A Robust Block-Jacobi Algorithm for Quadratic Programming under Lossy Communications

M. Todescato * **G. Cavraro** * **R. Carli** * **L. Schenato** *

* *Department of Information Engineering, Padova, 35131, Italy. (e-mail: todescat|cavraro|carlirug|schenato @ dei.unipd.it).*

Abstract: We address the problem distributed quadratic programming under lossy communications where the global cost function is the sum of coupled local cost functions, typical in localization problems and partition-based state estimation. We propose a novel solution based on a generalized gradient descent strategy, namely a Block-Jacobi descent algorithm, which is amenable for a distributed implementation and which is provably robust to communication failure if the step size is sufficiently small. Interestingly, robustness to packet loss, implies also robustness of the algorithm to broadcast communication protocols, asynchronous computation and bounded random communication delays. The theoretical analysis relies on the separation of time scales and singular perturbation theory. Our algorithm is numerically studied in the context of partition-based state estimation in smart grids based on the IEEE 123 nodes distribution feeder benchmark. The proposed algorithm is observed to exhibit a similar convergence rate when compared with the well known ADMM algorithm with no packet losses, while it has considerably better performance when including moderate packet losses.

Keywords: Multi-agent systems. Distributed optimization. Packet loss. Quadratic programming. Asynchronous. Broadcast. Singular perturbation Theory.

1. INTRODUCTION

The proliferation of wireless and the IP interconnection of smart electronic devices is creating large-scale cyber-physical systems which promise a new revolution in many fields. However, these systems require the development of new engineering design and computation paradigms due to the sheer amount of devices and data to be managed. For example, many problems have been shown to be cast as optimization problems. As so there has been a growing attention in the last years to distributed optimization tools. Distributed optimization has become so important for two different reasons: the first reason is that with the advent of Big Data, it is unconceivable to run optimization algorithms on a single (super)-computer, but it is necessary to *parallelise* computation among many processors. The second reason is that many optimization problems are *sparse* by nature since correlation between data is local. One of the major difficulty to deal with distributed optimization using multiple processing units is to guarantee reliable synchronisation and communication since communication can be wireless and CPU execution times might not be known in advance as in the context of cloud-computing. Although distributed optimization algorithms have a long history in the parallel and distributed computation literature, see, e.g. Bertsekas and Tsitsiklis (1989), it has mainly focused on synchronous algorithms. However, in the past years it has been reconsidered in a new peer-to-peer set-up which is more suitable for today's problems. The first class of algorithms appearing in this new literature relies on primal sub-gradient or descent iterations as in Nedic and Ozdaglar (2009); Nedic et al. (2010); Marelli and Fu (2015) which have the advantage to be easy to implement and suitable for asynchronous computation. In order to induce robustness in the computation and improve convergence speed, augmented lagrangian algorithms such as the Alternating Direction Methods of Multipliers (ADMM) have been recently proposed. A first distributed ADMM algorithm was proposed in Schizas et al. (2008); Kekatos and Giannakis (2013); Bolognani et al. (2014), while a survey on this technique is Boyd et al. (2011). A common drawback of this technique, is that each node must store in its local memory a copy of the entire state vector. To avoid this problem, a recent partition-based and scalable approach applied to the ADMM algorithm is presented in Erseghe (2012), while to comply with asynchronous computation, suitable modification of the ADMM algorithm have been proposed in Iutzeler et al. (2013); Bianchi et al. (2014). Finally, distributed algorithms based on Newton methods have been proposed to speed-up the computation (Zargham et al., 2014; Zanella et al., 2011).

In this paper we propose a strategy to implement an algorithm for distributed optimization based on a modified version of the generalized gradient descent method where the cost function is quadratic and exhibit a specific structure that can be solved by partitioning the state-space as in Schizas et al. (2008); Kekatos and Giannakis (2013); Bolognani et al. (2014). In particular, we focus on a class of quadratic optimization problems which can be encountered on a large variety of application such as electric grid state estimation (Bolognani et al. (2014)), multi-robot localization (Carron et al. (2014)) and Network Utility Maximization (Palomar and Chiang (2006)). The main contribution of the paper is a provably convergent and distributed algorithm, under suitable assumptions on the step size, which is robust to the presence of packet losses in the communication channel. To the best of the authors knowledge, this is one of the first provably convergent algorithms in the presence of packet losses, since both ADMM algorithms and distributed sub gradient methods (DSM) require reliable communication. Interestingly,

the proposed algorithm is also suitable for fully parallel computation, i.e. multiple agents can communicate and update their local variable simultaneously, and for broadcast communication, i.e. nodes do not need to enforce a bidirectional communication such as in gossip algorithms, therefore very attractive from a practical point of view.

1.1 Mathematical Preliminaries

In this paper, $\mathcal{G}(\mathcal{V},\mathcal{E})$ denotes a directed graph where $\mathcal{V}=\{1,...,N\}$ is the set of vertices and $\mathcal{E}$ is the set of directed edges, i.e., a subset of $\mathcal{V}\times\mathcal{V}$. More precisely the edge (i,j) is incident on node i and node j and is assumed to be directed away from i and directed toward j. The graph $\mathcal{G}$ is said to be bidirected if $(i,j)\in\mathcal{E}$ implies $(j,i)\in\mathcal{E}$. Given a directed graph $\mathcal{G}(\mathcal{V},\mathcal{E})$, a directed path in $\mathcal{G}$ consists of a sequence of vertices $(i_1,i_2,\dots,i_r)$ such that $(i_j,i_j+1)\in\mathcal{E}$ for every $j\in\{1,\dots,r-1\}$. The directed graph $\mathcal{G}$ is said to be *strongly connected* if for any pair of vertices (i,j) there exists a directed path connecting i to j. Given the directed graph $\mathcal{G}$, the set of neighbors of node i, denoted by $\mathcal{N}_i$, is given by $\mathcal{N}_i=\{j\in\mathcal{V}\,|\,(i,j)\in\mathcal{E}\}$. Moreover, $\mathcal{N}_i^+=\mathcal{N}_i\cup i$. Given a directed graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ with $|\mathcal{E}|=M$ let the *incidence matrix* $\mathcal{A}\in\mathbb{R}^{M\times N}$ of $\mathcal{G}$ be defined as $\mathcal{A}=[a_{ei}]$, where $a_{ei}=1,-1,0$, if edge e is incident on node i and directed away from it, is incident on node i and directed toward it, or is not incident on node i, respectively. Given a vector v with v^T we denote its transpose. Moreover, we denote with A_d the *adjacency matrix* or *laplacian matrix* of the graph which is defined as $\mathcal{A}_d:=\mathcal{A}^T\mathcal{A}$, which has the property that $[\mathcal{A}_d]_{ij}\neq 0$ if and only if $(i,j)\in\mathcal{E}$. If we associate to each edge a weight different from one, then it is possible to define the *weighted laplacian matrix* as $\mathcal{L}=\mathcal{A}^TW\mathcal{A}$ where $W\in\mathbb{R}^{M\times M}$ represents the diagonal matrix containing in its ℓ-th element the weight associated to the ℓ-th edge. Given a vector v with the symbols $\Re(v)$ and $\Im(v)$ we denote its real and imaginary parts, respectively. Finally, with the symbols $\mathbb{E}$ and $\mathbb{P}$ we denote, respectively, the expectation operator and the probability of an event.

2. PROBLEM FORMULATION

Consider the set of N agents $\mathcal{V}=\{1,\dots,N\}$, where each agent i is described by its state vector $x_i\in\mathbb{R}^{n_i}$. Assume the agents can communicate among themselves through a bidirected strongly connected *communication graph* $\mathcal{G}(\mathcal{V},\mathcal{E})$. Assume each agent collects a set of measurements $b_i\in\mathbb{R}^{m_i}$ which are noisy linear combinations of its own state and those of its neighboring agents, i.e.,

$$b_i=\sum_{j\in\mathcal{N}_i^+}A_{ij}x_j+w_i=A_{ii}x_i+\sum_{j\in\mathcal{N}_i}A_{ij}x_j+w_i$$

where $A_{ij}\in\mathbb{R}^{m_i\times n_j}$ and where w_i is white noise of zero mean and variance R_i independent of the other w_j. We consider the problem of estimating the entire state of the network from the knowledge of the noisy measurements. By collecting all the agents state and measurements in the vectors $x=[x_1^T,\dots,x_N^T]^T\in\mathbb{R}^n$ (where $n=\sum_i n_i$) and $b:=[b_1^T,\dots,b_N^T]\in\mathbb{R}^m$ (where $m=\sum_i m_i$), respectively, it is possible to formulate the problem as a classical *weighted least square* problem. That is

$$\min_x J(x), \tag{1}$$

where

$$J(x)=\frac{1}{2}(Ax-b)^TR^{-1}(Ax-b)\,. \tag{2}$$

The matrix $A\in\mathbb{R}^{m\times n}$ represents the measurements matrix, whose ij-th block is simply defined as $[A]_{ij}=A_{ij}$, while $R\in\mathbb{R}^{m\times m}$ is the block diagonal matrix defined as

$$R=\text{blkdiag}\{R_1,\dots,R_N\}\,,$$

which represents the noise variance.
From now on, we assume that $n\leq m$ and that A is full rank. Under these assumptions, it is well known that the solution of (1) is unique and given by

$$x^*=(A^TR^{-1}A)^{-1}A^TR^{-1}b\,. \tag{3}$$

To compute the value of x^* directly as in (3), one needs all the measurements, the matrix A and the noise variance R, i.e., full knowledge of the network is required. On the contrary, we aim at solving Problem (1) in a distributed fashion. To this end, note that it is possible to rewrite Problem (1) as

$$\min_{x_1,\dots,x_N}\ \sum_{i=1}^N J_i(x_i,\{x_j\}_{j\in\mathcal{N}_i})\,, \tag{4}$$

where

$$J_i(x_i,\{x_j\}_{j\in\mathcal{N}_i})=$$
$$\frac{1}{2}(A_{ii}x_i+\sum_{j\in\mathcal{N}_i}A_{ij}x_j-b_i)^TR_i^{-1}(A_{ii}x_i+\sum_{j\in\mathcal{N}_i}A_{ij}x_j-b_i).$$

The above equation highlights the local dependence of each cost function J_i on information regarding only agent i and its neighbors $j\in\mathcal{N}_i$. In next section, we present a distributed algorithm to solve optimization problems in the separable quadratic form (4). Firstly, we review an implementation which requires synchronous communications among the agents and basically coincides with a *Block-Jacobi* algorithm. Secondly, we present a modified version which is amenable for a distributed implementation and is robust to the presence of packet losses in the communication channel. We will compare the proposed algorithm with the classical ADMM algorithm (Kekatos and Giannakis, 2013; Bolognani et al., 2014) which we briefly review in Section 4.

3. BLOCK JACOBI ALGORITHM

3.1 Distributed Block Jacobi Algorithm

Consider the *generalized gradient* descent strategy

$$x(t+1)=x(t)-\epsilon D^{-1}\nabla J(x(t)) \tag{5}$$

where $\nabla J(x(t))$ is the gradient of J, i.e., $\nabla J(x(t))=[\partial J(x(t))/\partial x]^T$, D is a generic positive definite matrix and ϵ a suitable positive constant, usually referred as *step size*. The algorithm we propose is a particular case of (5), where D is a block diagonal matrix whose i-th diagonal block is defined as

$$D_i=\sum_{j\in\mathcal{N}_i^+}A_{ji}^TR_j^{-1}A_{ji}. \tag{6}$$

With a little abuse of notation, let us denote the Hessian of the cost function as $\nabla^2J(x)$. From standard algebraic computations, it follows that $\nabla^2J(x)=A^TR^{-1}A$. The matrix $\nabla^2J(x)$ can be partitioned as an $N\times N$ block matrix, where the i,j-th block $\left[\nabla^2J(x)\right]_{ij}$ is given by $\frac{\partial J(x)}{\partial x_i\partial x_j}$. One can see that the block $[\nabla^2J(x)]_{ij}$ is different from zero either if $j\in\mathcal{N}_i^+$ or if i and j are *two step neighbors* (i.e. there exists a agent k such that k is neighbor of both i and j). Furthermore, it can be shown that

$D_i = [\nabla^2 J(x)]_{ii}$ is the i-th diagonal block of the cost function Hessian.
From (2), we can compute the gradient of the cost function

$$\nabla J(x(t)) = A^T R^{-1}(Ax(t) - b), \tag{7}$$

whose component associated with the i-th agent is

$$[\nabla J(x(t))]_i = \sum_{j \in \mathcal{N}_i^+} A_{ji}^T R_j^{-1} n_j(t+1) \tag{8}$$

where the variable $n_j(t+1)$ is defined as

$$n_j(t+1) = \sum_{k \in \mathcal{N}_j^+} A_{jk} x_k(t) - b_j. \tag{9}$$

By plugging (6), (8) into (5), we can write the updating step performed by agent i as

$$x_i(t+1) = x_i(t) - \epsilon D_i^{-1} \sum_{j \in \mathcal{N}_i^+} A_{ji}^T R_j^{-1} n_j(t+1), \tag{10}$$

which, in vector form, leads to

$$x(t+1) = (I - \epsilon D^{-1} A^T R^{-1} A)x(t) + \epsilon D^{-1} A^T R^{-1} b. \tag{11}$$

Observe that agent i, in order to perform (10), needs information coming from the neighbours of its neighbors, i.e. the two-step neighbours. As so, to each iteration of the algorithm it is necessary to perform two communications, the first to compute the $n_i(t+1)$'s and the second to compute the $x_i(t+1)$'s. The *distributed Block Jacobi algorithm* (denoted hereafter as the BJ algorithm) for quadratic functions is formally described as in Algorithm 1. Next, the convergence properties of the BJ algorithm are established.

Lemma 1. Consider Problem (1) and the BJ algorithm. Assume

$$\epsilon \leq \frac{2}{\|D^{-\frac{1}{2}} A^T R^{-1} A D^{-\frac{1}{2}}\|}. \tag{12}$$

Then, for any $x(0) \in \mathbb{R}^n$, the trajectory $x(t)$, generated by the BJ algorithm, converges exponentially fast to the minimizer of Problem (1), i.e.,

$$\|x(t) - x^*\| \leq C\rho^t$$

for some constants $C > 0$ and $0 < \rho < 1$.

Remark 2. It is worth noticing that to compute the step size upper bound of Eq.12 one needs complete knowledge of the network. It will be part of future research to find a possible distributed implementation of Eq.12.

In the interest of space, the proof can be found in technical report Todescato et al. (2015).

Algorithm 1 Distributed Block Jacobi algorithm.

Require: $\forall i \in \mathcal{V}$, store $A_{ij}, A_{ji}, R_j, j \in \mathcal{N}_i^+$.
1: **for** $t \in \mathbb{N}$ each $i \in \mathcal{V}$ **do**
2: sends $x_i(t)$ to $j \in \mathcal{N}_i$;
3: receives $x_j(t)$ from $j \in \mathcal{N}_i$;
4: updates $n_i(t)$ by using (9)
5: sends $n_i(t)$ to $j \in \mathcal{N}_i$;
6: receives $n_j(t)$ from $j \in \mathcal{N}_i$;
7: updates $x_i(t)$ by using (10)
8: **end for**

3.2 Robust Block Jacobi Algorithm

Algorithm 1 has been designed for the ideal case with no lossy communication. In the following, we generalize Algorithm 1 for the case with lossy communication, e.g. agent i could not receive information sent by some of its neighbors, due to communication failures. The modification of the algorithm is apparently naive, since we simply perform the same algorithm by using the last received data from its neighbours if a packet is not received.
To model the packet losses, it is convenient to introduce the indicator function

$$\gamma_j^{(i)}(t) = \begin{cases} 1 & \text{if } i \text{ received the information sent by } j \\ 0 & \text{otherwise.} \end{cases}$$

with the assumption that $\gamma_i^{(i)}(t) = 1$, since node i has always access to its local variables $n_i(t)$ and $x_i(t)$. We assume the following property.

Assumption 3. There exists a constant T such that, for all $t \geq 0$, for all $i \in \mathcal{V}$ and for all $j \in \mathcal{N}_i$,

$$\mathbb{P}\left[\{\gamma_j^{(i)}(t), \ldots, \gamma_j^{(i)}(t+T)\} = \{0, \ldots, 0\}\right] = 0.$$

Roughly speaking, Assumption 3 states that agent i receives, at least once, information coming from agent j within any window of T iterations of the algorithm. Observe that, if agent i does not receive some of the packets transmitted by its neighbors, then it does not have the necessary information to perform the updates (9) and (10). To overcome this fact, we assume agent i stores in memory the auxiliary variables $x_j^{(i)}, n_j^{(i)}$, $j \in \mathcal{N}_i$, which are equal, respectively, to the last packets x_j and n_j received by agent i from agent j; specifically, the dynamics of $x_j^{(i)}, n_j^{(i)}$ are

$$n_j^{(i)}(t+1) = \begin{cases} n_j(t) & \text{if } \gamma_j^{(i)}(t) = 1 \\ n_j^{(i)}(t) & \text{if } \gamma_j^{(i)}(t) = 0 \end{cases} \tag{13}$$

$$x_j^{(i)}(t+1) = \begin{cases} x_i(t) & \text{if } \gamma_j^{(i)}(t) = 1 \\ x_j^{(i)}(t) & \text{if } \gamma_j^{(i)}(t) = 0 \end{cases}. \tag{14}$$

As mentioned in the previous section, Algorithm 1 requires two communication rounds every iteration. In a lossy environment, in order to reduce the communication burden and the number of communication failures, we modify Algorithm 1 by letting the agents to communicate just once every iteration, transmitting together the x_i's and the n_i's. Agents i exploit $x_j^{(i)}(t)$ and $n_j^{(i)}(t)$ to update n_i and x_i as

$$n_i(t+1) = \sum_{j \in \mathcal{N}_i^+} A_{ij} x_j^{(i)}(t) - b_i \tag{15}$$

$$x_i(t+1) = x_i(t) - \epsilon D_i^{-1} \sum_{j \in \mathcal{N}_i^+} A_{ji}^T R_j^{-1} n_j^{(i)}(t) \tag{16}$$

As so, even in the scenario with no packet losses, this new algorithm does not exactly coincide with Algorithm 1, since a one-step delay is introduced in the computation of the variables $n_i(t)$. The *robust block Jacobi algorithm* (hereafter referred to as r-BJ algorithm) for quadratic functions is formally described as in Algorithm 2. The left panel of Figure 1 provides a pictorial representation of the stored and communicated variables by each node. The convergence properties of the r-BJ algorithm are next established.

Theorem 4. Let Assumption 3 hold. Consider Problem (1) and the r-BJ algorithm. There exists $\bar{\epsilon}$ such that, if $0 < \epsilon < \bar{\epsilon}$, then, for any $x(0) \in \mathbb{R}^n$, the trajectory $x(t)$, generated by the BJ algorithm, converges exponentially fast to the minimizer of Problem (1), i.e.,

$$\|x(t) - x^*\| \leq C\rho^t$$

for some constants $C > 0$ and $0 < \rho < 1$.

Algorithm 2 Robust Block Jacobi algorithm.

Require: $\forall i \in \mathcal{V}$, store $A_{ij}, A_{ji}, R_j, j \in \mathcal{N}_i^+$.
1: **for** $t \in \mathbb{N}$ each $i \in \mathcal{V}$ **do**
2: sends $x_i(t), n_i(t)$ to $j \in \mathcal{N}_i$;
3: **if** $\gamma_j^{(i)}(t) = 1$ **then**
4: receives $x_j(t)$ and $n_j(t)$ from $j \in \mathcal{N}_i$
5: **end if**
6: updates $n_i(t)$ by using (15)
7: updates $n_j^{(i)}(t)$ by using (13)
8: updates $x_j^{(i)}(t)$ by using (14)
9: updates $x_i(t)$ by using (16)
10: **end for**

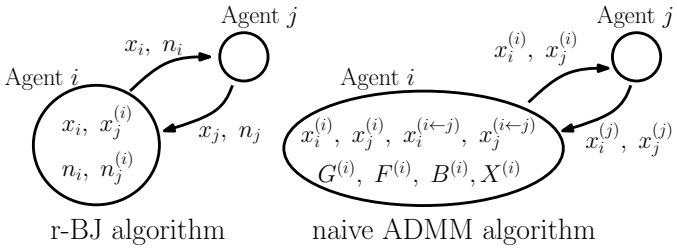

Fig. 1. Communication scheme for the BJ algorithm (left) and for the ADMM algorithm (right).

The proof of Theorem 4 can be found in Todescato et al. (2015), and basically relies on separation of time scales principle between the dynamics of the states x_i's and the auxiliary variables $x_j^{(i)}$'s, n_i's and $n_j^{(i)}$'s. Loosely speaking, if the update step-size parameter ϵ is small enough, the variation of the true states x_i's is so slow that, despite the lossy communication, the values of the copies $x_j^{(i)}$'s and $n_j^{(i)}$'s are essentially equal to the x_i's and the n_i's, respectively.

Remark 5. We would like to emphasize that this general model of packet losses includes as special cases asynchronous updates. In fact, asynchronous updates where only one node i updates its local variables based on the information received form its neighbours can be recovered by our algorithm assuming that all packets are lost except those from the neighbours of node i to node i itself. Also, our algorithm allows for multiple agents to communicate and perform updates at the same time, thus requiring no coordination. Finally, broadcast communication can be used since nodes do not need to establish reliable bidirectional communication as in gossip protocols.

4. ADMM ALGORITHM

In this section we briefly review the ADMM algorithm which was first presented in the mid 70's and later reviewed in Bertsekas (2014); Boyd et al. (2011). The algorithm has then been adapted for distributed optimization (Kekatos and Giannakis, 2013). Here we recall a partition based implementation of the algorithm which is specific for quadratic cost function and has been presented in Bolognani et al. (2014). It is well known that the ADMM algorithm is widely used for its good convergence properties. However, to the best of our knowledge, no implementation of the ADMM with theoretical guarantees has been presented in the literature to deal with the presence of unreliable communications.

4.1 Partition Based ADMM Algorithm

In this section we shortly review a partition-based version of the ADMM algorithm for quadratic functions of the type considered in this paper. We use the same notation used in Bolognani et al. (2014) to which we refer the reader. Let us consider agent i and, without loss of generality, assume $\mathcal{N}_i = \{j_1, \dots, j_{|\mathcal{N}_i|}\}$. Then let

$$X^{(i)} = \begin{bmatrix} x_i^{(i)} \\ \left\{x_j^{(i)}\right\}_{j\in\mathcal{N}_i} \end{bmatrix}, \quad A_i = \begin{bmatrix} A_{ii} & A_{ij_1} & \dots & A_{ij_{|\mathcal{N}_i|}} \end{bmatrix},$$

$$M_i = \operatorname{diag}\left\{ |\mathcal{N}_i|\, I_{n_i}, I_{n_{j_1}}, \dots, I_{n_{j_{|\mathcal{N}_i|}}} \right\}.$$

Additionally let

$$G^{(i)} = \begin{bmatrix} G_i^{(i)} \\ G_{j_1}^{(i)} \\ \vdots \\ G_{j_{|\mathcal{N}_i|}}^{(i)} \end{bmatrix}, F^{(i)} = \begin{bmatrix} F_i^{(i)} \\ F_{j_1}^{(i)} \\ \vdots \\ F_{j_{|\mathcal{N}_i|}}^{(i)} \end{bmatrix}, B^{(i)} = \begin{bmatrix} B_i^{(i)} \\ B_{j_1}^{(i)} \\ \vdots \\ B_{j_{|\mathcal{N}_i|}}^{(i)} \end{bmatrix},$$

where $G_i^{(i)}, F_i^{(i)}, B_i^{(i)} \in \mathbb{R}^{n_i}$ and $G_{j_h}^{(i)}, F_{j_h}^{(i)}, B_{j_h}^{(i)} \in \mathbb{R}^{n_{j_h}}$. It turns out that $A_i \in \mathbb{R}^{m_i \times \gamma_i}$, $M_i \in \mathbb{R}^{\gamma_i \times \gamma_i}$ and $G^{(i)}, F^{(i)}, B^{(i)} \in \mathbb{R}^{\gamma_i}$, where $\gamma_i = n_i + \sum_{h=1}^{|\mathcal{N}_i|} n_{j_h}$.

The partition-based *ADMM algorithm* (which hereafter we refer to as ADMM) for quadratic functions is formally described in Algorithm 3. The standing assumption is that all the matrices $A_i^T R_i^{-1} A_i + M_i$, $i \in \mathcal{V}$ are invertible. As

Algorithm 3 Partitioned Based ADMM algorithm.

Require: $\forall i \in \mathcal{V}$, store and initialize to 0 $X^{(i)}, G^{(i)}, F^{(i)}, B^{(i)}$.
1: **for** $t \in \mathbb{N}$ each $i \in \mathcal{V}$ **do**
2: sends $x_i^{(i)}(t), x_j^{(i)}(t)$ to $j \in \mathcal{N}_i$;
3: receives $x_i^{(j)}(t), x_j^{(j)}(t)$ from $j \in \mathcal{N}_i$;
4: updates the memory and the estimate as

$$G_i^{(i)}(t) = \frac{\rho}{2} \sum_{j\in\mathcal{N}_i} \left(x_i^{(i)}(t) - x_i^{(j)}(t) \right)$$

$$G_{j_h}^{(i)}(t) = \frac{\rho}{2} \left(x_{j_h}^{(i)}(t) - x_{j_h}^{(j_h)}(t) \right), \quad 1 \le h \le |\mathcal{N}_i|$$

$$F^{(i)}(t+1) = F^{(i)}(t) + G^{(i)}(t)$$

$$B^{(i)}(t+1) = 2\rho M_i X^{(i)}(t) - G^{(i)}(t) - 2F^{(i)}(t+1)$$

$$X^{(i)}(t+1) = \left[A_i^T R_i^{-1} \ A_i + M_i \right]^{-1} \left[A_i^T R_i^{-1} b_i + \frac{1}{2} B^{(i)}(t+1) \right]$$

5: **end for**

shown in Bolognani et al. (2014) and reference therein, the algorithm is provably convergent for quadratic cost functions for any value of the parameter ρ.

4.2 Partition Based ADMM Algorithm with packet losses

To the best of our knowledge there is no ADMM convergent implementation in presence of communication failures. We propose here a possible naive implementation of the ADMM algorithm to deal with packet losses which adopt the very simple idea used in r-BJ algorithm proposed in Section 3.2. More precisely, we simply use the last received information from a node if a packet loss occurs. To do so, we equip each agent with additional slots of

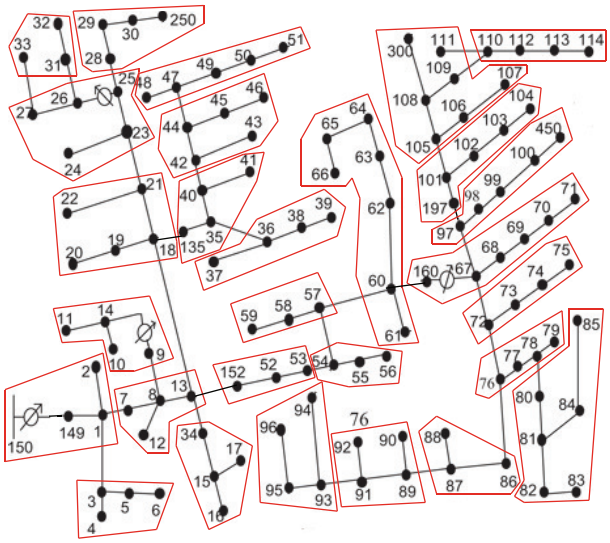

Fig. 2. IEEE 123 node split into 28 non overlapping areas.

memory. Specifically, agent i, for $j \in \mathcal{N}_i$, stores in memory the additional variables $x_i^{(i\leftarrow j)}, x_j^{(i\leftarrow j)}$, which are updated as follows:

$$x_i^{(i\leftarrow j)}(t) = \begin{cases} x_i^{(j)}(t) & \text{if } \gamma_j^{(i)}(t) = 1 \\ x_i^{(i\leftarrow j)}(t-1) & \text{if } \gamma_j^{(i)}(t) = 0 \end{cases} \tag{17}$$

$$x_j^{(i\leftarrow j)}(t) = \begin{cases} x_j^{(j)}(t) & \text{if } \gamma_j^{(i)}(t) = 1 \\ x_j^{(i\leftarrow j)}(t-1) & \text{if } \gamma_j^{(i)}(t) = 0 \end{cases} . \tag{18}$$

By exploiting these additional variables, it is possible to modify Algorithm 3 by updating $G_i^{(i)}(t)$ and $G_j^{(i)}(t)$ as

$$G_i^{(i)}(t) = \frac{\rho}{2} \sum_{j\in\mathcal{N}_i} \left(x_i^{(i)}(t) - x_i^{(i\leftarrow j)}(t) \right),$$

$$G_j^{(i)}(t) = \frac{\rho}{2} \left(x_j^{(i)} - x_j^{(i\leftarrow j)} \right).$$

while the remaining part of the algorithm is the same as before. The right panel of Figure 1 provides a pictorial representation of the stored and communicated variables by each node for this modified algorithm that we refer to as *naive ADMM*.

5. SIMULATIONS

In this section we compare the Block Jacobi and the ADMM algorithms in the ideal synchronous implementation with no packet losses and in the more realistic scenario with random packet losses.

The algorithms have been tested on the IEEE 123 nodes distribution grid benchmark (see Bolognani et al. (2014)). For our purpose, the feeder has been divided into non overlapping areas as shown in Figure 2. The areas identify the agents of the communication graph $\mathcal{G}$. In particular, we assume there exists, for each area, a smart monitor able to sense the physical part of the grid which it has been assigned to, and to communicate with the monitors of the areas which are physically connected to it.

Here, the algorithms presented in Sections 3 and 4 are exploited to estimate the state of the electric grid which consists in the voltage real and imaginary parts at every node of the grid. We assume each node v of the grid is described by its bus voltage $u_v \in \mathbb{C}$ and its injected current $i_v \in \mathbb{C}$. Let us stack together all the voltages and all the currents in the vectors $u = [u_1, \dots, u_{N_{e\ell}}]^T$ and

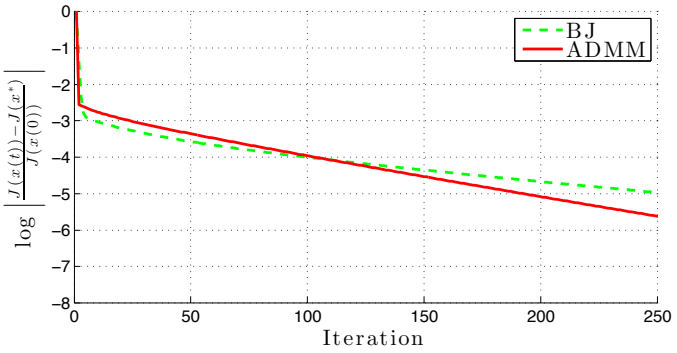

Fig. 3. Comparison between the optimized BJ algorithm and the ADMM with no packet losses, i.e. $\overline{\gamma} = 0$.

$i = [i_1, \dots, i_{N_{e\ell}}]^T$, respectively, where $N_{e\ell}$ is the number of nodes. Then, thanks to Kirchhoff's current and voltage laws, the relation among currents and voltages is

$$i = \mathcal{L}_{e\ell} u \,. \tag{19}$$

where $\mathcal{L}_{e\ell}$ is the electrical Laplacian of the grid which depends on the impedances of the electric lines connecting the nodes. Equation (19) can be expanded into its real and imaginary parts leading to

$$\begin{bmatrix} \Re(i) \\ \Im(i) \end{bmatrix} = \begin{bmatrix} \Re(\mathcal{L}_{e\ell}) & -\Im(\mathcal{L}_{e\ell}) \\ \Im(\mathcal{L}_{e\ell}) & \Re(\mathcal{L}_{e\ell}) \end{bmatrix} \begin{bmatrix} \Re(u) \\ \Im(u) \end{bmatrix}.$$

We assume to have at our disposal noisy measurements of the real and imaginary parts of voltage and current at every node of the grid. These are retrieved with phasor measurement units (PMUs) placed at each node. By defining the measurements vector b, the observation matrix A and the state vector x, respectively, as

$$b = \begin{bmatrix} b_{\Re(u)} \\ b_{\Im(u)} \\ b_{\Re(i)} \\ b_{\Im(i)} \end{bmatrix}, \quad A = \begin{bmatrix} I & 0 \\ 0 & I \\ \Re(\mathcal{L}_{e\ell}) & -\Im(\mathcal{L}_{e\ell}) \\ \Im(\mathcal{L}_{e\ell}) & \Re(\mathcal{L}_{e\ell}) \end{bmatrix}, \quad x = \begin{bmatrix} \Re(u) \\ \Im(u) \end{bmatrix}, \tag{20}$$

the measurements model reads as

$$b = Ax + w \,, \tag{21}$$

where w is the noise vector and $R = \mathbb{E}[ww^T]$ is its correlation matrix.

For a detailed analysis, we refer the interested reader to Bolognani et al. (2014) where an exhaustive modeling of the grid and of the measurements is given.

We partition A, b, x, w according to the splitting showed in Figure 2 and we formulate the state estimation problem as in (4). Next, we compare the performance of the Block-Jacobi algorithm with the performance of the ADMM algorithm. In particular, we compare the algorithms in terms of the evolution of the normalized distance of the cost function from its minimal value, i.e.,

$$\left| \frac{J(x(t)) - J(x^*)}{J(x(0))} \right| \tag{22}$$

where

$$J(x(t)) = \frac{1}{2}(Ax(t) - b)^T R^{-1}(Ax(t) - b)$$

being $x(t)$ the estimated state at iteration t obtained by either the Block Jacobi or the ADMM algorithm. In Figure 3, we depict the behavior of the two algorithms in the ideal scenario of reliable communications, i.e., no packet losses occur. The results reported have been obtained optimizing the performance of both algorithms over the parameters ρ and ϵ. We can see that the performance of the ADMM and BJ algorithms are comparable.

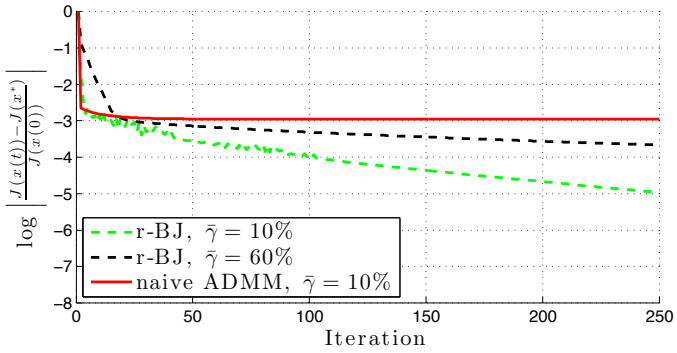

Fig. 4. Comparison between the r-BJ and the naive ADMM algorithm for different packet loss probabilities $\bar{\gamma}$.

Instead, in Figure 4 we compare the two algorithms in the lossy scenario. The parameters ϵ and ρ have been again manually optimised for best performance in both algorithms. In particular we assume that for all $i, j \in \mathcal{V}$ and for all $t \geq 0$

$$\mathbb{P}[\gamma_j^{(i)}(t) = 1] = 1 - \bar{\gamma},$$

where $\bar{\gamma}$ denotes the failure probability in the communication channel. Note that, according to the above probabilistic model, Assumption 3 is not necessarily satisfied, since we cannot ensure that within every T iterations every agent receives at least one packet from each of its neighbors. Nonetheless, the r-BJ algorithm still exhibit exponential convergence to the optimal solution even for high packet loss rates (60%). Differently, the ADMM fails to converge to the optimal point already in the presence of a moderate packet loss (10%).

6. CONCLUSIONS AND FUTURE DIRECTION

In this work we addressed the problem of distributed quadratic programming in presence of communication failure. We presented a modified version of the Block Jacobi algorithm which is robust to packet losses in the communication channel if the step-size is sufficiently small. We compared the algorithm with the well known ADMM algorithm in the context of smart grid state estimation based on the IEEE 123 nodes distribution feeder benchmark, showing comparable or better performance.
As possible future research directions, we intend to extend the algorithm to general smooth convex cost functions and to estimate the critical step-size for stability of the algorithm and possibly to optimize it for best convergence speed.

REFERENCES

Bertsekas, D.P. (2014). *Constrained optimization and Lagrange multiplier methods*. Academic press.

Bertsekas, D.P. and Tsitsiklis, J.N. (1989). *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Bianchi, P., Hachem, W., and Iutzeler, F. (2014). A stochastic coordinate descent primal-dual algorithm and applications to large-scale composite optimization. *arXiv preprint arXiv:1407.0898*.

Bolognani, S., Carli, R., and Todescato, M. (2014). State estimation in power distribution networks with poorly synchronized measurements. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, 2579–2584.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1), 1–122.

Carron, A., Todescato, M., Carli, R., and Schenato, L. (2014). An asynchronous consensus-based algorithm for estimation from noisy relative measurements. *IEEE Transactions on Control of Network Systems*, 1(3), 283 – 295.

Erseghe, T. (2012). A distributed and scalable processing method based upon admm. *Signal Processing Letters, IEEE*, 19(9), 563–566.

Iutzeler, F., Bianchi, P., Ciblat, P., and Hachem, W. (2013). Asynchronous distributed optimization using a randomized alternating direction method of multipliers. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, 3671–3676. IEEE.

Kekatos, V. and Giannakis, G.B. (2013). Distributed robust power system state estimation. *Power Systems, IEEE Transactions on*, 28(2), 1617–1626.

Marelli, D.E. and Fu, M. (2015). Distributed weighted least-squares estimation with fast convergence for large-scale systems. *Automatica*, 51, 27–39.

Nedic, A. and Ozdaglar, A. (2009). Distributed subgradient methods for multi-agent optimization. *Automatic Control, IEEE Transactions on*, 54(1), 48–61.

Nedic, A., Ozdaglar, A., and Parrilo, P.A. (2010). Constrained consensus and optimization in multi-agent networks. *Automatic Control, IEEE Transactions on*, 55(4), 922–938.

Palomar, D.P. and Chiang, M. (2006). A tutorial on decomposition methods for network utility maximization. *Selected Areas in Communications, IEEE Journal on*, 24(8), 1439–1451.

Schizas, I.D., Ribeiro, A., and Giannakis, G.B. (2008). Consensus in ad hoc wsns with noisy linkspart i: Distributed estimation of deterministic signals. *Signal Processing, IEEE Transactions on*, 56(1), 350–364.

Todescato, M., Cavraro, G., Carli, R., and Schenato, L. (2015). Convergence of the robust block jacobi algorithm in presence of packet losses. Technical report. URL `http://automatica.dei.unipd.it/people/todescato/publications.html`.

Zanella, F., Varagnolo, D., Cenedese, A., Pillonetto, G., and Schenato, L. (2011). Newton-raphson consensus for distributed convex optimization. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, 5917–5922. IEEE.

Zargham, M., Ribeiro, A., Ozdaglar, A., and Jadbabaie, A. (2014). Accelerated dual descent for network flow optimization. *Automatic Control, IEEE Transactions on*, 59(4), 905–920.