{\text{old}}(t) = a_i b_i, \quad h_i(t) = h_i^{\text{old}}(t) = a_i^2.$$

It follows that $\boldsymbol{\omega}_y(t) = \frac{\boldsymbol{y}(t)}{\boldsymbol{v}(t)}, \quad \boldsymbol{\omega}_z(t) = \frac{\boldsymbol{z}(t)}{\boldsymbol{v}(t)}$ can be suitably seen as two instances of the consensus algorithm 3 described in Appendix and with initial condition $\boldsymbol{y}(0) = [a_1 b_1, \dots, a_N b_N]$, $\boldsymbol{z}(0) = \left[a_1^2, \dots, a_N^2\right]$ and $\boldsymbol{v}(0) = \mathbf{1}$. Thus, since Assumptions II.1 and IV.1 hold true, it follows that

$$\lim_{t\to\infty} \boldsymbol{\omega}_y(t) = \left(\frac{1}{N}\sum_{i=1}^{N} a_i b_i\right)\mathbf{1},$$

$$\lim_{t\to\infty} \boldsymbol{\omega}_z(t) = \left(\frac{1}{N}\sum_{i=1}^{N} a_i^2\right)\mathbf{1},$$

that is sufficient for our claims. ■

V. Numerical experiments

Aims: Proposition IV.2 ensures the global convergence of Algorithm 2 for the quadratic case and for every $\varepsilon \in (0, 1]$. Numerical simulations nonetheless suggest that Algorithm 2 is converging also for non-quadratic local costs, for opportunely tuned $\varepsilon \in (0, 1]$, as for the original synchronous version in [20].

In this section we thus aim at describing qualitatively the behavior of the single nodes while running the ra-NRC, and comment the effects of choosing different ε's on the convergence speed / properties of the algorithm for non-quadratic costs.

Notice that we do not compare the ra-NRC with the two currently most famous distributed optimization techniques present in literature, namely ADMM [4], [5] and subgradient schemes [10], since: *i)* as for the ADMM, at the best of our knowledge there are no competing algorithms, i.e., there are no ADMM-based schemes that can perform broadcast asynchronous optimization tasks while being robust to packet losses issues. *ii)* as for subgradient schemes, it has already been numerically shown in [20] that these algorithms are outperformed by Newton-Raphson (NR)-based procedures. This indeed mimics the situation of centralized optimization procedures, where exploiting information on higher derivatives generally improves the convergence properties of the optimization routine.

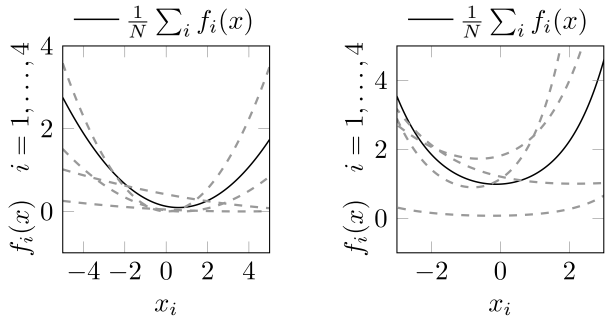
Numerical setup: as local costs we consider quadratic functions, i.e.,

$$f_i(x) = \frac{1}{2}\left(\alpha_i' x - \alpha_i''\right)^2 \tag{4}$$

sums of exponentials, i.e.,

$$f_i(x) = \alpha_i' \exp\left(\alpha_i'' x\right) + \alpha_i''' \exp\left(-\alpha_i'''' x\right) \tag{5}$$

with parameters randomly generated as either $[\alpha_i', \alpha_i''] \sim \mathbb{U}\left[0,1\right]^2$ or $[\alpha_i', \dots, \alpha_i''''] \sim \mathbb{U}\left[0,1\right]^4$.



(a) realizations of (4) (b) realizations of (5)

Fig. 1. Examples of the local costs considered for the numerical experiments (dashed lines) and of the relative global costs (solid lines).

The considered network is instead the random geometric network shown in Figure 2.

Communications are broadcast, asynchronous and with packet losses that occur independently on each link time with probability 0.2. In other words, a packet sent simultaneously to nodes i and j may reach i but not j.

Results: Figure 3 describes the effect of the choice of the design parameter ε on the convergence speed of the algorithm by considering how fast the average guess $\frac{1}{N}\sum_i x_i(t)$ approaches the optimum x^* both under quadratic and exponential local costs.

As expected, increasing ε leads to faster convergence speeds. Nonetheless, for non-quadratic cases too big ε's may lead to instability and diverging phenomena (a common issue of schemes that are based on separation of time-scales concepts). We remark that dynamically finding the best ε (that depends on several factors, mainly the curvature of the local costs and the topology of the communication network) is still an open issue.

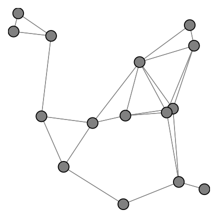
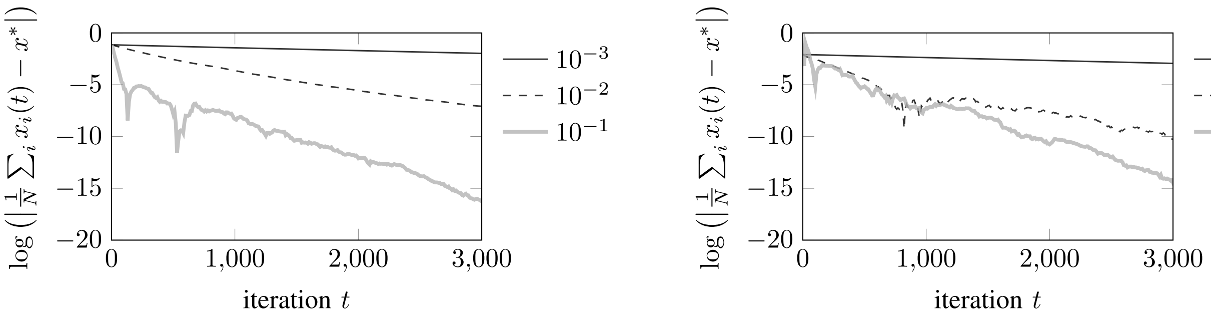


Fig. 2. The random geometric network considered for the numerical experiments of this section. It is composed by $N = 15$ nodes uniformly deployed in $[0, 1]^2$ and with communication radius 0.35.

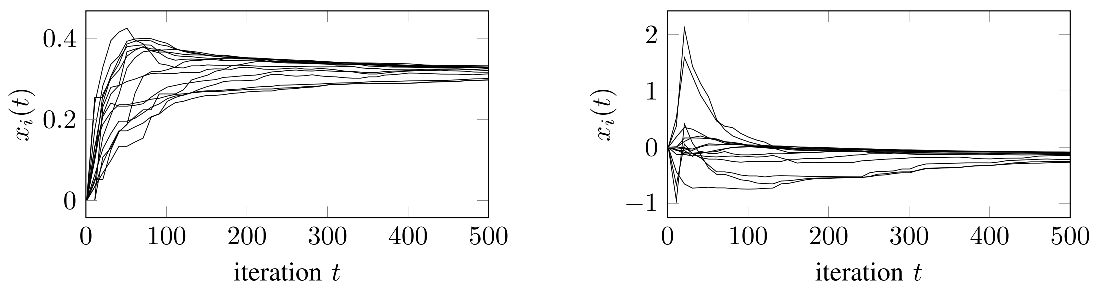


(a) evolutions under costs (4)

(b) evolutions under costs (5)

Fig. 3. Comparisons of the dependence of the convergence speed of the algorithm on ε for different cost functions.

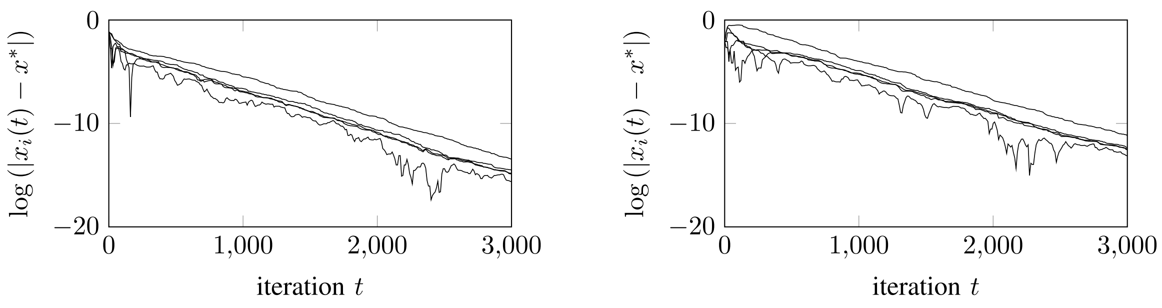
Regarding the behavior of the single nodes, Figure 4 plots the evolutions of the local the relative errors for $\varepsilon = 0.1$. We can notice that the qualitative behavior of the various nodes is the same, independently of the fact of being in the periphery of the network or not. It is also possible to notice that the algorithm has linear convergence time (fact that is driven by the linearity of the consensus algorithm underlying the information exchange process).



(a) evolutions under costs (4)

(b) evolutions under costs (5)

Fig. 4. Evolution of the local states of the various nodes for $\varepsilon = 0.1$.



(a) evolutions under costs (4)

(b) evolutions under costs (5)

Fig. 5. Evolution of the local errors of the various nodes for $\varepsilon = 0.1$.

VI. Conclusions

To be able to arrive to real-world implementations, distributed algorithms are required to seamlessly cope with packet-losses, asynchronous communications, and directed links. At the same time, optimization algorithms are supposed to be fast, i.e., return accurate estimates of the optimum after a limited amount of exchanged information.

These two considerations drove the development of this paper, presenting a robustification of the distributed Newton-Raphson algorithm proposed initially in [20]. More specifically, we added to the original procedure a set of features that enable this algorithm to work even with asynchronous, unreliable and broadcast communication protocols. This constitutes in our opinion an advantage with respect to ADMM schemes, that at the best of our knowledge do not tolerate these working conditions.

We then notice that this paper opens more questions than how many it closes, since the proofs of convergence are only for quadratic local costs. Thus proving convergence properties under general costs and unreliable communications scenarios is still an open question.

Moreover the algorithm, that in our vision may become the heart of a truly distributed interior point method, still lacks of important capabilities: *i)* tuning ε on line, that requires nodes to be able to detect diverging behaviors; *ii)* updating the state x with partition-based approaches, meaning that (in the same spirit of [24]) each node keeps and updates only some of the components; *iii)* accounting for equality constraints in the state of the form $Ax = b$.

References

[1] A. Nedic and A. Olshevsky, "Distributed optimization over time-varying directed graphs," in *IEEE Conference on Decision and Control*, Dec 2013, pp. 6855–6860.

[2] E. Wei, A. Ozdaglar, and A. Jadbabaie, "A distributed newton method for network utility maximization - i: Algorithm," *IEEE Transactions on Automatic Control*, vol. 58, no. 9, pp. 2162–2175, 2013.

[3] ——, "A distributed newton method for network utility maximization - part ii: Convergence," *IEEE Transactions on Automatic Control*, vol. 58, no. 9, pp. 2176 – 2188, 2013.

[4] 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.

[5] 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.

[6] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computations: Numerical Methods*. Belmont, Massachusetts: Athena Scientific, 1997.

[7] B. Yang and M. Johansson, "Distributed optimization and games: A tutorial overview," *Networked Control Systems*, pp. 109–148, 2011.

[8] M. Zhu and S. Martinez, "On distributed convex optimization under inequality and equality constraints," *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 151–164, 2012.

[9] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, "Consensus in ad hoc WSNs with noisy links - part I: Distributed estimation of deterministic signals," *IEEE Transactions on Signal Processing*, vol. 56, no. 1, pp. 350 – 364, 2008.

[10] 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*, vol. 3, no. 1, pp. 1–122, 2011.

[11] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson, "Optimal parameter selection for the alternating direction method of multipliers (ADMM): quadratic problems," *IEEE Transactions on Automatic Control*, vol. PP, no. 99, pp. 1–1, 2014.

[12] A. Teixeira, E. Ghadimi, I. Shames, H. Sandberg, and M. Johansson, "Optimal scaling of the ADMM algorithm for distributed quadratic programming," in *IEEE Conference on Decision and Control*. IEEE, 2013, pp. 6868–6873.

[13] P. Lin and W. Ren, "Distributed subgradient projection algorithm for multi-agent optimization with nonidentical constraints and switching topologies," in *IEEE Conference on Decision and Control*. IEEE, 2012, pp. 6813–6818.

[14] B. Gharesifard and J. Cortes, "Distributed continuous-time convex optimization on weight-balanced digraphs," *IEEE Transactions on Automatic Control*, vol. 59, no. 3, pp. 781–786, 2014.

[15] S. S. Kia, J. Cortes, and S. Martinez, "Distributed convex optimization via continuous-time coordination algorithms with discrete-time communication," in *arXiv*, 2014.

[16] G. Notarstefano and F. Bullo, "Distributed abstract optimization via constraints consensus: Theory and applications," *IEEE Transactions on Automatic Control*, vol. 56, no. 10, pp. 2247–2261, October 2011.

[17] ——, "Network abstract linear programming with application to minimum-time formation control," in *IEEE Conference on Decision and Control*, New Orleans, LA, December 2007, pp. 927–932.

[18] M. Bürger, G. Notarstefano, and F. Allgöwer, "A polyhedral approximation framework for convex and robust distributed optimization," *IEEE Transactions on Automatic Control*, vol. 59, no. 2, pp. 384–395, Feb 2014.

[19] C. Fischione, "F-Lipschitz Optimization with Wireless Sensor Networks Applications," *IEEE Transactions on Automatic Control*, vol. 56, no. 10, pp. 2319 – 2331, 2011.

[20] F. Zanella, D. Varagnolo, A. Cenedese, P. Gianluigi, and L. Scenato, "Newton-raphson consensus for distributed convex optimization," in *Proc. 50th IEEE Conf. on Decision and Control*, Orlando, Florida, December 2011, pp. 5917 – 5922.

[21] F. Zanella, D. Varagnolo, A. Cenedese, G. Pillonetto, and L. Schenato, "Asynchronous newton-raphson consensus for distributed convex optimization," in *3rd IFAC Workshop on Distributed Estimation and Control in Networked Systems*, 2012.

[22] F. Bénézit, V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli, "Weighted gossip: Distributed averaging using non-doubly stochastic matrices," in *IEEE International Symposium on Information Theory Proceedings (ISIT)*. IEEE, 2010, pp. 1753–1757.

[23] M. A. D. Dominguez-Garcis, C. N. Hadjicostis, and N. H. Vaidya, "Distributed algorithms for consensus and coordination in the presence of packet-dropping communication links. part i: Statistical moments analysis approach," *arXiv:1109.6391v1 [cs.SY] 29 Sep 2011*, 2011.

[24] R. Carli and G. Notarstefano, "Distributed partition-based optimization via dual decomposition," in *IEEE Conference on Decision and Control*, Firenze, Italy, 2013.

[25] N. H. Vaidya, C. N. Hadjicostis, and A. D. Dominguez-Garcia, "Distributed algorithms for consensus and coordination in the presence of packet-dropping communication links part ii: Coefficients of ergodicity analysis approach," in *arXiv*, 2011.

Appendix

We here propose a robust asynchronous distributed average consensus algorithm that is based on lossy broadcast communications. Due to lack of space we recall but omit to prove its convergence properties; the proof can nonetheless be easily obtained by following the techniques proposed in [25].

Consider then the strongly connected graph $\mathcal{G}$ of Section II, and assume that each node $i = 1, \ldots, N$ to be endowed with a deterministic initial value v_i. Nodes then aim to reach consensus to the average of these initial values, i.e., compute $v^* = \frac{1}{N}\sum_{i=1}^{N} v_i$, in a distributed and iterative fashion.

As for the previous algorithms, we assume that during each iteration of the algorithm there is only one node, say i, transmitting information to its neighbors. For $j \in \mathcal{N}_i^{\text{out}}$, we assume that node j can either receive the information sent by node i or do not receive it, if a packet loss occurs.

Additionally assume that every node i stores in its memory the variables x_i, z_i, y_i, $b_{i,y}$, $b_{i,z}$ and the variables $r_{i,y}^{(j)}$, and $r_{i,z}^{(j)}$ for $j \in \mathcal{N}_i^{\text{in}}$, where $x_i(0) = 0$, $z_i(0) = 1$, $y_i(0) = v_i$, $b_{i,y}(0) = b_{i,z}(0) = 0$ and $r_{i,y}^{(j)}(0) = r_{i,z}^{(j)}(0) = 0$ for $j \in \mathcal{N}_i^{\text{in}}$.

Without loss of generality, suppose node i is the transmitting node during the t-th iteration. Then the robust asynchronous Average Consensus (ra-AC) algorithm reads as:

Algorithm 3 robust asynchronous Average Consensus (ra-AC)

```
1: before transmission, node $i$ updates its local variables as
       $y_i \leftarrow \frac{1}{|\mathcal{N}_i^{\text{out}}| + 1} y_i$
       $z_i \leftarrow \frac{1}{|\mathcal{N}_i^{\text{out}}| + 1} z_i$
       $b_{i,y} \leftarrow b_{i,y} + y_i$
       $b_{i,z} \leftarrow b_{i,z} + z_i$
2: node $i$ then broadcasts to its neighbors $b_{i,y}$ and $b_{i,z}$;
3: each neighbor $j \in \mathcal{N}_i^{\text{out}}$ updates (if receiving the packet, otherwise it does nothing) its local variables as
       $y_j \leftarrow b_{i,y} - r_{j,y}^{(i)} + y_j$
       $z_j \leftarrow b_{i,z} - r_{j,z}^{(i)} + z_j$
       $x_j \leftarrow \frac{y_j}{z_j}$
       $r_{j,y}^{(i)} \leftarrow b_{i,y}$
       $r_{j,z}^{(i)} \leftarrow b_{i,z}$
   where the update of $x_j$ is performed only if $z_j \neq 0$, otherwise $x_j$ is left unchanged.
```

Proposition VI.1 *Let Assumptions II.1 and IV.1 hold true, and let $\boldsymbol{x} := [x_1, \ldots, x_N]^T$. Then the trajectory $t \to \boldsymbol{x}(t)$ reaches almost surely the asymptotic consensus on the value v^*, i.e.,*

$$\mathbb{P}\left[\lim_{t\to\infty} \boldsymbol{x}(t) = v^* \mathbf{1}\right] = 1.$$