
F. Farina, A. Garulli, A. Giannitrapani and G. Notarstefano, "Asynchronous Distributed Method of Multipliers for Constrained Nonconvex optimization," 2018 European Control Conference (ECC), Limassol, 2018, pp. 2535-2540.

---

https://doi.org/10.1109/ECC.2018.8550098

Asynchronous Distributed Method of Multipliers for Constrained Nonconvex Optimization

Francesco Farina[1], Andrea Garulli[1], Antonio Giannitrapani[1], Giuseppe Notarstefano[2]

***Abstract*— This paper addresses a class of constrained optimization problems over networks in which local cost functions and constraints can be nonconvex. We propose an asynchronous distributed optimization algorithm, relying on the centralized Method of Multipliers, in which each node wakes up in an uncoordinated fashion and performs either a descent step on a local Augmented Lagrangian or an ascent step on the local multiplier vector. These two phases are regulated by a distributed logic-AND, which allows nodes to understand when the descent on the (whole) Augmented Lagrangian is sufficiently small. We show that this distributed algorithm is equivalent to a block coordinate descent algorithm for the minimization of the Augmented Lagrangian followed by an update of the whole multiplier vector. Thus, the proposed algorithm inherits the convergence properties of the Method of Multipliers.**

I. Introduction

In several cyber-physical network contexts, ranging from control to estimation and learning, nonconvex optimization problems frequently arise. In these contexts, typically each device knows only a portion of the whole objective function and a subset of the constraints, so that, to avoid the presence of a central coordinator, distributed algorithms are needed.

Distributed optimization methods relevant for our paper can be divided in two groups: those handling nonconvex functions but not local constraints, and methods handling local constraints, typically designed for convex problems. Distributed algorithms for nonconvex optimization have started to appear in the literature only recently. Regarding constrained problems, most of the proposed methods usually do not deal with local constraints. In [1] a stochastic gradient method is proposed, while in [2] a decentralized Frank-Wolfe method is presented. In [3], [4] the authors propose distributed algorithms, based on the idea of tracking the whole function gradient and performing successive convex approximations of the nonconvex cost function. A perturbed push-sum algorithm for minimizing the sum of nonconvex functions is presented in [5].

Regarding distributed optimization algorithms handling local constraints, in [6] the authors propose a distributed random projection algorithm, while a proximal based algorithm is presented in [7]. In [8] the author proposes randomized block-coordinate descent methods. As for Lagrangian based distributed algorithms, in [9] an asynchronous distributed ADMM is proposed for a separable, constrained optimization problem. Other ADMM based approaches are presented in [10], [11], [12], while an asynchronous proximal dual algorithm is proposed in [13]. Finally, a distributed algorithm for a structured class of nonconvex optimization problems with local constrains is presented in [14].

The main contribution of this paper is a *fully* distributed optimization algorithm called ASYnchronous Method of Multipliers (ASYMM), addressing constrained optimization problems over networks in which both local cost functions and local constraints may be nonconvex. ASYMM is designed for asynchronous networks. It features two types of local updates at each node, a primal and a multiplier one, which are regulated by an asynchronous distributed logic-AND algorithm. When awake, nodes start performing a descent step on a local Augmented Lagrangian. By means of a distributed logic-AND algorithm, they realize when all of them have reached a given tolerance on their local gradient. Thus, still asynchronously, they start performing the multiplier updates. An interesting feature of ASYMM is that a node just needs to receive all neighboring multipliers to start again its primal descent. The analysis shows that ASYMM implements a suitable inexact version of the Method of Multipliers and hence it inherits all the convergence properties of the centralized method (see [15], [16]). It is worth mentioning that our algorithm implicitly leverages on some results given in [17], [18] to handle non-regularity of local minima (due to the presence of local copies of the common decision variable). In this respect, ASYMM presents three main novelties with respect to the distributed algorithm proposed in [17], [18]: the network model is asynchronous; primal minimization is performed approximately; switching from a primal to a multiplier update is performed in a distributed way.

Due to space limitations all the proofs of the presented results are omitted and will be provided in a forthcoming document.

Organization: In Section II-A we present the distributed optimization set-up. The proposed distributed algorithm is presented in Section III and analyzed in Section IV. Finally, a numerical application is presented in Section V.

Notation and definitions: Given a matrix $A \in \mathbb{R}^{n\times m}$ we denote by $A[i,j]$ the (i,j)-th element of A, $A[:,i]$ its i-th column and $A[i,:]$ its i-th row. We write $A\,[i,:] = b$ to assign the value b to all the elements in the i-th row of A. Given two vectors $A, B \in \mathbb{R}^{n\times 1}$, we write $A > c$ if all elements of A are greater than c and $A > B$ if $A[i] > B[i]$ for all i. We denote by $\mathbf{0}_{n\times m}$ the $n \times m$ zero matrix. If $J = \{j_1, ..., j_m\}$ is a set

[1]F. Farina, A. Garulli and A. Giannitrapani are with the Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche, Università di Siena, Siena, Italy.

[2]G. Notarstefano is with the Department of Engineering, Università del Salento, Lecce, Italy.

This result is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 638992 - OPT4SMART).

of indexes, we denote by $[z_j]_{j\in J}$ the vector $[z_{j_1}, ..., z_{j_m}]$. A function $\Psi(x)$ has *Lipschitz continuous gradient* if there exists a constant L such that $\|\nabla\Psi(x)-\nabla\Psi(y)\| \leq L\|x-y\|$ for all x, y. It is *σ-strongly convex* if $(\nabla\Psi(x)-\nabla\Psi(y))^\top(x-y) \geq \sigma\|x-y\|^2$. Let $\mathbf{x} = [x_1^\top, ..., x_N^\top]^\top$, with $x_i \in \mathbb{R}^{n_i}$ and $\sum_{i=1}^N n_i = m$, and let $U^{m\times m}$ be a column partition of the identity matrix such that $\mathbf{x} = \sum_{i=1}^N U_i x_i$ and $x_i = U_i^\top \mathbf{x}$. The function $\Phi(\mathbf{x})$ has *block component-wise Lipschitz continuous gradient* if there are constants $L_i \geq 0$ such that $\|\nabla_{x_i}\Phi(\mathbf{x}+U_i s_i) - \nabla_{x_i}\Phi(\mathbf{x})\| \leq L_i\|s_i\|$ for all $\mathbf{x} \in \mathbb{R}^n$ and $s_i \in \mathbb{R}^{n_i}$.

II. Set-up and Preliminaries

A. Distributed Optimization Set-up

Consider the following optimization problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{i=1}^N f_i(x) \\ \text{subject to} \quad & h_i(x) = 0, \quad i = 1, ..., N, \\ & g_i(x) \leq 0, \quad i = 1, ..., N, \end{aligned} \tag{1}$$

with $f_i, h_i, g_i : \mathbb{R}^n \to \mathbb{R}$ satisfying the following assumptions.

Assumption 2.1: Functions $f_i, h_i, g_i \in C^2$, $i \in \{1, ..., N\}$ have bounded Hessian and Lipschitz continuous gradients. Moreover, problem (1) has at least one feasible solution. □

Assumption 2.2: Every local minimum of (1) is a regular point and satisfies the second order sufficiency conditions.□

Problem (1) is to be solved in a distributed way by a network of N peer processors without a central coordinator. Each processor has a local memory, a local computation capability and can exchange information with neighboring nodes. Moreover, functions f_i, h_i and g_i are private to node i. The network is described by a fixed, undirected and connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, ..., N\}$ is the set of nodes and $\mathcal{E} \subseteq \{1, ..., N\} \times \{1, ..., N\}$ is the set of edges. We denote by $\mathcal{N}_i = \{j \in \mathcal{V} \mid (i, j) \in \mathcal{E}\} \cup \{i\}$ the set of the neighbors of node i (including i itself) and by $d_i = |\mathcal{N}_i|$ the cardinality of $\mathcal{N}_i$. Also, we denote by d_G the diameter of $\mathcal{G}$.

We consider a generalized version of the asynchronous communication protocol presented in [13]. Each node has its own concept of time defined by a local timer, which triggers when the node has awake, independently of the other nodes. Between two triggering events each node is in *IDLE* mode, i.e., it listens for messages from neighboring nodes and, if needed, updates some local variables. When a trigger occurs, it switches into *AWAKE* mode, performs local computations and broadcasts the updated information to neighbors. Formally, the triggering process is modeled by means of a local clock $\tau_i \in \mathbb{R}_{>0}$ and a certain waiting time T_i. As long as $\tau_i < T_i$ the node is in *IDLE* . When $\tau_i = T_i$ the node switches to the *AWAKE* mode and, after running the local computations, resets $\tau_i = 0$ and selects a new waiting time T_i, according to the following assumption.

Assumption 2.3 (Local timers): For each node i, there exists $\bar{T}_i$ such that $T_i \leq \bar{T}_i$ for each awakening cycle. □

Assumption 2.4 (No simultaneous awakening): Only one node can be awake at each time instant. □

B. Equivalent Formulation and Method of Multipliers

Due to the connectedness of $\mathcal{G}$, problem (1), can be rewritten in the equivalent form

$$\begin{aligned} \underset{x_1,...,x_N}{\text{minimize}} \quad & \sum_{i=1}^N f_i(x_i) \\ \text{subject to} \quad & x_i = x_j, \qquad \forall (i,j) \in \mathcal{E}, \\ & h_i(x_i) = 0, \qquad i \in \mathcal{V}, \\ & g_i(x_i) \leq 0, \qquad i \in \mathcal{V}. \end{aligned} \tag{2}$$

Next we define the Augmented Lagrangian associated to problem (2). Let $\nu_{ij} \in \mathbb{R}^n$ and ρ_{ij} be the multiplier and penalty parameter associated to the equality constraint $x_i = x_j$. Similarly, let λ_i and ρ_{I_i}, respectively μ_i and ρ_{E_i}, be the multiplier and penalty parameter associated to the i-th inequality, respectively equality, constraint. Moreover, let $\mathbf{x} = [x_1^\top, ..., x_N^\top]^\top$, denote by $\mathbf{P}$ the vector stacking all the penalty parameters, ν, λ and μ the vectors stacking the corresponding multipliers, and, consistently let $\mathbf{\Lambda} = [\nu^\top, \lambda^\top, \mu^\top]^\top$. Thus, the Augmented Lagrangian associated to (2) is

$$\begin{aligned} \mathcal{L}_{\mathbf{P}}(\mathbf{x}, \mathbf{\Lambda}) = \sum_{i=1}^N \Bigg\{ & f_i(x_i) \\ & + \sum_{j\in\mathcal{N}_i\setminus i} \left[\nu_{ij}^\top(x_i - x_j) + \frac{\rho_{ij}}{2}\|x_i - x_j\|^2\right] + \\ & + \lambda_i h_i(x_i) + \frac{\rho_{E_i}}{2}\|h_i(x_i)\|^2 + \\ & + \frac{1}{2\rho_{I_i}}\left(\max\{0,\, \mu_i + \rho_{I_i} g_i(x_i)\}^2 - \mu_i^2\right) \Bigg\}. \end{aligned} \tag{3}$$

A powerful method to solve problem (2) is the well known Method of Multipliers, which consists of the following steps (see e.g. [15], [16]),

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \mathcal{L}_{\mathbf{P}^k}(\mathbf{x}, \mathbf{\Lambda}^k) \tag{4}$$

$$\nu_{ij}^{k+1} = \nu_{ij}^k + \rho_{ij}^k(x_i^{k+1} - x_j^{k+1}), \qquad \forall (i,i) \in \mathcal{E}, \tag{5}$$

$$\lambda_i^{k+1} = \lambda_i^k + \rho_{E_i}^k h_i(x_i^{k+1}), \qquad i \in \mathcal{V}, \tag{6}$$

$$\mu_i^{k+1} = \max\{0,\, \mu_i^k + \rho_{I_i}^k g_i(x_i^{k+1})\}, \qquad i \in \mathcal{V}, \tag{7}$$

where $\mathbf{P}^{k+1} \geq \mathbf{P}^k \geq ... \geq \mathbf{P}^0 > 0$. Sufficient conditions guaranteeing the convergence of this method to a local minimum of problem (2) have been given, e.g., in [16]. One of these conditions involves the regularity of the local minima of the optimization problem. In general, such a condition is not verified in problem (2) due to the constraints $x_i = x_j$. In [17], [18] the results in [16] have been extended to deal with the non regularity of the local minima of problem (2).

We want to stress that the Augmented Lagrangian defined in (3) is not separable in the local decision variables x_i. Thus, the minimization step in (4) cannot be performed by independently minimizing with respect to each variable.

III. Asynchronous Method of Multipliers Distributed Algorithm

In this section, the Asynchronous Method of Multipliers (ASYMM) for solving problem (2) in an asynchronous and

distributed way is presented. We start by introducing a distributed algorithm allowing nodes in an asynchronous network to agree that all of them have set a local flag to 1.

A. Asynchronous distributed logic-AND

Each node in the network is assigned a flag C_i that is initially set to 0 and is then changed to 1 in finite time. We propose an asynchronous distributed logic-AND algorithm, based on the synchronous algorithm proposed in [19], for checking if all the nodes have $C_i = 1$.

Each node i stores a matrix $S_i \in \{0,1\}^{d_G \times d_i}$ which contains information about the status of the node itself and its neighbors. Let $S_i[l, j|_i]$ denote the element in the l-th row and $j|_i$-th column of S_i, where $j|_i$ is the index associated to node j by node i. Then, the elements $S_i[1, j|_i]$ for $j \in \mathcal{N}_i \backslash i$ represent the values of the flags of nodes $j \in \mathcal{N}_i \backslash i$ and $S_i[1, d_i]$ the one of node i itself. This means that $S_i[1, d_i] = C_i$. Moreover, for $l = 2, \dots, d_G$, the element $S_i[l, j|_i]$, $j \in \mathcal{N}_i \setminus i$, represents the status of the $(l-1)$-th row of S_j defined as the product of all its entries. Similarly, $S_i[l, d_i]$ represents the status of the $(l-1)$-th row of S_i and can be computed as

$$S_i[l, d_i] = \prod_{b=1}^{d_i} S_i[l-1, b]. \tag{8}$$

Hence, $S_i[l, d_i] = 1$ if and only if $S_i[l-1, j|_i] = 1 \ \forall j \in \mathcal{N}_i$. A pseudo code of the distributed logic-AND algorithm is reported in Algorithm 1. Notice, in particular, that node i has to broadcast to all its neighbors only the last column of S_i, i.e. $S_i[:, d_i]$. Moreover it stores only the d_j-th column of the matrices S_j of its neighbors $j \in \mathcal{N}_i \backslash i$.

Algorithm 1 Asynchronous distributed logic-AND

Initialization: $C_i \leftarrow 0$, $S_i \leftarrow \mathbf{0}_{d_G \times d_i}$

AWAKE

if $\prod_{l=1}^{d_i} S_i[d_G, l] \neq 1$ **then**
  $S_i[1, d_i] \leftarrow C_i$
  $S_i[l, d_i] \leftarrow \prod_{b=1}^{d_i} S_i[l-1, b]$ for $l = 2, ..., d_G$
  BROADCAST $S_i[:, d_i]$ to all $j \in \mathcal{N}_i \backslash i$

if $\prod_{l=1}^{d_i} S_i[d_G, l] = 1$ **then**
  STOP and send **STOP** signal to all $j \in \mathcal{N}_i \backslash i$

IDLE

if $S_j[:, d_j]$ received from $j \in \mathcal{N}_i \backslash i$ and not received a **STOP** signal **then**
  $S_i[l, j|_i] \leftarrow S_j[l, d_j]$ for $l = 1, ..., d_G$

if STOP received, set $S_i[d_G, :] \leftarrow 1$

It can be seen that a node will stop only when the last row of its stopping matrix is composed by all 1s, i.e. when

$$\prod_{b=1}^{d_i} S_i[d_G, b] = 1. \tag{9}$$

The following result states that (9) can be satisfied at some node if and only if $C_i = 1$ for all i.

Proposition 3.1: Let Assumption 2.3 holds. Suppose that when a flag C_i switches from 0 to 1, it remains equal to 1 indefinitely. Then, a node i satisfies $\prod_{b=1}^{d_i} S_i[d_G, b] = 1$ in finite time if and only if at a certain time instant one has $C_j = 1$ for all $j \in \mathcal{V}$. □

B. ASYMM Algorithm

We now present the proposed ASYMM algorithm for solving problem (2) in an asynchronous and distributed way.

Before describing the algorithm, we need to introduce a "local Augmented Lagrangian", whose minimization with respect to the decision variable x_i is equivalent to the minimization of the entire Augmented Lagrangian (3). To this end, define $x_{\mathcal{N}_i} = [x_j]_{j \in \mathcal{N}_i}$, $\mathbf{\Lambda}_{\mathcal{N}_i} = [\lambda_i, \mu_i, [\nu_{ij}]_{j \in \mathcal{N}_i \backslash i}, [\nu_{ji}]_{j \in \mathcal{N}_i \backslash i}]$, and $\mathbf{P}_{\mathcal{N}_i} = [\rho_{I_i}, \rho_{E_i}, [\rho_{ij}]_{j \in \mathcal{N}_i \backslash i}, [\rho_{ji}]_{j \in \mathcal{N}_i \backslash i}]$. Then we can introduce the *local Augmented Lagrangian*

$$\begin{aligned}
&\tilde{\mathcal{L}}_{\mathbf{P}_{\mathcal{N}_i}}(x_{\mathcal{N}_i}, \mathbf{\Lambda}_{\mathcal{N}_i}) = \\
&\quad = f_i(x_i) + \\
&\quad + \sum_{j \in \mathcal{N}_i \backslash i} \left[x_i^\top (\nu_{ij} - \nu_{ji}) + \frac{\rho_{ij} + \rho_{ji}}{2} \|x_i - x_j\|^2 \right] + \\
&\quad + \lambda_i h_i(x_i) + \frac{\rho_{E_i}}{2} \|h_i(x_i)\|^2 + \\
&\quad + \frac{1}{2\rho_{I_i}} \left(\max\{0, \ \mu_i + \rho_{I_i} g_i(x_i)\}^2 - \mu_i^2 \right).
\end{aligned} \tag{10}$$

It can be easily verified that, for fixed values of x_j, $j \neq i$, minimizing (3) with respect to x_i is equivalent to minimizing (10) with respect to x_i, i.e.

$$\arg\min_{x_i} \mathcal{L}_{\mathbf{P}}(\mathbf{x}, \mathbf{\Lambda}) = \arg\min_{x_i} \tilde{\mathcal{L}}_{\mathbf{P}_{\mathcal{N}_i}}(x_{\mathcal{N}_i}, \mathbf{\Lambda}_{\mathcal{N}_i}).$$

Moreover, the gradients with respect to x_i are equal, i.e.

$$\nabla_{x_i} \mathcal{L}_{\mathbf{P}}(\mathbf{x}, \mathbf{\Lambda}) = \nabla_{x_i} \tilde{\mathcal{L}}_{\mathbf{P}_{\mathcal{N}_i}}(x_{\mathcal{N}_i}, \mathbf{\Lambda}_{\mathcal{N}_i}). \tag{11}$$

We are now ready to introduce ASYMM (which we report in Algorithm 2), whose rationale is the following. When a node wakes up, it performs a gradient descent step on its local Augmented Lagrangian until every node has reached a suitable accuracy. This check is performed by nodes themselves in a distributed way. When a node gets aware of this condition, it performs (only) one ascent step on its local multiplier vector. Then, it gets back to the primal update when it has received the updated multipliers from all its neighbors.

More formally, when node i wakes up, it checks through a flag, called M_{done}, if its multiplier vector and the neighboring ones are up to date. If this is the case (which corresponds to $M_{done} = 0$), it performs one of the following two tasks:

T1. If $\prod_{l=1}^{d_i} S_i[d_G, l] \neq 1$, it performs a gradient descent step on its local Augmented Lagrangian and checks if the local tolerance on the gradient has been reached (if the latter is true this corresponds to setting $C_i \leftarrow 1$ in the distributed logic-AND). Then it updates matrix S_i, and broadcasts the updated x_i and $S_i[:, d_i]$ to its neighbors.

T2. If $\prod_{l=1}^{d_i} S_i[d_G, l] = 1$, it performs an ascent step on the local multiplier vector and updates the local penalty parameters, then sets $M_{done} = 1$ and broadcasts the updated multipliers and penalty parameters ν_{ij} and ρ_{ij} (associated to constraints $x_i = x_j$) to its neighbors.

When in *IDLE* , node i continuously listens for messages from its neighbors, but does not broadcast any information. Received messages may contain either the local optimization variable and variables for the logic-AND, or multiplier vectors and penalty parameters. If necessary, node i suitably updates local variables of the logic-AND or the flag M_{done}. We note that, for node i, sending a new multiplier ν_{ij} or receiving a new ν_{ji} corresponds to sending or receiving a **STOP** signal in the asynchronous logic-AND. Finally, regarding the rule used for updating the penalty parameters, the heuristics presented, e.g., in [16, Chapters 2 and 3] can be used.

Algorithm 2 ASYMM

Initialization: Initialize x_i, $\mathbf{\Lambda}_i$, $\boldsymbol{\nu}_i$, $\mathbf{P}_i$, $\boldsymbol{\rho}_i$, $S_i = \mathbf{0}_{d_G \times d_i}$, M_{done} = 0.

AWAKE

if $\prod_{l=1}^{d_i} S_i[d_G, l] \neq 1$ **and not** M_{done} **then**

$x_i \leftarrow x_i - \frac{1}{L_i} \nabla_{x_i} \tilde{\mathcal{L}}_{\mathbf{P}_{\mathcal{N}_i}}(x_{\mathcal{N}_i}, \mathbf{\Lambda}_{\mathcal{N}_i})$

if $\|\nabla_{x_i} \tilde{\mathcal{L}}_{\mathbf{P}_{\mathcal{N}_i}}(x_{\mathcal{N}_i}, \mathbf{\Lambda}_{\mathcal{N}_i})\| \leq \epsilon_i$ **then** $S_i[1, d_i] \leftarrow 1$

$S_i[l, d_i] \leftarrow \prod_{b=1}^{d_i} S_i[l-1, b]$ for $l = 2, ..., d_G$

BROADCAST x_i, $S_i[:, d_i]$ to all $j \in \mathcal{N}_i \backslash i$

if $\prod_{l=1}^{d_i} S_i[d_G, l] = 1$ **and not** M_{done} **then**

$\nu_{ij} \leftarrow \nu_{ij} + \rho_{ij}(x_i - x_j)$ for $j \in \mathcal{N}_i \backslash i$

$\lambda_i \leftarrow \lambda_i + \rho_{E_i} h_i(x_i)$

$\mu_i \leftarrow \max\{0,\, \mu_i + \rho_{I_i} g_i(x_i)\}$

update ρ_{E_i}, ρ_{I_i}, ρ_{ij} for all $j \in \mathcal{N}_i \backslash i$

$M_{done} \leftarrow 1$

BROADCAST ν_{ij}, ρ_{ij} to $j \in \mathcal{N}_i \backslash i$

IDLE

if $S_j[:, d_j]$ received from $j \in \mathcal{N}_i \backslash i$ and not already received some new ν_{ji} **then** $S_i[l, j|_i] \leftarrow S_j[l, d_j]$ for $l = 1, ..., d_G$

if ν_{ji} and ρ_{ji} received from $j \in \mathcal{N}_i \backslash i$ set $S_i[d_G, :] \leftarrow 1$

if x_j^{new} received from $j \in \mathcal{N}_i \backslash i$, update $x_j \leftarrow x_j^{new}$

if M_{done} **and** ν_{ji} received from all $j \in \mathcal{N}_i \backslash i$ **then**

$M_{done} \leftarrow 0$, $S_i \leftarrow \mathbf{0}_{d_G \times d_i}$, update ϵ_i

IV. ASYMM Convergence Analysis

In order to analyze ASYMM, we start by noting that under Assumption 2.3, from a global perspective, the local asynchronous updates result into an algorithmic evolution in which, at each iteration, only one node wakes up in an essentially cyclic fashion[1]. Hence, we can associate to each triggering an iteration of the distributed algorithm. We denote by $t \in \mathbb{N}$ a discrete, *universal* time indicating the t-th iteration of the algorithm and define as $i_t \in \mathcal{V}$ the index of the node triggered at iteration t.

In the following we show that: (i) there is an *equivalence* relationship between ASYMM and an inexact Method of Multipliers and (ii) the *convergence* of ASYMM to a local minimum of problem (1) is guaranteed under suitable conditions inherited from the (centralized) optimization literature.

A. Equivalence with an inexact Method of Multipliers

We consider an inexact Method of Multipliers which consists of solving the k-th instance of the Augmented Lagrangian minimization by means of a block-coordinate gradient descent algorithm (see, e.g., [20] for a survey), which runs for a certain number of iterations h^k. A pseudo code of this inexact Method of Multipliers is given in the following table, where i_h is the index of the block chosen at iteration h and the penalty parameters are updated as in [16].

Algorithm 3 Inexact MM

for $k = 0, 1, ...$ **do**

$\hat{\mathbf{x}}^0 = \mathbf{x}^k$

for $h = 1, ..., h^k$ **do**

$\hat{\mathbf{x}}^{h+1} = \hat{\mathbf{x}}^h - \frac{1}{L_{i_h}} U_{i_h} \nabla_{x_{i_h}} \mathcal{L}_{\mathbf{P}}(\hat{\mathbf{x}}^h, \mathbf{\Lambda}^k)$

$\mathbf{x}^{k+1} = \hat{\mathbf{x}}^{h^k+1}$

$\nu_{ij}^{k+1} = \nu_{ij}^k + \rho_{ij}^k (x_i^{k+1} - x_j^{k+1})$, $\forall (i,i) \in \mathcal{E}$

$\lambda_i^{k+1} = \lambda_i^k + \rho_{E_i}^k h_i(x_i^{k+1})$, $i \in \mathcal{V}$

$\mu_i^{k+1} = \max\{0,\, \mu_i^k + \rho_{I_i}^k g_i(x_i^{k+1})\}$, $i \in \mathcal{V}$

We would like to clarify that the ordered sequence of indexes h and k used in Algorithm 3 does not coincide with the sequence in the universal time t of ASYMM algorithm. We will rather show that a (possibly reordered) subsequence of iterations (in the universal time t) of ASYMM will give rise to the h and k sequences in Algorithm 3.

Let $t_1, t_2, ...$ be a subsequence of $\{t\}$ such that at each t_ℓ a multiplier update (task T2) has been performed by node i_{t_ℓ} and let t_1 be the time instant of the first multiplier update. Then, the following result holds.

Lemma 4.1: Each sequence $(i_{t_{kN+1}}, ..., i_{t_{(k+1)N}})$, for $k = 0, 1, ...$, is a permutation of $\{1, ..., N\}$. Moreover, if $\epsilon_i > 0$ $\forall i \in \mathcal{V}$, multiplier updates occur infinitely many times. □

Define $\mathbf{\Lambda}_i = [\lambda_i, \mu_i, [\nu_{ij}]_{j \in \mathcal{N}_i \backslash i}]$ and let $\tilde{x}_i^t$ and $\tilde{\Lambda}_i^t$ be the value of the state vector and of the multiplier vector at node i, at iteration t, computed according to ASYMM. Then, the following Corollary follows immediately from Lemma 4.1.

Corollary 4.2: For all $\tau \in \{t_{kN+1}, ..., t_{(k+1)N}\}$, $k = 0, 1, ...$, it holds $\tilde{\mathbf{\Lambda}}_{i_\tau}^t = \tilde{\mathbf{\Lambda}}_{i_\tau}^\tau$ $\forall t \in \{\tau, \tau+1, \ldots, t_{(k+1)N}\}$. □

For all $\tau \in \{t_{kN+1}, ..., t_{(k+1)N}\}$, $k = 0, 1, ...$, define

$$x_{i_\tau}^{k+1} = \tilde{x}_{i_\tau}^\tau, \quad \mathbf{\Lambda}_{i_\tau}^{k+1} = \tilde{\mathbf{\Lambda}}_{i_\tau}^\tau.$$

[1]Indexes in $\{1, \ldots, N\}$ are drawn according to an *essentially cyclic* rule if there exists $M \geq N$ such that every $i \in \{1, \ldots, N\}$ is drawn at least once every M extractions.

By using Lemma 4.1 and reordering the indexes i_τ, one can define

$$\mathbf{x}^{k+1} = \left[\left(x_1^{k+1}\right)^\top, ..., \left(x_N^{k+1}\right)^\top\right]^\top,$$
$$\mathbf{\Lambda}^{k+1} = \left[\left(\mathbf{\Lambda}_1^{k+1}\right)^\top, ..., \left(\mathbf{\Lambda}_N^{k+1}\right)^\top\right]^\top.$$

The next two lemmas show that a local primal (resp. multiplier) update is performed according to a common multiplier (resp. primal) variable.

Lemma 4.3: For all $\tau \in \{t_{kN+1}, ..., t_{(k+1)N}\}$, $k = 0,1, ...$, every multiplier update $\mathbf{\Lambda}_{i_\tau}^{k+1}$ is performed using $\mathbf{x}^{k+1}$. □

Lemma 4.4: Let τ_i^k be the time instant in which x_i^{k+1} is computed, i.e., $\tau_i^k \in \{t_{kN+1}, ..., t_{(k+1)N}\}$ such that node i is awake at time τ_i^k. Then, for all $t \in \{\tau_i^k, \tau_i^k + 1, \ldots, \tau_i^{k+1}\}$, every descent step on the Augmented Lagrangian with respect to x_i is performed using the multiplier vector $\mathbf{\Lambda}^{k+1}$. □

Next lemma states that every node performs at least one primal update between the beginning of two consecutive cycles of multiplier updates.

Lemma 4.5: Between t_{kN+1} and $t_{(k+1)N+1}$ every node performs task $T1$ at least once. □

The equivalence of ASYMM and Algorithm 3 is stated in the next theorem, whose proof relies on the previous Lemmas.

Theorem 4.6: Let Assumptions 2.1, 2.2, 2.3 and 2.4 hold. Then, ASYMM is equivalent to an instance of Algorithm 3 in which the selection of nodes i_h satisfies an essentially cyclic rule. Moreover, if in Algorithm 2, $\epsilon_i > 0\ \forall i \in \mathcal{V}$, the total number of primal descent steps h^k is finite. □

B. Local Convergence

Let us first introduce a result that allows us to bound the norm of the gradient of a strongly convex function with block component-wise Lipschitz continuous gradient, during the evolution of a block coordinate descent algorithm. Specifically, we relate this bound to given local tolerances ϵ_i on the norm of the gradient with respect to block i.

Lemma 4.7: Let $\Phi(y_1, \ldots, y_N)$ be a σ-strongly convex function with block component-wise Lipschitz continuous gradients (with L_i being the Lipschitz constant with respect to block y_i) in a subset $Y \subseteq \mathbb{R}^n$. Let $\{\mathbf{y}^h\}$ be a sequence generated starting according to $\mathbf{y}^{h+1} = \mathbf{y}^h - \frac{1}{L_{i_h}} U_{i_h} \nabla_{y_{i_h}} \Phi(\mathbf{y}^h)$, where $\mathbf{y}^0 \in Y$ and indexes $i_h \in \{1, \ldots, N\}$ are drawn in an essentially cyclic way. If, for some $\bar{h} > 0$, $\|\nabla_{y_i} \Phi(\mathbf{y}^{\bar{h}})\| \leq \epsilon_i,\ \forall i \in \{1, \ldots, N\}$, then

$$\|\nabla_{\mathbf{y}} \Phi(\mathbf{y}^h)\| \leq \sqrt{\sum_{i=1}^{N} \left(\frac{L_i \epsilon_i}{\sigma}\right)^2}$$

for all $h \geq \bar{h}$. □

In order to show the local convergence, we need an additional assumption.

Assumption 4.8: There exists some $\bar{k} > 0$, such that for all $k \geq \bar{k}$, the sequence $\mathbf{x}^k$ generated by ASYMM belongs to a σ^k-strongly convex neighborhood of a local minimum of $\mathcal{L}_{\mathbf{P}^k}(\mathbf{x}, \mathbf{\Lambda}^k)$. □

Assumption 4.8 is indeed strong, but it is somehow standard in the optimization literature. As pointed out, e.g., in [16], while it is not possible to guarantee that such an assumption holds a priori, in practice it turns out to be satisfied after a sufficient number of iterations of the primal minimization and multiplier/penalties update. The next Theorem, whose proof relies on Lemma 4.7, shows that ASYMM guarantees $\|\nabla_{\mathbf{x}} \mathcal{L}_{\mathbf{P}^k}(\mathbf{x}^{k+1}, \mathbf{\Lambda}^k)\| \leq \varepsilon^k$, with ε^k depending on the local thresholds ϵ_i.

Theorem 4.9: Let Assumptions 2.1, 2.2, 2.3, 2.4 and 4.8 hold. Then, there exists $\bar{k} > 0$, such that for all $k \geq \bar{k}$, it holds $\|\nabla_{\mathbf{x}} \mathcal{L}_{\mathbf{P}^k}(\mathbf{x}^{k+1}, \mathbf{\Lambda}^k)\| \leq \varepsilon^k$ with

$$\varepsilon^k = \sqrt{\sum_{i=1}^{N} \left(\frac{L_i^k \epsilon_i^k}{\sigma^k}\right)^2},$$

where ϵ_i^k is the local tolerance set by node i for the primal descent related to multiplier $\mathbf{\Lambda}^k$ and L_i^k is the Lipschitz constant of $\nabla_{\mathbf{x}} \mathcal{L}_{\mathbf{P}^k}(\mathbf{x}, \mathbf{\Lambda}^k)$ with respect to x_i. □

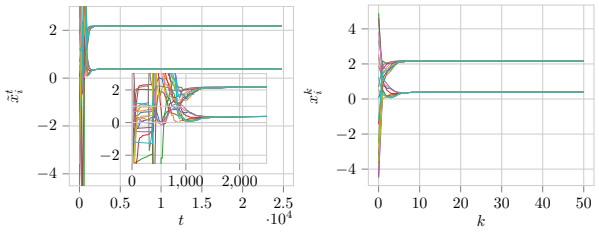
We want to remark that the only global parameter σ^k appearing in our analysis does not need to be known by the nodes, because it is not required for the execution of ASYMM.

We stress that ASYMM is equivalent to Algorithm 3 even without Assumption 4.8, which is needed to guarantee the local convergence to a (strict) local minimum of problem (1). In fact, for $k < \bar{k}$ in Assumption 4.8 the Augmented Lagrangian can be nonconvex, thus Lemma 4.7 cannot be invoked. However, the block coordinate descent algorithm is guaranteed to converge (at least) to a stationary point as shown in [21]. This means that multiplier updates in ASYMM will surely occur after a finite number of primal minimizations. Moreover, as $\mathbf{P}^k$ grows, the Augmented Lagrangian typically becomes locally strongly convex, see, e.g., [16]. If this happens, the block coordinate descent algorithm approaches the corresponding minimum of the Augmented Lagrangian and, provided that for $k \geq \bar{k}$ the local tolerances ϵ_i^k vanish as $k \to \infty$, the minimization of the Augmented Lagrangian will be asymptotically exact. This guarantees a convergence result for ASYMM in the same sense as the one in the centralized case: the convergence of the algorithm is contingent upon the generation of (possibly local) minima of the Augmented Lagrangian that, after some index $\hat{k}$, stay in the neighborhood of the same local minimum $x^\star$ of problem (2). As reported in [16], extensive numerical experience has shown that, from a practical point of view, choosing the obtained $\mathbf{x}^k$ as the initial condition for the $(k+1)$-th minimization usually generates sequences $\{\mathbf{x}^k\}$ within a neighborhood of the same local minimum $\mathbf{x}^\star$.

V. Numerical Results

Consider a network of N sensors, deployed over a certain region, communicating according to a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, which have to solve the optimization problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{i=1}^{N} f_i(x) \\ \text{subject to} \quad & \|x - c_i\| - R_i \leq 0, \quad i = 1, ..., N \\ & r_i - \|x - c_i\| \leq 0, \quad i = 1, ..., N, \end{aligned}$$



(a) (b)

Fig. 1: Evolution of the local decision variables x_i.

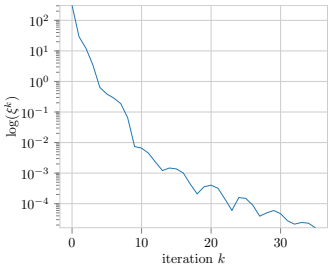


Fig. 2: Logarithm of the measure of infeasibility ξ^k.

which can be rewritten in the form of problem (2).

Such a problem naturally arises, for example, in the context of source localization under the assumption of unknown but bounded (UBB) noise, in which each agent knows its own absolute location c_i and takes a noisy measurement y_i of its own distance from an an emitting source located at an unknown location $x^\star$ as $y_i = \|x^\star - c_i\| + w_i$, with $|w_i| \leq \kappa_i$ for some $\kappa_i \geq 0$.

Suppose $f_i(x_i) = x_i^\top x_i$ for all $i \in \mathcal{V}$. We report a simulation with $N = 10$ nodes and $n = 2$, in which $x^\star \in U[-2.5, 2.5]^n$, $c_i \in U[-2.5, 2.5]^n$ and $\kappa_i = U[0, 0.3]$ for all $i \in \mathcal{V}$, where $U[a, b]$ denotes the uniform distribution in the interval $[a, b]$. The graph is modeled through a connected Watts-Strogatz model in which nodes have mean degree $K = 2$. Let us define the measure of infeasibility at iteration k as $\xi^k = \sum_{i=1}^N [\max(0, \|x_i^k - c_i\| - R_i) + \max(0, r_i - \|x_i^k - c_i\|) + \sum_{j \in \mathcal{N}_i \setminus i} \|x_i^k - x_j^k\|]$. We run ASYMM for 25000 iterations. In Fig. 1a the values $\tilde{x}_i^t$ are shown for each $t = 1, ..., 25000$ and each $i \in \mathcal{V}$. A magnification of the first 2500 iterations is reported as a subplot. Fig. 1b shows the evolution of x_i^k for each $i \in \mathcal{V}$. As it can be seen, the nodes performed 50 multiplier updates along the 25000 iterations. Finally, in Fig. 2 the values of ξ^k are reported.

VI. Conclusions

In this paper we proposed ASYMM, an asynchronous distributed algorithm for a class of constrained optimization problems in which both local cost functions and constraints can be nonconvex. ASYMM has been proved to be equivalent to an inexact Method of Multipliers from which it inherits all the convergence properties.

References

[1] P. Bianchi and J. Jakubowicz, "Convergence of a multi-agent projected stochastic gradient algorithm for non-convex optimization," *IEEE Transactions on Automatic Control*, vol. 58, no. 2, pp. 391–405, 2013.

[2] H.-T. Wai, A. Scaglione, J. Lafond, and E. Moulines, "A projection-free decentralized algorithm for non-convex optimization," in *Signal and Information Processing (GlobalSIP), 2016 IEEE Global Conference on*. IEEE, 2016, pp. 475–479.

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

[4] Y. Sun, G. Scutari, and D. Palomar, "Distributed nonconvex multiagent optimization over time-varying networks," in *Signals, Systems and Computers, 2016 50th Asilomar Conference on*. IEEE, 2016, pp. 788–794.

[5] T. Tatarenko and B. Touri, "Non-convex distributed optimization," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3744–3757, 2017.

[6] S. Lee and A. Nedic, "Distributed random projection algorithm for convex optimization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 2, pp. 221–229, 2013.

[7] K. Margellos, A. Falsone, S. Garatti, and M. Prandini, "Distributed constrained optimization and consensus in uncertain networks via proximal minimization," *IEEE Transactions on Automatic Control*, vol. PP, no. 99, pp. 1–1, 2017.

[8] I. Necoara, "Random coordinate descent algorithms for multi-agent convex optimization over networks," *IEEE Transactions on Automatic Control*, vol. 58, no. 8, pp. 2001–2012, 2013.

[9] E. Wei and A. Ozdaglar, "On the $O(1{=}k)$ convergence of asynchronous distributed alternating direction method of multipliers," in *Global conference on signal and information processing (GlobalSIP), 2013 IEEE*. IEEE, 2013, pp. 551–554.

[10] 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.

[11] P. Bianchi, W. Hachem, and I. Franck, "A stochastic coordinate descent primal-dual algorithm and applications," in *Machine Learning for Signal Processing (MLSP), 2014 IEEE International Workshop on*. IEEE, 2014, pp. 1–6.

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

[13] I. Notarnicola and G. Notarstefano, "Asynchronous distributed optimization via randomized dual proximal gradient," *IEEE Transactions on Automatic Control*, vol. 62, no. 5, pp. 2095–2106, 2017.

[14] ——, "A randomized primal distributed algorithm for partitioned and big-data non-convex optimization," in *Decision and Control (CDC), 2016 IEEE 55th Conference on*. IEEE, 2016, pp. 153–158.

[15] R. T. Rockafellar, "Augmented lagrange multiplier functions and duality in nonconvex programming," *SIAM Journal on Control*, vol. 12, no. 2, pp. 268–285, 1974.

[16] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[17] I. Matei, J. S. Baras, M. Nabi, and T. Kurtoglu, "An extension of the method of multipliers for distributed nonlinear programming," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*. IEEE, 2014, pp. 6951–6956.

[18] I. Matei and J. S. Baras, "Nonlinear programming methods for distributed optimization," *arXiv preprint arXiv:1707.04598*, 2017.

[19] T. Ayken and J.-I. Imura, "Diffusion based stopping criterion for event-triggered distributed optimization," *SICE Journal of Control, Measurement, and System Integration*, vol. 8, no. 6, pp. 371–379, 2015.

[20] S. J. Wright, "Coordinate descent algorithms," *Mathematical Programming*, vol. 151, no. 1, pp. 3–34, 2015.

[21] Y. Xu and W. Yin, "A globally convergent algorithm for nonconvex optimization based on block coordinate update," *Journal of Scientific Computing*, pp. 1–35, 2017.