Asynchronous Newton-Raphson Consensus for Robust Distributed Convex Optimization

Ruggero Carli Giuseppe Notarstefano Luca Schenato Damiano Varagnolo

***Abstract*— A general trend in the development of distributed convex optimization procedures is to robustify existing algorithms so that they can tolerate the characteristics and conditions of communications among real devices. This manuscript follows this tendency by robustifying a promising distributed convex optimization procedure known as Newton-Raphson consensus. More specifically, we modify this algorithm so that it can cope with asynchronous, broadcast and unreliable communications. We prove the convergence properties of the modified algorithm under the assumption that the local costs are quadratic, and support with numerical simulations the intuition that this robustified algorithm converges to the true optimum as soon as the local costs satisfy some mild smoothness conditions.**

I. INTRODUCTION

The research area of distributed optimization has recently received significant attentions in the distributed control and estimation literature. In fact, distributed optimization algorithms are important building blocks in several estimation and control problems, specially in peer-to-peer networks. But, despite being the literature on distributed optimization quite 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 class of completely distributed optimization algorithms appearing in the literature relied on primal subgradient iterations [1], [2]. Following the dual decomposition approach proposed in the large-scale optimization literature [3], purely distributed dual decomposition methods have been proposed in peer-to-peer networks. In [4] a tutorial on network optimization via dual decomposition can be found. A recent reference handling equality and inequality constraints is [5]. To induce robustness in the computation and improve convergence in the case of non-strictly convex functions it has been proposed to use Alternating Direction Methods of Multipliers (ADMM) schemes. A first distributed ADMM algorithm was proposed in [6], while a survey on this technique is [7]. Notice that recently some efforts have been posed to increase the convergence speed of this technique by means of accelerated consensus schemes [8]. All these algorithms have been proved to converge to the global optimum under fixed and undirected topologies assumptions. Recently sub-gradient based algorithms for switching topologies have been proposed in [9] and [10].

Another class of algorithms exploits the exchange of active constraints among the network nodes. A constraints consensus has been proposed in [11] to solve linear, convex and general abstract programs, see also [12]. 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 [13]. 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). An other technique that exploits contraction maps is the one proposed in [14], but we notice that it requires strong assumptions on the structure of the cost functions.

An alternative approach for unconstrained optimization is to exploit the Newton-Raphson consensus approach that has been recently proposed in [15]. These algorithms show very interesting convergence properties and are proved to work under synchronous communication. However, in the algorithm proposed in [15] the communication is required to be undirected and reliable, in the sense that there are no mechanisms to handle packet losses.

Inspired by this algorithmic idea, in this paper we propose a novel methodology working under asynchronous broadcast communications over a directed graph. Specifically, the contributions of the paper are as follows. First, we combine the Newton-Raphson consensus idea introduced in [15] with a push-sum consensus method proposed in [16] to achieve average consensus in directed networks. The intuition behind the proposed algorithm is the following: the Newton-Raphson consensus solves the distributed optimization problem by estimating a Newton-Raphson descent update. The convergence is guaranteed through a time-scale separation between the iteration computing the Newton-Raphson update and the average consensus that forces the nodes to share the common Newton direction. Here we introduce the push-sum idea to replace the aforementioned consensus protocol, so to regain convergence to the average even under direct topologies assumptions, and moreover add a technique that allows to handle packets losses. Second, we show that for distributed quadratic programs the push-sum update guarantees

This work is supported by the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement n. 257462 HYCON2 Network of excellence.

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.

the convergence of all the agents to the global optimum. The result is proved by showing that the proposed update rule is a forward product of column stochastic matrices which, under the broadcast communication, is shown to be a stationary and ergodic process.

The manuscript is organized as follows: Section II formulates the problem and our working assumptions. Section III then introduces the proposed algorithm and its proof of convergence. Section IV adds to it the robustness to packet losses. Section V collects some numerical experiments corroborating our results, and eventually Section VI concludes the manuscript with some remarks and indications of future research directions.

II. Problem Formulation

We consider a network with set of nodes $V = \{1, \ldots, N\}$ and a fixed directed communication graph $\mathcal{G} = (V, \mathcal{E})$. In our definitions $\mathcal{E}$ is the set of edges, i.e., $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ and $(i, j) \in \mathcal{E}$ if there is an edge going from node i to node j. In our context, 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., $\mathcal{N}_i^{\text{out}} := \{j \in \mathcal{V} | (i, j) \in \mathcal{E}\}$ is the set of agents receiving messages from i. Similarly, $\mathcal{N}_i^{\text{in}}$ denotes the set of in-neighbors of node i, i.e., $\mathcal{N}_i^{\text{in}} := \{j \in \mathcal{V} | (j, i) \in \mathcal{E}\}$. The graph $\mathcal{G}$ is assumed to be strongly connected.

We start dealing with the scalar case, and consider the solution of the separable optimization problem

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

under the assumptions that each f_i is known only to agent i and is $\mathcal{C}^2$, coercive over $\mathbb{R}$ and strictly convex with second derivative bounded from below, i.e., $f_i''(x) > c$ for all x.

We are interested into algorithms solving (1) with the following two features:

(i) being *distributed*, as opposed to centralized: namely, we assume that there is no central unit that, by knowing all the f_i's and by having global knowledge of the graph $\mathcal{G}$, may compute x^* directly. Instead we assume that each node has limited computational and memory resources and that it is allowed to communicate directly only with its out-neighbors;

(ii) being *asynchronous*, as opposed to synchronous: namely, agents do not share a common reference time with which it is possible to synchronize all the updating and transmitting actions.

In what follows we introduce a distributed algorithm which is based on a Newton-Raphson consensus strategy and which employs an *asynchronous broadcast* communication protocol. Specifically, during each iteration of the algorithm there is just one node transmitting information to all its neighbors in the graph $\mathcal{G}$, while the others either merely receive the information or do nothing.

In the following we will refer to this procedure as the *asynchronous Newton-Raphson Consensus* algorithm (denoted hereafter as a-NRC algorithm). Our a-NRC scheme is reminiscent of the Newton-Raphson procedure introduced in [15] in a completely synchronous scenario.

We thus assume that, to solve (1), each agent i stores in its memory a copy of x, say x_i. We thus can reformulate problem in (1) as

$$\begin{aligned} &\min_{x_1, \ldots, x_N} \sum_{i=1}^{N} f_i(x_i) \\ &\text{subj. to } x_i = x_j \quad \text{for all } (i, j) \in \mathcal{E} \end{aligned} \tag{2}$$

The strongly connectedness 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.

Instrumental to our aims, we assume that each node has its local concept of time. Each node has thus its individual timer that randomly triggers the associated nodes to transmit, eventually triggering an iteration of the algorithm. How often these local timers ticks is described by the following assumption:

Assumption II.1 *Let $\{T^{(i)}(h)\}$, $h \in \mathbb{N}$, be the time instants in which the node i is triggered by its own timer. We assume that the timer ticks with exponentially distributed waiting times, identically distributed for all the nodes in $\{1, \ldots, N\}$.*

With this machinery we thus introduce an artificial concept of time driving the sequence of iterations t of the algorithm.

Notice then that, if the random sequence $\sigma(t) \in \{1, \ldots, N\}$ defines which node has been triggered at iteration t, Assumption II.1 implies that $\sigma(t)$ is an i.i.d. uniform process on the alphabet $\{1, \ldots, N\}$.

We also define the following operator: assuming the scalar $c > 0$ bounding the second derivatives of the local costs to be known, we let

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

III. The asynchronous Newton-Raphson Consensus algorithm

We assume that each node i stores in its memory the variables x_i, g_i, g_i^{old}, h_i, h_i^{old}, z_i and y_i, initialized as

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

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. Then the following actions are performed in order:

(i) node i starts by updating its local variables as

$$
\begin{aligned}
y_i &\leftarrow \frac{1}{|\mathcal{N}_i^{\text{out}}|+1}\left[y_i + g_i - g_i^{\text{old}}\right]\\
z_i &\leftarrow \frac{1}{|\mathcal{N}_i^{\text{out}}|+1}\left[z_i + h_i - h_i^{\text{old}}\right]\\
g_i^{\text{old}} &\leftarrow g_i\\
h_i^{\text{old}} &\leftarrow h_i\\
x_i &\leftarrow (1-\varepsilon)x_i + \varepsilon\frac{y_i}{[z_i]_c}\\
g_i &\leftarrow f_i''(x_i)x_i - f_i'(x_i)\\
h_i &\leftarrow f_i''(x_i)
\end{aligned}
$$

(ii) node i then broadcasts y_i and z_i to its neighbors;

(iii) each neighbor $j \in \mathcal{N}_i^{\text{out}}$ updates its local variables as

$$
\begin{aligned}
y_j &\leftarrow y_i + y_j + g(x_j) - g(x_j^{\text{old}})\\
z_j &\leftarrow z_i + z_j + h(x_j) - h(x_j^{\text{old}})\\
g_j^{\text{old}} &\leftarrow g_j\\
h_j^{\text{old}} &\leftarrow h_j\\
x_j &\leftarrow (1-\varepsilon)x_j + \varepsilon\frac{y_j}{[z_j]_c}\\
g_j &\leftarrow f_j''(x_j)x_j - f_j'(x_j)\\
h_j &\leftarrow f_j''(x_j)
\end{aligned}
$$

To describe the a-NRC algorithm in a compact form let

$$
\begin{aligned}
\boldsymbol{x} &:= [x_1,\ldots,x_N]^T\\
\boldsymbol{g}^{\text{old}} &:= \left[g_1^{\text{old}},\ldots,g_N^{\text{old}}\right]^T\\
\boldsymbol{h}^{\text{old}} &:= \left[h_1^{\text{old}},\ldots,h_N^{\text{old}}\right]^T\\
\boldsymbol{g} &:= [g_1,\ldots,g_N]^T\\
\boldsymbol{h} &:= [h_1,\ldots,h_N]^T\\
\boldsymbol{y} &:= [y_1,\ldots,y_N]^T\\
\boldsymbol{z} &:= [z_1,\ldots,z_N]^T\\
\boldsymbol{f}'(x) &:= [f_1'(x_1),\ldots,f_N'(x_N)]^T\\
\boldsymbol{f}''(x) &:= [f_1''(x_1),\ldots,f_N''(x_N)]^T
\end{aligned}
$$

and let also the notation $\boldsymbol{f}''(\boldsymbol{x}(t))\boldsymbol{x}(t)$ 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 every matrix $P_i \in \mathbb{R}^{N\times N}$, $i \in \{1,\ldots,N\}$, be

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

where e_h is the N-dimensional vector having all the components equal to zero except the h-th component which is equal to 1. Let $\mathbf{1}$ be the N-dimensional vector with all the components equal to one and observe that that, since every P_i has nonnegative elements and is s.t. $\mathbf{1}^T P_i = \mathbf{1}^T$, every P_i is column stochastic.

With this notation, and recalling that $\sigma(t) \in \{1,\ldots,N\}$ denotes the node triggering iteration t, the generic t-th iteration of the a-NRC can equivalently be described as

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

Observe that, since $\sigma(t)$ is an i.i.d. process on the alphabet $\{1,\ldots,N\}$, it follows that also the sequence $\left\{P_{\sigma(t)}\right\}_{t\geq 1}$ is i.i.d. on the alphabet $\{P_1,\ldots,P_N\}$.

Remark III.1 As already highlighted, the distributed Newton-Raphson algorithm proposed in [15] works only for undirected graphs and in a completely synchronous scenario, in the sense that all the nodes are assumed to perform the transmissions and the updates at the same time. The currently proposed scheme instead generalizes the previous ones, that can be retrieved from the currently proposed formalism simply employing a time-invariant and doubly stochastic matrix P.

Remark III.2 The design parameter ε dictates how much each node i trusts $\dfrac{y_i(t-1)}{[z_i(t-1)]_c}$ as a valid descent direction. As mentioned in [15], the synchronous NRC algorithm follows a separation of time scales, i.e., it is possible to recognize, in the dynamics of the system, two different time scales: one is related to how fast the network reaches consensus over the variables y_i and $[z_i]_c$. The other one is instead related to how fast the local guesses x_i evolve. ε then dictates the relative speed of these two dynamics. Moreover, if the consensus process is much faster than the evolution of the guesses, then the latter process approximately follows the dynamics of continuous Newton-Raphson algorithms.

As in all ordinary singularly perturbed systems, the stability of the overall system is not guaranteed for every ε. Indeed it can be numerically shown that there may exist $\varepsilon^* \in (0,1]$ (dependent on the structure of the local costs f_i and on the topology of the communication network) s.t. if $\varepsilon > \varepsilon^*$ then the overall system diverges. Unfortunately this conflicts with the practical necessity of having high ε's, since the higher its value, the faster the algorithm converges (if converging) to the optimum.

We notice that how to choose ε distributedly and dynamically is still an open question.

A. The quadratic case

We now give insights on the convergence properties of the a-NRC algorithm by restricting our attention to the quadratic case. More specifically we 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

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

Proposition III.3 *Let the local costs f_i be as in* (3)*, Assumption II.1 hold true, and $\varepsilon \in (0, 1]$. Then the trajectory $t \to \boldsymbol{x}(t)$ 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: In the quadratic case, for any $t \geq 1$

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

while, for $t = 0$,

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

For $t \geq 1$, thus, the evolution of $\boldsymbol{y}$ coincides with the evolution of that $\widetilde{\boldsymbol{y}}$ whose dynamic is described by the column-stochastic consensus algorithm

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

Moreover, since the $[\cdot]_c$ operator is never active in this quadratic case, in a similar way we have that $\boldsymbol{z}(t) = \widetilde{\boldsymbol{z}}(t)$ for $t \geq 1$, with $\widetilde{\boldsymbol{z}}(t)$ evolving as

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

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

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

and let

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

Inspired by [16], manuscript in the context of computing average consensus using non-doubly stochastic matrices, 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

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

and

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

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 [16, 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 $\{P_1, \ldots, P_N\}$, 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 $(j, i) \in \mathcal{E}$. 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,

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

where the division is once again considered element-wise, i.e.,

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

■

We remark that the previous proof ensures convergence only for the quadratic case. Nonetheless in our numerical simulations we never found a set of valid local costs leading to diverging behaviors for every $\varepsilon \in (0, 1]$. This supports our belief that, as for the original synchronous version in [15], the algorithm exhibits global convergence properties.

B. The multidimensional case

Let now $x \in \mathbb{R}^m$, $m \geq 1$, so that the local costs are defined on a multidimensional domain, i.e., $f_i : \mathbb{R}^m \to \mathbb{R}$. The extension of the scalar algorithm to the multidimensional scenario can now be immediately obtained by replacing $f_i'(x)$ with the gradient $\nabla f_i(x) \in \mathbb{R}^m$, $f_i''(x)$ with the full Hessian $\nabla^2 f_i(x) \in \mathbb{R}^{m\times m}$, by letting z_i, y_i, g_i, g_i^{old} be m-dimensional vectors, and the variables h_i, h_i^{old} be $m \times m$-square matrices.

Let now the local costs f_i be

$$f_i(x) = \frac{1}{2}(A_i^T x - b_i)^T Q_i (A_i x - b_i) \tag{4}$$

with $A_i \in \mathbb{R}^{m_i \times m}$, $Q_i \in \mathbb{R}^{m_i \times m_i}$, $b_i \in \mathbb{R}^{m_i}$, and assume the matrix $\sum_{i=1}^N A_i^T Q_i A_i$ to be invertible. In this case it is easy to show that the optimal solution of the (multidimensional) problem (1) is

$$x^* = \left(\sum_{i=1}^{N} A_i^T Q_i A_i \right)^{-1} \left(\sum_{i=1}^{N} A_i^T Q_i b_i \right).$$

Repeating the same steps performed in the proof of Proposition III.3 it is thus immediate to prove the following Proposition. With the symbol $\otimes$ we denote the Kronecker product and we observe that, in this multidimensional case, $\boldsymbol{x} = \left[x_1^T, \ldots, x_N^T\right]^T \in \mathbb{R}^{mN}$.

Proposition III.4 *Let the local costs f_i be as in* (4)*, Assumption II.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.,*

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

It is worth remarking that there are significant examples for which the optimization problem can be cast as in (4), i.e., as the sum of quadratic functions. E.g., static state estimation in power networks [17], distributed localization in sensor networks [18], and network utility maximization and resource allocation [19].

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

We now consider the realistic situation where some communication links might fail, in the sense that when node i performs a broadcast communication, not every out-neighbor receives the transmitted information. This models situations where, e.g., wireless communications fail due to packets corruption phenomena.

The aim is to suitably modify the previously presented a-NRC algorithm and make it robust against this type of communication failures. To this aim we inherit the technique proposed in [20], where authors obtain average consensus algorithms that converge to the right point 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 $j \in \mathcal{N}^{\text{in}}$. The meanings of these variables are the following:

- $b_{i,y}$ and $b_{i,z}$ are quantities owned by node i and keep track *inside node* i of the total mass of (respectively) states y_i and z_i. They are the quantities that (in this robustified version of the algorithm) are actually broadcast by node i to its out-neighbors;
- $r_{j,y}^{(i)}$ and $r_{j,z}^{(i)}$ are instead quantities owned by node j and keep track *inside node* j of the total mass of (respectively) states y_i and z_i. 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.

Thus, letting again w.l.o.g. $\sigma(t) = i$ (i.e., node i be the node triggering iteration t), the robust a-NRC becomes:

(i) node i starts by updating its local variables as

$$\begin{aligned}
y_i &\leftarrow \frac{1}{|\mathcal{N}_i^{\text{out}}| + 1}\left[y_i + g_i - g_i^{\text{old}}\right] \\
z_i &\leftarrow \frac{1}{|\mathcal{N}_i^{\text{out}}| + 1}\left[z_i + h_i - h_i^{\text{old}}\right] \\
g_i^{\text{old}} &\leftarrow g_i \\
h_i^{\text{old}} &\leftarrow h_i \\
x_i &\leftarrow (1-\varepsilon)x_i + \varepsilon \frac{y_i}{[z_i]_c} \\
g_i &\leftarrow f_i''(x_i)x_i - f_i'(x_i) \\
h_i &\leftarrow f_i''(x_i) \\
b_{i,y} &\leftarrow b_{i,y} + y_i \\
b_{i,z} &\leftarrow b_{i,z} + z_i
\end{aligned}$$

(ii) node i then broadcasts to its neighbors $b_{i,y}$ and $b_{i,z}$;

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

$$\begin{aligned}
y_j &\leftarrow b_{i,y} - r_{j,y}^{(i)} + y_j + g_j - g_j^{\text{old}} \\
z_j &\leftarrow b_{i,z} - r_{j,z}^{(i)} + z_j + h(x_j) - h(x_j^{\text{old}}) \\
g_j^{\text{old}} &\leftarrow g_j \\
h_j^{\text{old}} &\leftarrow h_j \\
x_j &\leftarrow (1-\varepsilon)x_j + \varepsilon \frac{y_j}{[z_j]_c} \\
g_j &\leftarrow f_i''(x_j)x_i - f_i'(x_j) \\
h_j &\leftarrow f_i''(x_j) \\
r_{j,y}^{(i)} &\leftarrow b_{i,y} \\
r_{j,z}^{(i)} &\leftarrow b_{i,z}
\end{aligned}$$

As shown in the following Section V, numerical evidence show that this robustification makes the algorithm able to converge to the optimal solution even in presence of a significant number of communication failures.

V. Simulations

Aims: the principal aims are to describe qualitatively the behavior of the single agents while running the procedure, and comment the effects of choosing different ε's on the convergence speed / properties of the algorithm.

We do not compare our robust a-NRC with the two currently main distributed optimization techniques present in literature, namely ADMM [1], [2] and subgradient schemes [7], 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 [15] that these algorithms are outperformed by 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.

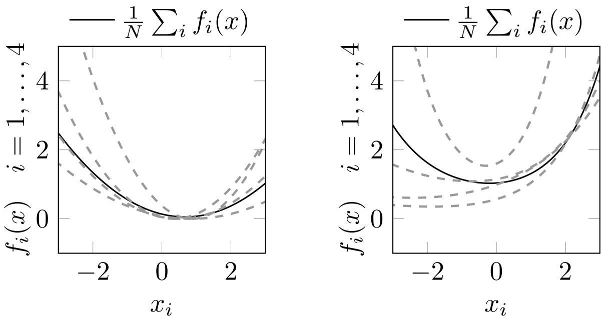
Numerical setup: to fulfill the previous aims we consider either quadratic, i.e.,

$$f_i(x) = \frac{1}{2}\left(\alpha_i' x - \alpha_i''\right)^2 \tag{5}$$

or sums of exponentials, i.e.,

$$f_i(x) = \alpha_i' \exp\left(\alpha_i'' x\right) + \alpha_i''' \exp\left(-\alpha_i'''' x\right) \tag{6}$$

local costs, with parameters randomly generated as either $[\alpha_i', \alpha_i''] \sim \mathbb{U}[0,1]^2$ or $[\alpha_i', \ldots, \alpha_i''''] \sim \mathbb{U}[0,1]^4$. The considered network is instead the random geometric network shown in Figure 2.



(a) realizations of (5) (b) realizations of (6)

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

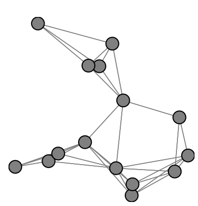
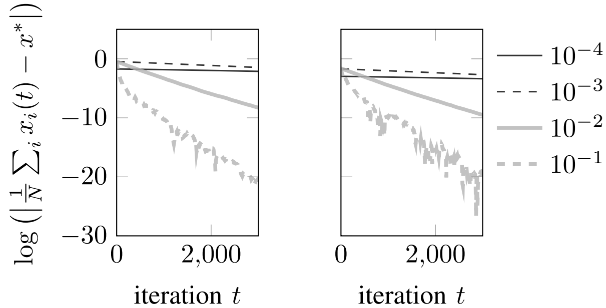


Fig. 2. The random geometric network considered for the numerical experiments of this section. It is composed by $N = 15$ nodes uniformly deployed in $[0,1]^2$ and with communication radius 0.35.

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 agents 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.

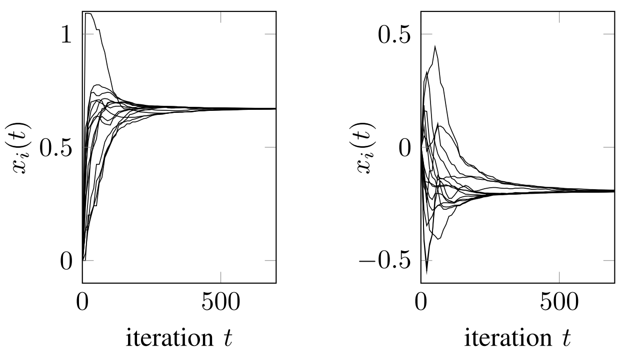
As expected, increasing ε leads to faster convergence speeds. Nonetheless, too big ε's may lead to instability and diverging phenomena (a common issue of schemes that are based on separation of time-scales concepts). We remark that dynamically finding the best ε (that depends on several factors, mainly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



(a) evolutions under costs (5) (b) evolutions under costs (6)

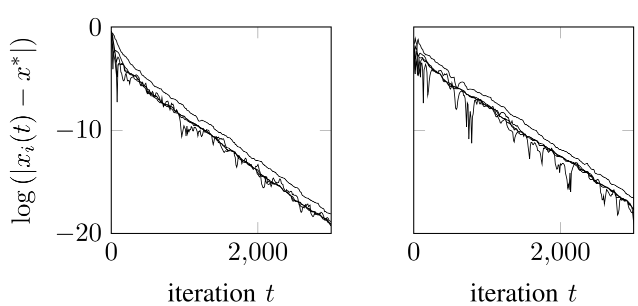
Fig. 3. Comparisons of the dependence of the convergence speed of the algorithm on ε for different cost functions.

Regarding the behavior of the single agents, Figures 4 and 5 plot respectively the evolutions of the local guesses and of 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 (5) (b) evolutions under costs (6)

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



(a) evolutions under costs (5) (b) evolutions under costs (6)

Fig. 5. Evolution of the local errors of the various agents 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 [15]. 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. More specifically, our proofs of convergence exploit asynchronous, broadcast, reliable communications and quadratic local costs. Thus proving its 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 agents to be able to detect diverging behaviors; *ii)* updating the state x with partition-based approaches, meaning that (in the same spirit of [21]) each agent 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. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[2] 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.

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

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

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

[6] 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.

[7] 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.

[8] E. Ghadimi, M. Johansson, and I. Shames, "Accelerated gradient methods for networked optimization," in *American Control Conference (ACC), 2011*. IEEE, 2011, pp. 1668–1673.

[9] P. Lin and W. Ren, "Distributed subgradient projection algorithm for multi-agent optimization with nonidentical constraints and switching topologies," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*. IEEE, 2012, pp. 6813–6818.

[10] A. Nedic and A. Olshevsky, "Distributed optimization over time-varying directed graphs," in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, Dec 2013, pp. 6855–6860.

[11] 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.

[12] ——, "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.

[13] 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.

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

[15] 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.

[16] 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.

[17] F. Pasqualetti, R. Carli, and F. Bullo, "Distributed estimation via iterative projections with application to power network monitoring," *Automatica*, vol. 48, no. 5, pp. 747–758, 2012.

[18] S. Bolognani, S. D. Favero, L. Schenato, and D. Varagnolo, "Consensus-based distributed sensor calibration and least-square parameter identification in wsns," *International Journal of Robust and Nonlinear Control*, vol. 20, no. 2, pp. 176–193, 2010.

[19] S. H. Low and D. E. Lapsley, "Optimization flow control - i: Basic algorithm and convergence," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 861–874, 1999.

[20] 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.

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