Newton-Raphson Consensus under asynchronous and lossy communications for peer-to-peer 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 proved to be locally exponentially stable with respect to the optimal global solution. Finally, we complement the theoretical analysis with numerical simulations that are based on real datasets.

I. Introduction

Recently, we have been witnessing a surge of interest in distributed optimization, and in particular in distributed convex optimization. The reason is twofold: the first is due to the advent of Big-Data analytics, whose problems can be often cast as a large-scale convex optimization problems via Machine Learning tools [1]. As so, parallelization of computation is ought in order to obtain rapid solutions. The second 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 [2]. Even in this second scenario, several estimation and control problems such as localization, map-building, sensor calibration, power flow optimization can be cast as large-scale convex optimization problems. The main difference between these two scenarios is that in the former the bottleneck is mainly given by computation time and therefore the typical architecture adopted is server-client (i.e., memory is centralized at a master node or redundant via synchronized cloud architectures, and computation is parallelized among many nodes). Our work will mainly focus on the second scenario; however, since the boundary between the two is sometimes blurred, we will briefly overview the most relevant literature on distributed convex optimization in general.

To cope with real-world requirements, distributed convex algorithms need to be designed to work under asynchronous, directed, faulty and time-varying communications.

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 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.

A popular class of algorithms that are able to cope with asynchronous updates and lossy communication is the one of *distributed subgradient methods*. 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 [3], [4]. Recent works based on this approach have extended these results to directed and possibly time-varying communication in both discrete-time [5], [6] and continuous-time settings [7], [8], however the use of a diminishing step-size tacitly implies that the communication is synchronous (since the step-size is designed as a function of the global time that triggers the algorithm). Moreover, the underlying assumption for guaranteeing convergence is that the transmitter nodes should know which packets are transmitted successfully. This assumption corresponds to employing communication protocols with reliable packet transmission-acknowledge mechanisms, which might be difficult or expensive to implement over wireless media. The recent work [9] proposes an asynchronous algorithm, based on random projections, in which the step-size (both diminishing and constant) is uncoordinated among agents.

Another popular class of distributed optimization algorithms is the one of *dual decomposition schemes*. In this case the related literature is very large and we refer to [10] 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. This algorithm was originally proposed in mid '70s as a general convex optimization strategy, then exploited in the context of networked optimization [11], and recently popularized by the survey [12]. Substantial research has been dedicated in optimizing the free parameters of ADMM in order to obtain fastest convergence rates, but these are mainly restricted to synchronous implementations over undirected communication graphs [13], [14], [15], [16]. Some recent exceptions extend dual decomposition, [17], and ADMM, [18], [19], [20], to asynchronous scenarios with edge-based or node-based activation schemes.

A third class of optimization algorithms, usually referred to as *Newton-based methods*, consists of strategies that exploit second-order derivatives, i.e., the Hessians of the cost functions for computing descent directions. For example in [21], [22] the authors apply quasi-Newton distributed descent schemes to general time-varying directed graphs. Another approach, based on computing Newton-Raphson directions through average consensus algorithms, has been proposed in [23]. Even if initially proposed for synchronous implementations, this scheme has been later extended to cope with asynchronous symmetric gossip communication schemes [24].

Finally, a different approach, based on the exchange of active constraints, has been proposed in [25] for convex (and abstract) optimization problems and extended in [26] by means of cutting-plane methods. The proposed algorithms work under asynchronous, directed and unreliable communication.

Although there exists a large body of literature on distributed convex optimization schemes employing synchronous and asynchronous communications, no work has directly addressed situations where the communications are unreliable and lossy. Unfortunately, trying to make the aforementioned algorithms cope with packet losses using naïve modifications (e.g., using the most recently received message from the neighboring nodes, interpretable as using delayed information in the algorithms) may destroy some of the hypotheses that guarantee the convergence of the original algorithms (e.g., the doubly stochasticity or the invariance of some quantities such as the global averages). Distributed convex optimization in the presence of lossy communications is thus a non-trivial task, and recently some works have specifically addressed this problem in ADMM schemes [27], [28], [20]; however these strategies are restricted to networks with server-client communication topologies.

The main contribution of this work is to propose a Newton-based algorithm which is robust to both asynchronous updates and packet losses and which is suitable for general peer-to-peer networks. More specifically, we robustify the Newton-Raphson approach initially proposed in [23] by introducing a new consensus algorithm, which is an ad hoc merging of two known schemes for consensus: *i)* the *ratio* or *push-sum* consensus, useful to compute averages in networks with directed communication graphs (i.e., networks using broadcast protocols [29]); *ii)* the *robust consensus* algorithm, which allows for a robust computation of arithmetic averages over networks with lossy communication [30]. The new scheme is then able to deal with asynchronous and lossy communication protocols. Under mild conditions, i.e., persistency of (asynchronous) node updates, uniformly bounded consecutive communication link failures, and connectivity of the communication graph, we then show that the optimization algorithm is locally exponentially stable with respect to the global solution as long as the step-size of the updates is smaller than a certain critical value and the cost functions are sufficiently smooth. The proof is based on time-scale separation and Lyapunov theory, and extends the results in [31], 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. It is worth mentioning that the algorithm we propose not only handles asynchronous updates, as some recent references, but also is robust to packet losses.

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 VI. Finally, Section VII collects some numerical experiments corroborating the theoretical results, while Section VIII draws some concluding remarks and future research directions.

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 and is $\mathcal{C}^2$ and 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].

We also define the following operator that will be useful in the description of the main algorithm in the next section:

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

where $z \in \mathbb{R}^{n \times n}$ is a positive semidefinite matrix, and I_n is the identity matrix of dimension n.

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

[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.

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 $\mathcal{V} = \{1, \dots, N\}$ 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.

Remark II.3 In some distributed systems, as 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 unsynchronized 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 global minimizer that asymptotically converges to the optimal solution x^*.

F2) *Peer-to-peer (leaderless)*: each node has limited computational and memory resources and it is allowed to communicate directly only with its neighbors; moreover there is no leader/master node, and the communication graph is arbitrary (but strongly connected).

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; in other words, no multi-hop information exchange is allowed.

F4) *Asynchronous*: events as the update of the local variables, and the transmission/reception of messages do not need to be synchronized within the node itself nor with its neighbors, i.e., any communication and update protocol can be used (e.g., time-triggered, event-triggered or hybrid). Therefore, 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 mechanisms and allow for packet losses (due to ambient noise, collisions, or other effects) without impairing the convergence properties of the algorithm.

To the best of authors' knowledge, none of the previously cited works possesses all the previous features.

III. Building blocks

The algorithm we propose consists of three different building blocks: *i) Newton-Raphson Consensus*, proposed in [23] to solve problem (1), *ii) the push-sum algorithm*, initially proposed

in [29] as an asynchronous average consensus protocol, and *iii) the robust ratio consensus algorithm*, initially proposed in [30] as a robust average consensus protocol. While possessing the first three features mentioned above (i.e., F1, F2, and F3), 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. 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 algorithm, we offer a brief description of the three aforementioned algorithms.

A. Newton-Raphson Consensus

Newton-Raphson Consensus [23] is based on the observation that the standard Newton-Raphson update in the standard centralized scenario with a single agent can be written as

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

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. 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:

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

The dynamics of the N local systems is 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 variable 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 [23] 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 [24] 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

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

where $\eta_1, \dots, \eta_N$ 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 aim 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 [29] 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[2]

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

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 [29]). Let at any time k only one node i activate, update its variables, and

[2]These well-known results can also be readily derived from standard theories on Markov Chains [32].

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 [29], if the activation of the nodes is randomized and i.i.d. then the local variables converge to

$$\begin{array}{l} y_i \to \eta_i(k) \sum_i y_i(0) = \eta_i(k) \sum_i g_i(x_i) = \eta_i(k) \overline{g}(x_1, \ldots, x_N) \\ z_i \to \eta_i(k) \sum_i z_i(0) = \eta_i(k) \sum_i h_i(x_i) = \eta_i(k) \overline{h}(x_1, \ldots, x_N) \end{array} \tag{7}$$

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

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

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 [30]. More specifically, in this way the synchronous update (3) transforms into

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

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

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

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 a counter $\sigma_{i,y}(k)$ to keep track of the total y-mass sent to its neighbors up to iteration k, and, for each neighbor $j \in \mathcal{N}_i^{\text{in}}$, a counter $\rho_{i,y}^{(j)}(k)$ 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, such virtual mass is zero, while each time a packet loss occurs, this variable accumulates the additional mass that node j wants to transfer to node i, and therefore it 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 $\boldsymbol{\nu}_y$ be the vectors collecting, respectively, the variables y_i, $i \in V$, and $\nu_{i,y}^{(j)}$, $i \in V$ and $j \in \mathcal{N}_i^{\text{out}}$, and, 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, from weak ergodicity theory, that local variables y_i, z_i, converge asymptotically as in (7) [30]. As it will be clear in the next sections, matrix $M(k)$ will be a building block for the design and analysis of our 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.

We propose a "meta distributed algorithm" which can result in different distributed algorithms depending on the (possibly asynchronous and packet-lossy) communication protocol 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 corresponds to a one-time execution at startup, the blocks can be executed asynchronously, 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 node is executing Estimate Update and a new packet is incoming, this packet is either dropped or placed in a buffer till 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

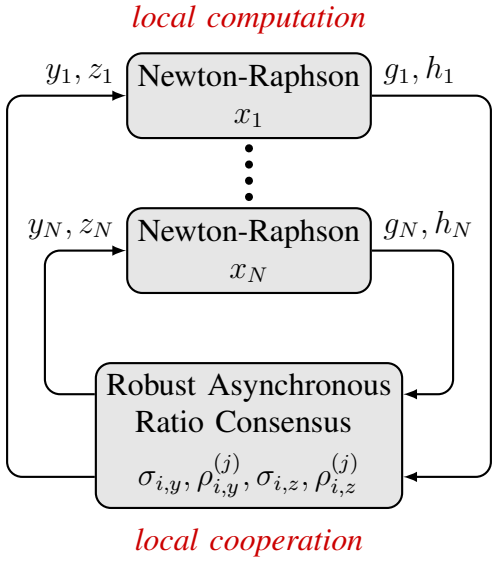

Fig. 1. Graphical representation of the robust asynchronous Newton-Raphson Consensus (ra-NRC).

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.

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 set is the initial estimate x^o for the global optimization, while all other variables are set to zero or to identity matrices of the proper dimension.

The blocks *Data Transmission* (lines 8-15) and *Data Reception* (lines 17-24) implement a new Robust Asynchronous Ratio Consensus (see bottom block in Figure 1), which merges the benefits of the push-sum algorithm, with its asynchronous nature, and the robust ratio consensus with 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 Eqn. (5). The update for $\sigma_{i,y}$ in the algorithm (line 12) is identical to Eqn. (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, 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 Eqn. (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 Eqn. (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

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

Require: x^o, ε, c

Initialization (atomic)

$x_i \leftarrow x^o$

$y_i \leftarrow 0, \;\; g_i \leftarrow 0, \;\; g_i^{old} \leftarrow 0$

$z_i \leftarrow I_n, \;\; h_i \leftarrow I_n, \;\; h_i^{old} \leftarrow I_n$

$\sigma_{i,y} \leftarrow 0, \;\; \sigma_{i,z} \leftarrow 0$

$\rho_{i,y}^{(j)} \leftarrow 0, \;\; \rho_{i,z}^{(j)} \leftarrow 0, \quad \forall j \in \mathcal{N}_i^{\text{in}}$

$\texttt{flag}_{\texttt{reception},i} \leftarrow 0, \;\; \texttt{flag}_{\texttt{update},i} \leftarrow 0$

$\texttt{flag}_{\texttt{transmission},i} \leftarrow 1$

Data Transmission (atomic)

if $\texttt{flag}_{\texttt{transmission},i} = 1$ **then**

  $\texttt{transmitter\_node\_ID} \leftarrow i$

  $y_i \leftarrow \frac{1}{|\mathcal{N}_i^{\text{out}}|+1} y_i$

  $z_i \leftarrow \frac{1}{|\mathcal{N}_i^{\text{out}}|+1} z_i$

  $\sigma_{i,y} \leftarrow \sigma_{i,y} + y_i$

  $\sigma_{i,z} \leftarrow \sigma_{i,z} + z_i$

  Broadcast: $\texttt{transmitter\_node\_ID}, \sigma_{i,y}, \sigma_{i,z}$

  $\texttt{flag}_{\texttt{transmission},i} \leftarrow 0$

end if

Data Reception (atomic)

if $\texttt{flag}_{\texttt{reception},i} = 1$ **then**

  $j \leftarrow \texttt{transmitter\_node\_ID}, \quad (j \in \mathcal{N}_i^{\text{in}})$

  $y_i \leftarrow y_i + \sigma_{j,y} - \rho_{i,y}^{(j)}$

  $z_i \leftarrow z_i + \sigma_{j,z} - \rho_{i,z}^{(j)}$

  $\rho_{i,y}^{(j)} \leftarrow \sigma_{j,y}, \quad \forall j \in \mathcal{N}_i$

  $\rho_{i,z}^{(j)} \leftarrow \sigma_{j,z}, \quad \forall j \in \mathcal{N}_i$

  $\texttt{flag}_{\texttt{reception},i} \leftarrow 0$

  $\texttt{flag}_{\texttt{update},i} \leftarrow 1$ (optional)

end if

Estimate Update (atomic)

if $\texttt{flag}_{\texttt{update},i} = 1$ **then**

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

  $g_i^{\text{old}} \leftarrow g_i$

  $h_i^{\text{old}} \leftarrow h_i$

  $h_i \leftarrow \nabla^2 f_i(x_i)$

  $g_i \leftarrow h_i x_i - \nabla f_i(x_i)$

  $y_i \leftarrow y_i + g_i - g_i^{\text{old}}$

  $z_i \leftarrow z_i + h_i - h_i^{\text{old}}$

  $\texttt{flag}_{\texttt{update},i} \leftarrow 0$

  $\texttt{flag}_{\texttt{transmission},i} \leftarrow 1$ (optional)

end if

only the variables $\sigma_{i,y}, \sigma_{i,z}$ and its ID to its neighbors. After transmission, the node returns 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 updated similarly to Eqn. (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 Eqn. (10)[3]. The local variable $\rho_{i,y}^{(j)}$ is then updated (line 21) similarly to Eqn. (9) in the robust ratio consensus.

The last block *Estimate Update* is responsible for implementing a local version of Newton-Raphson. 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 Eqn. (2), the proposed algorithm uses these variables to implement an approximated Newton-Raphson (line 27), where the operator $[\cdot]_c$ is used to avoid divide-by-zero numerical issues during the transient, and 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 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} e h_i^{old} in lines (28-29) and g_i e 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 Eqn. (11), the following invariant are preserved:

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

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

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. The intuition behind the convergence of the algorithm, is that if the local estimates x_i change slower than the rate at which the asynchronous robust ratio consensus converges, which can be achieved by choosing a sufficiently small stepsize ε, then we would expect that

$$y_i(k) \rightarrow \eta_i(k) \sum_i g_i(k), \tag{14}$$

$$z_i(k) \rightarrow \eta_i(k) \sum_i h_i(k) \tag{15}$$

A formal proof 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 $\sigma_{i,z}$ needs 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 [23], it is possible to modify the proposed algorithm to use Jacobi or Gradient descents which have

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

reduced communication and computational requirements. More specifically, the only modification needed is to substitute line (37) with the following ones

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

where the operator $\operatorname{diag}(A)$ 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 consensus only one scalar needs to be transmitted and inverted. Of course, the price to pay with these choices is a likely slower convergence rate.

V. Dynamical system interpretation of ra-NRC

In this Section, we introduce an asynchronous and lossy communication protocol that defines the evolution of the flags in Algorithm 1, in order to carry out the convergence analysis of the algorithm itself, and we will also show that this choice of communication protocol is not restricting. The protocol selected allows us to rewrite the resulting ra-NRC as a dynamical system of the form:

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

where proper definitions of variables $\boldsymbol{x}$, $\boldsymbol{\xi}$ and maps ϕ and φ can be found in Corollary V.2.

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 (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* block, and in the *Estimate Update* block. 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. Here, with a slight abuse of notation, we denote within the parentheses after the flag variables the owner of the corresponding variable. 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*

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, e.g., f_i' and f_i'' respectively the first and second derivatives of the function f_i.

Next, for the sake of analysis, we provide a sequential description of the ra-NRC algorithm when the communication protocol in Algorithm 2 is adopted. 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.,

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

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 $\sigma_{i,y}(k+1)$, $\sigma_{i,z}(k+1)$ 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[4]

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

[4]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.

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

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

Next, we provide a suitable vector-form description of the Asymmetric broadcast ra-NRC algorithm.

To do so, similarly to the *robust ratio consensus algorithm* revisited in Section III-C, we first need to build an *augmented* network that contains all the nodes in V and also some additional virtual nodes; precisely, a virtual node for each link in $\mathcal{E}$. Let us denote the augmented network by $\mathcal{G}_a = (V_a, \mathcal{E}_a)$, where $V_a = V \cup \mathcal{E}$ and

$$\mathcal{E}_a = \mathcal{E} \cup \{((i,j),j) \mid (i,j) \in \mathcal{E}\} \cup \{(i,(i,j)) \mid (i,j) \in \mathcal{E}\}.$$

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, which has not been received due to packet losses. Accordingly, let $\boldsymbol{\nu}_y$ and $\boldsymbol{\nu}_z$ be the vectors that collect, respectively, all the variables $\nu_{j,y}^{(i)}$ and $\nu_{j,z}^{(i)}$, $i \in V$ and $j \in \mathcal{N}_i^{\text{out}}$. Assuming that $|\mathcal{E}| = N_{\mathcal{E}}$, then $\boldsymbol{\nu}_y$, $\boldsymbol{\nu}_z \in \mathbb{R}^{N_{\mathcal{E}}}$. Now let

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \qquad \boldsymbol{z} = \begin{bmatrix} z_1 \\ \vdots \\ z_N \end{bmatrix},$$

and, based on these vectors, let us build the augmented vectors $\boldsymbol{y}_a$, $\boldsymbol{z}_a \in \mathbb{R}^{N+N_{\mathcal{E}}}$ as

$$\boldsymbol{y}_a = \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{\nu}_y \end{bmatrix}, \qquad \boldsymbol{z}_a = \begin{bmatrix} \boldsymbol{z} \\ \boldsymbol{\nu}_z \end{bmatrix}.$$

Moreover let

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

Since we are considering a lossy scenario, it might happen that the packet transmitted by node i is either received or not received by node $j \in \mathcal{N}_i^{\text{out}}$. For this reason, it is convenient to introduce the sets

$$\tilde{\mathcal{N}}_i(k) = \{j \in \mathcal{N}_i^{\text{out}} \text{ such that } (i,j) \text{ is reliable at time } k\},$$

and, its complement on $\mathcal{N}_i^{\text{out}}$,

$$\bar{\mathcal{N}}_i(k) = \mathcal{N}_i^{\text{out}} \setminus \tilde{\mathcal{N}}_i(k).$$

To state Proposition V.1, where we provide a vector form description of Algorithm 2, it is convenient to resort to the following notational convention. When referring to an N-dimensional vector, we assume its components to be indexed according to the nodes in V, while when referring to an $N_{\mathcal{E}}$-dimensional vector, we assume its components to be indexed according to the edges in $\mathcal{E}$. In particular, $e_i \in \mathbb{R}^N$ and $e_{(i,j)} \in \mathbb{R}^{N\varepsilon}$ denote the vectors with all the components equal to zero, except, respectively, the one related to node i and the one related to edge (i,j), which are equal to one; that is e_i, $i \in V$ and $(e_{i,j})$, $(i,j) \in \mathcal{E}$, are the vectors of the canonical basis of, respectively, $\mathbb{R}^N$ and $R^{N\varepsilon}$.

Proposition V.1 *The ra-NRC algorithm with asymmetric broadcast (Algorithm 1 and Algorithm 2), can be written in vector form as*[5]

$$\begin{aligned}
\boldsymbol{x}(k+1) &= \boldsymbol{x}(k) + \varepsilon\, S(k)\ (\boldsymbol{p}(k) - \boldsymbol{x}(k)) \qquad (16)\\
\boldsymbol{g}^{old}(k+1) &= S(k)\boldsymbol{g}(k) + (I - S(k))\boldsymbol{g}^{old}(k)\\
\boldsymbol{g}(k+1) &= \boldsymbol{f}''(\boldsymbol{x}(k+1))\boldsymbol{x}(k+1) - \boldsymbol{f}'(\boldsymbol{x}(k+1))\\
\boldsymbol{h}^{old}(k+1) &= S(k)\boldsymbol{h}(k) + (I - S(k))\boldsymbol{h}^{old}(k)\\
\boldsymbol{h}(k+1) &= \boldsymbol{f}''(\boldsymbol{x}(k+1))\\
\boldsymbol{y}_a(k+1) &= M(k)\boldsymbol{y}_a(k) +\\
&\qquad T(k)\ \left(\boldsymbol{g}(k+1) - \boldsymbol{g}^{old}(k+1)\right)\\
\boldsymbol{z}_a(k+1) &= M(k)\boldsymbol{z}_a(k) +\\
&\qquad T(k)\ \left(\boldsymbol{h}(k+1) - \boldsymbol{h}^{old}(k+1)\right)\\
\boldsymbol{p}(k+1) &= \frac{\boldsymbol{y}(k+1)}{[\boldsymbol{z}(k+1)]_c}
\end{aligned}$$

where

$$S(k) = e_i e_i^T + \sum_{j \in \tilde{\mathcal{N}}_i(k)} e_j e_j^T \quad \textit{and} \quad T(k) = \begin{bmatrix} T_V(k) \\ T_{\mathcal{E}}(k) \end{bmatrix},$$

[5]We observe that the matrices S, S_a and M depend on which node is activated, and on which edges between this node and its out-neighbors are reliable. In order to keep the notation lighter, we do not make this dependency explicit (for instance using some superscript or subscript); instead, we limit ourselves to emphasize only the time-varying nature of these matrices, i.e., just writing $S(k)$, $S_a(k)$ and $M(k)$.

with

$$T_V(k) = \frac{1}{|\mathcal{N}_i^{out}|+1}\left(e_i e_i^T + \sum_{j\in\tilde{\mathcal{N}}_i(k)} e_j e_i^T\right) + \sum_{j\in\tilde{\mathcal{N}}_i(k)} e_j e_j^T$$

$$T_{\mathcal{E}}(k) = \frac{1}{|\mathcal{N}_i^{out}|+1}\sum_{j\in\tilde{\mathcal{N}}_i} e_{(i,j)} e_i^T$$

and where $M(k)$ is a column stochastic matrix such that

$$M(k) = \begin{bmatrix} M_{VV}(k) & M_{V\mathcal{E}}(k) \\ M_{\mathcal{E}V}(k) & M_{\mathcal{E}\mathcal{E}}(k) \end{bmatrix}$$

with

$$M_{VV}(k) = \frac{1}{|\mathcal{N}_i^{out}|+1}\left(e_i e_i^T + \sum_{j\in\tilde{\mathcal{N}}_i} e_j e_i^T\right) + \sum_{h\neq i} e_h e_h^T$$

$$M_{V\mathcal{E}}(k) = \sum_{j\in\tilde{\mathcal{N}}_i} e_j e_{(i,j)}^T$$

$$M_{\mathcal{E}V}(k) = \frac{1}{|\mathcal{N}_i^{out}|+1}\sum_{j\in\tilde{\mathcal{N}}_i} e_{(i,j)} e_i^T \quad (= T_{\mathcal{E}}(k))$$

$$M_{\mathcal{E}\mathcal{E}}(k) = \sum_{j\in\tilde{\mathcal{N}}_i} e_{(i,j)} e_{(i,j)}^T + \sum_{(r,s):\, r\neq i} e_{(r,s)} e_{(r,s)}^T.$$

Proof: We start by observing that, only nodes in $\tilde{N}_i(k)\cup\{i\}$ update the variables x, g^{old}, g, h^{old}, h. Moreover, observe that the matrix $S(k)$ can be seen as a *selection* matrix which selects the nodes in $\tilde{N}_i(k)\cup\{i\}$. This explains the vector form of the first five equations in (16).

Now, to each edge (i,j), $j\in\mathcal{N}_i^{\text{out}}$, we associate the indicator function variable $X_{i,j}(k)$ as follows:

$$X_{i,j}(k) = \begin{cases} 1, & \text{if } (i,j) \text{ reliable at time } k \\ 0, & \text{if } (i,j) \text{ not reliable at time } k. \end{cases}$$

In the following, for the sake of simplicity, we consider only the update of $\boldsymbol{y}_a$ (the update of $\boldsymbol{z}_a$ is similar). Recall that

$$y_i(k+1) = \frac{1}{|\mathcal{N}_i^{\text{out}}|+1}\left(y_i(k) + g_i(k+1) - g_i^{\text{old}}(k+1)\right). \tag{17}$$

Observe that, for $j\in\mathcal{N}_i^{\text{out}}$, by using the indicator function defined above, we can write that

$$\rho_{j,y}^{(i)}(k+1) = X_{i,j}(k)\sigma_{i,y}(k+1) + (1-X_{i,j}(k))\,\rho_{j,y}^{(i)}(k).$$

Since

$$\nu_{j,y}^{(i)}(k) = \sigma_{i,y}(k) - \rho_{j,y}^{(i)}(k),$$

and

$$\sigma_{i,y}(k+1) = \sigma_{i,y}(k) + y_i(k+1),$$

it follows that

$$\nu_{j,y}^{(i)}(k+1) = (1 - X_{i,j}(k)) \Big(\nu_{j,y}^{(i)}(k) + \frac{1}{|\mathcal{N}_i^{\text{out}}| + 1} \left[y_i(k) + g_i(k+1) - g_i^{\text{old}}(k+1) \right] \Big) \tag{18}$$

and that

$$y_j(k+1) = y_j(k) + \\ + X_{i,j}(k) \left[y_i(k+1) + g_j(k+1) - g_j^{\text{old}}(k+1) + \nu_{j,y}^{(i)}(k) \right]$$

and, in turn, that

$$y_j(k+1) = y_j(k) + X_{i,j}(k) \frac{1}{|\mathcal{N}_i^{\text{out}}| + 1} y_i(k) + \\ + X_{i,j}(k) \Big[g_j(k+1) - g_j^{\text{old}}(k+1) + \nu_{j,y}^{(i)}(k) + \\ \frac{1}{|\mathcal{N}_i^{\text{out}}| + 1} \left[g_i(k+1) - g_i^{\text{old}}(k+1) \right] \Big]. \tag{19}$$

From (17) and (19) we can write that

$$\begin{aligned} \boldsymbol{y}(k+1) = \\ & \left[\frac{1}{|\mathcal{N}_i^{\text{out}}| + 1} \left(e_i + \sum_{j \in \tilde{\mathcal{N}}_i(k)} e_j \right) e_i^T + \sum_{h \neq i} e_h e_h^T \right] \boldsymbol{y}(k) + \\ & + \sum_{j \in \tilde{\mathcal{N}}_i} e_j e_{(i,j)}^T \boldsymbol{\nu} + \left[\frac{1}{|\mathcal{N}_i^{\text{out}}| + 1} \left(e_i + \sum_{j \in \tilde{\mathcal{N}}_i(k)} e_j \right) e_i^T + \right. \\ & \left. + \sum_{j \in \tilde{\mathcal{N}}_i(k)} e_j e_j^T \right] \left(\boldsymbol{g}(k+1) - \boldsymbol{g}^{\text{old}}(k+1) \right) \\ & = M_{VV}(k) \boldsymbol{y} + M_{V\mathcal{E}}(k) \boldsymbol{\nu}_y(k) + \\ & + T_V(k) \left(\boldsymbol{g}(k+1) - \boldsymbol{g}^{\text{old}}(k+1) \right). \end{aligned}$$

From (18), we have that

$$\begin{aligned} \boldsymbol{\nu}_y(k+1) = & \left[\sum_{j \in \bar{\mathcal{N}}_i} e_{(i,j)} e_{(i,j)}^T + \sum_{(r,s) \,:\, r \neq i} e_{(r,s)} e_{(r,s)}^T \right] \boldsymbol{\nu}_y + \\ & + \left[\frac{1}{|\mathcal{N}_i^{\text{out}}| + 1} \sum_{j \in \bar{\mathcal{N}}_i} e_{(i,j)} e_i^T \right] \left(\boldsymbol{y} + \boldsymbol{g}(k+1) - \boldsymbol{g}^{\text{old}}(k+1) \right) \\ & = M_{\mathcal{E}V}(k) \boldsymbol{y}(k) + M_{\mathcal{E}\mathcal{E}}(k) \boldsymbol{\nu}_y(k) + \\ & + T_{\mathcal{E}}(k) \left(\boldsymbol{g}(k+1) - \boldsymbol{g}^{\text{old}}(k+1) \right). \end{aligned}$$

The above computations explain the vector-form illustrated in equations (16).

The fact that $M(k)$ is a column-stochastic matrix can be shown by simply verifying that the sum of the elements of each column is equal to one. ■

Observe that variables $\boldsymbol{y}_a, \boldsymbol{z}_a$ are trajectories of a linear, time-varying algorithm with column-stochastic state-matrix, driven by the differences $\boldsymbol{g} - \boldsymbol{g}^{\text{old}}$, $\boldsymbol{h} - \boldsymbol{h}^{\text{old}}$.

From the previous proposition, the next fact follows directly.

Corollary V.2 *Let* $\boldsymbol{\xi} = \left[\boldsymbol{g}^T, \boldsymbol{g}^{old^T}, \boldsymbol{h}^T, \boldsymbol{h}^{old^T}, \boldsymbol{y}_a^T, \boldsymbol{z}_a^T, \boldsymbol{p}^T\right]^T$*, then, system in* (16) *can be written as:*

$$\begin{cases} \boldsymbol{x}(k+1) = \boldsymbol{x}(k) + \epsilon\, \phi(k, \boldsymbol{x}(k), \boldsymbol{\xi}(k)) \\ \boldsymbol{\xi}(k+1) = \varphi(k, \boldsymbol{x}(k)), \boldsymbol{\xi}(k)), \end{cases} \tag{20}$$

where $\boldsymbol{x} \in \mathbb{R}^N$, $\boldsymbol{\xi} \in \mathbb{R}^{7N+2|\mathcal{E}|}$, $\phi : \mathbb{N} \times \mathbb{R}^N \times \mathbb{R}^{7N+2|\mathcal{E}|} \to \mathbb{R}^N$, $\varphi : \mathbb{N} \times \mathbb{R}^N \times \mathbb{R}^{7N+2|\mathcal{E}|} \to \mathbb{R}^{7N+2|\mathcal{E}|}$, $\epsilon > 0$ *and where equations in* (16) *properly define the maps* ϕ *and* φ*.*

Finally, we characterize a *mass conservation* property of system in (16), which will be useful in next section.

Lemma V.3 *Consider system in* (16)*. Then, for all* $k \in \mathbb{N}$*, the following equalities hold true*

$$\sum_{\ell=1}^{N} \left(y_\ell(k) + \sum_{j \in \mathcal{N}_\ell^{out}} \nu_{j,y}^{(\ell)}(k) \right) = \sum_{\ell=1}^{N} g_\ell(k)$$

$$\sum_{\ell=1}^{N} \left(z_\ell(k) + \sum_{j \in \mathcal{N}_\ell^{out}} \nu_{j,z}^{(\ell)}(k) \right) = \sum_{\ell=1}^{N} h_\ell(k).$$

Proof: We provide the proof of only the first equality; the second one can be proved analogously. We proceed by induction. The property is trivially true for $k = 0$. Indeed according to the Initialization block we have that $y_\ell(0) = g_\ell(0) = g_\ell^{\text{old}}(0) = \nu_{j,y}^{(\ell)}(0) = 0$ for all ℓ and $j \in \mathcal{N}_\ell^{\text{out}}$; the fact that $g_\ell(0) = g_\ell^{\text{old}}(0) = 0$ implies that also $g_\ell^{\text{old}}(1) = 0$ for all ℓ. Now, we assume the property to be true for k and we show that it holds also for $k+1$. Without loss of generality, assume that node i is activated at iteration k. Then we have

$$\begin{aligned}
&\sum_{\ell=1}^{N} \left(y_\ell(k+1) + \sum_{j \in \mathcal{N}_\ell^{\text{out}}} \nu_{j,y}^{(\ell)}(k+1) \right) = \mathbf{1}^T \boldsymbol{y}_a(k+1) \\
&= \mathbf{1}^T M(k) \boldsymbol{y}_a(k) + \mathbf{1}^T T(k) \left(\boldsymbol{g}(k+1) - \boldsymbol{g}^{\text{old}}(k+1) \right) \\
&= \sum_{\ell=1}^{N} \left(y_\ell(k) + \sum_{j \in \mathcal{N}_\ell^{\text{out}}} \nu_{j,y}^{(\ell)}(k) \right) + g_i(k+1) - g_i^{\text{old}}(k+1) \\
&\qquad + \sum_{j \in \mathcal{N}_i^{\text{out}}} X_{i,j}(k) \left(g_j(k+1) - g_j^{\text{old}}(k+1) \right) \\
&= \sum_{\ell=1}^{N} g_\ell(k) + \sum_{j \in \tilde{\mathcal{N}}_i(k) \cup \{i\}} \left(g_j(k+1) - g_j^{\text{old}}(k+1) \right),
\end{aligned}$$

where, in the above equalities, we have used the properties

$$\mathbf{1}^T M(k) = \mathbf{1}^T, \qquad \mathbf{1}^T T(k) = e_i^T + \sum_{j \in \tilde{\mathcal{N}}_i} e_j^T$$

and the inductive hypothesis

$$\sum_{\ell=1}^{N} \left(y_\ell(k) + \sum_{j \in \mathcal{N}_\ell^{\text{out}}} \nu_{j,y}^{(\ell)}(k) \right) = \sum_{\ell=1}^{N} g_\ell(k).$$

By simple algebraic manipulations, we can write

$$\begin{aligned}
&\sum_{\ell=1}^{N} g_\ell(k) + \sum_{j \in \tilde{\mathcal{N}}_i(k) \cup \{i\}} \left(g_j(k+1) - g_j^{\text{old}}(k+1) \right) \\
&= \sum_{j \in \tilde{\mathcal{N}}_i(k) \cup \{i\}} g_j(k) + \sum_{j \notin \tilde{\mathcal{N}}_i(k) \cup \{i\}} g_j(k) + \\
&\qquad + \sum_{j \in \tilde{\mathcal{N}}_i(k) \cup \{i\}} \left(g_j(k+1) - g_j^{\text{old}}(k+1) \right) \\
&= \sum_{j \in \tilde{\mathcal{N}}_i(k) \cup \{i\}} g_j(k+1) + \sum_{j \notin \tilde{\mathcal{N}}_i(k) \cup \{i\}} g_j(k) + \\
&\qquad + \sum_{j \in \tilde{\mathcal{N}}_i(k) \cup \{i\}} \left(g_j(k) - g_j^{\text{old}}(k+1) \right).
\end{aligned}$$

Now, observe that, if $\ell \notin \tilde{\mathcal{N}}_i(k) \cup \{i\}$ then $g_\ell(k+1) = g_\ell(k)$, and, if $\ell \in \tilde{\mathcal{N}}_i(k) \cup \{i\}$ then $g_\ell^{\text{old}}(k+1) = g_\ell(k)$. Then, from the previous expression, it follows

$$\begin{aligned}
&\sum_{\ell=1}^{N} g_\ell(k) + \sum_{j \in \tilde{\mathcal{N}}_i(k) \cup \{i\}} \left(g_j(k+1) - g_j^{\text{old}}(k+1) \right) \\
&= \sum_{\ell=1}^{N} g_\ell(k+1).
\end{aligned}$$

This concludes the proof. ■

Remark V.4 In this Section, we have provided a dynamical system description of ra-NRC algorithm, assuming the asymmetric broadcast communication protocol has been adopted. However, it is worth stressing that similar computations hold also for other communication protocols like *symmetric gossip*, *asymmetric gossip*, *coordinated broadcast*[6]. When adopting one of the above aforementioned communication protocols it turns out that ra-NRC algorithm can again be described as in (16), with the only difference related to the matrix $M(k)$ which is still a column stochastic matrix but with a slight different structure, and to the selection matrix $S(k)$. This justifies the fact that the convergence results we provide in the next Section, which are specifically tailored to the scenario considered in this Section, can be technically extended to also other types of communication protocols.

[6]For a concise but effective description of the aforementioned protocols we refer the interested reader to [33].

VI. THEORETICAL ANALYSIS OF THE RA-NRC

We now provide a theoretical analysis of the Asymmetric broadcast ra-NRC algorithm, described in Algorithm 2. In particular, we provide 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 VI.1 (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 VI.2 (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 $[k, k+L\tau]$.

We now want to characterize the convergence properties of the Asymmetric broadcast ra-NRC algorithm. To do so, we start by introducing two Lemmas which will be later used.

Let $\boldsymbol{x} = [x_1, \dots, x_N]^T$ and $\boldsymbol{x}^0 = [x_1^0, \dots, x_N^0]^T$. In the first lemma we show that if the variable $\boldsymbol{x}$ is kept constant, then the components of the vector $\boldsymbol{p}$ achieve consensus to the ratio $\overline{h}(x_1, \dots, x_N)/\overline{g}(x_1, \dots, x_N)$. Viceversa, in the second lemma, we show that if the components of $\boldsymbol{p}$ have reached consensus, then the vector $\boldsymbol{x}$ exponentially converges to the global minimizer.

Formally, to state the first result, for a given $\bar{k}$, we consider the following dynamics, for $k \geq \bar{k}$,

$$\boldsymbol{\xi}_{\bar{k}}(k+1) = \varphi\left(k, \boldsymbol{x}(\bar{k}), \boldsymbol{\xi}_{\bar{k}}(k)\right), \tag{21}$$

initialized by $\boldsymbol{\xi}_{\bar{k}}(\bar{k}) = \boldsymbol{\xi}(\bar{k})$. Observe that, $\boldsymbol{\xi}_{\bar{k}}$ describes the evolution of the variable $\boldsymbol{\xi}$, starting at iteration $\bar{k}$, assuming that the variable $\boldsymbol{x}$ is kept constant for $k \geq \bar{k}$, that is, $\boldsymbol{x}(k) = \boldsymbol{x}(\bar{k})$ for all $k \geq \bar{k}$. In particular, in this scenario, we are interested in the behavior of the variable $\boldsymbol{p}$, that is, of the last block of components of $\boldsymbol{\xi}_{\bar{k}}$, that, in this case, similarly to $\boldsymbol{\xi}_{\bar{k}}$, we denote as $\boldsymbol{p}_{\bar{k}}$. We have the following result.

Lemma VI.3 *For a given $\bar{k}$, consider, for $k \geq \bar{k}$, the dynamics in* (21)*. Then, under Assumptions VI.1, VI.2, we have that the point*

$$\frac{\sum_\ell \boldsymbol{g}_\ell(\boldsymbol{x}_\ell(\bar{k}))}{\sum_\ell \boldsymbol{h}_\ell(\boldsymbol{x}_\ell(\bar{k}))}\mathbf{1}$$

is exponentially stable for the variable $\boldsymbol{p}_{\bar{k}}$, that is, defined

$$\tilde{\boldsymbol{p}}_{\bar{k}}(k) := \boldsymbol{p}_{\bar{k}}(k) - \frac{\sum_\ell \boldsymbol{g}_\ell(\boldsymbol{x}_\ell(\bar{k}))}{\sum_\ell \boldsymbol{h}_\ell(\boldsymbol{x}_\ell(\bar{k}))}\mathbf{1},$$

there exists $C_{\bar{k}} > 0$ and $0 \leq \rho_{\bar{k}} < 1$ such that

$$\|\tilde{\boldsymbol{p}}_{\bar{k}}(k)\| \leq C_{\bar{k}}\, \rho_{\bar{k}}^{k-\bar{k}}\, \|\tilde{\boldsymbol{p}}_{\bar{k}}(\bar{k})\|. \tag{22}$$

Proof: In the following we denote by $\boldsymbol{y}_{a;\bar{k}}$, $\boldsymbol{z}_{a;\bar{k}}$ the block components of $\boldsymbol{\xi}_{\bar{k}}$ corresponding to $\boldsymbol{y}_a$, $\boldsymbol{z}_a$. To study the evolution of $\boldsymbol{p}_{\bar{k}}(k)$, we analyze the behavior of the variables $\boldsymbol{y}_{a;\bar{k}}(k)$, $\boldsymbol{z}_{a;\bar{k}}(k)$, separately. Consider $\boldsymbol{y}_{a;\bar{k}}(k)$. Observe that, since $\boldsymbol{x}(k) = \boldsymbol{x}(\bar{k})$, $k \geq \bar{k}$, and according to

Assumptions VI.1 and VI.2, we have that there exists $\bar{k}' > \bar{k}$ such that $\boldsymbol{g}(k) = \boldsymbol{g}^{\text{old}}(k)$ for all $k \geq \bar{k}'$ and, hence,

$$\boldsymbol{y}_{a,\bar{k}}(k+1) = M(k)\boldsymbol{y}_{a,\bar{k}}(k),$$

for $k \geq \bar{k}'$. A similar reasoning holds for $\boldsymbol{z}_{a,\bar{k}}(k)$. It follows that the variables $\boldsymbol{y}_{a,\bar{k}}(k)$, $\boldsymbol{z}_{a,\bar{k}}(k)$ and, in turn, the variables $\boldsymbol{y}_{\bar{k}}(k)$, $\boldsymbol{z}_{\bar{k}}(k)$ run the iterations of a ratio-consensus algorithm for $k \geq \bar{k}'$, as described in [33].

From Lemma V.3, we have that, for $k \geq \bar{k}$ and, in particular, for $k \geq \bar{k}'$

$$\mathbf{1}^T \boldsymbol{y}_{a,\bar{k}}(k) = \sum_{\ell=1}^{N} g_\ell\left(x_\ell(\bar{k})\right)$$

$$\mathbf{1}^T \boldsymbol{z}_{a,\bar{k}}(k) = \sum_{\ell=1}^{N} h_\ell\left(x_\ell(\bar{k})\right).$$

From Theorem 3 in [33], it follows that $\frac{\boldsymbol{y}_{\bar{k}}(k)}{[\boldsymbol{z}_{\bar{k}}(k)]_c}$ converges exponentially to $\frac{\sum_\ell \boldsymbol{g}_\ell(\boldsymbol{x}_\ell(\bar{k}))}{\sum_\ell \boldsymbol{h}_\ell(\boldsymbol{x}_\ell(\bar{k}))}\mathbf{1}$. ■

Now, let us assume that, for each k, the variable $\boldsymbol{p}(k)$ has reached consensus and consider the following dynamics for the variable $\boldsymbol{x}$,

$$\begin{aligned}\boldsymbol{x}(k+1) &= \boldsymbol{x}(k) + \varepsilon\, S(k)\left(\frac{\sum_\ell g_\ell(x_\ell(k))}{\sum_\ell h_\ell(x_\ell(k))}\mathbf{1} - \boldsymbol{x}(k)\right) \\ &= \boldsymbol{x}(k) + \varepsilon\, \tilde{\phi}(k;\boldsymbol{x}(k))\end{aligned} \tag{23}$$

where

$$\tilde{\phi}(k;\boldsymbol{x}(k)) = S(k)\left(\frac{\sum_\ell g_\ell(x_\ell(k))}{\sum_\ell h_\ell(x_\ell(k))}\mathbf{1} - \boldsymbol{x}(k)\right).$$

Let

$$\boldsymbol{x}^* = x^*\mathbf{1}, \tag{24}$$

where we recall that x^* is the minimizer of the optimization problem in (1). By standard algebraic manipulations, one can see that $\boldsymbol{x}^*$ is an equilibrium of (23). The following result states that the linearized version of (23) around $\boldsymbol{x}^*$ is an exponentially stable system.

Lemma VI.4 *Consider system in* (23) *and let* $\boldsymbol{x}^*$ *be as in* (24)*. Let*

$$A(k) = I + \epsilon\frac{\partial\tilde{\phi}}{\partial\boldsymbol{x}}(k;\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}^*},$$

and, accordingly, consider the auxiliary system

$$\tilde{\boldsymbol{x}}(k+1) = A(k)\tilde{\boldsymbol{x}}(k). \tag{25}$$

Then, under Assumptions VI.1, VI.2, $\tilde{\boldsymbol{x}} = 0$ *is exponentially stable equilibrium point for* (25)*.*

Proof: Let

$$\alpha(\boldsymbol{x}(k)) = \frac{\sum_\ell g_\ell(x_\ell(k))}{\sum_\ell h_\ell(x_\ell(k))}.$$

Computing the partial derivative of α with the respect to x_i we get

$$\left[\frac{\partial \alpha}{\partial x_i}\right]|_{\boldsymbol{x}=\boldsymbol{x}^*} = \frac{g_i'(x^*)\ \sum_{\ell=1}^N h_\ell\,(x^*) - h_i'\,(x^*)\ \sum_{\ell=1}^N g_\ell\,(x^*)}{\left(\sum_{\ell=1}^N h_\ell\,(x^*)\right)^2}$$

with

$$\begin{aligned}
& g_i'(x^*)\ \sum_{\ell=1}^N h_\ell\,(x^*)\ -\ h_i'\,(x^*)\ \sum_{\ell=1}^N g_\ell\,(x^*) \\
&= (\ f_i'''(x^*)\,x^*\ + f_i''(x^*)\ -\ f_i''(x^*)\ )\ \sum_{\ell=1}^N f_\ell''\,(x^*) \\
&\qquad - f_i'''(x^*)\ \sum_{\ell=1}^N (f_\ell''\,(x^*)\ x^*\ - f_\ell'(x^*)) \\
&= f_i'''(x^*)\,x^*\ \sum_{\ell=1}^N f_\ell''\,(x^*)\ -\ f_i'''(x^*)\,x^*\ \sum_{\ell=1}^N f_\ell''\,(x^*) \\
&\qquad + f_i'''(x^*)\ \sum_{\ell=1}^N f_\ell'(x^*) \\
&= 0,
\end{aligned}$$

where , in the last equality, we have used the fact that $\sum_{\ell=1}^N f_\ell'(x^*) = 0$. From the previous calculations, it turns out that

$$A(k) = I - \varepsilon S(k).$$

By Assumption VI.1, we have that the matrix

$$\bar{A}_{k,\tau} = \prod_{s=k}^{k+\tau} A(k),$$

is a diagonal matrix such that $0 < \left[\bar{A}_{k,\tau}\right]_{ii} < 1-\varepsilon$, for all i. Then, system in (23) satisfies the stated property. ■

Intuitively, one would conclude that when the parameter ε is small the results of the two lemma can be combined to simultaneously obtain asymptotic consensus and convergence to the global minimizer. This is formally shown in the next theorem which characterizes the convergence properties of the Asymmetric broadcast ra-NRC algorithm.

Theorem VI.5 *Under Assumptions VI.1, VI.2 and the assumptions posed in Section II, there exist some positive scalars ε_c and δ such that, if the initial conditions $\boldsymbol{x}^0 \in \mathbb{R}^N$ 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^*.*

Proof: The proof of the result is based on showing that the system in (20) satisfies the assumptions of Proposition A.2. To do so, we start by defining, for $k \geq \bar{k}$,

$$\boldsymbol{\xi}^*_{\boldsymbol{x}(\bar{k}),\boldsymbol{\xi}(\bar{k})}(k) = \tilde{I}\,\boldsymbol{\xi}_{\bar{k}}(k)\ +\ \tilde{u}, \tag{26}$$

where $\boldsymbol{\xi}_{\bar{k}}(k)$ is defined as in (21) and where

$$\tilde{I} = \begin{bmatrix} I_{(6N+2|\mathcal{E}|)\times(6N+2|\mathcal{E}|)} & 0_{(6N+2|\mathcal{E}|)\times N} \\ 0_{N\times(6N+2|\mathcal{E}|)} & 0_{N\times N} \end{bmatrix}$$

and

$$\tilde{u} = \begin{bmatrix} 0_{(6N+2|\mathcal{E}|)\times 1} \\ \boldsymbol{p}^*_{\boldsymbol{x}(\bar{k})} \end{bmatrix},$$

with

$$\boldsymbol{p}^*_{\boldsymbol{x}(\bar{k})} = \frac{\sum_\ell \boldsymbol{g}_\ell(x_\ell(\bar{k}))}{\sum_\ell \boldsymbol{h}_\ell(x_\ell(\bar{k}))}\mathbf{1}.$$

Observe that the first six blocks components of $\boldsymbol{\xi}^*_{\boldsymbol{x}(\bar{k}),\boldsymbol{\xi}(\bar{k})}(k)$ coincide with the first six blocks components of $\boldsymbol{\xi}_{\bar{k}}(k)$, while the last block component is constant for all $k \geq \bar{k}$. Moreover, for $k \geq \bar{k}$, let

$$\tilde{\boldsymbol{\xi}}_{\bar{k}}(k) := \boldsymbol{\xi}_{\bar{k}}(k) - \boldsymbol{\xi}^*_{\boldsymbol{x}(\bar{k}),\boldsymbol{\xi}(\bar{k})}(k).$$

Based on the previous observation, we have that the first six blocks components of $\tilde{\boldsymbol{\xi}}_{\bar{k}}(k)$ are equal to zero, while the last block component is equal to

$$\boldsymbol{p}_{\bar{k}}(k) - \boldsymbol{p}^*_{\boldsymbol{x}(\bar{k})}.$$

From Lemma VI.3, it easily follows that there exists $C_{\bar{k}} > 0$ and $0 \leq \rho_{\bar{k}} < 1$ such that

$$\|\tilde{\boldsymbol{\xi}}_{\bar{k}}(k)\| \leq C_{\bar{k}}\, \rho_{\bar{k}}^{k-\bar{k}}\, \|\tilde{\boldsymbol{\xi}}_{\bar{k}}(\bar{k})\|. \tag{27}$$

This shows that system in (20) satisfies property in (35), in Appendix A

Consider now the system

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{x}(k) + \varepsilon\, \phi\left(k, \boldsymbol{x}(k), \boldsymbol{\xi}^*_{\boldsymbol{x}(k),\boldsymbol{\xi}(k)}(k)\right) && (28)\\ &= \boldsymbol{x}(k) + \varepsilon\, S(k)\, \left(\frac{\sum_\ell g_\ell(x_\ell(k))}{\sum_\ell h_\ell(x_\ell(k))}\mathbf{1} - \boldsymbol{x}(k)\right) \\ &= \boldsymbol{x}(k) + \varepsilon\, \tilde{\phi}(k; \boldsymbol{x}(k)) && (29) \end{aligned}$$

In Lemma VI.4, it is established that the previous system satisfies Assumption A.1, in Appendix A. Hence, Proposition A.2, in Appendix A can be applied to system in (20), yielding the result of the statement. ■

Remark VI.6 Algorithm 1 assumes the initial conditions of the local variable x_i to be all identical to x^o. Although not being a very stringent requirement, this assumption can be relaxed, that is, slightly modified versions of Theorem VI.5 would hold even in the case $x_i = x_i^o$ as soon as all the initial conditions are sufficiently close to the global minimizer x^*, i.e., as soon as $|x_i^o - x^*| < \delta$ for all $i = 1, \ldots, N$.

Remark VI.7 The initial conditions on the local variables $y_i = g_i^{\text{old}} = g_i = f_i''(x^o)x^o - f_i'(x^o)$ and $z_i = h_i^{\text{old}} = h_i = f_i''(x^o)$ are instead more critical for the convergence of the local variables x_i to the true minimizer x^*. As shown in [34], small perturbations of these initial conditions can lead to convergence to a point $\overline{x} \neq x^*$ (notice that these perturbations do not affect the stability of the algorithm, so that possible small numerical errors due to the computation and data quantization do not disrupt the convergence properties of the algorithm). Moreover, the map

from the amplitude of these perturbations and the distance $\|\overline{x} - x^*\|$ is continuous, so that if these perturbations are small then $\overline{x} \approx x^*$.

Remark VI.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, local stability implies global stability [31].

Remark VI.9 The major challenges in proving the main results are related to proving that the ra-NRC algorithm satisfy a number of technical conditions required by standard theory of separation of time-scales. Different conditions and theorems are available for continuous time dynamical systems (we refer the interested reader to Chapter 11 in [35]). 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 [35]. 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. As for the first requirement, we were able to guaranteed it by extending (see [33]) the work by [36], which only provided convergence in probability. As for the second one, we are able to prove local exponential stability of the slow dynamics which is not trivial since the dynamics is non-autonomous. As for the last requirement on the *bounded interconnection conditions*, very much depends on cost functions and in the discrete-time domain it is difficult to provide *global* guarantees. However, under some mild smoothness conditions, we were able to show that the conditions on *bounded interconnection conditions* are locally satisfied, and, in turn, to prove local exponential stability.

VII. 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 functions, we consider the distributed training of a Binomial-Deviance based spam-nonspam classifier [37, Chap. 10.5] where the training set is a database of E emails with j the email index, $y_j = -1, 1$ indicating if email j is spam or not, $\chi_j \in \mathbb{R}^{n-1}$ summarizing the $n-1$ features of the j-th email (in our case the frequency of words "make", "address", and "all"). Letting $x = (x', x_0) \in \mathbb{R}^{n-1} \times \mathbb{R}$ represent a generic classification hyperplane, distributedly training a Binomial-Deviance based classifier corresponds to solve the distributed optimization problem with local costs defined by

$$f_i(x) := \sum_{j \in E_i} \log\left(1 + \exp\left(-y_j\left(\chi_j^T x' + x_0\right)\right)\right) + \gamma \|x'\|_2^2 \tag{30}$$

where E_i is the set of emails available to agent i, $E = \cup_{i=1}^N E_i$, and γ is a global regularization parameter. In our experiments we consider $|E| = 5000$ emails from the spam-nonspam UCI

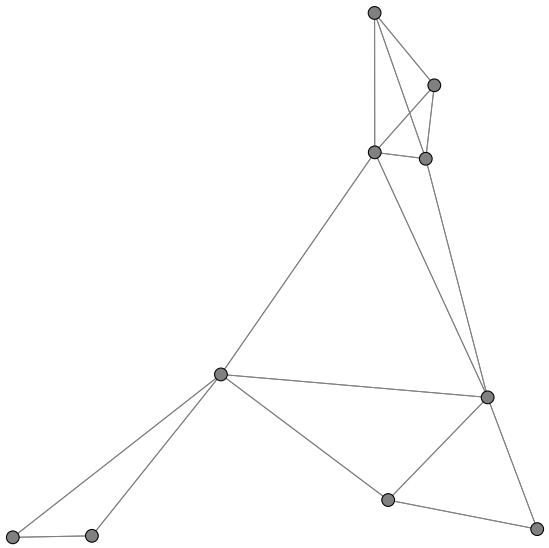

Fig. 2. The random geometric network considered in the simulations.

repository[7], randomly assigned to the 10 nodes users communicating as in Figure 2. As a performance index, we consider the Mean Squared Error (MSE)

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

as a function of the iteration index x.

Figure 3 then plots the evolution of the MSE of a typical realization of the optimization process as a function of the iteration index k, for a fixed packet loss probability equal to 0.1, and for different values of ε. The figure confirms the intuition that increasing ε may lead to faster convergence properties, but only up to a certain value; too aggressive ε's may indeed hinder the convergence property of the algorithm.

Figure 4 instead inspects the effect of varying the probability of packets losses on the MSEs of single realizations for a fixed ε. This figure confirms the intuition that, independently of ε, increasing the chances of packet losses leads to initially slower convergence properties and eventually divergent behaviors.

VIII. Conclusions

Implementations of distributed optimization methods in real-world scenarios require strategies that are both able to cope with real-world problematics (like unreliable, asynchronous and directed communications), and converge sufficiently fast so to produce usable results in meaningful times. Here we worked towards this direction, and improved an already existing

[7] `http://archive.ics.uci.edu/ml/datasets/Spambase`

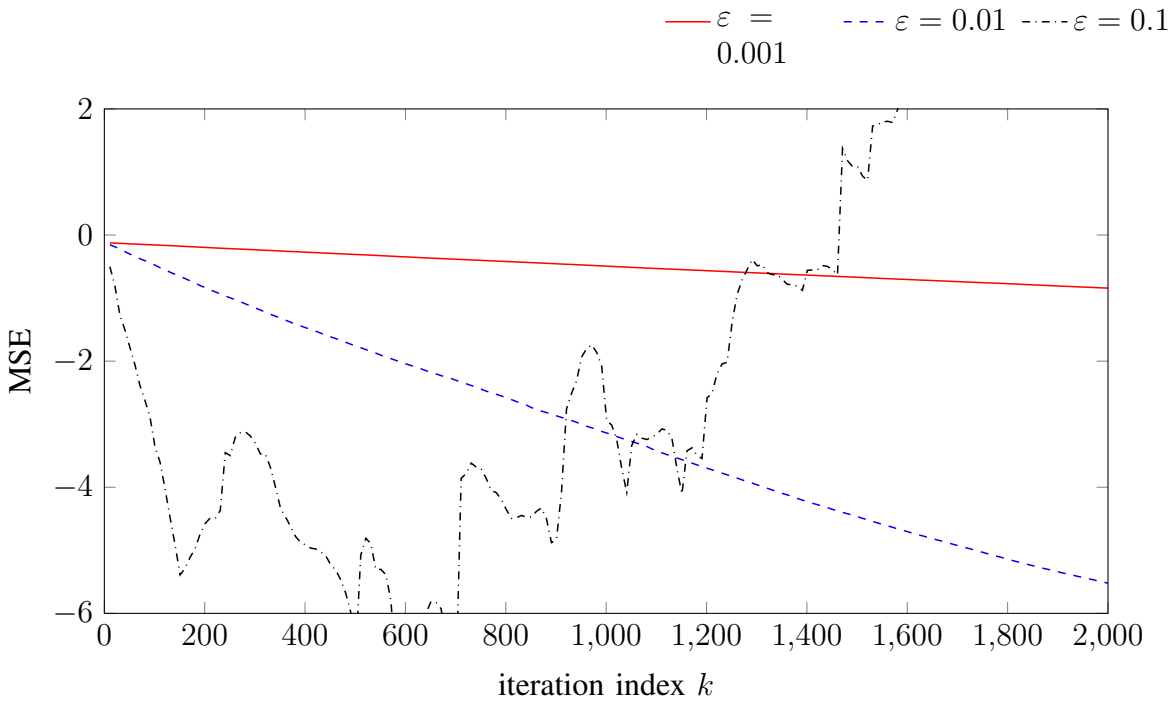

Fig. 3. Evolution of the MSE in time for a typical realization of the optimization process for a fixed packet loss probability of 0.1 and different values of ε.

distributed optimization strategy, previously shown to have fast convergence properties, so to make it tolerate the previously mentioned real-world problematics.

More specifically, we considered a robustified version of the Newton-Raphson consensus algorithm originally proposed in [34] and proved its convergence properties under some general mild assumptions on the local costs. From technical perspectives we shown that under suitable assumptions on the initial conditions, on the step-size parameter, on the connectivity of the communication graph and on the boundedness of the number of consecutive packet losses, the considered optimization strategy is locally exponentially stable around the global optimum as soon as the local costs are $\mathcal{C}^2$ and strongly convex with second derivative bounded from below.

We also shown how the strategy can be applied to real world scenarios and datasets, and be used to successfully compute optima in a distributed way.

We then notice that the results offered in this manuscript do not deplete the set of open questions and plausible extensions of the Newton Raphson consensus strategy. We indeed devise that the algorithm is potentially usable as a building block for distributed interior point methods, but that some lacking features prevent this development. Indeed it is still not clear how to tune the parameter ε online so that the convergence speed is dynamically adjusted (and maximized), how to account for equality constraints of the form $Ax = b$, and how to update the local variables x_i using partition-based approaches so that each agent keeps and updates only a subset of the components of x.

APPENDIX A
GENERAL RESULTS ON DISCRETE-TIME NONLINEAR SYSTEMS

The proofs and results of this appendix can be found in the technical report [38].

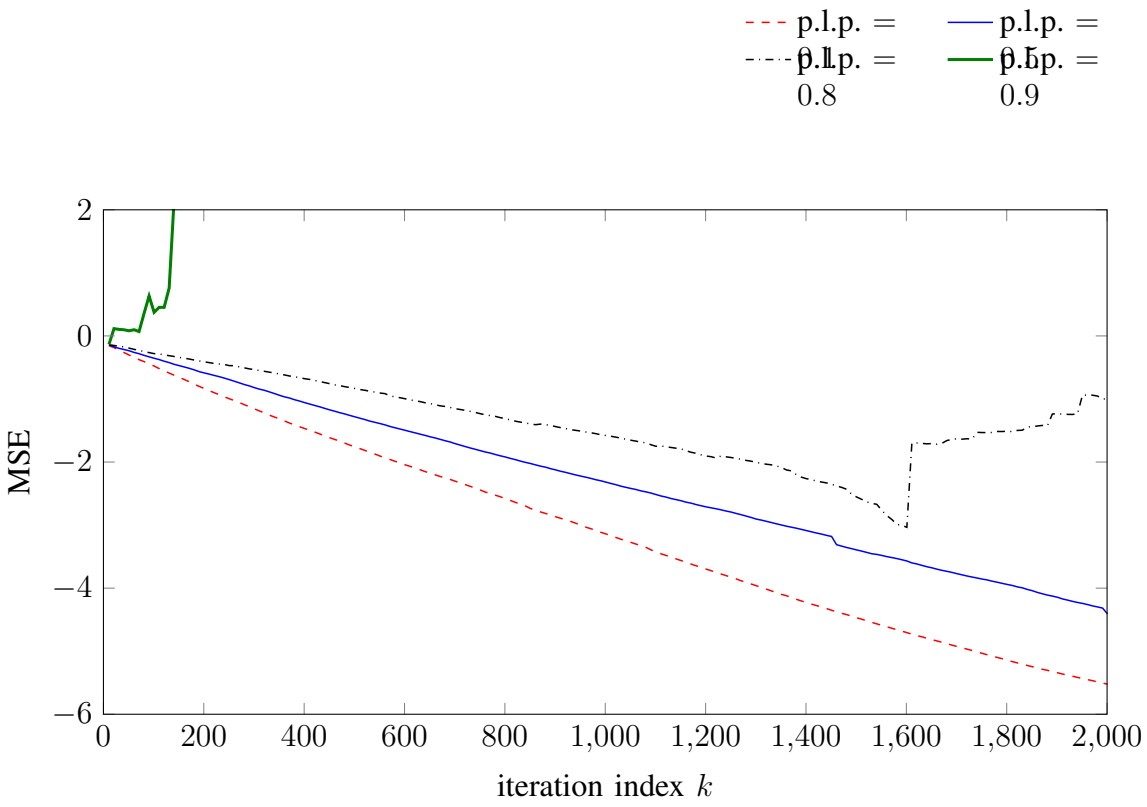

Fig. 4. Effect of varying the probability of packets losses on the MSE of single realizations of the optimization process for $\varepsilon = 0.01$.

Consider the system

$$\begin{cases} \boldsymbol{x}(k+1) = \boldsymbol{x}(k) + \varepsilon\phi(k, \boldsymbol{x}(k), \boldsymbol{\xi}(k)) \\ \boldsymbol{\xi}(k+1) = \varphi(k, \boldsymbol{x}(k)), \boldsymbol{\xi}(k)) \end{cases} \tag{31}$$

where $\boldsymbol{x} \in \mathbb{R}^{n_1}$, $\boldsymbol{\xi} \in \mathbb{R}^{n_2}$, $\phi : \mathbb{N} \times \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}^{n_1}$, $\varphi : \mathbb{N} \times \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \to \mathbb{R}^{n_2}$, $\varepsilon > 0$ and with given initial conditions $\boldsymbol{x}(0)$, $\boldsymbol{\xi}(0)$.
For a given $\bar{k} \in \mathbb{N}$, consider the system, for $k \geq \bar{k}$,

$$\boldsymbol{\xi}_{\bar{k}}(k+1) = \varphi\left(k,\, \boldsymbol{x}(\bar{k}),\, \tilde{\boldsymbol{\xi}}_{\bar{k}}(k)\right), \tag{32}$$

initialized by $\boldsymbol{\xi}_{\bar{k}}(\bar{k}) = \boldsymbol{\xi}(\bar{k})$, where $\boldsymbol{\xi}(\bar{k})$ is obtained ruling system (31) up to $\bar{k}$.
Given $\bar{k}$, assume that, for $k \geq \bar{k}$, there exists a sequence

$$k \;\to\; \boldsymbol{\xi}^*_{\boldsymbol{x}(\bar{k}),\boldsymbol{\xi}(\bar{k})}(k)\,, \tag{33}$$

in general dependent on $\boldsymbol{x}(\bar{k})$ and $\boldsymbol{\xi}(\bar{k})$, such that the evolution

$$\tilde{\boldsymbol{\xi}}_{\bar{k}}(k) := \boldsymbol{\xi}_{\bar{k}}(k) - \boldsymbol{\xi}^*_{\boldsymbol{x}(\bar{k}),\boldsymbol{\xi}(\bar{k})}(k) \tag{34}$$

satisfies the property

$$\|\tilde{\boldsymbol{\xi}}_{\bar{k}}(k)\| \leq C_{\bar{k}}\, \rho_{\bar{k}}^{k-\bar{k}}\, \|\tilde{\boldsymbol{\xi}}_{\bar{k}}(\bar{k})\|, \tag{35}$$

for suitable $C_{\bar{k}} > 0$ and $0 \leq \rho_{\bar{k}} < 1$, that is $\tilde{\boldsymbol{\xi}}'_{\bar{k}} = 0$ is an exponentially stable point for the evolution in (34). Basically, the property in (35) establishes that there exists a trajectory $\boldsymbol{\xi}^*$ to

which the trajectory of the variable $\boldsymbol{\xi}$, generated keeping the variable $\boldsymbol{x}$ constant, converges asymptotically.

Next, let us assume that, for each k, the variable $\boldsymbol{\xi}$ has already reached the asymptotic convergence to the corresponding trajectory $\boldsymbol{\xi}^*$. More precisely, observe that there exists a family of sequences of the type (33), where each sequence starts from a different index k. From this family we can build the following new sequence

$$k \;\rightarrow\; \boldsymbol{\xi}^*_{\boldsymbol{x}(k),\boldsymbol{\xi}(k)}\,(k)\,, \tag{36}$$

where, to the index k, we have associated the first element of the sequence which starts at k. Based on (36), we consider the system

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) + \varepsilon\,\phi\left(k, \boldsymbol{x}(k), \boldsymbol{\xi}^*_{\boldsymbol{x}(k),\boldsymbol{\xi}(k)}(k)\right). \tag{37}$$

Assume that $\boldsymbol{\xi}^*_{\boldsymbol{x}(k),\boldsymbol{\xi}(k)}(k)$ is such that there exists a suitable map $\tilde{\phi} : \mathbb{N} \times \mathbb{R}^{n_1} \rightarrow \mathbb{R}^{n_1}$ such that (37) can be, equivalently, rewritten as

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) + \varepsilon\,\tilde{\phi}\,(k, \boldsymbol{x}(k))\,, \tag{38}$$

that is, $\tilde{\phi}\,(k, \boldsymbol{x}(k)) = \phi\left(k, \boldsymbol{x}(k), \boldsymbol{\xi}^*_{\boldsymbol{x}(k),\boldsymbol{\xi}(k)}(k)\right)$. We make the following assumption.

Assumption A.1 *Let $\boldsymbol{x}^*$ be an equilibrium point for* (38)*. We assume that, there exists $r > 0$ such that $\tilde{\phi}$ is continuously differentiable on $D = \{\boldsymbol{x} \in \mathbb{R}^{n_1} \mid \|\boldsymbol{x} - \boldsymbol{x}^*\| < r\}$ and the Jacobian matrix $[\partial\tilde{\phi}/\partial\boldsymbol{x}]$ is bounded and Lipschitz on D, uniformly in k. In addition, defining*

$$A(k) = I + \varepsilon\frac{\partial\tilde{\phi}}{\partial\boldsymbol{x}}(k;\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}^*},$$

and considering the auxiliary system

$$\tilde{\boldsymbol{x}}(k+1) = A(k)\tilde{\boldsymbol{x}}(k), \tag{39}$$

we assume that $\tilde{\boldsymbol{x}} = 0$ is exponentially stable equilibrium point for (39)*.*

The following Proposition characterizes the convergence properties of system (31).

Proposition A.2 *Consider system in* (31)*. For any $\bar{k}$, assume that there exists a sequence as in* (33) *such that property* (35) *is satisfied. Consider system in* (37)*. Let $\boldsymbol{x}^*$ be an equilibrium point for* (37)*. Assume Assumption* (A.1) *holds true. Then, there exist $r > 0$ and $\varepsilon^* > 0$, such that, for all $\varepsilon \in (0, \varepsilon^*]$ and for all $x(0) \in B^n_r = \{\boldsymbol{x} \in \mathbb{R}^n : \|\boldsymbol{x} - \boldsymbol{x}^*\| < r\}$, the trajectory $\boldsymbol{x}(t)$ generated by* (31)*, converges exponentially to $\boldsymbol{x}^*$, i.e., there exist $C > 0$ and $0 < \lambda < 1$ such that*

$$\|\boldsymbol{x}(k) - \boldsymbol{x}^*\| \leq C\lambda^k\|\boldsymbol{x}(0) - \boldsymbol{x}^*\|.$$

REFERENCES

[1] K. Slavakis, G. B. Giannakis, and G. Mateos, "Modeling and optimization for big data analytics," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 18–31, 2014.

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

[3] A. Nedic and A. Ozdaglar, *Convex Optimization in Signal Processing and Communications*. Cambridge University Press, 2010, ch. Cooperative Distributed Multi-Agent Optimization, pp. 340–386.

[4] 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.

[5] 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.

[6] 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.

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

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

[9] S. Lee and A. Nedić, "Asynchronous gossip-based random projection algorithms over networks," *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 953–968, 2016.

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

[11] 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.

[12] 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.

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

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

[15] R. Nishihara, L. Lessart, B. Recht, A. Packard, and M. I. Jordan, "A general analysis of the convergence of ADMM," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.

[16] F. Iutzeler, P. Bianchi, P. Ciblat, and W. Hachem, "Explicit convergence rate of a distributed Alternating Direction Method of Multipliers," *IEEE Transactions on Automatic Control*, vol. 61, no. 4, pp. 892–904, 2016.

[17] I. Notarnicola and G. Notarstefano, "Asynchronous distributed optimization via randomized dual proximal gradient," *IEEE Transactions on Automatic Control*, 2016.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] ——, "A Distributed Newton Method for Network Utility Maximization - Part II: Convergence," *IEEE Transactions on Automatic Control*, vol. 58, no. 9, pp. 2176 – 2188, 2013.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.
[31] 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.
[32] E. Seneta, *Non-negative Matrices and Markov Chains*. Springer: John Wiley & Sons, Inc., 2006.
[33] N. Bof, R. Carli, and L.Schenato, "Average consensus with asynchronous updates and unreliable communication," in *Submitted to IFAC World Congress (IFAC'17)*, 2017.
[34] F. Zanella, D. Varagnolo, A. Cenedese, P. Gianluigi, and L. Schenato, "Newton-Raphson Consensus for Distributed Convex Optimization," in *Proc. 50th IEEE Conf. on Decision and Control*, Orlando, Florida, December 2011, pp. 5917 – 5922.
[35] H. K. Khalil, *Noninear Systems*. Prentice-Hall, New Jersey, 1996.
[36] N. H. Vaidya, C. N. Hadjicostis, and A. D. Dominguez-Garcia, "Distributed Algorithms for Consensus and Coordination in the Presence of Packet-Dropping Communication Links Part II: Coefficients of Ergodicity Analysis Approach," in *arXiv*, 2011.
[37] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics Springer, Berlin, 2001, vol. 1.
[38] N. Bof, R. Carli, and L. Schenato, "Lyapunov theory for discrete time systems," Tech. Rep., 2017, [Online] Available at http://automatica.dei.unipd.it/tl_files/utenti2/bof/Papers/NoteDiscreteLyapunov.pdf.