ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

ARCHIVIO ISTITUZIONALE
DELLA RICERCA

Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

Multi-Agent Newton-Raphson Optimizaton Over Lossy Networks

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version:
Bof, N., Carli, R., Notarstefano, G., Schenato, L., Varagnolo, D. (2019). Multi-Agent Newton-Raphson Optimizaton Over Lossy Networks. IEEE TRANSACTIONS ON AUTOMATIC CONTROL, 64(7), 2983-2990 [10.1109/TAC.2018.2874748].

Availability:
This version is available at: https://hdl.handle.net/11585/671895 since: 2020-02-23

Published:
DOI: http://doi.org/10.1109/TAC.2018.2874748

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/).
When citing, please refer to the published version.

(Article begins on next page)

12 October 2024

This is the final peer-reviewed accepted manuscript of:

N. Bof, R. Carli, G. Notarstefano, L. Schenato and D. Varagnolo, "Multiagent Newton–Raphson Optimization Over Lossy Networks," in IEEE Transactions on Automatic Control, vol. 64, no. 7, pp. 2983-2990, July 2019.

The final published version is available online at: https://doi.org/10.1109/TAC.2018.2874748

Rights / License:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/)

When citing, please refer to the published version.

Multiagent Newton-Raphson Optimization over lossy networks

Nicoletta Bof Ruggero Carli Giuseppe Notarstefano Luca Schenato Damiano Varagnolo

***Abstract*—In this work we study the problem of unconstrained convex-optimization in a fully distributed multi-agent setting, which includes asynchronous computation and lossy communication. In particular, we extend a recently proposed algorithm named Newton-Raphson Consensus by integrating it with a broadcast-based average consensus algorithm which is robust to packet losses. We show via the separation of time scales principle that under mild conditions (i.e., persistency of the agents activation and bounded consecutive communication failures) the proposed algorithm is provably locally exponentially stable with respect to the optimal global solution. Finally, we complement the theoretical analysis with numerical simulations and comparisons based on real datasets.**

I. Introduction

Recently, we have been witnessing a surge of interest in multi-agent distributed optimization in peer-to-peer networks. The main reason is the advent of Internet-of-Things and Smart Cyber-physical Systems, where a large multitude of electronic devices are capable of sensing, communicating, and of autonomous decision making through cooperation [1]. To cope with real-world requirements, distributed convex algorithms need to be designed to work under asynchronous and directed communication with possibly random delay. Moreover, in peer-to-peer networks where communication is wireless, packet loss often becomes one of the major bottlenecks.

A large body of literature has addressed the problem of distributed optimization subject to asynchronous communication and updates. A popular class of algorithms that are able to cope with *asynchronous* updates and time-varying graphs is the one of Distributed Subgradient Methods (DSMs). They are simple to implement, can cope with non-differentiable convex cost functions, and require only the computation of local (sub)-gradients. However, these algorithms exhibit sub-linear converge rates even if the cost functions are smooth [2], [3], [4] and do not take into account packet loss explicitly.

Another popular class of distributed optimization algorithms is based on dual decomposition schemes. In this case the related literature is very large and we refer to [5] for a comprehensive tutorial. Among these algorithms, the Alternating Direction Method of Multipliers (ADMM) has attracted the attention of the scientific community for its simple distributed implementation and good convergence speed (see for example the survey [6]). Most of the literature on ADMM is based on synchronous implementation with reliable communication, however some recent works consider asynchronous (but reliable) communication [7], [8], [9], and asynchronous scenarios with edge-based or node-based activation schemes [10]. Some recent works have moreover addressed the problem of random delay in the communication/updates rounds in ADMM schemes [11], [12], [9]. However these strategies are restricted to networks with server-client communication topologies and do not explicitly address packet losses.

A third class of optimization algorithms, usually referred to as Newton-based methods (NBMs), consists of strategies that exploit second-order derivatives, i.e., the Hessians of the cost functions for computing descent directions. These methods are however limited to synchronous reliable communication [13], [14] or reliable symmetric gossip communication schemes [15]. The works [15] and [14] have introduced for the first time the idea of tracking the gradient of the whole cost function as a mean to replace the diminishing stepsize with a constant one. This idea has been reconsidered with different averaging schemes and formalized in [16] to handle nonconvex optimization (combined with a successive convex approximation approach) and in [17] and [18] to show linear convergence with constant stepsize.

All the aforementioned literature has not directly addressed situations where the communications are unreliable and lossy. Differently from what one might believe, packet loss is not equivalent to asynchronous communication nor to random delay. In fact in the literature considering asynchronous communication there is the tacit assumption that the transmitter knows which neighbours successfully received the transmitted packet. In the literature considering random delay all variables are stored at a single location (server) and only computation is distributed (clients), therefore memory consistency is not a concern, and instead is the case in peer-to-peer networks. *The main contribution of this work is to propose a set of distributed optimization algorithms that are robust to packet losses for general asynchronous peer-to-peer networks and that are guaranteed to have (local) exponential convergence.*

More specifically, we robustify the Newton-Raphson approach initially proposed in [14] by introducing a new consensus algorithm that is an ad-hoc combination of two known consensus schemes: *i)* the *ratio* or *push-sum* consensus, useful to compute averages in networks with directed communication graphs (i.e., networks using broadcast protocols [19]); *ii)* the *robust consensus* algorithm, which allows for a robust computation of arithmetic averages over networks with lossy communication [20]. The new scheme is the first distributed optimization algorithm able to deal with simultaneously asynchronous and lossy communication protocols. The proof of its convergence is based on time-scale separation and Lyapunov theories, and extends the results in [21], where the convergence was proved only for quadratic cost functions. We complement the theoretical results with numerical simulations based on real datasets under lossy, broadcast communication. We also numerically compared our algorithms against two algorithms recently proposed in [22], [17] under the special case of asynchronous lossless communication, showing the better performance of our proposed Newton-Raphson approach.

The paper is organized as follows: Section II formulates our problem and working assumptions. Section III presents the building blocks of the scheme proposed in this manuscript. Section IV then introduces the main distributed optimization algorithm and gives some intuitions on the convergence properties of the scheme, summarized then in Section V. Finally, Section VI collects some numerical experiments corroborating the theoretical results, while Section VII draws some concluding remarks and future research directions.

This work is partially supported by the Celtic Plus project *SENDATE-Extend* (C2015/3-3), the Swedish research council Norrbottens Forskningsråd project *DISTRACT*, and by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 638992 - OPT4SMART).

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

Giuseppe Notarstefano is with the Department of Electrical, Electronic and Information Engineering, University of Bologna, Bologna, Italy, giuseppe.notarstefano@unibo.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.

II. Problem Formulation and Assumptions

We consider the separable optimization problem

$$x^* := \operatorname*{argmin}_x f(x) = \operatorname*{argmin}_x \sum_{i=1}^{N} f_i(x), \tag{1}$$

where $x \in \mathbb{R}^n$ and where the local costs $f_i : \mathbb{R}^n \mapsto \mathbb{R}$ satisfy:

Assumption II.1 (Cost smoothness) *Each f_i is known only to node i, is $\mathcal{C}^3$, and is strongly convex, i.e., its Hessian is bounded from below, $\nabla^2 f_i(x) > cI_n$ for all x, with $c > 0$ some positive scalar*[1].

The communication among nodes is modeled via a communication graph that satisfies the following:

Assumption II.2 (Network connectivity) *The communication graph among the nodes is fixed, directed and strongly connected, i.e., for each pair of nodes there is at least one directed path connecting them.*

More formally, the communication graph is represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V} = \{1, \ldots, N\}$ and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ so that $(i, j) \in \mathcal{E}$ iff node j can directly receive information from node i. With $\mathcal{N}_i^{\text{out}}$ we denote the set of *out-neighbors* of node i, i.e., $\mathcal{N}_i^{\text{out}} := \{j \in \mathcal{V} \mid (i, j) \in \mathcal{E}, i \neq j\}$ is the set of nodes receiving messages from i. Similarly, with $\mathcal{N}_i^{\text{in}}$ we denote the set of *in-neighbors* of i, i.e., $\mathcal{N}_i^{\text{in}} := \{j \in \mathcal{V} \mid (j, i) \in \mathcal{E}\ i \neq j\}$. Their cardinality is indicated by $|\mathcal{N}_i^{\text{out}}|$ and $|\mathcal{N}_i^{\text{in}}|$ respectively.

Notice that in some distributed systems, as in Wireless Sensor Networks, the communication graph is often undirected, in the sense that a node can transmit to any node from which it can receive. However, communication is typically only half-duplex, i.e., two nodes cannot communicate simultaneously, so that protocols with multiple communication rounds and reliable acknowledge (ACK) mechanisms are needed for bidirectional communication. This, in turn, requires pairwise synchronization and results in substantial delays. As so, algorithms that are suitable for broadcast-based (directed) communication without ACK, such as UDP, are extremely valuable also for undirected graphs.

As for the concept of time, we assume that the local variables at each node are updated at discrete time instants (e.g., based on local and possibly non-synchronized clocks, or based on events like receiving a packet). Thus, from a global perspective, we collect and order all time instants when at least one variable in one node is updated and refer to it as the sequence $\{t_k\}_{k=1}^{\infty}$. With a little abuse of notation we will then write $x(k) = x(t_k)$ and we will study the time evolution of the nodes variables as a discrete-time system.

Our objective is to design an algorithm solving (1) with the following features: *F1): Asymptotic global estimation*: each agent wants to obtain an estimate of the global minimizer that asymptotically converges to the optimal solution x^*; *F2): Peer-to-peer (leaderless)*: each node has limited computational and memory resources and is allowed to communicate directly only with its neighbors; *F3): Distributed*: the update-rule of the local variables at each node depends only on the variables stored by the local node and by its neighbors; *F4): Asynchronous*: none, one or multiple nodes can communicate or update their variables at any given time; *F5): Lossy broadcast communication without ACK*: communication can be broadcast-based with no ACK mechanism. To the best of authors' knowledge, none of the previously cited works possesses all the previous features simultaneously.

[1]With a little abuse of notation we use the symbols $\nabla f(\cdot)$ and $\nabla^2 f(\cdot)$ to indicate the gradient and Hessian of the cost function $f(\cdot)$, respectively. Also, given two matrices $A, B \in \mathbb{R}^{n \times n}$, the notation $A > B$ means that $A - B$ is a positive definite matrix.

III. Building blocks

The here proposed algorithm consists of three different building blocks: *i) Newton-Raphson Consensus*, proposed in [14] to solve problem (1); *ii) the push-sum algorithm*, initially proposed in [19] as an asynchronous average consensus protocol; and *iii) the robust ratio consensus algorithm*, initially proposed in [20] as a robust average consensus protocol. While possessing the first three features mentioned above (i.e., F1-F3), the Newton-Raphson Consensus is nonetheless limited since it assumes synchronous and reliable communications. The two adopted consensus schemes (ratio consensus and its robust version) are nonetheless limited since assume respectively reliable communications and synchronous updates. The major contribution of this work is to suitably modify and integrate the three schemes above to design a distributed optimization algorithm that solves problem (1) and that exhibits all the features F1-F5 above. The main challenge in doing this is that the interaction between these algorithms might lead to instability unless some suitable assumptions are considered and used to establish opportune convergence properties. The key mathematical machinery that will be used to this means is Lyapunov theory and separation of time-scales.

Before providing the description of the proposed overall optimization scheme, we offer a brief description of its three aforementioned building blocks.

A. Newton-Raphson Consensus

Newton-Raphson Consensus [14] is based on the observation that the standard Newton-Raphson update in the standard centralized scenario with a single agent can be written as

$$\begin{aligned} x^+ &= x - \varepsilon(\nabla^2 f(x))^{-1} \nabla f(x) \\ &= (1-\varepsilon)x + \varepsilon\big(\nabla^2 f(x)\big)^{-1}\big(\nabla^2 f(x)\, x - \nabla f(x)\big) \\ &= (1-\varepsilon)x + \varepsilon\underbrace{\big(\sum_i \nabla^2 f_i(x)\big)}_{=:h(x)}{}^{-1}\underbrace{\big(\sum_i (\nabla^2 f_i(x)\, x - \nabla f_i(x))\big)}_{=:g(x)}, \end{aligned}$$

where we used the simplified notation x^+ to indicate $x(k+1)$ and x to indicate $x(k)$. This system is exponentially stable as long as the parameter $\varepsilon > 0$, which acts as a stepsize, is chosen in a proper way [14]. If we now assume that all agents can have a different value of x_i and we mimic the previous algorithm, we get the N local updates:

$$x_i^+ = (1-\epsilon)x_i + \varepsilon\underbrace{\big(\sum_j \underbrace{\nabla^2 f_j(x_j)}_{=:h_j(x_j)}\big)}_{=:\overline{h}(x_1,\ldots,x_N)}{}^{-1}\underbrace{\big(\sum_j \underbrace{(\nabla^2 f_j(x_j)\, x_j - \nabla f_j(x_j))}_{=:g_j(x_j)}\big)}_{=:\overline{g}(x_1,\ldots,x_N)}. \tag{2}$$

The dynamics of the N local systems are identical and exponentially stable; therefore, since they are all driven by the same forcing term $\kappa(x_1, \ldots, x_n) = (\overline{h}(x_1, \ldots, x_N))^{-1}\overline{g}(x_1, \ldots, x_N)$, intuitively we expect that

$$x_i - x_j \to 0, \quad \forall i, j\ ,$$

which implies that all local variables will be identical. If this is the case, then the dynamics of each local system will eventually become the dynamics of a standard centralized Newton-Raphson algorithm. This algorithm, however, requires each agent to be able to instantaneously compute the two sums $\overline{h}, \overline{g}$, which is obviously not possible in a distributed computation set-up. The original paper [14] extends the standard Newton-Raphson algorithm into a distributed scenario via the use of synchronous lossless average consensus protocols that compute these sums asymptotically, while [15] extends it to the case of asynchronous gossip-based lossless average consensus strategies.

B. Push-sum Consensus

The Newton-Raphson Consensus scheme described in Section III-A requires each node to compute the two sums $y_i = \overline{g}$ and $z_i = \overline{h}$ at least asymptotically in order to apply a Newton-Raphson descent. In fact, since the ratio of the two quantities is needed, each agent can asymptotically converge to a scaled version of the two. That is, assuming each variable x_i, $i \in \mathcal{V}$, to be fixed, we require

$$\begin{aligned} y_i &\rightarrow \eta_i \overline{g}(x_1, \ldots, x_N) = \eta_i \sum_j g_j(x_j) \\ z_i &\rightarrow \eta_i \overline{h}(x_1, \ldots, x_N) = \eta_i \sum_j h_j(x_j), \end{aligned}$$

where $\eta_1, \ldots, \eta_N$ are possibly time-dependent non-zero scalars. Here the right arrow means that the difference between left and right hand-sides goes to zero as the iteration counter goes to infinity. Having identified our aim, we first describe the push-sum algorithm, which is able to solve the given problem in an asynchronous communication scenario. Then, we describe the robust ratio consensus which is able to solve the problem in a scenario where the communication is unreliable but the protocol is synchronous. One of the aims of this work will be the merging of these two schemes to obtain a robust and asynchronous consensus algorithm.

Under synchronous communication, the local updates of the *push-sum* or *ratio consensus* introduced in [19] are, for each $i \in \mathcal{V}$,

$$y_i^+ = \frac{1}{|\mathcal{N}_i^{\text{out}}| + 1} y_i + \sum_{j \in \mathcal{N}_i^{\text{in}}} \frac{1}{|\mathcal{N}_j^{\text{out}}| + 1} y_j \tag{3}$$

$$z_i^+ = \frac{1}{|\mathcal{N}_i^{\text{out}}| + 1} z_i + \sum_{j \in \mathcal{N}_i^{\text{in}}} \frac{1}{|\mathcal{N}_j^{\text{out}}| + 1} z_j, \tag{4}$$

paired with the initialization $y_i(0) = g_i(x_i)$, $z_i(0) = h_i(x_i)$. Assuming for notation simplicity a scalar optimization problem, the previous update can be written as

$$\begin{aligned} \boldsymbol{y}^+ &= P\boldsymbol{y} \\ \boldsymbol{z}^+ &= P\boldsymbol{z}, \end{aligned}$$

where $\boldsymbol{y} = [y_1 \cdots y_N]^T$, $\boldsymbol{z} = [y_1 \cdots z_N]^T$. In this way the matrix P results to be column-stochastic and its induced graph $\mathcal{G}_P$ (i.e., $(i,j) \in \mathcal{G}_P$ if $[P]_{ji} \neq 0$) coincides with the original communication graph (i.e., $\mathcal{G}_P = \mathcal{G}$). Since we assume $\mathcal{G}$ to be strongly connected, this guarantees that

$$\begin{aligned} y_i &\rightarrow \eta_i \textstyle\sum_i y_i(0) = \eta_i \sum_i g_i(x_i) = \eta_i \overline{g}(x_1, \ldots, x_N) \\ z_i &\rightarrow \eta_i \textstyle\sum_i z_i(0) = \eta_i \sum_i h_i(x_i) = \eta_i \overline{h}(x_1, \ldots, x_N), \end{aligned}$$

where $\eta = [\eta_1 \cdots \eta_N]^T$ is the right eigenvector of P relative to the unique unitary eigenvalue, i.e., $P\eta = \eta$ and $\eta_i > 0, \forall i$.

The ratio consensus described above can then be extended to asynchronous implementations (as proposed in [19]). Let at any time k only one node i activate, update its variables, and broadcast them to its out-neighbors, and then, consistently, let the generic receiving node j update its local variables. The update rules for y_i and y_j therefore become

$$y_i^+ = \frac{1}{|\mathcal{N}_i^{\text{out}}| + 1} y_i \tag{5}$$

$$y_j^+ = y_j + \frac{1}{|\mathcal{N}_i^{\text{out}}| + 1} y_i = y_j + y_i^+ \quad \forall j \in \mathcal{N}_i^{\text{out}}, \tag{6}$$

(the rules for z_i and z_j being equal in structure). In this scenario, the global dynamics can be described by a time-varying consensus matrix that depends on the specific node that is activated, i.e., $P(k) \in \{P_1, \ldots, P_N\}$, where the matrices P_i are still column-stochastic. As shown via weak ergodic theory considerations in [19], if the activation of the nodes is randomized and i.i.d. then the local variables converge to

$$\begin{aligned} y_i &\rightarrow \eta_i(k) \textstyle\sum_i y_i(0) = \eta_i(k) \sum_i g_i(x_i) = \eta_i(k) \overline{g}(x_1, \ldots, x_N) \\ z_i &\rightarrow \eta_i(k) \textstyle\sum_i z_i(0) = \eta_i(k) \sum_i h_i(x_i) = \eta_i(k) \overline{h}(x_1, \ldots, x_N), \end{aligned} \tag{7}$$

where $\eta_i(k) > 0$ is time-varying and depends on the activation sequence of the nodes.

C. Robust ratio consensus

The synchronous ratio-consensus strategy defined by iterations (3) and (4) in Section (III-B) loses its convergence properties in case of lossy communications. A naïve attempt to solve this problem is then to use a buffer such that when i does not receive a message from j then i updates its local variables by using the latest values that it has received from j. Focusing only on (3) to avoid repetitions, mathematically this corresponds to add an additional local variable $y_i^{(j)}$ representing the latest y_j received by i from j, and to transform the update rule (3) into

$$\begin{aligned} y_i^{(j)+} &= \begin{cases} y_j & \text{if } y_j \text{ is received} \\ y_j^{(i)} & \text{otherwise} \end{cases} \quad \forall i \in \mathcal{V}, \forall j \in \mathcal{N}_i^{\text{in}} \\ y_i^+ &= \frac{1}{|\mathcal{N}_i^{\text{out}}| + 1} y_i + \sum_{j \in \mathcal{N}_i^{\text{in}}} \frac{1}{|\mathcal{N}_j^{\text{out}}| + 1} y_i^{(j)}, \quad \forall i \in \mathcal{V}. \end{aligned}$$

However this solution does not preserve the total mass of the variables y_i during the progress of the algorithm, i.e., $\sum_i y_i(k) \neq \sum_i y_i(0)$, differently from the original lossless ratio consensus. This eventually leads the average consensus algorithm not to converge to the desired value.

To overcome this issue, it is possible to add some additional "mass counter" variables $\sigma_{i,y}, \rho_{i,y}^{(j)}$ that guarantee the preservation of the masses even in the presence of packet losses [20]. More specifically, in this way the synchronous update (3) transforms into

$$\sigma_{i,y}^+ = \sigma_{i,y} + y_i, \quad \forall i \in \mathcal{V} \tag{8}$$

$$\rho_{i,y}^{(j)+} = \begin{cases} \sigma_{j,y} & \text{If } \sigma_{j,y} \text{ is received} \\ \rho_{i,y}^{(j)} & \text{otherwise} \end{cases} \quad \forall j \in \mathcal{N}_i^{\text{in}} \tag{9}$$

$$y_i^+ = \frac{1}{|\mathcal{N}_i^{\text{out}}| + 1} y_i + \sum_{j \in \mathcal{N}_i^{\text{in}}} \frac{1}{|\mathcal{N}_j^{\text{out}}| + 1} (\rho_{i,y}^{(j)+} - \rho_{i,y}^{(j)}), \tag{10}$$

where the "mass counter" variables are initialized to zero, i.e., $\sigma_{i,y}(0) = \rho_{i,y}^{(j)}(0) = 0$ for every i and j.

Observe that each node i has two types of counters: $\sigma_{i,y}(k)$ to keep track of the total y-mass sent to its neighbors up to iteration k, and a $\rho_{i,y}^{(j)}(k)$ for each in-neighbor $j \in \mathcal{N}_i^{\text{in}}$ to take into account the total y-mass received from j up to iteration k. If during iteration k node i receives information from node j, the information related to node j used in the update of the variable y_i is $\nu_{i,y}^{(j)}(k) = \sigma_{j,y} - \rho_{i,y}^{(j)}$. The fictitious variable $\nu_{i,y}^{(j)}(k)$ corresponds to a "virtual mass" stored on edge $(j,i) \in \mathcal{E}$. Under reliable transmission this virtual mass is zero. If packet losses occur, instead, then this variable accumulates the additional mass that node j wants to transfer to node i, so that therefore this mass is not lost. As so, the total mass stored on the nodes and the edges is preserved, regardless of the packet loss sequence. Thus, for each time instant k,

$$\sum_i \left(y_i(k) + \sum_{j \in \mathcal{N}_i^{\text{in}}} \nu_{i,y}^{(j)}(k) \right) = \sum_i y_i(0). \tag{11}$$

Let $\boldsymbol{y}$ and $\boldsymbol{\nu}_y$ then be the vectors collecting respectively the variables y_i, $i \in V$, and $\nu_{i,y}^{(j)}$, $i \in V$ and $j \in \mathcal{N}_i^{\text{in}}$. Accordingly, let

$\boldsymbol{y}_a$ be the augmented variable defined as $\boldsymbol{y}_a = \left[\boldsymbol{y}^T \ \boldsymbol{\nu}_y^T\right]^T$. Similarly, let $\boldsymbol{z}_a = \left[\boldsymbol{z}^T \ \boldsymbol{\nu}_z^T\right]^T$. It can be shown that

$$\boldsymbol{y}_a^+ = M(k)\boldsymbol{y}_a, \qquad \boldsymbol{z}_a^+ = M(k)\boldsymbol{z}_a,$$

where $M(k)$ is an augmented column-stochastic matrix, and that, from weak ergodicity theory, the local variables y_i and z_i converge asymptotically as in (7) [20]. As it will be clear in the next sections, the matrix $M(k)$ will be a building block for the design and analysis of the here proposed distributed optimization algorithm.

IV. The Robust Asynchronous Newton-Raphson Consensus (ra-NRC)

This section merges the three building blocks *Newton-Raphson Consensus*, *push-sum consensus* and *robust ratio consensus* into one algorithm, called robust asynchronous Newton-Raphson Consensus (ra-NRC), that solves problem (1) and exhibits all the features listed in Section II. The algorithm can be organized in a block scheme as in Figure 1.

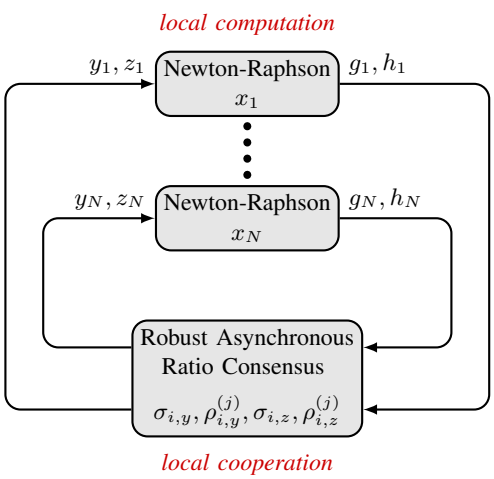

Fig. 1. Graphical representation of the robust asynchronous Newton-Raphson Consensus (ra-NRC).

We propose a "meta distributed algorithm" which can result in different distributed algorithms depending on the (possibly asynchronous and packet-lossy) communication protocol that is desired to be implemented in the network. The meta algorithm consists of four main blocks of code implemented by each node $i \in \mathcal{V}$ in the network: *Initialization* (at startup), *Data Transmission*, *Data Reception* and *Estimate Update*.

Except for the first block, which is executed only once at startup, the blocks can be executed asynchronously and with possibly different execution rates. The scheduling of these three blocks, for each agent i, is determined by three binary variables $\texttt{flag}_{\texttt{transmission},i}, \texttt{flag}_{\texttt{reception},i}, \texttt{flag}_{\texttt{update},i}$ whose evolutions are determined by the communication protocol. Each code block is assumed to be executed *sequentially* and *atomically*, i.e., the local variables and flags cannot be changed by any other process. For example, if a node is executing *Estimate Update* and a new packet is incoming, this packet is either dropped or placed in a buffer until *Estimate Update* is not completed. Thus, a distributed algorithm will be simply the combination of the given meta scheme with a communication protocol defining how the flags are activated. For example, in an event-triggered communication protocol the reception of a packet may sequentially trigger (if no other block is being executed) the *Data Reception* block, which then triggers the *Estimate Update* block, and that finally triggers the *Data Transmission* block. In the following we assume that when an agent is idle, it is always ready to receive a new packet and when a packet is received by the i-th node then $\texttt{flag}_{\texttt{reception},i}$ is set to one.

One of the strengths of the proposed algorithm is that it is independent of the specific communication protocol as long as it satisfies some mild assumptions in terms of minimum scheduling rate of each block and maximum consecutive packet losses, which will be formally stated in the next section. We are, then, ready to provide a pseudo-code description of ra-NRC as in Algorithm 1. Notice that the local variables in the algorithm mimic the variable names and purpose of the ones defined in the previous section.

Algorithm 1 robust asynchronous Newton-Raphson Consensus (ra-NRC) for node i

Require: x^o, ε, c

Initialization (atomic)

1: $x_i \leftarrow x^o$
2: $y_i \leftarrow 0, \ \ g_i \leftarrow 0, \ \ g_i^{old} \leftarrow 0$
3: $z_i \leftarrow I_n, \ \ h_i \leftarrow I_n, \ \ h_i^{old} \leftarrow I_n$
4: $\sigma_{i,y} \leftarrow 0, \ \ \sigma_{i,z} \leftarrow 0$
5: $\rho_{i,y}^{(j)} \leftarrow 0, \ \ \rho_{i,z}^{(j)} \leftarrow 0, \quad \forall j \in \mathcal{N}_i^{\text{in}}$
6: $\texttt{flag}_{\texttt{reception},i} \leftarrow 0, \ \ \texttt{flag}_{\texttt{update},i} \leftarrow 0$
7: $\texttt{flag}_{\texttt{transmission},i} \leftarrow 1$

Data Transmission (atomic)

8: **if** $\texttt{flag}_{\texttt{transmission},i} = 1$ **then**
9: $\quad \texttt{transmitter_node_ID} \leftarrow i$
10: $\quad y_i \leftarrow \frac{1}{|\mathcal{N}_i^{\text{out}}|+1} y_i$
11: $\quad z_i \leftarrow \frac{1}{|\mathcal{N}_i^{\text{out}}|+1} z_i$
12: $\quad \sigma_{i,y} \leftarrow \sigma_{i,y} + y_i$
13: $\quad \sigma_{i,z} \leftarrow \sigma_{i,z} + z_i$
14: $\quad$ Broadcast: $\texttt{transmitter_node_ID}, \sigma_{i,y}, \sigma_{i,z}$
15: $\quad \texttt{flag}_{\texttt{transmission},i} \leftarrow 0$
16: **end if**

Data Reception (atomic)

17: **if** $\texttt{flag}_{\texttt{reception},i} = 1$ **then**
18: $\quad j \leftarrow \texttt{transmitter_node_ID}, \quad (j \in \mathcal{N}_i^{\text{in}})$
19: $\quad y_i \leftarrow y_i + \sigma_{j,y} - \rho_{i,y}^{(j)}$
20: $\quad z_i \leftarrow z_i + \sigma_{j,z} - \rho_{i,z}^{(j)}$
21: $\quad \rho_{i,y}^{(j)} \leftarrow \sigma_{j,y}$
22: $\quad \rho_{i,z}^{(j)} \leftarrow \sigma_{j,z}$
23: $\quad \texttt{flag}_{\texttt{reception},i} \leftarrow 0$
24: $\quad \texttt{flag}_{\texttt{update},i} \leftarrow 1$ (optional)
25: **end if**

Estimate Update (atomic)

26: **if** $\texttt{flag}_{\texttt{update},i} = 1$ **then**
27: $\quad x_i \leftarrow (1-\varepsilon)x_i + \varepsilon {z_i}^{-1} y_i$
28: $\quad g_i^{\text{old}} \leftarrow g_i$
29: $\quad h_i^{\text{old}} \leftarrow h_i$
30: $\quad h_i \leftarrow \nabla^2 f_i(x_i)$
31: $\quad g_i \leftarrow h_i x_i - \nabla f_i(x_i)$
32: $\quad y_i \leftarrow y_i + g_i - g_i^{\text{old}}$
33: $\quad z_i \leftarrow z_i + h_i - h_i^{\text{old}}$
34: $\quad \texttt{flag}_{\texttt{update},i} \leftarrow 0$
35: $\quad \texttt{flag}_{\texttt{transmission},i} \leftarrow 1$ (optional)
36: **end if**

We now provide a detailed explanation of the pseudo-code.

The first block *Initialization* (lines 1-7) is a one-time operation preformed by each node at the beginning of the algorithm. The only free parameter to be set is the initial estimate x^o for the global optimization, while all the other variables are set to zero or to identity matrices with proper dimensions.

The blocks *Data Transmission* (lines 8-16) and *Data Reception* (lines 17-25) implement a new Robust Asynchronous Ratio Consensus (see the bottom block in Figure 1) that merges the benefits of the push-sum algorithm (and its asynchronous nature) with the robust ratio consensus (and its resilience to packet losses). Moreover, our proposed Robust Asynchronous Ratio Consensus has the advantage to be *fully parallel*, in the sense that multiple nodes can transmit at the same time, since any potential collision will result in a packet loss already handled by the algorithm. Specifically, the update of variables y_i, z_i at the time of transmission (line 10-11) are the same as in the push-sum consensus given by (5). The update for $\sigma_{i,y}$ in the algorithm (line 12) is identical to (8), however the variable $\sigma_{i,y}$ in our algorithm is based on the value of y_i that has been updated above (line 10). Therefore the various variables $\sigma_{i,y}$'s in Algorithm 1 are scaled by a factor $\frac{1}{|\mathcal{N}_i^{\text{out}}|+1}$ as compared to those in (8). Since the variables $\rho_{j,y}^{(i)}$ will be just (possibly delayed) copies of the variable $\sigma_{i,y}$ (line 21), also these variables are scaled by a factor $\frac{1}{|\mathcal{N}_i^{\text{out}}|+1}$ as compared to those appearing in (9). Similar arguments apply for the variables related to $\sigma_{i,z}$, $\rho_{j,z}^{(i)}$. Once the update of the variables has been completed, the transmitting node broadcasts only the variables $\sigma_{i,y}$ and $\sigma_{i,z}$ plus its ID to its neighbors. After transmitting the node returns then to an idle-mode (line 15). If a neighboring node i is in the receiving mode and actually receives a message (line 17), then it extracts the transmitter node ID j and the corresponding variables $\sigma_{j,y}$, $\sigma_{i,z}$ (line 18). The variable y_i is then updated similarly to (6), where y_i^+ is replaced by the term $\sigma_{j,y} - \rho_{i,y}^{(j)}$, which is the same as the last term appearing in (10)[2]. The local variable $\rho_{i,y}^{(j)}$ is then updated (line 21) similarly to (9) in the robust ratio consensus.

The last block, i.e., *Estimate Update*, is responsible for implementing a local Newton-Raphson step. The update of the local estimate x_i of the global optimizer, available at each node i, is performed via the Newton-Raphson Consensus described in the previous section. In practice, the roles of y_i and z_i are those of (scaled) local approximations of the global functions $\overline{g}(x_1, \ldots, x_N)$ and $\overline{h}(x_1, \ldots, x_N)$ defined above. As so, mimicking (2), the proposed algorithm uses these variables to implement an approximated Newton-Raphson (line 27), where the variable ε corresponds to the stepsize. Since the local variables x_i are continuously updated, also the global functions $\overline{g}(x_1, \ldots, x_N) = \sum_i g_i(x_i)$ and $\overline{h}(x_1, \ldots, x_N) = \sum_i h_i(x_i)$ need to be updated accordingly. This cannot be done instantaneously due to the networked nature of the framework and has to be achieved through the asynchronous robust ratio consensus (see Figure 1). In order to be able to track the continuously changing signals g_i and h_i, each node has to compute these signals before and after updating the x_i (g_i^{old} and h_i^{old} in lines (28-29), plus g_i and h_i lines (30-31), respectively) and then update the "consensus" variables y_i and z_i in order to track the current sums $\overline{g}(x_1, \ldots, x_N)$ and $\overline{h}(x_1, \ldots, x_N)$ (lines 32-33). In fact, this operation guarantees that, similarly to (11), the following invariants are preserved:

$$\sum_i (y_i(k) + \sum_{j \in \mathcal{N}_i^{\text{in}}} (\sigma_{j,y}(k) - \rho_{i,y}^{(j)}(k))) = \sum_i g_i(k), \quad (12)$$

$$\sum_i (z_i(k) + \sum_{j \in \mathcal{N}_i^{\text{in}}} (\sigma_{j,z}(k) - \rho_{i,z}^{(j)}(k))) = \sum_i h_i(k), \quad (13)$$

where, with a slight abuse of notation, with $g_i(k)$ and $h_i(k)$ we denote $g_i(x_i(k))$ and $h_i(x_i(k))$ respectively. Intuitively he algorithm will converge if the local estimates x_i change more slowly than the rate at which the asynchronous robust ratio consensus converges. This separation of time scales can then be achieved by choosing a sufficiently small stepsize ε, so that we can expect that

$$y_i(k) \quad \rightarrow \quad \eta_i(k) \sum_i g_i(k), \quad (14)$$

$$z_i(k) \quad \rightarrow \quad \eta_i(k) \sum_i h_i(k). \quad (15)$$

A formal characterization of the ra-NRC algorithm and the necessary conditions in terms of node activation and packet loss frequencies, when a particular communication protocol is adopted, are given in the next section.

Remark IV.1 The robust asynchronous Newton-Raphson Consensus has the demanding requirements that full matrices $\sigma_{i,z}$ need to be transmitted and inverted, which could be rather demanding if the feature space dimension n is large. Similarly to what has been proposed in [14], it is possible to modify the proposed algorithm to use Jacobi or Gradient descents which have reduced communication and computational requirements. More specifically, the only modification needed is to substitute line (37) with the following ones:

$$\begin{array}{lcll} h_i & \leftarrow & \text{diag}(\nabla^2 f_i(x_i)), & \text{Jacobi Descent Consensus} \\ h_i & \leftarrow & I_n, & \text{Gradient Descent Consensus,} \end{array}$$

where the operator $\text{diag}(A)$ returns a diagonal matrix whose diagonal elements coincide with the diagonal elements of A. As so, for the Jacobi Descent Consensus it is necessary to invert n scalars and to transmit only the n diagonal elements, while for the Gradient Descent consensus no transmission is required as long as the Hessian is concerned. Of course, the price to pay with these choices is a likely slower convergence rate. Note that in the Gradient Descent consensus the cost functions are required to be only $\mathcal{C}^2$.

Remark IV.2 The results presented in this work are concerned with the local stability and guarantee that $z_i(k) > cI_n$ for all times k. However, we have observed that, in order to increase the basin of attraction and the robustness of the algorithm, it is suitable to force $z_i(k) \geq cI_n$. A simple solution is to replace line (27) with

$$x_i \leftarrow (1-\varepsilon)x_i + \varepsilon [z_i]_c^{-1} y_i,$$

where the operator $[\cdot]_c$ is defined as

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

where $z \in \mathbb{R}^{n \times n}$ is a positive semidefinite matrix, and I_n is the identity matrix of dimension n. This does not impair the local stability analysis provided below since close to the equilibrium point we have $[z_i(k)]_c^{-1} = z_i(k)^{-1}$.

V. Theoretical analysis of the ra-NRC

In this Section, in order to carry out the convergence analysis of the algorithm itself, we introduce an asynchronous and lossy communication protocol that defines the evolution of the variables defined in Algorithm 1. In particular, we focus our analysis on an *asymmetric broadcast* communication protocol subject to packet losses, which represents a widely used communication protocol in wireless sensor networks applications. Specifically, let $t_0, t_1, t_2, \ldots$ be an ordered sequence of time instants, i.e., $t_0 < t_1 < t_2 < \ldots$. We assume that at each time instant one node, say i, is activated. Then, node i performs in order the operations in the *Estimate Update* block and in the *Data Transmission* block, broadcasting to all its out-neighbors in $\mathcal{G}$ the updated variables $\sigma_{i,y}, \sigma_{i,z}$. The transmitted packet might be received or not by $j \in \mathcal{N}_i^{\text{out}}$, depending whether

[2]Note that since the packet is received, $\rho_{i,y}^{(j)+} = \sigma_{j,y}$.

the communication link (i,j) is reliable or not at the time of transmission. If (i,j) is reliable, then node j performs, in order, the operations in the *Data Reception* and *Estimate Update* blocks. Since there is no risk of confusion, in the following we denote t_k only by the index k, referring to it as the k-th iteration of the ra-NRC algorithm.

An algorithmic description of the *asymmetric broadcast* communication protocol with packet losses (for the ra-NRC Algorithm 1) is provided in Algorithm 2. Moreover, in the following, without loss of generality, we assume that the node performing the transmission step during the k-th iteration is node i.

Algorithm 2 Asymmetric broadcast for ra-NRC algorithm

Node i is activated

1: $\texttt{flag}_{\texttt{update},i} \leftarrow 1$ (line 26) : *Estimate Update*

2: $\texttt{flag}_{\texttt{transmission},i} \leftarrow 1$ (line 8) : *Data transmission*

For $j \in \mathcal{N}_i^{\text{out}}$, if (i,j) is reliable

3: $\texttt{flag}_{\texttt{reception},j} \leftarrow 1$ (line 17) : *Data reception*

4: $\texttt{flag}_{\texttt{update},j} \leftarrow 1$ (line 26) : *Estimate Update*

Next, for the sake of clarity, we provide a sequential description of the ra-NRC algorithm when the communication protocol in Algorithm 2 is adopted. In order to keep the notation lighter, from now on, we restrict to the scalar case, i.e., $x_i \in \mathbb{R}$ for all i. Consistently we will denote with f_i', f_i'' and f_i''' respectively the first, second and third derivatives of the function f_i.

Observe that, once activated, node i updates x_i, g_i^{old}, h_i^{old}, g_i, h_i according to lines 27, 28, 29, 30, 31, i.e.,

$$\begin{aligned}
x_i(k+1) &= (1-\varepsilon)x_i(k) + \varepsilon z_i(k)^{-1} y_i(k)\\
g_i^{\text{old}}(k+1) &= g_i(k)\\
h_i^{\text{old}}(k+1) &= h_i(k)\\
g_i(k+1) &= f_i''(x_i(k+1))x_i(k+1) - f_i'(x_i(k+1))\\
h_i(k+1) &= f_i''(x_i(k+1)).
\end{aligned}$$

Based on $g_i(k+1)$ and $h_i(k+1)$, the variables y_i and z_i are updated performing in order the steps in lines 32, 10, and 33, 11, respectively, which result in

$$\begin{aligned}
y_i(k+1) &= \frac{1}{|\mathcal{N}_i^{\text{out}}|+1}\left(y_i(k) + g_i(k+1) - g_i^{\text{old}}(k+1)\right)\\
z_i(k+1) &= \frac{1}{|\mathcal{N}_i^{\text{out}}|+1}\left(z_i(k) + h_i(k+1) - h_i^{\text{old}}(k+1)\right),
\end{aligned}$$

and, in turn, from lines 12, 13, we have that

$$\begin{aligned}
\sigma_{i,y}(k+1) &= \sigma_{i,y}(k) + y_i(k+1)\\
\sigma_{i,z}(k+1) &= \sigma_{i,z}(k) + z_i(k+1).
\end{aligned}$$

The quantities $\sigma_{i,y}(k+1)$ and $\sigma_{i,z}(k+1)$ are transmitted by node i to its out-neighbors. If (i,j) is reliable then node j, based on the *Data Reception* packet, updates the local variables y_j, z_j, $\rho_{j,y}^{(i)}$, $\rho_{j,z}^{(i)}$ as[3]

$$\begin{aligned}
y_j' &= y_j(k) + \sigma_{i,y}(k+1) - \rho_{j,y}^{(i)}(k)\\
z_j' &= z_j(k) + \sigma_{i,z}(k+1) - \rho_{j,z}^{(i)}(k)\\
\rho_{j,y}^{(i)}(k+1) &= \sigma_{i,y}(k+1)\\
\rho_{j,z}^{(i)}(k+1) &= \sigma_{i,z}(k+1),
\end{aligned}$$

and, subsequently, based on the *Data Update* packet, updates the local variables x_j, g_j^{old}, h_j^{old}, g_j, h_j, y_j, z_j as

$$\begin{aligned}
x_j(k+1) &= (1-\varepsilon)x_j(k) + \varepsilon\frac{y_j(k)}{z_j(k)}\\
g_j^{\text{old}}(k+1) &= g_j(k)\\
h_j^{\text{old}}(k+1) &= h_j(k)\\
g_j(k+1) &= f_j''(x_j(k+1))x_j(k+1) - f_j'(x_j(k+1))\\
h_j(k+1) &= f_j''(x_j(k+1))\\
y_j(k+1) &= y_j' + g_j(k+1) - g_j^{\text{old}}(k+1)\\
z_j(k+1) &= z_j' + h_j(k+1) - h_j^{\text{old}}(k+1).
\end{aligned}$$

Next, we characterize a *mass conservation* property of Algorithm 2. Similarly to what done in Section III-C, for each $(i,j) \in \mathcal{E}$ we introduce the auxiliary variables $\nu_{j,y}^{(i)}(k)$, $\nu_{j,z}^{(i)}(k)$, defined as

$$\begin{aligned}
\nu_{j,y}^{(i)}(k) &= \sigma_{i,y}(k) - \rho_{j,y}^{(i)}(k)\\
\nu_{j,z}^{(i)}(k) &= \sigma_{i,z}(k) - \rho_{j,z}^{(i)}(k).
\end{aligned}$$

Recall that the role of the above variables is to keep track of the transmitted mass that has not been received due to packet losses. We have the following result:

Lemma V.1 *Consider Algorithm 2. Then, for all $k \in \mathbb{N}$, the following equalities hold true*

$$\begin{aligned}
\sum_{\ell=1}^{N}\left(y_\ell(k) + \sum_{j\in\mathcal{N}_\ell^{out}} \nu_{j,y}^{(\ell)}(k)\right) &= \sum_{\ell=1}^{N} g_\ell(k)\\
\sum_{\ell=1}^{N}\left(z_\ell(k) + \sum_{j\in\mathcal{N}_\ell^{out}} \nu_{j,z}^{(\ell)}(k)\right) &= \sum_{\ell=1}^{N} h_\ell(k).
\end{aligned}$$

The proof of the above Lemma can be found in [23]. To establish the convergence of the Asymmetric broadcast ra-NRC algorithm described in Algorithm 2 we introduce some sufficient conditions that guarantee local exponential stability under the assumptions posed in Section II. Informally, we assume that each node updates its local variables and communicates with its neighbors infinitely often, and that the number of consecutive packet losses is bounded. Formally, we make the following assumptions:

Assumption V.2 (Communications are persistent) *For any iteration $k \in \mathbb{N}$ there exists a positive integer number τ such that each node performs at least one broadcast transmission within the interval $[k, k+\tau]$, i.e., for each $i \in \{1,\dots,N\}$ there exists $h \in [k, k+\tau]$ such that node i is activated at iteration h.*

Assumption V.3 (Packet losses are bounded) *There exists a positive integer L such that the number of consecutive communication failures over every directed edge in the communication graph is smaller than L.*

From the above two assumptions, it follows that, given $i \in V$ and $j \in \mathcal{N}_i^{\text{out}}$, node j receives information from node i at least once within the interval $[k, k+L\tau]$.

Remark V.4 The previous two assumptions are standard in the context of consensus-based algorithms. They are necessary to guarantee deterministic exponential convergence as shown in [24] in the sense that if they do not hold, it is possible to construct communication and packet loss sequences that do not guarantee exponential convergence.

[3] As far as the variables y_j and z_j are concerned, to denote their updates in the *Data Reception packet* we introduce the auxiliary variables y_j', z_j', since the overall updates of the current values of y_j and z_j are performed in the subsequent *Data Update* packet.

The following theorem characterizes the convergence properties of the Asymmetric broadcast ra-NRC algorithm:

Theorem V.5 *Under Assumptions V.2, V.3 and the assumptions posed in Section II, there exist some positive scalars ε_c and δ such that, if the initial conditions $\boldsymbol{x}^o = [x_1^o, \ldots, x_N^o]^T \in \mathbb{R}^N$ satisfy $\|\boldsymbol{x}^o - x^* \mathbf{1}\| < \delta$ and if ε satisfies $0 < \varepsilon < \varepsilon_c$, then the local variables x_i in Algorithm 1 are exponentially stable with respect to the global minimizer x^*.*

The proof of the above Theorem can be found in [23].

Remark V.6 In this Section, we have provided a theoretical analysis of the ra-NRC algorithm, assuming that the asymmetric broadcast communication protocol has been adopted. However, it is worth stressing that similar results hold also for other communication protocols like *symmetric gossip*, *asymmetric gossip*, and *coordinated broadcast*[4].

Remark V.7 It is possible to show that the selected protocol allows us to rewrite the resulting ra-NRC as a dynamical system of the form

$$\begin{cases} \boldsymbol{x}(k+1) = \boldsymbol{x}(k) + \epsilon \phi(k, \boldsymbol{x}(k), \boldsymbol{\xi}(k)) \\ \boldsymbol{\xi}(k+1) = \varphi(k, \boldsymbol{x}(k), \boldsymbol{\xi}(k)), \end{cases}$$

under proper definitions of variables $\boldsymbol{x}$, $\boldsymbol{\xi}$ and maps ϕ and φ. The major challenges in proving the main results are related to proving that the ra-NRC algorithm satisfies a number of technical conditions required by standard theory of separation of time-scales. In particular, we are interested in proving exponential stability for a non-autonomous discrete time dynamical system whose closest counterpart in the continuous time is given by Theorem 11.4 in [26]. Besides some standard conditions on smoothness and uniformity of the dynamical flows involved, there are three major requirements that need to be satisfied: the first is that the fast dynamics converges exponentially to an equilibrium manifold, the second is that the slow dynamics restricted to this manifold is exponentially stable, and the third is that a number of *bounded interconnection conditions* which represent the perturbation of the slow dynamics into the fast dynamics and vice-versa are satisfied. We refer the interested reader to [23].

Remark V.8 Although the previous theorem guarantees only local exponential convergence, numerical simulations on real datasets seem to indicate that the basin of attraction is rather large and stability is mostly dictated by the choice of the parameter ε. However, for the special but relevant case when the cost functions $f_i(x)$ are quadratic, as in distributed least-squares problems, it has been proved that local stability implies global stability [21].

VI. Numerical Experiments

We consider a random geometric network with 10 nodes in $[0,1]^2$ and with communication radius $r = 0.5$ as in Figure 2.

As cost function, we consider a regression problem inspired by the UCI Housing dataset available at `http://archive.ics.uci.edu/ml/datasets/Housing`. In this task, an example $\chi_j \in \mathbb{R}^{n-1}$ is a vector representing some features of a house (e.g., per-capita crime rate by town, index of accessibility to radial highways, etc.), and $y_j \in \mathbb{R}$ denotes the corresponding median monetary value of the house. The objective is to obtain a predictor of house value based on these data. If the datasets come from different users that do not want to disclose their private information, then one may formulate the regression problem as a distributed problem defined on the local costs

$$f_i(x) := \sum_{j \in E_i} |y_j - \chi_j^T x' - x_0|_{1,\beta} + \gamma \left\|x'\right\|_2^2. \qquad (16)$$

where $|z|_{1,\beta} = \sqrt{z^2+\beta} - \sqrt{\beta}$, $z \in \mathbb{R}$ and $x = (x', x_0) \in \mathbb{R}^{n-1} \times \mathbb{R}$ is the vector of coefficients for the linear predictor $\widehat{y} = \chi^T x' + x_0$ and γ is a common regularization parameter known to all agents. The loss function $|\cdot|_{1,\beta}$ corresponds to a smooth $\mathcal{C}^\infty$ version of the Huber loss, a loss function that is usually employed to minimize the effects of outliers. In our case β dictates for which arguments the loss is pseudo-linear or pseudo-quadratic and has been manually chosen to minimize the effects of outliers. In our experiments we used 3 features, $\beta = \sqrt{25}$, $\gamma = 1$, and $|E| = 506$ total number of examples in the dataset randomly assigned to the $N = 10$ users communicating as in the graph of Figure 2. As a performance index, we consider the Mean Squared Error (MSE)

$$\frac{1}{N} \sum_{i=1}^{N} \|x_i(k) - x^*\|^2.$$

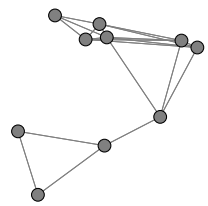

Fig. 2. The random geometric network considered in the simulations.

In the first experiment we assume lossless communications and compare the convergence speed of five schemes: the ra-NRC Algorithm 1 and its Jacobi and Gradient Descent versions defined in Remark IV.1, the Push-DIGing [17], and the Push-sum distributed dual averaging [22]. Notice that, as explained in the introduction, these two last schemes are based on broadcast-like communications but cannot cope with lossy communications. Figure 3 displays the evolution of the MSEs for the different algorithms subject to the same activation sequence (i.e., at the same time instant k the same node i awakens) and with the stepsize (referred to as α in [17], [22]) individually tuned by choosing (a posteriori) that particular one giving the fastest convergence speed among a grid of 20 logarithmically spaced potential values between 10^{-4} and 1.

Figure 4 instead inspects the effect of varying the probability of packets losses on the MSEs of single realizations for the ra-NRC scheme. In this figure the algorithm is tuned with a slower $\varepsilon = 0.01$ so to highlight the intuition that increasing the chances of packet losses leads to slower convergence properties. No comparison is possible with alternative algorithms available in the literature as none can cope with packet losses.

VII. Conclusions

In this work we addressed the problem of distributed unconstrained convex optimization in the context of lossy communication, a scenario that has not been considered before in the literature. More specifically, we considered a robustified version of the Newton-Raphson consensus algorithm proposed in [14], and we proved its (local) convergence properties under some general mild assumptions on the local costs and on the communication persistency. We also showed how the strategy can be applied to real world scenarios and datasets, and that it compares favorably against some alternative algorithms available in the literature even under the special case of lossless

[4]For a concise but effective description of the aforementioned protocols we refer the interested reader to [25].

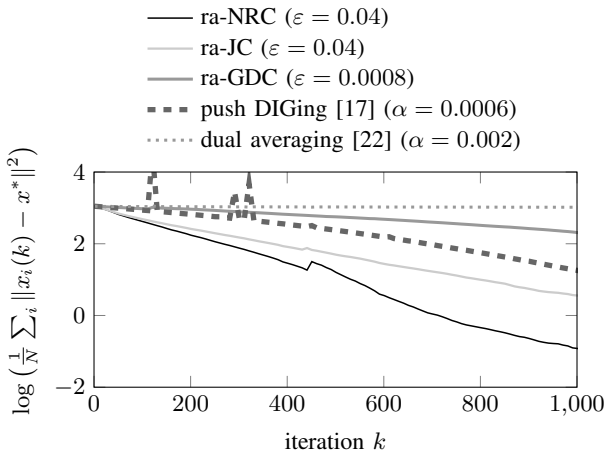

Fig. 3. Evolution of the MSE in time for different optimization algorithms. The legend indicates the hyperparameter for which the algorithm achieves fastest convergence speed in this specific experiment.

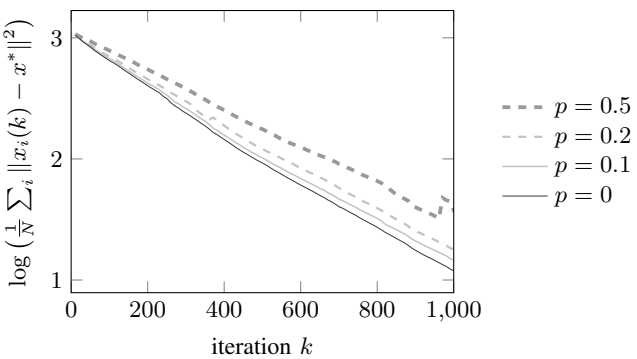

Fig. 4. Effect of varying the probability of packets losses on the speed of convergence of the ra-NRC scheme considered in Figure 3.

asynchronous scenario. Possible future research directions include adaptive strategies to tune the step-size ε on-line, the inclusion of local constraints, and the extension to partition-based approaches, where each agent is interested in computing only some components of the global minimizer vector.

References

[1] K. Kyoung-Dae and P. R. Kumar, "Cyber-physical systems: A perspective at the centennial," *Proceedings of IEEE*, vol. 100, pp. 1288–1308, 2012.

[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] P. Lin, W. Ren, and Y. Song, "Distributed multi-agent optimization subject to nonidentical constraints and communication delays," *Automatica*, vol. 65, pp. 120–131, 2016.

[4] A. Nedić and A. Olshevsky, "Distributed optimization over time-varying directed graphs," *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 601–615, 2015.

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

[6] 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.

[7] E. Wei and A. Ozdaglar, "On the O(1/k) convergence of asynchronous distributed Alternating Direction Method of Multipliers," in *Proceedings of IEEE Global Conference on Signal and Information Processing*, 2013.

[8] P. Bianchi, W. Hachem, and F. Iutzeler, "A stochastic coordinate descent primal-dual algorithm and applications to distributed optimization," *IEEE Transactions on Automatic Control*, vol. 61, no. 10, pp. 2947–2957, 2016.

[9] T.-H. Chang, M. Hong, W.-C. Liao, and X. Wang, "Asynchronous distributed ADMM for large-scale optimization- part I: Algorithm and convergence analysis," *IEEE Transactions on Signal Processing*, vol. 64, no. 12, pp. 3118–3130, 2016.

[10] I. Notarnicola, R. Carli, and G. Notarstefano, "Partitioned big-data optimization via asynchronous dual decomposition," *IEEE Transactions on Control Networks Systems*, 2018 (To appear).

[11] R. Zhang and J. Kwok, "Asynchronous distributed ADMM for consensus optimization," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 1701–1709.

[12] Z. Peng, Y. Xu, M. Yan, and W. Yin, "ARock: an algorithmic framework for asynchronous parallel coordinate updates," *SIAM Journal on Scientific Computing*, vol. 38, no. 5, pp. A2851–A2879, 2016.

[13] 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.

[14] D. Varagnolo, F. Zanella, A. Cenedese, P. Gianluigi, and L. Schenato, "Newton-Raphson Consensus for Distributed Convex Optimization," *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 994 – 1009, 2016.

[15] 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.

[16] P. Di Lorenzo and G. Scutari, "Next: In-network nonconvex optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 2, pp. 120–136, 2016.

[17] A. Nedich, A. Olshevsky, and W. Shi, "Achieving geometric convergence for distributed optimization over time-varying graphs," *arXiv preprint arXiv:1607.03218*, 2016.

[18] G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," *IEEE Transactions on Control of Network Systems*, 2017.

[19] 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.

[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, G. Notarstefano, L. Schenato, and D. Varagnolo, "Distributed quadratic programming under Asynchronous and Lossy Communications via Newton-Raphson Consensus," in *European Control Conference*, 2015.

[22] K. I. Tsianos, S. Lawlor, and M. G. Rabbat, "Push-sum distributed dual averaging for convex optimization," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*. IEEE, 2012, pp. 5453–5458.

[23] N. Bof, R. Carli, G. Notarstefano, L. Schenato, and D. Varagnolo, "Newton-raphson consensus under asynchronous and lossy communications for peer-to-peer networks," *arXiv preprint arXiv:1707.09178*, 2017.

[24] L. Moreau, "Stability of multiagent systems with time-dependent communication links," *IEEE Transactions on Automatic Control*, vol. 50, no. 2, pp. 169–182, 2005.

[25] N. Bof, R. Carli, and L.Schenato, "Average consensus with asynchronous updates and unreliable communication," in *IFAC World Congress (IFAC'17)*, vol. 50, no. 1, 2017, pp. 601–606.

[26] H. K. Khalil, *Nonlinear Systems*. Prentice-Hall, New Jersey, third edition, 2001.