A centre–free approach for resource allocation with lower bounds

Germán Obando¹,², Nicanor Quijano¹ and Naly Rakoto–Ravalontsalama²

¹Departamento de Ingeniería Eléctrica y Electrónica, Universidad de los Andes, Colombia; ²Ecole des Mines de Nantes, and IRCCyN, 44307 Nantes Cedex 03, France

ABSTRACT
Since complexity and scale of systems are continuously increasing, there is a growing interest in developing distributed algorithms that are capable to address information constraints, specially for solving optimisation and decision-making problems. In this paper, we propose a novel method to solve distributed resource allocation problems that include lower bound constraints. The optimisation process is carried out by a set of agents that use a communication network to coordinate their decisions. Convergence and optimality of the method are guaranteed under some mild assumptions related to the convexity of the problem and the connectivity of the underlying graph. Finally, we compare our approach with other techniques reported in the literature, and we present some engineering applications.

1. Introduction
The increasing scale and complexity of systems have motivated the development of distributed methods to deal with situations where optimisation and decision-making are required. An important issue within this field is the optimal resource allocation over networks of agents, a problem that is closely related to network utility maximisation (NUM) problems (Palomar & Chiang, 2006; Tan, Zhu, Ge, & Xiong, 2015). Resource allocation arises when there is a limited amount of a certain resource (e.g. electric power, computing capacity or execution time), and it is necessary to establish an optimal distribution policy between some entities (e.g. loads, processors or controllers) that are connected by a communication network. This kind of problems has a large number of applications in economics (Ayesta, Erasquin, Ferreira, & Jacko, 2016; Conrad, 2010), smart energy systems (Hansen, Roche, Suryanarayanan, Maciejewski, & Siegel, 2015; Pantoja & Quijano, 2012), cloud computing (Pietrabissa et al., 2016; Pillai & Rao, 2016), and communications (H. Lee, K.J. Lee, Kim, Clerckx, & I. Lee, 2016; Tan et al., 2015).

Although there exists an extensive literature regarding distributed methods for solving resource allocation problems, this field still attracts considerable research attention (Cherukuri & Cortés, 2015; Obando, Pantoja, & Quijano, 2014; Pantoja, Quijano, & Passino, 2014; Poveda & Quijano, 2015; Ramirez-Llanos & Martinez, 2015; Tan, Yang, & Xu, 2013). Most of the solution methods are based on multi-agent systems (e.g. a survey that deals with the general class of NUM problems can be found in Palomar & Chiang, 2006), where the agents make decisions based on local information in order to obtain a desirable global behaviour. Appropriate coordination of agents is crucial because it avoids converging to sub-optimal solutions. In order to ensure this coordination, a large number of methods require either the inclusion of a centralised agent or the use of restrictive information structures (as it is pointed out in Mosk-Aoyama, Roughgarden, & Shah, 2010). For instance, in classic decomposition techniques (Bemporad, Heemels, & Johansson, 2010; Boyd, Parikh, Chu, Pelleato, & Eckstein, 2010; Palomar & Chiang, 2006), the Lagrange multiplier related to the ‘price’ of the resource is centrally adjusted to reach the optimum. By contrast, other methods are fully decentralised (e.g. Barreiro-Gomez, Obando, & Quijano, 2016; Xiao & Boyd, 2006; Zhu & Martinez, 2012). These methods exploit the communication capabilities of the agents to coordinate their decisions based on the information received from their neighbours. Fully decentralised methodologies have important advantages, among which we highlight the increase of the autonomy and resilience of the whole system since the dependence on a central authority is avoided.

In this paper, we propose a distributed resource allocation algorithm that does not require a central coordinator. An important characteristic of our method is the capability of handling lower bounds on the decision variables. This feature is crucial in a large number of
practical applications, e.g. in Conrad (2010), Pantoja and Quijano (2012), and Lee et al. (2016), where it is required to capture the non-negativity of the resource allocated to each entity. We use a Lyapunov-based analysis in order to prove that the proposed algorithm asymptotically converges to the optimal solution under some mild assumptions related to the convexity of the cost function, and the connectivity of the graph that represents the communication topology. In order to illustrate our theoretical results, we perform some simulations and compare our method with other techniques reported in the literature. Finally, we present two engineering applications of the proposed algorithm. The first one seeks to improve the energy efficiency in large-scale air-conditioning systems. The second one is related to the distributed computation of the Euclidean projection onto a given set.

Our approach is based on a continuous time version of the centre-free algorithm presented in Xiao and Boyd (2006). The key difference is that the method in Xiao and Boyd (2006) does not allow the explicit inclusion of lower bounds on the decision variables, unless they are added by means of barrier functions (either logarithmic or exact; Cherukuri & Cortés, 2015). The problem of using barrier functions is that they can adversely affect the convergence time (in the case of using exact barrier functions) and the accuracy of the solution (in the case of using classic logarithmic barrier functions), especially for large-scale problems (Jensen & Bard, 2003). There are other methods that consider lower bound constraints in the problem formulation. For instance, Dominguez-Garcia, Cady, and Hadjicostis (2012) and Tan et al. (2013) have developed a decentralised technique based on broadcasting and consensus to optimally distribute a resource considering capacity constraints on each entity in the network. Nonetheless, compared to our algorithm, the approach in Dominguez-Garcia et al. (2012) and Tan et al. (2013) is only applicable to quadratic cost functions. On the other hand, Pantoja and Quijano (2012) propose a novel methodology based on population dynamics. The main drawback of this technique is that its performance is seriously degraded when the number of communication links decreases. We point out the fact that other distributed optimisation algorithms can be applied to solve resource allocation problems, as those presented in Nedic, Ozdaglar, and Parrilo (2010), Yi, Hong, and Liu (2015), and Johansson and Johansson (2009). Nevertheless, the underlying idea in these methods is different from the one used in our work, i.e. Nedic et al. (2010), Yi et al. (2015), and Johansson and Johansson (2009) use consensus steps to refine an estimation of the system state, while in our approach, consensus is used to equalise a quantity that depends on both the marginal cost perceived by each agent in the network and the Karush–Kuhn–Tucker (KKT) multiplier related to the corresponding resource’s lower bound. In this regard, it is worth noting that the method studied in this paper requires less computational capability than the methods mentioned above. Finally, there are other techniques based on game theory and mechanism design (Kakhbod & Teneketzis, 2012; Sharma & Teneketzis, 2009) that decompose and solve resource allocation problems. Nonetheless, those techniques need that each agent broadcasts a variable to all the other agents, i.e. a communication topology given by a complete graph is required. In contrast, the method developed in this paper only uses a communication topology given by a connected graph, which generally requires lower infrastructure.

The remainder of this paper is organised as follows. Section 2 shows preliminary concepts related to graph theory. In Section 3, the resource allocation problem is stated. Then, in Section 4, we present our distributed algorithm and the main results on convergence and optimality. A comparison with other techniques reported in the literature is performed in Section 5. In Section 6, we describe two applications of the proposed method: (i) the optimal chiller loading problem in large-scale air-conditioning systems, and (ii) the distributed computation of Euclidean projections. Finally, in Sections 7 and 8, arguments and conclusions of the developed work are presented.

2. Preliminaries

First, we describe the notation used throughout the paper and presents some preliminary results on graph theory that are used in the proofs of our main contributions.

In the multi-agent framework considered in this article, we use a graph to model the communication network that allows the agents to coordinate their decisions. A graph is mathematically represented by the pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, n\}$ is the set of nodes, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges connecting the nodes. $\mathcal{G}$ is also characterised by its adjacency matrix $A = [a_{ij}]$. The adjacency matrix $A$ is an $n \times n$ non-negative matrix that satisfies: $a_{ij} = 1$ if and only if $(i, j) \in \mathcal{E}$, and $a_{ij} = 0$ if and only if $(i, j) \notin \mathcal{E}$. Each node of the graph corresponds to an agent of the multi-agent system, and the edges represent the available communication channels (i.e. $(i, j) \in \mathcal{E}$ if and only if agents $i$ and $j$ can share information). We assume that there is no edges connecting a node with itself, i.e. $a_{ii} = 0$, for all $i \in \mathcal{V}$; and that the communication channels are bidirectional, i.e. $a_{ij} = a_{ji}$. The last assumption implies that $\mathcal{G}$ is undirected. Additionally, we denote by $\mathcal{N}_i = \{ j \in \mathcal{V}: (i, j) \in \mathcal{E}\}$, the set of neighbours...
of node \(i\), i.e. the set of nodes that are able to receive/send information from/to node \(i\).

Let us define the \(n \times n\) matrix \(L(G) = [l_{ij}]\), known as the graph Laplacian of \(G\), as follows:

\[
l_{ij} = \begin{cases} 
  \sum_{j \in V} a_{ij} & \text{if } i = j \\
  -a_{ij} & \text{if } i \neq j.
\end{cases}
\]  

(1)

Properties of \(L(G)\) are related to connectivity characteristics of \(G\) as shown in the following theorem. We remark that a graph \(G\) is said to be connected if there exists a path connecting any pair of nodes.

**Theorem 2.1** (adapted from Godsil & Royle, 2001): An undirected graph \(G\) of order \(n\) is connected if and only if \(\text{rank}(L(G)) = n - 1\).

From Equation (1), it can be verified that \(L(G)\) \(\mathbb{I} = 0\), where \(\mathbb{I} = [1, \ldots, 1]^T, 0 = [0, \ldots, 0]^T\). A consequence of this fact is that \(L(G)\) is a singular matrix. However, we can modify \(L(G)\) to obtain a nonsingular matrix as shown in the following lemma.

**Lemma 2.1**: Let \(L^k(G) \in \mathbb{R}^{(n-1) \times n}\) be the submatrix obtained by removing the \(k\)-th row of the graph Laplacian \(L(G)\), and let \(L^k(G) \in \mathbb{R}^{(n-1) \times (n-1)}\) be the submatrix obtained by removing the \(k\)-th column of \(L^k(G)\). If \(G\) is connected, then \(L^k(G)\) is positive definite. Furthermore, the inverse matrix of \(L^k(G)\) satisfies \((L^k(G))^{-1} \mathbb{I}_k^k = -\mathbb{I}\), where \(\mathbb{I}_k^k\) is the \(k\)-th column of the matrix \(L^k(G)\).

**Proof**: First, notice that \(L(G)\) is a symmetric matrix because \(G\) is an undirected graph. Moreover, notice that according to Equation (1), \(L(G)\) is diagonally dominant with non-negative diagonal entries. The same holds for \(L^k(G)\) since this is a sub-matrix obtained by removing the \(k\)-th row and column of \(L(G)\). Thus, to show that \(L^k(G)\) is positive definite, it is sufficient to prove that \(L^k(G)\) is nonsingular.

According to Theorem 2.1, since \(G\) is connected, \(L(G)\) has exactly \(n-1\) linearly independent columns (resp. rows). Let us show that the \(k\)-th column (resp. row) of \(L(G)\) can be obtained by a linear combination of the other columns (resp. rows), i.e. the \(k\)-th column (resp. row) is not linearly independent of the rest of the columns (resp. rows).

Since \(L(G)\) \(\mathbb{I} = 0\), notice that \(l_{ik} = -\sum_{j \in V, j \neq k} l_{ij}\), for all \(i \in V\), i.e. the \(k\)-th column can be obtained by a linear combination of the rest of the columns. Furthermore, since \(L(G)\) is a symmetric matrix, the same occurs with the \(k\)-th row. Therefore, the submatrix \(L^k(G)\) is nonsingular since its \(n-1\) columns (resp. rows) are linearly independent.

Now, let us prove that \((L^k(G))^{-1} \mathbb{I}_k^k = -\mathbb{I}\). To do so, we use the fact that \((L^k(G))^{-1} L^k(G) = I\), where \(I\) is the identity matrix. Hence, by the definition of matrix multiplication, we have that

\[
\sum_{m=1}^{n-1} \bar{p}_{im} \bar{p}_{mj} = \begin{cases} 
  0 & \text{if } i = j \\
  1 & \text{if } i \neq j.
\end{cases}
\]  

(2)

where \(\bar{p}_{ij}\) and \(\bar{p}_{ik}\) are the elements located in the \(i\)-th row and \(j\)-th column of the matrices \(L(G)\) and \((L^k(G))^{-1}\), respectively. Thus,

\[
\sum_{m=1}^{n-1} \bar{p}_{im} \bar{p}_{mj} = 1, \text{ for } i = 1, \ldots, n. \tag{3}
\]

Let \(\mathbb{I}_k^k\) be the \(m\)-th entry of the vector \(\mathbb{I}_k^k\). Notice that, according to the definition of \(L^k(G)\) and since \(L(G)\) \(\mathbb{I} = 0\),

\[
l_{mi} = -\sum_{j=1, j \neq i}^{n-1} \bar{x}_{mj} - \bar{x}_{mi}. \tag{4a}
\]

Replacing this value in Equation (3), we obtain

\[
\sum_{m=1}^{n-1} \bar{p}_{im} \bar{p}_{mj} = 1, \text{ for } i = 1, \ldots, n. \tag{4b}
\]

According to Equation (2), \(\sum_{m=1}^{n-1} \bar{p}_{im} \bar{p}_{mj} = 0\). This implies that \(\sum_{m=1}^{n-1} \bar{p}_{im} \bar{p}_{mj} = -1\), for all \(i = 1, \ldots, n - 1\). Therefore, \((L^k(G))^{-1} \mathbb{I}_k^k = -\mathbb{I}\).

Theorem 2.1 and Lemma 2.1 will be used in the analysis of the method proposed in this paper.

### 3. Problem statement

In general terms, a resource allocation problem can be formulated as follows (Patriksson, 2008; Patriksson & Strömberg, 2015):

\[
\min_x \phi(x) := \sum_{i=1}^{n} \phi_i(x_i) \tag{4a}
\]

subject to

\[
\sum_{i=1}^{n} x_i = X \tag{4b}
\]

\[
x_i \geq \bar{x}_i, \text{ for all } i = 1, \ldots, n, \tag{4c}
\]

where \(x_i \in \mathbb{R}\) is the resource allocated to the \(i\)-th zone; \(x = [x_1, \ldots, x_n]^T\); \(\phi_i : \mathbb{R} \mapsto \mathbb{R}\) is a strictly convex and differentiable cost function; \(X\) is the available resource; and \(\bar{x}_i\), is the lower bound of \(x_i\), i.e. the minimum amount of resource that has to be allocated in the \(i\)-th zone.

Given the fact that we are interested in distributed algorithms to solve the problem stated in Equation (4),
we consider a multi-agent network, where the $i$th agent is responsible for managing the resource allocated to the $i$th zone. Moreover, we assume that the agents have limited communication capabilities, so they can only share information with their neighbours. This constraint can be represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as it was explained in Section 2.

Avoiding the individual inequality constraints (4c), KKT conditions establish that at the optimal solution $x^* = [x_1^*, \ldots, x_n^*]^T$ of the problem given in Equation (4a–4b), the marginal costs $\phi'_i(x_i^*) = \frac{\partial \phi_i}{\partial x_i}$ must be equal, i.e. $\phi'_i(x_i^*) = \lambda$, for all $i = 1, \ldots, n$, where $\lambda \in \mathbb{R}$. Hence, a valid alternative to solve (4a–4b) is the use of consensus methods. For instance, we can adapt the algorithm presented in Xiao and Boyd (2006), which is described as follows:

$$\dot{x}_i = \sum_{j \in \mathcal{N}_i} \left( \phi'_i(x_j) - \phi'_i(x_i) \right), \text{ for all } i \in \mathcal{V}. \quad (5)$$

This algorithm has two main properties: (i) at equilibrium, $\phi'_i(x_j^*) = \phi'_i(x_i^*)$ if the nodes $i$ and $j$ are connected by a path; (ii) $\sum_{i=1}^n x_i^* = \sum_{i=1}^n x_i(0)$, where $x_i(0)$ is the initial condition of $x_i$. Therefore, if the graph $\mathcal{G}$ is connected and the initial condition is feasible (i.e. $\sum_{i=1}^n x_i(0) = X$), $x$ asymptotically reaches the optimal solution of (4a–4b) under (5). However, the same method cannot be applied to solve (4) (the problem that considers lower bounds in the resource allocated to each zone) since some feasibility issues related with the constraints (4c) arise.

In the following section, we propose a novel method that extends the algorithm in Equation (5) to deal with the individual inequality constraints given in Equation (4c).

4. Centre-free resource allocation algorithm

4.1 Resource allocation among a subset of nodes in a graph

First, we consider the following subproblem: let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph comprised by a subset of active nodes $\mathcal{V}_a$ and a subset of passive nodes $\mathcal{V}_p$, such that $\mathcal{V}_a \cup \mathcal{V}_p = \mathcal{V}$. A certain amount of resource $X$ has to be split among those nodes to minimise the cost function $\phi(x)$ subject to each passive node is allocated with its corresponding lower bound $x_i$. Mathematically, we formulate this subproblem as:

$$\min_{X} \phi(x) \quad \text{subject to } \sum_{i=1}^n x_i = X \quad (6b)$$

$$x_i = x_i^*, \text{ for all } i \in \mathcal{V}_p. \quad (6c)$$

Feasibility of (6) is guaranteed by making the following assumption.

Assumption 4.1: At least one node is active, i.e. $\mathcal{V}_a \neq \emptyset$.

According to KKT conditions, the active nodes have to equalise their marginal costs at the optimal solution. Therefore, a consensus among the active nodes is required to solve (6). Nonetheless, classic consensus algorithms, as the one given in Equation (5), cannot be used directly. For instance, if all the nodes of $\mathcal{G}$ apply (5) and $\mathcal{G}$ is connected, the marginal costs of both passive and active nodes are driven to be equal in steady state. This implies that the resource allocated to passive nodes can violate the constraint (6c). Besides, if the resource allocated to passive nodes is forced to satisfy (6c) by setting $x_i^* = \bar{x}_i$, for all $i \in \mathcal{V}_p$, there is no guarantee that the new solution satisfies (6b). Another alternative, is to apply (5) to only active nodes (in this case, the neighbourhood of node $i \in \mathcal{V}_a$ in Equation (5) has to be taken as $\{ j \in \mathcal{V}_a : (i, j) \in \mathcal{E} \}$, and the initial condition must satisfy $\sum_{j \in \mathcal{V}_a} x_i(0) = X - \sum_{j \in \mathcal{V}_p} x_j(0)$. However, the sub-graph formed by the active nodes is not necessarily connected although $\mathcal{G}$ is connected. Hence, marginal cost of active nodes are not necessarily equalised at equilibrium, which implies that the obtained solution is sub-optimal. In conclusion, modification of (5) to address (6) is not trivial. In order to deal with this problem, we propose the following algorithm:

$$\dot{x}_i = \sum_{j \in \mathcal{N}_i} (y_j - y_i), \text{ for all } i \in \mathcal{V} \quad (7a)$$

$$\dot{x}_i = (x_i - \bar{x}_i) + \sum_{j \in \mathcal{N}_i} (y_j - y_i), \text{ for all } i \in \mathcal{V}_p \quad (7b)$$

$$y_i = \begin{cases} 
\phi'_i(x_i), & \text{if } i \in \mathcal{V}_a \\
\phi'_i(x_i) + \dot{x}_i, & \text{if } i \in \mathcal{V}_p. 
\end{cases} \quad (7c)$$

In the same way as in (5), the variables $\{x_i, i \in \mathcal{V}\}$ in Equation (7) correspond to the resource allocated to both active and passive nodes. Notice that we have added auxiliary variables $\{\dot{x}_i, i \in \mathcal{V}_p\}$ that allow the passive nodes to interact with their neighbours taking into account the constraint (6c). On the other hand, the term $\sum_{j \in \mathcal{N}_i} (y_j - y_i)$, in Equations (7a)–(7b), leads to a consensus among the elements of the vector $y = [y_1, \ldots, y_n]^T$, which are given in Equation (7c). For active nodes, $y_i$ only depends on the marginal cost $\phi'_i(x_i)$, while for passive nodes, $y_i$ depends on both the marginal cost and the state of the auxiliary variable $\dot{x}_i$. Therefore, if the $i$th node is passive,
it has to compute both variables $x_i$ and $\hat{x}_i$. Furthermore, it can be seen that, if all the nodes are active, i.e. ($\mathcal{V}_a = \mathcal{V}$), then the proposed algorithm becomes the one stated in Equation (5).

Notice that the $i$th node only needs to know $y_i$ and the values $\{y_j : j \in \mathcal{N}_i\}$ to compute $\sum_{j \in \mathcal{N}_i} (y_j - y_i)$ in (7a)-(7b). In other words, $L(\hat{G})y = \sum_{j \in \mathcal{N}_i} (y_j - y_i)$. 

\[
\ldots \sum_{j \in \mathcal{N}_i} (y_j - y_i)\right)^T
\]

is a distributed map over the graph $\hat{G}$ (Cortés, 2008). This implies that the dynamics given in Equation (7) can be computed by each node using only local information. In fact, the message that the $i$th node must send to its neighbours is solely composed by the variable $y_i$.

### 4.1.1 Feasibility

Let us prove that, under the multi-agent system proposed in Equation (7), $x(t)$ satisfies the first constraint of the problem given by Equation (6), for all $t \geq 0$, provided that $\sum_{i=1}^n x_i(0) = X$.

**Lemma 4.1:** The quantity $\sum_{i=1}^n x_i(t)$ is invariant under Equation (7), i.e. if $\sum_{i=1}^n x_i(0) = X$, then $\sum_{i=1}^n x_i(t) = X$, for all $t \geq 0$.

**Proof:** It is sufficient to prove that $\Delta = 0$, where $\Delta = \sum_{i=1}^n \dot{x}_i$. Notice that $\Delta = \sum_{i=1}^n \dot{x}_i = 1^T \dot{x}$, where $\dot{x} = [x_1, \ldots, x_n]^T$. Moreover, according to Equation (7), $1^T \dot{x} = -1^T L(\hat{G})y$. Since $\hat{G}$ is undirected, $1^T L(\hat{G}) = L(\hat{G})1 = 0$. Therefore, $\Delta = 0$.

The above lemma does not guarantee that $x(t)$ is always feasible because of the second constraint in Equation (6), i.e. $x_i = x_i^p$, for all $i \in \mathcal{V}_p$. However, it is possible to prove that, at equilibrium, this constraint is properly satisfied.

### 4.1.2 Equilibrium point

The next proposition characterises the equilibrium point of the multi-agent system given in Equation (7).

**Proposition 4.1:** If $\hat{G}$ is connected, the system in Equation (7) has an equilibrium point $x^*$, $\{\hat{x}_i^*, i \in \mathcal{V}_p\}$, such that: $\phi_i'(x_i^*) = \lambda$, for all $i \in \mathcal{V}_a$, where $\lambda \in \mathbb{R}$ is a constant; and $x_i^* = x_i^p$, for all $i \in \mathcal{V}_p$. Moreover, $\hat{x}_i^* = \lambda - \phi_i'(x_i^*)$, for all $i \in \mathcal{V}_p$.

**Proof:** Let $x^*$, $\{\hat{x}_i^*, i \in \mathcal{V}_p\}$ be the equilibrium point of Equation (7). Since $\hat{G}$ is connected by assumption, it follows from Equation (7a) that $y_i^* = \lambda$, for all $i \in \mathcal{V}_a$, where $\lambda$ is a constant. Thus, $y_i^* = \phi_i'(x_i^*)$ if $i \in \mathcal{V}_a$, and $y_i^* = \phi_i'(x_i^*) + \hat{x}_i^*$, if $i \in \mathcal{V}_p$. Hence, $\phi_i'(x_i^*) = \lambda$, for all $i \in \mathcal{V}_a$, and $\hat{x}_i^* = \lambda - \phi_i'(x_i^*)$, for all $i \in \mathcal{V}_p$. Moreover, given the fact that $\sum_{j \in \mathcal{N}_i} (y_j - y_i^*) = 0$, it follows from Equation (7b) that $x_i^* = x_i^p$, for all $i \in \mathcal{V}_p$.

**Remark 4.1:** Proposition 4.1 states that, at the equilibrium point of (7), the active nodes equalise their marginal costs, while each passive node is allocated with an amount of resource equal to its corresponding lower bound. In conclusion, if $\sum_{i=1}^n x_i^* = X$, then it follows from Proposition 4.1, that $x^*$ minimises the optimisation problem given in Equation (6). Additionally, notice that the values $\{\hat{x}_i^*, i \in \mathcal{V}_p\}$ are equal to the KKT multipliers associated with the constraint (6c).

### 4.1.3 Convergence

Let us prove that the dynamics in Equation (7) converge to $x^*$, $\{\hat{x}_i^*, i \in \mathcal{V}_p\}$, provided that each $\phi_i(x_i)$ is strictly convex.

**Proposition 4.2:** Assume that $\phi_i(x_i)$ is a strictly convex cost function, for all $i \in \mathcal{V}$. If $\hat{G}$ is connected, $\sum_{i=1}^n x_i(0) = X$, and Assumption 4.1 holds, then $x(t)$ converges to $x^*$ under Equation (7), where $x^*$ is the solution of the optimisation problem stated in Equation (6), i.e. $x^*$ is the same given in Proposition 4.1. Furthermore, $\hat{x}_i$ converges to $\hat{x}_i^*$, for all $i \in \mathcal{V}_p$.

**Proof:** According to Lemma 4.1, since $\sum_{i=1}^n x_i(0) = X$, then $x(t)$ satisfies the first constraint of the problem stated in Equation (6), for all $t \geq 0$. Therefore, it is sufficient to prove that the equilibrium point $x^*$, $\{\hat{x}_i^*, i \in \mathcal{V}_p\}$ (which is given in Proposition 4.1) of the system proposed in Equation (7) is asymptotically stable (AS). In order to do that, let us express our multi-agent system in error coordinates, as follows:

\[
\begin{align*}
\dot{e}_i &= -L(\hat{G}) e_y & \forall i \in \mathcal{V}_p \\
\dot{\hat{e}}_i &= e_i - \left(L(\hat{G}) e_y\right)_i, & \forall i \in \mathcal{V}_p \\
e_y &= \begin{cases} 
\phi_i'(x_i) - \phi_i'(x_i^*) & \text{if } i \in \mathcal{V}_a \\
\phi_i'(x_i) - \phi_i'(x_i^*) + \hat{e}_i & \text{if } i \in \mathcal{V}_p
\end{cases}
\end{align*}
\]

where $L(\hat{G})$ is the graph Laplacian of $\hat{G}$; $e_i = x_i - x_i^*$, and $e_y = y_i - y_i^*$, for all $i \in \mathcal{V}$; $\hat{e}_i = \hat{x}_i - \hat{x}_i^*$, for all $i \in \mathcal{V}_p$; $e = [e_1, \ldots, e_n]^T$; $e_y = [e_1, \ldots, e_n]^T$; and $(L(\hat{G}) e_y)_i$ represents the $i$th element of the vector $L(\hat{G}) e_y$.

Since Assumption 4.1 holds, $\mathcal{V}_a \neq \emptyset$. Let $k$ be an active node, i.e. $k \in \mathcal{V}_a$, and let $e_k^1, e_k^2$ be the vectors obtained by removing the $k$th element from vectors $e$ and $e_y$, respectively. We notice that, according to Lemma 4.1, $e_k(t) = -\sum_{i \neq k} e_i(t)$, for all $t \geq 0$. Therefore, Equation (8) can be expressed as

\[
\begin{align*}
\dot{e}_k &= -L(\hat{G}) e_y - I_{k}^T e_y \\
\dot{e}_y &= -\sum_{i \neq k} e_y \\
\dot{\hat{e}}_i &= e_i - \left(L(\hat{G}) e_y\right)_i, & \forall i \in \mathcal{V}_p \\
e_y &= \begin{cases} 
\phi_i'(x_i) - \phi_i'(x_i^*) & \text{if } i \in \mathcal{V}_a \\
\phi_i'(x_i) - \phi_i'(x_i^*) + \hat{e}_i & \text{if } i \in \mathcal{V}_p
\end{cases}
\end{align*}
\]

where $L(\hat{G})$ and $I_{k}$ are defined in Lemma 2.1. In order to prove that the origin of the above system is AS, let us define the following Lyapunov function (adapted from...
Obando, Quijano, & Rakoto-Ravalontsalama, 2014):
\[
V = \frac{1}{2} e^T \left( L^k(\mathcal{G}) \right)^{-1} e + \frac{1}{2} \sum_{i \in \mathcal{V}_p} (e_i - \hat{e}_i)^2. \tag{10}
\]

The function \( V \) is positive definite since \( \mathcal{G} \) is connected (the reason of this fact is that, according to Lemma 2.1, \( L^k(\mathcal{G}) \) and its inverse are positive definite matrices if \( \mathcal{G} \) is connected). The derivative of \( V \) along the trajectories of the system stated in Equation (9) is given by,
\[
\dot{V} = -e^T e - e^T \left( L(\mathcal{G}) \right)^{-1} \sum_{i \in \mathcal{V}_p} e_i (e_i - \hat{e}_i).
\]

Taking into account that \( \left( L^k(\mathcal{G}) \right)^{-1} 1 = -1 \) (cf. Lemma 2.1), we obtain
\[
\dot{V} = -e^T e + \sum_{i \in \mathcal{V}, i \neq k} e_i - \sum_{i \in \mathcal{V}_p} e_i (e_i - \hat{e}_i) = -\sum_{i=1}^{n} \left( (x_i - x_i^*) (\phi'_i(x_i) - \phi'_i(x_i^*)) \right) - \sum_{i \in \mathcal{V}_p} e_i^2,
\]

where \( \phi'_i \) is strictly increasing given the fact that \( \phi_i \) is strictly convex, for all \( i \in \mathcal{V} \). Therefore, \( (x_i - x_i^*) (\phi'_i(x_i) - \phi'_i(x_i^*)) \geq 0 \), for all \( i \in \mathcal{V} \), and thus \( \dot{V} \leq 0 \).

Since \( V \) does not depend on \( \{\hat{e}_i, i \in \mathcal{V}_p\} \), it is negative semidefinite. Let \( S = \{\{e_i, i \in \mathcal{V} \}, \{\hat{e}_i, i \in \mathcal{V}_p \} : V = 0\} \), i.e., \( S = \{\{e_i, i \in \mathcal{V} \}, \{\hat{e}_i, i \in \mathcal{V}_p \} : e_i = 0, \text{ for all } i \in \mathcal{V}\} \).

Given the fact that \( \mathcal{G} \) is connected and \( \mathcal{V} \neq \mathcal{V}_p \) (by Assumption 4.1), then \( e = 0 \) if \( \hat{e}_p = 0 \) (see Equation (8)). Therefore, the only solution that stays identically in \( S \) is the trivial solution, i.e. \( e_i(t) = 0 \), for all \( i \in \mathcal{V}, \hat{e}_i(t) = 0 \), for all \( i \in \mathcal{V}_p \). Hence, we can conclude that the origin is AS by applying the Lasalle’s invariance principle.

In summary, we have shown that the algorithm described in Equation (7) asymptotically solves the subproblem in Equation (6), i.e. (7) guarantees that the resource allocated to each passive node is equal to its corresponding lower bound, while the remaining resource \( X - \sum_{i \in \mathcal{V}_p} \xi_i \) is optimally allocated to active nodes.

### 4.2 Optimal resource allocation with lower bounds

Now, let us consider our original problem stated in Equation (4), i.e. the resource allocation problem that includes lower bound constraints. Let \( x^* = [x_1^*, \ldots, x_n^*]^T \) be the optimal solution of this problem. Notice that, if we know in advance which nodes will satisfy the constraint (4c) with strict equality after making the optimal resource allocation process, i.e. \( \mathcal{I} := \{i \in \mathcal{V} : x_i^* = \xi_i\} \), we can mark these nodes as passive and reformulate (4) as a subproblem of the form (6). Based on this idea, we propose a solution method for (4), which is divided in two stages: in the first one, the nodes that belong to \( \mathcal{I} \) are identified and marked as passive; in the second one, the resulting subproblem of the form (6) is solved by using (7).

Protocol (7) can be also used in the first stage of the method as follows: in order to identify the nodes that will satisfy (4c) with strict equality at the optimal allocation, we start marking all nodes as active and apply the resource allocation process given by (7). The nodes that are allocated with an amount of resource below their lower bounds at equilibrium are marked as passive, and then (7) is newly applied (in this way, passive nodes are forced to meet (4c)). This iterative process is performed until all nodes satisfy their lower bound constraints. Notice that the last iteration of this procedure corresponds to solve a subproblem of the form (6) where the set of passive nodes is equal to the set \( \mathcal{I} \). Therefore, this last iteration is equivalent to the second stage of the proposed method.

Summarising, our method relies on an iterative process that uses the continuous-time protocol (7) as a subroutine. The main idea of this methodology is to identify in each step the nodes that have an allocated resource out of their lower bounds. These nodes are marked as passive, so they are forced to satisfy their constraints in subsequent iterations, while active nodes seek to equalise their marginal costs using the remaining resource. In the worst case scenario, the classification between active and passive nodes requires \( |\mathcal{V}| \) iterations, where \( |\mathcal{V}| \) is the number of nodes in the network. This fact arises when only one active node becomes passive at each iteration.

The proposed method is formally described in Algorithm 1. Notice that this algorithm is fully decentralised since Steps 4–6 can be computed by each agent using only local information. Step 4 corresponds to solve Equation (7), while Steps 5 and 6 describe the conditions for converting an active node into passive. Let us note that Steps 4–6 have to be performed \( |\mathcal{V}| \) times since we are considering the worst case scenario. Therefore, each agent needs to know the total number of nodes in the network. This requirement can be computed in a distributed way by using the method proposed in Garin and Schenato (2010, p. 90). We also notice the fact that the agents have to be synchronised (as usual in several distributed algorithms; Cortés, 2008; Garin & Schenato, 2010; Xiao & Boyd, 2006) in order to apply the Step 4 of Algorithm 1,
i.e. all agents must start solving Equation (7) at the same time.

**Algorithm 1**: Resource allocation with lower bounds

**Input**:
- Parameters of the problem in Equation (4).
- An initial value \( x^{(0)} \), such that \( \sum_{i=1}^{n} x_i^{(0)} = X \).

**Output**: Optimal allocation \( \tilde{x}^* \)

1. Mark all nodes as active, i.e. \( \tilde{V}_{a,0} \leftarrow \mathcal{V}, \tilde{V}_{p,0} \leftarrow \emptyset \).
2. \( \tilde{x}_{i,0} \leftarrow x_i^{(0)} \), for all \( i \in \mathcal{V} \).
3. for \( l \rightarrow 1 \) to |\( \mathcal{V} \)| do
   4. \( \tilde{x}_{i,l} \leftarrow x_{i}(t_{l}) \), for all \( i \in \mathcal{V} \), where \( x_{i}(t_{l}) \) is the solution of Equation (7a) at time \( t_{l} \), with initial conditions \( x(0) = [\tilde{x}_{1,l=1}, \ldots, \tilde{x}_{n,l=1}]^{T}, \ V_{a} = \tilde{V}_{a,l=1}, \ V_{p} = \tilde{V}_{p,l=1} \), and \( \{\tilde{x}_{i}(0) = 0, \forall i \in \mathcal{V}_{p}\} \).
   5. \( \tilde{V}_{a,l} \leftarrow \tilde{V}_{a,l-1} \cup \{i \in \tilde{V}_{a,l-1} : \tilde{x}_{i} < \tilde{x}_{j}\} \), and \( \tilde{V}_{a,l} \leftarrow \tilde{V}_{a,l-1} \setminus \{i \in \tilde{V}_{a,l-1} : \tilde{x}_{i} = \tilde{x}_{j}\} \).
6. \( \tilde{x}^* \leftarrow [\tilde{x}_{1,l}, \ldots, \tilde{x}_{n,l}]^{T} \).
7. return \( \tilde{x}^* \).

According to the reasoning described at the beginning of this subsection, we ideally require to know the steady-state solution of Equation (7) at each iteration of Algorithm 1 (since we need to identify which nodes are allocated with an amount of resource below their lower bounds in steady state). This implies that the time \( t_{l} \) in Step 4 of Algorithm 1 goes to infinity. Under this requirement, each iteration would demand infinite time and the algorithm would not be implementable. Hence, to relax the infinite time condition, we state the following assumption on the time \( t_{l} \).

**Assumption 4.2**: Let \( x_{i,l}^{*} \) be the steady state of \( x_{i}(t) \) under Equation (7), with initial conditions \( x(0) = \tilde{x}_{i,l=1}, \ V_{a} = \tilde{V}_{a,l=1}, \ V_{p} = \tilde{V}_{p,l=1} \), and \( \{\tilde{x}_{i}(0) = 0, \forall i \in \mathcal{V}_{p}\} \). For each \( l = 1, \ldots, |\mathcal{V}|-1 \), the time \( t_{l} \) satisfies the following condition: \( x_{i}(t_{l}) < \tilde{x}_{j} \) if and only if \( x_{i,l}^{*} < \tilde{x}_{j} \), for all \( i \in \mathcal{V} \).

According to assumption 4.2, for the first \(|\mathcal{V}|-1\) iterations, we only need a solution of (7) that is close enough to the steady-state solution. We point out the fact that, if the conditions of Proposition 4.2 are met in the \( l \)th iteration of Algorithm 1, then \( x_{i}(t) \) asymptotically converges to \( x_{i,l}^{*} \), for all \( i \in \mathcal{V} \), under Equation (7). Therefore, Assumption 4.2 is satisfied for large values of \( t_{1}, \ldots, |\mathcal{V}|-1 \).

Taking into account all the previous considerations, the next theorem states our main result regarding the optimality of the output of Algorithm 1.

**Theorem 4.1**: Assume that \( \mathcal{G} \) is a connected graph. Moreover, assume that \( \phi_{i} \) is a strictly convex function for all \( i = 1, \ldots, n \). If \( t_{1}, \ldots, |\mathcal{V}|-1 \) satisfy Assumption 4.2, and the problem stated in Equation (4) is feasible, then the output of Algorithm 4 tends to the optimal solution of the problem given in Equation (4) as \( t_{|\mathcal{V}|} \rightarrow \infty \).

**Proof**: The \( i \)th component of the output of Algorithm 1 is equal to \( \tilde{x}_{i,|\mathcal{V}|} = x_{i}(t_{|\mathcal{V}|}) \), where \( x_{i}(t_{|\mathcal{V}|}) \) is the solution of Equation (7a) at time \( t_{|\mathcal{V}|} \), with initial conditions \( [\tilde{x}_{1,|\mathcal{V}|-1}, \ldots, \tilde{x}_{n,|\mathcal{V}|-1}]^{T}, \ V_{a} = \tilde{V}_{a,|\mathcal{V}|}, \ V_{p} = \tilde{V}_{p,|\mathcal{V}|} \). Hence, it is sufficient to prove that \( \{x_{1,|\mathcal{V}|}^{*}, \ldots, x_{n,|\mathcal{V}|}^{*}\} \) solves the problem in Equation (4). In order to do that, let us consider the following premises (proof of each premise is written in brackets).

P1: \( \{\tilde{x}_{1,1}, \ldots, \tilde{x}_{n,1}\} \) satisfies (4b), for all \( l = 1, \ldots, |\mathcal{V}| \) (this follows from Lemma 4.1, and form the fact that \( \sum_{i=1}^{n} \tilde{x}_{i,0} = X \)).

P2: \( x_{i,l}^{*} = \tilde{x}_{i,l} \), for all \( i \in \tilde{V}_{p,|\mathcal{V}|-1} \), and for all \( l = 1, \ldots, |\mathcal{V}| \) (this follows directly from Proposition 4.2).

P3: \( \tilde{V}_{a,l} \cup \tilde{V}_{p,l} = \mathcal{V}, \) for all \( l = 1, \ldots, |\mathcal{V}| \) (from P3, we know that \( \tilde{V}_{a,l} \cup \tilde{V}_{p,l} = \tilde{V}_{a,l-1} \cup \tilde{V}_{p,l-1} \), for all \( l = 1, \ldots, |\mathcal{V}| \). Moreover, given the fact that \( \tilde{V}_{p,0} = \emptyset \), and \( \tilde{V}_{a,|\mathcal{V}|} = \mathcal{V}, \) (see step 1 of Algorithm 1) we can conclude P5.

P6: Since the problem in Equation (4) is feasible by assumption, then \( |\tilde{V}_{p,l}| < |\mathcal{V}| \), for all \( l = 1, \ldots, |\mathcal{V}| \) (the fact that \( |\tilde{V}_{p,l}| \leq |\mathcal{V}| \), for all \( l = 1, \ldots, |\mathcal{V}| \), follows directly from P5. Let us prove that \( |\tilde{V}_{p,l}| \neq |\mathcal{V}| \), for all \( l = 1, \ldots, |\mathcal{V}| \). We proceed by contradiction: Assume that there exists some \( l \) such that \( |\tilde{V}_{p,l}| < |\mathcal{V}| \) and \( |\tilde{V}_{p,l}| = |\mathcal{V}| \). Hence, from P2 and P3, we know that \( x_{i,l}^{*} \leq \tilde{x}_{l,|\mathcal{V}|-1} \), for all \( i \in \mathcal{V} \); moreover, \( i \in \tilde{V}_{a,l-1} : x_{i,l}^{*} < \tilde{x}_{l,|\mathcal{V}|-1} \neq \emptyset \). Therefore, \( \sum_{i=1}^{n} x_{i,l}^{*} < \sum_{i=1}^{n} \tilde{x}_{l,|\mathcal{V}|-1} \). According to P1, we know that \( \sum_{i=1}^{n} x_{i,l}^{*} = X \); thus, \( X < \sum_{i=1}^{n} \tilde{x}_{l,|\mathcal{V}|-1} \), which contradicts the feasibility assumption.

P7: \( \{x_{1,|\mathcal{V}|-1}^{*}, \ldots, x_{n,|\mathcal{V}|-1}^{*}\} \) satisfies the constraints (4c) (in order to prove P7, we proceed by contradiction: assume that \( \{x_{1,|\mathcal{V}|-1}^{*}, \ldots, x_{n,|\mathcal{V}|-1}^{*}\} \) does not satisfy the constraints (4c). Since P2 holds, this assumption implies that \( i \in \tilde{V}_{a,|\mathcal{V}|-1} : x_{i,|\mathcal{V}|-1}^{*} < \tilde{x}_{|\mathcal{V}|-1} \neq \emptyset \). Therefore, \( \tilde{V}_{p,|\mathcal{V}|-1} \neq \tilde{V}_{p,|\mathcal{V}|-1} \) (see P3). Using P4, we can conclude that \( \tilde{V}_{p,|\mathcal{V}|-1} \neq \tilde{V}_{p,|\mathcal{V}|-1} \). This, according to P3, \( |\tilde{V}_{p,|\mathcal{V}|-1}| > |\tilde{V}_{p,|\mathcal{V}|-1}| > \cdots > |\tilde{V}_{p,0}| > 0 \). Hence, \( |\tilde{V}_{p,|\mathcal{V}|-1}| \geq |\mathcal{V}| \), which contradicts P6).

P8: \( \sum_{i=1}^{n} \tilde{V}_{a,i} x_{i,l}^{*} \geq \sum_{i=1}^{n} \tilde{V}_{a,i} x_{i,l+1}^{*} \) (we prove P8 as follows: using P1 and the result in Lemma
4.1, we know that ∑_{i=1}^{V} x_{i,l} = ∑_{i=V} x_{i,l+1} = X. Moreover, according to P5, V can be expressed as V = V_{l} \cup V_{l+1}, where V_{l+1} ⊂ V_{l} (see P3). Thus, we have that ∑_{i∈V_{l}} x_{i,l} = ∑_{i∈V_{l+1}} x_{i,l} + ∑_{i∈V_{l}} x_{i,l+1} = ∑_{i∈V_{l}} x_{i,l} + ∑_{i∈V_{l}} x_{i,l+1} + ∑_{i∈V_{l+1}} x_{i+1,l} = ∑_{i∈V_{l}} x_{i,l+1} + ∑_{i∈V_{l+1}} x_{i,l+1} = ∑_{i∈V_{l}} x_{i,l+1} + ∑_{i∈V_{l+1}} x_{i,l+1} + ∑_{i∈V_{l+1}} x_{i,l+1}. Furthermore, since P2 holds, we have that ∑_{i∈V_{l}} x_{i,l} + ∑_{i∈V_{l}} x_{i,l-1} = ∑_{i∈V_{l}} x_{i,l} + ∑_{i∈V_{l}} x_{i,l-1} + ∑_{i∈V_{l}} x_{i,l-1} = ∑_{i∈V_{l}} x_{i,l} + ∑_{i∈V_{l}} x_{i,l-1} + ∑_{i∈V_{l}} x_{i,l-1}. Therefore, ∑_{i∈V_{l}} x_{i,l} + ∑_{i∈V_{l}} x_{i,l-1} + ∑_{i∈V_{l}} x_{i,l-1} = ∑_{i∈V_{l}} x_{i,l} + ∑_{i∈V_{l}} x_{i,l-1} + ∑_{i∈V_{l}} x_{i,l-1}. Hence, P9 guarantees that Assumption 4.1 is satisfied at each iteration.

P10: φ_{l}(x_{i,l}) ≥ φ_{l}(x_{i,l+1}), for all i ∈ V_{l,l}, we prove P10 by contradiction: assume that φ_{l}(x_{i,l}) < φ_{l}(x_{i,l+1}), for some i ∈ V_{l,l}. According to Proposition 4.2, and since P1 and P9 hold, x_{i,l} has the characteristics given in Proposition 4.1, for all i ∈ V, and for all l = 1, ..., |V|. Hence, φ_{l}(x_{i,l}) has the same value for all i ∈ V_{l,l}, and φ_{l}(x_{i,l+1}) has the same value for all i ∈ V_{l,l}. Moreover, since V_{l,l} ⊂ V_{l,l} (according to P3), we have that φ_{l}(x_{i,l}) < φ_{l}(x_{i,l+1}), for all i ∈ V_{l,l}. Thus, x_{i,l} < x_{i,l+1}, for all i ∈ V_{l,l}, because φ_{l} is strictly increasing (this follows from the fact that φ_{l} is strictly convex by assumption). Therefore, ∑_{i∈V_{l}} x_{i,l} < ∑_{i∈V_{l}} x_{i,l+1}, which contradicts P8.

Now, let us prove that \{x_{1,|V|}, ..., x_{n,|V|}\} solves the Problem in Equation (4). First, the solution \{x_{1,|V|}, ..., x_{n,|V|}\} is feasible according to P1 and P7. On the other hand, from P9, it is known that ∃k: k ∈ V_{l}, for all l = 1, ..., |V|. Let φ_{k}(x_{k,|V|}) = λ, where λ ∈ R. Moreover, let us define \{x_{0,|V|}\} = \{j ∈ V : x_{j,|V|} > x_{k}\}, and \{x_{1,|V|}\} = \{j ∈ V : x_{j,|V|} ≤ x_{k}\}. If i ∈ \{x_{0,|V|}\}, then i ∈ V_{l,|V|}. (given the fact that, if i ∈ V_{l,|V|}, then xi,|V| = x_{k}). Hence, φ_{k}(x_{k,|V|}) = φ_{k}(x_{k,|V|}) = λ (this follows from the fact that φ_{l}(x_{i,l}) has the same value for all i ∈ V_{l,l}), which in turn follows directly from step 4 of Algorithm 1, and Proposition 4.2).

If i ∈ \{x_{1,|V|}\}, then i ∈ V_{l,|V|}. In the first case, φ_{l}(x_{i,|V|}) = φ_{k}(x_{k,|V|}) = λ (following the reasoning used when i ∈ \{x_{0,|V|}\}). In the second case, ∃k: i ∈ (V_{l}\{x_{i,|V|}\}); hence, φ_{l}(x_{i,|V|}) = φ_{l}(x_{k,|V|}) (this follows from the fact that, if i ∈ (V_{l}\{x_{i,|V|}\}), then i ∈ V_{l,l}). Furthermore, since i ∈ (V_{l}\{x_{i,|V|}\}, x_{i,l} < x_{l} (see P3), and given the fact that φ_{l} is strictly increasing, we have that φ_{l}(x_{i,l}) < φ_{l}(x_{l}). Moreover, according to P10, φ_{l}(x_{i,l}) ≥ φ_{k}(x_{k,|V|}). Hence, φ_{l}(x_{i,l}) > φ_{k}(x_{k,|V|}) = λ. In conclusion, if i ∈ \{x_{1,|V|}\}, then φ_{l}(x_{i,l}) ≥ λ.

Thus, we can choose μ_i ≥ 0, for all i ∈ V, such that φ_{l}(x_{i,l}) − μ_i = λ, where μ_i = 0 if i ∈ \{x_{0,|V|}\}. Hence, let us note that \frac{dφ_{l}}{dx_{i}}|_{x_{i}=x_{i,l}} = μ_i − λ = 0, for all i ∈ V, where \frac{dφ_{l}}{dx_{i}}|_{x_{i}=x_{i,l}} = φ_{l}(x_{i,l}). Therefore, \{x_{1,|V|}, ..., x_{n,|V|}\}, μ_1, ..., μ_n, satisfies the KKT conditions for the problem given in Equation (4). Furthermore, since φ(x) is a strictly convex function by assumption, then \{x_{1,|V|}, ..., x_{n,|V|}\} is the optimal solution to that problem.

**Early stopping criterion**

Notice that, if the set of passive nodes does not change in the kth iteration of Algorithm 1 because all active nodes satisfy the lower bound constraints (see step 5), then the steady state solutions x_{i,k} and x_{i,k+1} are the same, for all i ∈ V, which implies that the set of passive nodes also does not change in the (k + 1)th iteration. Following the same reasoning, we can conclude that x_{i,k} = x_{i,k+1} = ... = x_{i,|V|} for all i ∈ V. Therefore, in this case, \{x_{1,|V|}, ..., x_{n,|V|}\} is the solution of our resource allocation problem. Practically speaking, this implies that Algorithm 1 does not need to perform more iterations after the kth one. Thus, it is possible to implement a flag z_{i} (in a distributed way) that alerts the agents if all active nodes satisfy the lower bound constraints after step 4 of Algorithm 1. A way to do that is by applying a min–consensus protocol (Cortés, 2008) with initial conditions z_{i}(0) = 0 if the node i is active and does not satisfy its lower bound constraint, and z_{i}(0) = 1 otherwise. Hence, notice that our flag z_{i} (i.e. the result of the min–consensus protocol) is equal to one, for all i ∈ V, only if all the active nodes satisfy the lower bound constraints, which corresponds to the early stopping criterion described above.

5. Simulation results and comparison

In this section, we compare the performance of our algorithm with other continuous-time distributed techniques found in the literature. We have selected three techniques that are capable to address nonlinear problems and can handle lower bound constraints: (i) a distributed interior point method (Xiao & Boyd, 2006), (ii) the local replicator equation (Pantoja & Quijano, 2012), and (iii) a distributed interior point method with exact barrier functions (Cherukuri & Cortés, 2015). The first one is a traditional methodology that uses barrier functions; the second one is a novel technique based on
population dynamics; and the third one is a recently proposed method that follows the same ideas as the first one, but replaces classic logarithmic barrier functions by exact penalty functions. Below, we briefly describe the aforementioned algorithms.

### 5.1 Distributed interior point (DIP) method

This algorithm is a variation of the one presented in Equation (5) that includes strictly convex barrier functions to prevent the solution to flow outside the feasible region. The barrier functions $b_i(x_i)$ are added to the original cost function as follows:

$$
\phi_b(x) = \phi(x) + \epsilon \sum_{i=1}^{n} b_i(x_i)
$$

$$
b_i(x_i) = -\ln (x_i - \bar{x}_i), \quad \text{for all } i \in \mathcal{V},
$$

where $\phi_b(x)$ is the new cost function, and $\epsilon > 0$ is a constant that minimises the effect of the barrier function when the solution is far from the boundary of the feasible set. With this modification, the distributed algorithm is described by the following equation:

$$
\dot{x}_i = \sum_{j \in \mathcal{N}_i} (\phi'_b (x_j) - \phi'_b (x_i)), \quad \text{for all } i \in \mathcal{V},
$$

where $\phi'_b (x_i) = \frac{d\phi_b}{dx_i} - \epsilon \frac{db_i}{dx_i}$, i.e. $\phi'_b (x_i)$ is equal to the marginal cost plus a penalty term induced by the derivative of the corresponding barrier function.

### 5.2 Local replicator equation (LRE)

This methodology is based on the classical replicator dynamics from evolutionary game theory. In the LRE, the growth rate of a population that plays a certain strategy only depends on its own fitness function and on the fitness of its neighbours. Mathematically, the LRE is given by

$$
\dot{x}_i = \sum_{j \in \mathcal{N}_i} (x_i - \bar{x}_i)(x_j - \bar{x}_j)(v_i(x_i) - v_j(x_j)),
$$

$$
v_i = -\phi'_b (x_i), \quad \text{for all } i \in \mathcal{V},
$$

where $v_i$ is the fitness perceived by the individuals that play the $i$th strategy. In this case, the strategies correspond to the nodes of the network, and the fitness functions to the negative marginal costs (the minus appears because replicator dynamics are used to maximise utilities instead of minimise costs). On the other hand, it can be shown that, if the initial condition $x(0)$ is feasible for the problem given in Equation (4), then $x(t)$ remains feasible for all $t \geq 0$, under the LRE.

### 5.3 Distributed interior point method with exact barrier functions (DIPe)

This technique follows the same reasoning of the DIP algorithm. The difference is that DIPe uses exact barrier functions (Bertsekas, 1975) to guarantee satisfaction of the lower bound constraints. The exact barrier function for the $i$th node is given by:

$$
b'_i(x_i) = \frac{1}{\epsilon}[x_i - \bar{x}_i]_+,
$$

where $[\cdot]_+ = \max(\cdot, 0)$, $0 < \epsilon < \frac{1}{\max_{x \in \mathcal{F}} \|\nabla \phi(x)\|_{\infty}}$, and $\mathcal{F} = \{x \in \mathbb{R}^n : \sum_{i=1}^{n} x_i = 1, x_i \geq \bar{x}_i\}$ is the feasible region of $x$ for the problem (4). Using these exact barrier functions, the augmented cost function can be expressed as:

$$
\phi'_e(x) = \phi(x) + \sum_{i=1}^{n} b'_i(x_i).
$$

The DIPe algorithm is given in terms of the augmented cost function and its generalised gradient $\partial \phi'_e(x) = [\partial_1 \phi'_e(x), \ldots, \partial_n \phi'_e(x)]^\top$ as follows:

$$
\dot{x}_i \in \sum_{j \in \mathcal{N}_i} \left(\partial_j \phi'_e(x_j) - \partial_i \phi'_e(x_i)\right), \quad \text{for all } i \in \mathcal{V},
$$

where

$$
\partial_i \phi'_e(x) = \begin{cases} 
\{\phi'_i(x_i) - \frac{1}{\epsilon}\} & \text{if } x_i < \bar{x}_i \\
\{\phi'_i(x_i) - \frac{1}{\epsilon}, \phi'_i(x_i)\} & \text{if } x_i = \bar{x}_i \\
\{\phi'_i(x_i)\} & \text{if } x_i > \bar{x}_i
\end{cases}
$$

In Cherukuri and Cortés (2015), the authors show that the differential inclusion (13) converges to the optimal solution of the problem (4), provided that $x(0)$ is feasible.

### 5.4 Comparison

In order to compare the performance of our algorithm with the three methods described above, we use the following simulation scenario: a set of $n$ nodes connected as in Figure 1 (we use this topology to verify the behaviour of the different algorithms in the face of few communication channels since previous studies have shown that algorithms’ performance decreases with the number of
available communication links); a nonlinear cost function $\phi(x) = \sum_{i=1}^{n} e^{a_i(x_i - b_i)} + e^{-a_i(x_i - b_i)}$, where $a_i$ and $b_i$ are random numbers that belong to the intervals $[1, 2]$ and $[-\frac{1}{2}, \frac{1}{2}]$, respectively; a resource constraint $X = 1$; and a set of lower bounds $\{x_i = 0 : i \in \mathcal{V}\}$.

For each $n$, we generate 50 problems with the characteristics described above. The four distributed methods are implemented in Matlab employing the solver function ode23s. Moreover, we use the solution provided by a centralised technique as reference. The results on the average percentage decrease in the cost function reached with each algorithm and the average computation time (time taken by each algorithm for solving a problem) are summarised in Table 1. Results of DIPe for 100 and 200 nodes were not computed for practicality since the time required by this algorithm to solve a 100/200-nodes problem is very high.

We notice that the algorithm proposed in this paper always reaches the maximum reduction, regardless of the number of nodes that comprise the network. The same happens with the DIPe algorithm. This is an important advantage of our method compared to other techniques. In contrast, the algorithm based on the LRE performs far from the optimal solution. This unsatisfactory behaviour is due to the small number of links of the considered communication network. In Pantoja and Quijano (2012), the authors prove the optimality of the LRE in problems involving well connected networks; however, they also argue that this technique can converge to suboptimal solutions in other cases. On the other hand, the DIP method provides solutions close to the optimum. Nonetheless, its performance decreases when the number of nodes increases. This tendency is due to the influence of barrier functions on the original problem. Notice that, the larger the number of nodes, the bigger the effect of the barrier functions in Equation (11).

Regarding the computation time, although convergence of the proposed method is slower than the one shown by LRE and DIP, it is faster than the convergence of the method based on exact barrier functions, i.e. DIPe. Therefore, among the methods that guarantee optimality of the solution, our technique shows the best convergence speed. Computation time taken by DIPe is affected by the use of penalty terms that generate strong changes in the value of the cost function near to the boundaries of the feasible set. The drastic variations of the generalised gradient of exact barrier functions produces oscillations of numerical solvers around the lower bounds (a visual inspection of the results given in Figure 3 of Cherukuri and Cortés (2015) confirms this claim). These oscillations are the main responsible for the low convergence speed shown by DIPe. On the other hand, LRE and DIP exhibit the fastest convergence. Hence, LRE and DIP are appealing to be implemented in applications that require fast computation and tolerate suboptimal solutions.

### 6. Applications

This section describes the use of the approach developed in this paper to solve two engineering problems. First, we present an application for sharing load in multiple chillers plants. Although this is not a large-scale application (multi-chiller plants are typically comprised of less than ten chillers; Yu & Chan, 2007), it aims to illustrate the essence of the proposed method and shows algorithm’s performance in small-size problems. One of the reasons to use a distributed approach in small-medium-size systems is due to the need of enhancing systems resilience in the face of central failures (e.g. in multiple chiller plants, central failures can occur due to cyber-attacks (Manic, Wijayasekara, Amarasinghe, & Rodriguez-Andina, 2016) against building management systems (Yu & Chan, 2007)). The second application deals with the distributed computation of the Euclidean projection of a vector onto a given set. Particularly, we use the proposed algorithm as part of a distributed technique that computes optimal control inputs for plants composed of a large number of sub-systems. This application aims to illustrate the performance of the method proposed in this paper when coping with large-scale problems.

#### 6.1 Optimal chiller loading

The optimal chiller loading problem in multiple chiller systems arises in decoupled chilled-water plants, which
are widely used in large air-conditioning systems (Chang & Chen, 2009). The goal is to distribute the cooling load among the chillers that comprise the plant for minimising the total amount of power used by them. For a better understanding of the problem, below we present a brief description of the system.

A decoupled chilled-water plant comprised by \( n \) chillers is depicted in Figure 2. The purpose of this plant is to provide a water flow \( f_T \) at a certain temperature \( T_r \) to the rest of the air-conditioning system. In order to do this task the plant needs to meet a cooling load \( C_L \), that is given by the following expression:

\[
C_L = mf_T(T_r - T_i),
\]

where \( m > 0 \) is the specific heat of the water, and \( T_r \) is the temperature of the water returning to the chillers. Since there are multiple chillers, the total cooling load \( C_L \) is split among them, i.e. \( C_L = \sum_{i=1}^{n} Q_i \), where \( Q_i \) is the cooling power provided by the \( i \)th chiller, which, in turn, is given by

\[
Q_i = mf_i(T_r - T_i),
\]

where \( f_i > 0 \) and \( T_i \) are, respectively, the flow rate of chilled water and the water supply temperature of the \( i \)th chiller. As it is shown in Figure 2, we have that \( f_T = \sum_{i=1}^{n} f_i \). In order to meet the corresponding cooling load, the \( i \)th chiller consumes a power \( P_i \) that can be calculated using the following expression (Chang & Chen, 2009):

\[
P_i = (k_{0,i} + k_{1,i}mf_iT_r + k_{2,i}(mf_iT_r)^2+i) + (k_{3,i} - k_{4,i}mf_iT_r - k_{5,i}(mf_iT_r)^2_i)T_i + (k_{6,i} + k_{7,i}(mf_iT_r)^2 + k_{8,i}(mf_iT_r)^3)T_i^2,
\]

where \( k_{j,i} \), for \( j = 0, \ldots, 6 \), are constants related to the \( i \)th chiller. If we assume that the flow rate \( f_i \) of each chiller is constant, then \( P_i \) is a quadratic function of the temperature \( T_i \). The optimal chiller loading problem involves the calculation of the chillers’ water supply temperatures that meet the total cooling load given in Equation (14), and minimise the total amount of power consumed by the chillers, i.e. \( \sum_{i=1}^{n} P_i \). Moreover, given the fact that each chiller has a maximum cooling capacity, we have to consider the following additional constraints:

\[
m f_i(T_r - T_i) \leq Q_i \quad \text{for all } i = 1, \ldots, n,
\]

where \( Q_i \) is the maximum capacity (rated value) of the \( i \)th chiller.

Summarising, the optimal chiller loading problem can be expressed as follows:

\[
\min_{T_1, \ldots, T_n} \sum_{i=1}^{n} P_i(T_i) \quad \text{s.t.} \quad \sum_{i=1}^{n} m f_i(T_r - T_i) = C_L
\]

\[
T_i \geq T_r - \frac{Q_i}{m f_i}, \quad \text{for all } i = 1, \ldots, n.
\]

Now, let us consider that we want to solve the aforementioned problem in a distributed way by using a multi-agent system, in which each chiller is managed by an agent that decides the value of the water supply temperature. We assume that the \( i \)th agent knows (e.g. by measurements) the temperature of the water returning to the chillers, i.e. \( T_r \), and the flow rate of chilled water, i.e. \( f_i \). Moreover, agents can share their own information with their neighbours through a communication network with a topology given by the graph \( G \). If each \( P_i(T_i) \) is a convex function, then the problem can be solved by using the method proposed in Algorithm 1 (we take, in this case, \( x_i = f_iT_i \)). The main advantage of this approach is to increase the resilience of the whole system in the face of possible failures, due to the fact that the plant operation does not rely on a single control centre but on multiple individual controllers without the need for a centralised coordinator.

### 6.1.1 Illustrative example

We simulate a chilled-water plant comprised by \( 7 \) chillers.\(^3\) The cooling capacity and the water flow rate of each chiller are, respectively, \( Q_i = 1406.8 \text{ kW} \), and \( f_i = 65 \text{ kg.s}^{-1} \), for \( i = 1, \ldots, 7 \); the specific heat of the water is \( m = 4.19 \text{ kJ.kg}^{-1}.\text{degC}^{-1} \); the supply temperature of the system is \( T_r = 11 \text{ degC} \); and the coefficients \( k_{j,i} \) of Equation (16) are given in Table 2. We operate the system at two different cooling loads, the first one is 90% of the total capacity, i.e. \( C_L = 0.9 \sum_{i=1}^{n} Q_i \), and the second one is 60% of the total capacity, i.e. \( C_L = 0.6 \sum_{i=1}^{n} Q_i \). The \( P_i - T_i \) curves are shown in Figure 3(a) for both cases, it can be noticed that all functions are convex. In order to apply
the distributed solution presented in Algorithm 1, we use an agent per chiller (i.e. the $i$th agent controls the supply temperature $T_i$ of the $i$th chiller) and the communication network shown in Figure 1. In all cases the initial conditions of the chillers’ supply temperatures are $T_i(0) = T_e$, for $i = 1, \ldots, 7$. The results for the first cooling load, i.e. $C_L = 8862.8$ kW, are depicted in Figure 3(b), where it is shown that the cooling load is properly allocated among the chillers by adjusting the supply temperatures. More efficient chillers (i.e. chiller 3, chiller 6, and chiller 7 in Figure 3(a)) are more loaded than the less efficient ones (i.e. chiller 2 and chiller 5). This can be noticed from the fact that their supply temperatures, in steady state, reach the minimum value. Furthermore, the energy consumption is minimised and power saving reaches to 2.6%. The results for the second cooling load, i.e. $C_L = 5908.6$ kW, are shown in Figure 3(c), where it can be noticed a similar performance to that obtained with the first cooling load. However, in this case, it is not necessary that the supply temperatures reach the minimum value to meet the required load. Newly, energy consumption is minimised and power saving reaches to 2.8%. As it is stated in Section 4, convergence and optimality of the method is guaranteed under the conditions given in Theorem 4.1.

Table 2. Chillers’ parameters.

| $i$ | $i = 1$ | $i = 2$ | $i = 3$ | $i = 4$ | $i = 5$ | $i = 6$ | $i = 7$ |
|-----|---------|---------|---------|---------|---------|---------|---------|
| $k_{s,i}$ | 113.51 | 71.70 | 62.75 | 122.68 | 74.13 | 61.98 | 76.54 |
| $k_{r,i}$ | 0.21 | -0.45 | 0.49 | 0.18 | -0.44 | 0.55 | 0.34 |
| $k_{v,i}$ | 0.35 | 0.48 | 0.30 | 0.36 | 0.44 | 0.30 | 0.31 |
| $k_{q,i}$ | -8.19 | -5.13 | -4.53 | -8.13 | -5.30 | -4.48 | -5.527 |
| $k_{c,i}$ | 0.43 | -0.14 | 0.71 | 0.40 | -0.13 | 0.26 | 0.54 |
| $k_{d,i}$ | 0.14 | 0.09 | 0.08 | 0.14 | 0.09 | 0.08 | 0.10 |
| $k_{e,i}$ | 0.01 | 0.001 | 0.02 | 0.01 | 0.001 | 0.002 | 0.001 |

Figure 3. (a) $P_i-T_i$ curves for each chiller, (b) evolution of supply temperatures and total power consumed by the chillers, $C_L = 8862.8$ kW, (c) evolution of supply temperatures and total power consumed by the chillers, $C_L = 5908.6$ kW.
Although other techniques have been applied to solve the optimal chiller loading problem, e.g. the ones in Chang and Chen (2009), they require centralised information. In this regard, it is worth noting that the same objective is properly accomplished by using our approach, which is fully distributed.

### 6.2 Distributed computation of the Euclidean projection

Several applications require computing the Euclidean projection of a vector in a distributed way. These applications include matrix updates in quasi-Newton methods, balancing of traffic flows, and decomposition techniques for stochastic optimisation (Patriksson, 2008). The problem of finding the Euclidean projection of the vector $\xi$ onto a given set $\mathcal{X}$ is formulated as follows:

$$
\min_{\xi} \| \xi - \hat{\xi} \|_2^2 \quad \text{s.t.} \quad \xi \in \mathcal{X},
$$

(19)

where $\| \cdot \|_2$ is the Euclidean norm. The vector that minimises the above problem, which is denoted by $\xi^*$, is the Euclidean projection. Roughly speaking, $\xi^*$ can be seen as the closest vector to $\xi$ that belongs to the set $\mathcal{X}$.

In Barreiro-Gomez, Obando, Ocampo-Martinez, and Quijano (2015), the authors use a distributed computation of the Euclidean projection to decouple large-scale control problems. Specifically, they propose a discrete time method to address problems involving plants comprised of a large number of decoupled sub-systems whose control inputs are coupled by a constraint. The control inputs are associated with the power applied to the sub-systems, and the constraint limits the total power used to control the whole plant. At each time iteration, local controllers that manage the sub-systems compute optimal control inputs ignoring the coupled constraint (each local controller uses a model predictive control scheme that does not use global information since the sub-systems’ dynamics are decoupled). Once this is done, the coupled constraint is addressed by finding the Euclidean projection of the vector of local control inputs (i.e. the vector formed by all the control inputs computed by the local controllers) onto a domain that satisfies the constraint associated with the total power applied to the plant.

For a better explanation of the method, consider a plant comprised of $n$ sub-systems. Let $\hat{u}_i(k) \geq 0$ be the control input computed by the $i$th local controller at the $k$th iteration ignoring the coupled constraint (non-negativity of $\hat{u}_i(k)$ is required since the control signals correspond to an applied power). Let $\hat{u}(k) = [\hat{u}_1(k), \ldots, \hat{u}_n(k)]^T$ be the vector of local control inputs, and let $u^*(k)$ be the vector of control signals that are finally applied to the sub-systems. If the maximum allowed power to control the plant is $U > 0$, the power constraint that couples the control signals is given by $\sum_{i=1}^n u^*_i(k) \leq U$. The vector $u^*(k)$ is calculated by using the Euclidean projection of $\hat{u}(k)$ onto a domain that satisfies the power constraint, i.e. $u^*(k)$ is the solution of the following optimisation problem (cf. Equation (19)):

$$
\begin{align}
\min_{u(k)} & \| \hat{u}(k) - u(k) \|_2^2 \\
\text{s.t.} & \sum_{i=1}^n u_i(k) \leq U \\
& u_i(k) \geq 0, \text{ for all } i = 1, \ldots, n,
\end{align}
$$

(20)

where $u_i(k)$ denotes the $i$th entry of the vector $u(k)$. Notice that $u^*(k)$ satisfies the power constraint and minimises the Euclidean distance with respect to the control vector $\hat{u}_k$ that is initially calculated by the local controllers. Computation of $u^*(k)$ can be performed by using the approach proposed in this paper because the problem stated in Equation (20) is in the standard form given in Equation (4) except for the inequality constraint (20b). However, this constraint can be addressed by adding a slack variable.

#### 6.2.1 Illustrative example

Consider a plant composed of 100 sub-systems. Assume that, at the $k$th iteration of the discrete time method presented in Barreiro-Gomez, Obando, Ocampo-Martinez, and Quijano (2015), the control inputs that are initially computed by the local controllers are given by the entries of the vector $\hat{u}(k) = [\hat{u}_1(k), \ldots, \hat{u}_{100}(k)]^T$, where $\hat{u}_i(k)$ is a random number chosen from the interval $[0, 1]$ kW. Furthermore, assume that the maximum allowed power to control the plant is $U = 40$ kW. To satisfy this constraint, the Euclidean projection described in Equation (20) is computed in a distributed way using Algorithm 1 with the early stopping criterion described in Section 4. The results under a communication network with path topology (see Figure 1) are depicted in Figure 4. The curve at the top of Figure 4 describes the evolution of the Euclidean distance. Notice that the proposed algorithm minimises this distance and reaches the optimum value (dashed line), which has been calculated employing a centralised method. On the other hand, the curves at the bottom of Figure 4 illustrate the evolution of the values $\sum_{i=1}^{100} u_i(k)$ (solid line) and $\min \{u_i(k)\}$ (dash-dotted line). These curves show that the constraints of the problem stated in Equation (20) are properly satisfied in steady state, i.e. $\sum_{i=1}^{100} u_i^*(k) = 40$ kW and $\min \{u_i^*(k)\} = 0$ kW. As a final observation, our algorithm exhibits a suitable
performance even considering that the communication graph is sparse and the optimal solution is not in the interior of the feasible domain. As shown in Section 5, this characteristic is an advantage of Algorithm 1 over population dynamics techniques as the one proposed in Barreiro-Gomez, Obando, Ocampo-Martinez, and Quijano (2015) to compute the Euclidean projection in a distributed way.

7. Discussion

The method developed in this paper solves the problem of resource allocation with lower bounds given in Equation (4). The main advantage of the proposed technique is its distributed nature; indeed, our approach does not need the implementation of a centralised coordinator. This characteristic is appealing, especially in applications where communications are strongly limited. Moreover, fully distributed methodologies increase the autonomy and resilience of the system in the face of possible failures. In Section 5, we show by means of simulations that the performance of the method presented in this paper does not decrease when the number of nodes (which are related to the decision variables of the optimisation problem) is large, or the communication network that allows the nodes to share information has few channels.

In these cases, the behaviour of our approach is better than the behaviour of other techniques found in the literature, such as the DIP method, or the LRE. Moreover, it is worth noting that our technique addresses the constraints as hard. This fact has two important consequences: (i) in all cases, the solution satisfies the imposed constraints, and (ii) the objective function (and therefore the optimum) is not modified (contrary to the DIP method that includes the constraints in the objective function decreasing the quality of the solution as shown in Section 5.4).

Other advantage of the method proposed in this paper is that it does not require an initial feasible solution of the resource allocation problem (4). Similarly to the DIPe technique, our method only requires that the starting point satisfies the resource constraint (4b), i.e. we need that \( \sum_{i=1}^{n} x_i(0) = X \). Notice that an initial solution \( x(0) \) that satisfies (4b) is not hard to obtain in a distributed manner. For instance, if we assume that only the kth node has the information of the available resource X, we can use \( x_k(0) = X, \{ x_i(0) = 0 : i \in \mathcal{V}, i \neq k \} \) as our starting point. Thus, an initialisation phase is not required. In contrast, other distributed methods, such as DIP and LRE needs an initial feasible solution of the problem (4), i.e. a solution that satisfies (4b) and (4c). Finding this starting point is not a trivial problem for systems involving a
large number of variables. Therefore, for these methods, it is necessary to employ distributed constraint satisfaction algorithm (as the one described in Dominguez-Garcia & Hadjicostis, 2011) as a first step.

On the other hand, we notice that to implement the early stopping criterion presented at the end of Section 4, it is required to perform an additional min-consensus step in each iteration. Despite this fact, if the number of nodes is large, this criterion saves computational time, because in most of the cases, all passive nodes are identified during the first iterations of Algorithm 1.

8. Conclusions

In this paper, we have developed a distributed method that solves a class of resource allocation problems with lower bound constraints. The proposed approach is based on a multi-agent system, where coordination among agents is done by using a consensus protocol. We have proved that convergence and optimality of the method is guaranteed under some mild assumptions, specifically, we require that the cost function is strictly convex and the graph related to the communication network that enables the agents to share information is connected. The main advantage of our technique is that it does not need a centralised coordinator, which makes the method appropriate to be applied in large-scale distributed systems, where the inclusion of centralised agents is undesirable or infeasible. As future work, we propose to use a switched approach in order to eliminate the iterations in Algorithm 1. Moreover, we plan to include upper bound constraints in our original formulation.

Notes

1. As well as in Step 4 of Algorithm 1, we have initialised the auxiliary variables \( \hat{x}_i \) to zero by convention. If these variables are initialised to other value, convergence of (7) is not affected (cf. Proposition 4.2).
2. Algorithms were implemented in a computer with an Intel Core i5 processor.
3. Simulation parameters are adapted from Chang and Chen (2009).

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

G. Obando is supported in part by Convocatoria 528 Colciencias–Colfuturo and in part by OCAD–Fondo de CTel SGR, Colombia (ALTERNAR project, BPIN 20130001000089).

References

Ayesta, U., Eruausquin, M., Ferreira, E., & Jacko, P. (2016). Optimal dynamic resource allocation to prevent defaults. Operations Research Letters, 44(4), 451–456.
Barreiro-Gomez, J., Obando, G., Ocampo-Martinez, C., & Quijano, N. (2015). Making non-centralized a model predictive control scheme by using distributed smith dynamics. In Proceedings of the 5th IFAC Conference on Nonlinear Model Predictive Control (Vol. 48, no. 23, pp. 501–506). IFAC-PapersOnLine.
Barreiro-Gomez, J., Obando, G., & Quijano, N. (2016). Distributed population dynamics: Optimization and control applications. IEEE Transactions on Systems, Man, and Cybernetics: Systems, doi:10.1109/TSMC.2016.2523934
Bemporad, A., Heemels, M., & Johansson, M. (2010). Networked control systems. London: Springer.
Bertsekas, D.P. (1975). Necessary and sufficient conditions for infeasible. As future work, we propose to use a switched approach in order to eliminate the iterations in Algorithm 1. Moreover, we plan to include upper bound constraints in our original formulation.

1. As well as in Step 4 of Algorithm 1, we have initialised the auxiliary variables \( \hat{x}_i \) to zero by convention. If these variables are initialised to other value, convergence of (7) is not affected (cf. Proposition 4.2).
2. Algorithms were implemented in a computer with an Intel Core i5 processor.
3. Simulation parameters are adapted from Chang and Chen (2009).

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

G. Obando is supported in part by Convocatoria 528 Colciencias–Colfuturo and in part by OCAD–Fondo de CTel SGR, Colombia (ALTERNAR project, BPIN 20130001000089).
Decision and Control and 28th Chinese Control Conference (pp. 1678–1683). Shanghai: IEEE.
Kakhbod, A., & Teneketzis, D. (2012). Power allocation and spectrum sharing in multi-user, multi-channel systems with strategic users. *IEEE Transactions on Automatic Control*, 57(9), 2338–2342.
Lee, H., Lee, K.J., Kim, H., Clerckx, B., & Lee, I. (2016). Resource allocation techniques for wireless powered communication networks with energy storage constraint. *IEEE Transactions on Wireless Communications*, 15(4), 2619–2628.
Manic, M., Wijayasekara, D., Amarasinghe, K., & Rodriguez-Andina, J.J. (2016). Building energy management systems: The age of intelligent and adaptive buildings. *IEEE Industrial Electronics Magazine*, 10(1), 25–39.
Mosk-Aoyama, D., Roughgarden, T., & Shah, D. (2010). Fully distributed algorithms for convex optimization problems. *SIAM Journal on Optimization*, 20(6), 3260–3279.
Nedic, A., Ozdaglar, A., & Parrilo, P. (2010). Constrained consensus and optimization in multi-agent networks. *IEEE Transactions on Automatic Control*, 55(4), 922–938.
Obando, G., Pantoja, A., & Quijano, N. (2014). "Building temperature control based on population dynamics. *IEEE Transactions on Control Systems Technology*, 22(1), 404–412.
Obando, G., Quijano, N., & Rakoto-Ravalontsalama, N. (2014). Distributed building temperature control with power constraints. In *Proceedings of the 13th European Control Conference* (pp. 2857–2862).
Palomar, D., & Chiang, M. (2006). A tutorial on decomposition methods for network utility maximization. *IEEE Journal on Selected Areas in Communications*, 24(8), 1439–1451.
Pantoja, A., & Quijano, N. (2012). Distributed optimization using population dynamics with a local replicator equation. In Jay Farrell, et al. (Eds.) *Proceedings of the 51st IEEE Conference on Decision and Control* (pp. 3790–3795), Maui: IEEE.
Pantoja, A., Quijano, N., & Passino, K.M. (2014). Dispatch of distributed generators under local-information constraints. In Dawn Tilbury, et al. (Eds.) *Proceedings of the American Control Conference* (pp. 2682–2687), Portland: IEEE.
Patriksson, M. (2008). A survey on the continuous nonlinear resource allocation problem. *European Journal of Operational Research*, 185(1), 1–46.
Patriksson, M., & Strömberg, C. (2015). Algorithms for the continuous nonlinear resource allocation problem – New implementations and numerical studies. *European Journal of Operational Research*, 243(3), 703–722.
Pietrabissa, A., Delli Priscoli, F., Di Giorgio, A., Giuseppi, A., Panfili, M., & Suraci, V. (2016). An approximate dynamic programming approach to resource management in multi-cloud scenarios. *International Journal of Control*, doi:10.1080/00207179.2016.1185802
Pillai, P.S., & Rao, S. (2016). Resource allocation in cloud computing using the uncertainty principle of game theory. *IEEE Systems Journal*, 10(2), 637–648.
Poveda, J.I., & Quijano, N. (2015). Shahshahani gradient-like extremum seeking. *Automatica*, 58, 51–59.
Ramirez-Llanos, E., & Martinez, S. (2015). Distributed and robust resource allocation algorithms for multi-agent systems via discrete-time iterations. In Yoshito Ohta, et al. (Eds.) *Proceedings of the 54th IEEE Conference on Decision and Control* (pp. 1390–1395), Osaka: IEEE.
Sharma, S., & Teneketzis, D. (2009). An externalities-based decentralized optimal power allocation algorithm for wireless networks. *IEEE/ACM Transactions on Networking (TON)*, 17(6), 1819–1831.
Tan, L., Zhu, Z., Ge, F., & Xiong, N. (2015). Utility maximization resource allocation in wireless networks: Methods and algorithms. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(7), 1018–1034.
Tan, S., Yang, S., & Xu, J. (2013). Consensus based approach for economic dispatch problem in a smart grid. *IEEE Transactions on Power Systems*, 28(4), 4416–4426.
Xiao, L., & Boyd, S. (2006). Optimal scaling of a gradient method for distributed resource allocation. *Journal of optimization theory and applications*, 129(3), 469–488.
Yi, P., Hong, Y., & Liu, F. (2015). Distributed gradient algorithm for constrained optimization with application to load sharing in power systems. *Systems & Control Letters*, 83, 45–52.
Yu, F., & Chan, K. (2007). Optimum load sharing strategy for multiple-chiller systems serving air-conditioned buildings. *Building and Environment*, 42(4), 1581–1593.
Zhu, M., & Martinez, S. (2012). On distributed convex optimization under inequality and equality constraints. *IEEE Transactions on Automatic Control*, 57(1), 151–164.