Gossip based Asynchronous and Randomized Distributed Task Assignment with Guaranteed Performance on Heterogeneous Networks

Mauro Franceschelli[a], Alessandro Giua[b,a], Carla Seatzu[a]

[a] *Dept. of Electrical and Electronic Engineering, University of Cagliari, Italy (e-mail: {mauro.franceschelli,giua,seatzu}@diee.unica.it).*
[b] *Aix Marseille Université, CNRS, ENSAM, Université de Toulon, LSIS UMR 7296,13397, Marseille, France (e-mail: alessandro.giua@lsis.org)*

Abstract

The main contribution of this paper is a novel distributed algorithm based on asynchronous and randomized local interactions, i.e., gossip based, for task assignment on heterogeneous networks. We consider a set of tasks with heterogeneous cost to be assigned to a set of nodes with heterogeneous execution speed and interconnected by a network with unknown topology represented by an undirected graph. Our objective is to minimize the execution time of the set of tasks by the networked system. We propose a local interaction rule which allows the nodes of a network to cooperatively assign tasks among themselves with a guaranteed performance with respect to the optimal assignment exploiting a gossip based randomized interaction scheme. We first characterize the convergence properties of the proposed approach, then we propose an edge selection process and a distributed embedded stop criterion to terminate communications, not only task exchanges, while keeping the performance guarantee. Numerical simulations are finally presented to corroborate the theoretical results.

Keywords: Distributed task assignment, quantized consensus, gossip algorithms, distributed optimization, multi-agent systems.

1. Introduction

In this paper we deal with the problem of distributing evenly a set of indivisible tasks over a network of agents who have to process them. To keep the presentation general, we consider tasks with different costs and agents with different execution speeds[1]. This problem, which we called *discrete consensus* in

☆ The research leading to these results was partially supported by the Italian grant SIR "Scientific Independence of young Researchers" with project CoNetDomeSys, code RBSI14OF6H, funded by the Italian Ministry of Research and Education (MIUR)

[1] In [1] some preliminary results leading to this paper were presented. In this manuscript we provide extended proofs, an improved characterization of the convergence properties, a

[2], is a generalization of the *quantized consensus* problem proposed by Kashyap *et al.* in [3] where all tasks (or tokens) have equal weight (or cost) and task execution speeds are not considered, namely they are assumed to be the same for all nodes. Moreover, we say that the assignment is performed on heterogeneous networks to emphasize the fact that each agent has its own execution speed. Agents are assumed to be interconnected by a bidirectional communication network. In accordance with most of the literature in this area, such a network is assumed to be modeled by an undirected graph as opposed to a directed graph. The quantized consensus problem on directed graphs has been studied by Kai and Ishii in [4].

We investigate the distributed task assignment problem in the framework of gossip algorithms. These algorithms were popularized in the control literature by Boyd *et al.* [5] who proposed and elegantly characterized a simple, randomized and asynchronous scheme to solve the distributed average problem in a network of sensors. This communication scheme became popular in the development of distributed algorithms for multi-agent systems because it does not require network wide synchronization of the interagent interactions as fundamental assumption for its execution. Many results followed, for brevity's sake we mention the work by Ravazzi *et al.* in [6] which characterized randomized affine dynamics with several applications to multi-agent systems, social networks and sensor networks and the very popular survey on gossip algorithms by Dimakis *et al.* in [7] and references therein. We also mention applications of gossip algorithms proposed by Riazi *et al.* to the heterogeneous multi-vehicle routing problem in [8] and to home healthcare scheduling problems [9].

Our goal in this paper is that of minimizing the execution time of the set of tasks by the networked system. The assignment is performed according to a novel distributed algorithm based on gossip-like asynchronous local state updates between the nodes. As a result of the proposed local interaction mechanism, the achievement of an optimal task assignment is not guaranteed. However, we are able to prove almost sure convergence in finite time to a bounded set containing optimal solutions. We guarantee that the worst case difference between the optimal value of the execution time and the performance of the proposed algorithm is bounded by a constant which does not depend on the network size.

The quantized consensus algorithm by Kashyap *et al.* [3] and the discrete consensus algorithm proposed by Franceschelli *et al.* [2] are based on a local balancing rule to redistribute tasks or tokens among selected pairs of nodes and a so called "swap" rule which updates the state of the nodes by simply swapping their set of tasks or tokens. The swap rule is necessary to avoid blocking configurations. It "shakes" the network state to redistribute the load and allows loads composed by discrete tasks to travel in the network, reaching a situation in which a new balancing may occur. The study of the convergence

distributed mechanism to terminate interagent communications, and an extended simulation campaign.

time of this process depends upon the meeting time of two random walkers in a graph and has received a significant attention in [10, 11, 12] among others. When considering heterogeneous execution speed, methods developed for the homogeneous case may fail to converge to the desired convergence set. The approach in [2], which considers heterogeneous execution speeds, suffers the limitation that each pair of *swap domains*, namely connected subgraphs induced by nodes with the same speed, should be connected.

In this paper, instead of making use of *swaps* as in [2], we propose a novel randomized method to overcome local minima that prevented in [2] to reach the desired convergence set.

The main contribution of this paper is a novel distributed algorithm which allows to remove the mentioned limitation of [2] and considers arbitrary topologies modeled by connected undirected graphs with nodes of arbitrary speed. Furthermore, we characterize the convergence properties of the algorithm in terms of almost sure convergence to a set of task assignments which well approximates the optimal solution. We prove an absolute performance guarantee by showing that in the worst case scenario the execution time obtained by the proposed algorithm does not differ from the optimal one by more than a constant which does not depend on the network size. Thus, the proposed approach is well suited to address the task assignment problem in large networks. Furthermore, we propose a distributed self-triggered stop criterion to terminate a Poisson edge selection process while keeping the performance guarantee.

Finally, we provide numerical simulations to characterize the expected convergence time of the proposed algorithm in line graphs and random graphs and compare it with the performance of the algorithm in [2]. We conclude the Introduction with a brief overview of the state of the art in this area.

1.1. Literature review

One of the first major contributions to the problem of *quantized consensus* which inspired several works in the following years, was proposed in [3]. It consists in a gossip-based algorithm to steer a set of quantized state variables towards a common value. Other significant contributions, still consisting in randomized gossip algorithms, have been provided in [13]. In these papers state variables in the networks are not considered as indivisible tokens or tasks. More recently, in [4] it was proposed a quantized consensus algorithm that, unlike the previous approaches, applies to networks described by directed graphs.

The issue of providing a characterization of the convergence time of quantized consensus algorithms is investigated in depth in [10], [14] and more recently in [11].

A series of contributions in the framework of *discrete consensus* have been recently proposed. Apart from [2] that has already been recalled in the first part of this section, we want to mention [15] where a discrete consensus algorithm in networks affected by execution feasibility constraints has been considered. In [16] tasks of different cost and type with capacity and feasibility constraints are

considered. Furthermore, authors impose a constraint on the maximum number of tasks executable by individual nodes. Almost sure convergence to a feasible and time-invariant configuration is proved. However, only preliminary results on the converge time are proposed. In [17] a modification of the quantized consensus algorithm is proposed to solve a load balancing problem with tasks of identical size and nodes with limited capacity.

In [18] a discrete consensus algorithm with improved convergence time with respect to quantized consensus algorithms is proposed for Hamiltonian graphs. In [19] the expected convergence time of discrete and quantized consensus has been improved on arbitrary graphs to $\mathcal{O}(n)d(\mathcal{G})$, where n is the number of nodes and $d(\mathcal{G})$ is the diameter of graph $\mathcal{G}$ representing the network topology.

In [20] the distributed task assignment problem in a network of heterogeneous mobile robots with heterogeneous tasks is investigated. The authors exploit gossip based local optimizations to both assign tasks located in a plane and compute optimized routes for robots. Finally, Chopra and Egerstedt investigated in [21, 22] heterogeneity in multi-robot systems as the ability of robots with heterogeneous skill sets to serve spatially and temporally distributed tasks.

The paper is structured as follows. In Section 2 we introduce the proposed problem statement. In Section 3 we present the main result of the paper, namely the Heterogeneous Discrete Consensus (HDC) Algorithm. In Section 4 we characterize the convergence properties of local estimation variables exploited in the HDC algorithm. In Section 5 we characterize the convergence properties of Algorithm 1 with respect to the assignment of tasks. In Section 7 we corroborate the theoretical analysis with numerical simulations. Finally, in Section 8 we discuss concluding remarks.

2. Problem Statement

Consider a network of n nodes whose pattern of interconnections can be described by an undirected connected graph $\mathcal{G} = (\mathcal{V}, E)$, where $\mathcal{V} = \{1, \dots, n\}$ is the set of nodes and $E \subseteq \{\mathcal{V} \times \mathcal{V}\}$ is the set of edges. We consider K indivisible tasks to be assigned to the nodes with execution cost $c_j \in \mathbb{N}^+$, $j = 1, \dots, K$ associated with each task. We define the *maximal cost* $c_{\max} = \max_{j=1,\dots,K} c_j$, and the *average load* $c_{ave} = \frac{1}{n} \sum_{j=1,\dots,K} c_j$. The tasks assigned to each node can be specified by n binary vectors $y_i \in \{0,1\}^K$ such that $y_{i,j} = 1$ if the j-th task is assigned to node i, $y_{i,j} = 0$ otherwise. The *load* assigned to node i is $c^T y_i$, i.e., it represents the total cost of tasks assigned to it (here $c \in \mathbb{N}^K$ is a vector whose j-th component is equal to c_j). The current *task assignment* of the network is thus $Y = [y_1 \; y_2 \; \dots \; y_n] \in \{0,1\}^{K \times n}$. Each node has an associated execution speed $\gamma_i \in \mathbb{N}^+$ and we define the *minimal speed* $\gamma_{min} = \min_{i=1,\dots,n} \gamma_i$ and the *average speed* $\gamma_{ave} = \frac{1}{n} \sum_{i=1,\dots,n} \gamma_i$. Without loss of generality, we consider integer task costs and integer execution speeds because any real value can be quantized with arbitrary precision with fixed point notation, for all practical

purposes. The *execution time* of node i is therefore $x_i = \frac{c^T y_i}{\gamma_i}$. We define the *network execution time* as

$$F(Y) = \max_{i \in \mathcal{V}} \frac{c^T y_i}{\gamma_i}, \tag{1}$$

i.e., it corresponds to the maximum execution time among all nodes.

Our objective is to minimize the execution time of the network. An optimal assignment Y^* can be found as the solution of the following integer programming problem with binary variables

$$\begin{cases} \min & F(Y) = \max_{i \in \mathcal{V}} \frac{c^T y_i}{\gamma_i} \\ s.t. & Y \mathbf{1}_n = \mathbf{1}_n \\ & y_{i,j} \in \{0,1\} \quad \forall i \in \mathcal{V}, \;\; j = 1, \ldots, K. \end{cases} \tag{2}$$

We denote by $\mathbf{1}_n$ a vector of ones with n elements. The constraint $Y\mathbf{1}_n = \mathbf{1}_n$ imposes that tasks are indivisible and can be assigned only to one node. For large number of nodes and tasks, the computational complexity of the integer programming problem (2) can be extremely high. In particular, (2) is a formulation of the makespan minimization problem on heterogeneous parallel machines. The complexity of finding an exact solution to this problem is known to be NP-hard (see [23] for reference). Furthermore, it requires a centralized coordinator with full knowledge of the network state and ability to communicate with all the nodes.

In this manuscript, we aim to develop a distributed algorithm which by exploiting only local and asynchronous interactions between the nodes is able to achieve a task assignment with a guaranteed distance from the optimum. In particular we consider a (parametrized) target set

$$\mathcal{Y}_\varepsilon = \left\{ Y \;\; | \;\; F(Y) \leq F(Y^*) + \frac{c_{max}}{\gamma_{min}} + \varepsilon \right\}, \tag{3}$$

where ε is a given arbitrary small constant. As discussed in the following section, ε is a design parameter of the proposed algorithm. A solution $Y \in \mathcal{Y}_\varepsilon$ provides an absolute performance guarantee with bounded error which does not depend on the size of the network, i.e., on the number of nodes. We point out that in the case tasks have unitary cost and nodes have unitary speed, provided that ε is sufficiently small, the set in eq. (3) contains the same set of task assignments that correspond to the quantized consensus state in [3].

3. Proposed Algorithm

The triple $(\hat{c}_{ave,i}(t), \hat{\gamma}_{ave,i}(t), \mathcal{K}_i(t))$ represents the state of node i at time t, where:

- $\hat{c}_{ave,i}(t)$ denotes the current estimate at node i of the average load c_{ave} in the network;

- $\hat{\gamma}_{ave,i}(t)$ denotes the current estimate at node i of the average speed γ_{ave} in the network;
- $\mathcal{K}_i(t) = \{j \mid y_{i,j} = 1\}$ denotes the set of indices of tasks currently assigned to node i.

The first two components of the state are called *local estimation variables* while the last one is the *task assignment*. As it will appear clear in the following, considering the two local estimation variables as part of the state, allows to emphasize that the proposed method is gossip based. Indeed such variables are updated by the same process which updates the local task assignment.

We consider a gossip model of communication between agents, driven by a random edge selection process, described in Algorithm 1 (Heterogeneous Discrete Consensus). At each iteration an arbitrary edge (i, j) is selected, and nodes i and j communicate to update their state. First, the two nodes execute an averaging of the local estimation variables. In addition they execute the Balancing Rule described in Algorithm 2 to update their task assignment. We make the following common assumption concerning the network and the edge selection process.

Assumption 3.1. *The underlying undirected graph is connected and at each iteration all arcs have a non-null lower bounded probability of being selected.* $\diamond$

Note that in Algorithms 1 and 2 we denote by the superscript $^+$ the updated value of a variable at the generic time t and omit the absolute time t altogether. Therefore, the current state will be denoted by $(\hat{c}_{ave,i}, \hat{\gamma}_{ave,i}, \mathcal{K}_i)$, while the updated state will be denoted by $(\hat{c}^+_{ave,i}, \hat{\gamma}^+_{ave,i}, \mathcal{K}^+_i)$.

To simplify the presentation of our algorithms, we also denote the *execution time* of a node i with task assignment $\mathcal{K}_i$ as: $x_i(\mathcal{K}_i) = \dfrac{1}{\gamma_i} \displaystyle\sum_{r \in \mathcal{K}_i} c_r$.

We now discuss separately the two types of updates executed by Algorithm 1. The stopping condition will be discussed later.

Update of the local estimation variables. These variables are initialized, respectively, with the node initial load and with the node speed. The evolution of these two variables, that does not depend on the current task assignments, follows the well known *gossip averaging algorithm* whose properties have been investigated in [5]. The computed value $\hat{x}_{ave,i} = \hat{c}_{ave,i}/\hat{\gamma}_{ave,i}$ is the estimate of the average execution time assuming it may be possible to assign to each node a fraction of total load in the network proportional to its speed (but this may not be possible due to task discretization).

Update of the task assignments. The task assignments of communicating nodes are updated as described in Algorithm 2. Initially (step 2) a simple heuristic is used to average the load of two nodes incident on the selected edge. This heuristic is a modification of the very well known algorithm for the 2-machine N job problem by Johnson *et al.* in [24] and is completed in a number of steps proportional to the number of tasks contained in node i and j. Variations of this greedy and widely known heuristic have been investigated in the

Algorithm 1: Heterogeneous Discrete Consensus (HDC)

Input : Sets $\mathcal{K}_i(0)$, for $i \in \mathcal{V}$ (*initial assignment of tasks to nodes*).
Output: Sets $\mathcal{K}_{i,\infty}$, for $i \in \mathcal{V}$ (*final assignment of tasks to nodes*).
1 - Initialize: For $i \in \mathcal{V}$, let

$$\hat{\gamma}_{ave,i}(0) = \gamma_i \quad \text{and} \quad \hat{c}_{ave,i}(0) = \sum_{r \in \mathcal{K}_i(0)} c_r.$$

2 - while *NOT stop_criterion* **do**
3 - A random edge (i, j) is selected according to a given stochastic selection process.
4 - Update the local estimation variables according to

$$\begin{aligned} \hat{c}^+_{ave,i} &= \tfrac{1}{2}(\hat{c}_{ave,i} + \hat{c}_{ave,j}) \\ \hat{c}^+_{ave,j} &= \tfrac{1}{2}(\hat{c}_{ave,i} + \hat{c}_{ave,j}) \\ \hat{\gamma}^+_{ave,i} &= \tfrac{1}{2}(\hat{\gamma}_{ave,i} + \hat{\gamma}_{ave,j}) \\ \hat{\gamma}^+_{ave,j} &= \tfrac{1}{2}(\hat{\gamma}_{ave,i} + \hat{\gamma}_{ave,j}) \end{aligned} \tag{4}$$

and let $\hat{x}_{ave,i} = \hat{c}^+_{ave,i}/\hat{\gamma}^+_{ave,i}$.
5 - Update the task assignment of nodes i and j according to

$$(\mathcal{K}_i^+, \mathcal{K}_j^+) = \textbf{Balancing rule}(\mathcal{K}_i, \mathcal{K}_j, \gamma_i, \gamma_j, \hat{x}_{ave,i})$$

as described in Algorithm 2.

context of load distribution between two parallel machines and is a polynomial time approximation of the 2-partitioning problem [25]. This rule computes two updated assignments: $\mathcal{K}_i^+$, $\mathcal{K}_j^+$. If the new assignments do not yield a smaller local execution time we revert to the original assignments (step 3). However, in such a case we also check if the maximum local execution time exceeds the estimated average time by a quantity greater than $c_{\max}/\gamma_{\min} + \varepsilon/2$ (step 4): if this is true we move one random task from one node to the other one to shake the network configuration and avoid being stuck in local minima. Here ε is a design parameter that will be discussed in the following section. Note also that we assume the exact value of $c_{\max}$ and $\gamma_{\min}$ to be known to all nodes: if these parameters are not available, it is possible to estimate them with max-consensus algorithms such as those developed in [26].

The proposed algorithm has a straightforward embedded stopping criterion for what regards task exchanges among nodes: when all nodes have a sufficiently accurate estimation $\hat{x}_{ave,i}$ and a local execution time below the estimated threshold $\hat{x}_{ave,i} + \frac{c_{max}}{\gamma_{min}} + \frac{\varepsilon}{2}$, then task exchanges do not occur anymore. In Section 6 we discuss how to add a distributed self-triggered stop criterion to terminate the edge selection process once a satisfactory task assignment has been achieved.

Algorithm 2: Balancing rule

Input : $\mathcal{K}_i$, $\mathcal{K}_j$, γ_i, γ_j, $\hat{x}_{ave,i}$ *(current node task assignments, node speeds and estimated average execution time)*

Output: $\mathcal{K}_i^+$, $\mathcal{K}_j^+$ *(updated node task assignments)*

1 - Initialize: Let $\mathcal{K} = \mathcal{K}_i \cup \mathcal{K}_j$, let $\mathcal{K}_i^+ := \emptyset$ and $\mathcal{K}_j^+ := \emptyset$.

2 - while $\mathcal{K} \neq \emptyset$ **do**

let $\delta := \text{argmax}_{j \in \mathcal{K}} c_j$;

if $x_i(\mathcal{K}_i^+) + c_\delta/\gamma_i \leq x_j(\mathcal{K}_j^+) + c_\delta/\gamma_j$ **then**

let $\mathcal{K}_i^+ := \mathcal{K}_i^+ \cup \{\delta\}, \quad \mathcal{K}_j^+ := \mathcal{K}_j^+$;

else

let $\mathcal{K}_i^+ := \mathcal{K}_i^+, \quad \mathcal{K}_j^+ := \mathcal{K}_j^+ \cup \{\delta\}$.

(assign task δ so as to minimally increase the maximal execution time of the two nodes) $\mathcal{K} := \mathcal{K} \setminus \{\delta\}$.

3 - if $\max(x_i(\mathcal{K}_i^+), x_j(\mathcal{K}_j^+)) \geq \max(x_i(\mathcal{K}_i), x_j(\mathcal{K}_j))$ **then**

$\mathcal{K}_i^+ := \mathcal{K}_i, \quad \mathcal{K}_j^+ := \mathcal{K}_j$;

(the heuristic did not find a more balanced assignment and we revert to original one)

4 - if $\max(x_i(\mathcal{K}_i), x_j(\mathcal{K}_j)) > \hat{x}_{ave,i} + \dfrac{c_{max}}{\gamma_{min}} + \dfrac{\varepsilon}{2}$ **then**

Choose at random a task $\delta \in \mathcal{K}_i \cup \mathcal{K}_j$.

if $\delta \in \mathcal{K}_i$ **then**

$\mathcal{K}_i^+ = \mathcal{K}_i \setminus \{\delta\}, \quad \mathcal{K}_j^+ = \mathcal{K}_j \cup \{\delta\}$

else

$\mathcal{K}_i^+ = \mathcal{K}_i \cup \{\delta\}, \quad \mathcal{K}_j^+ = \mathcal{K}_j \setminus \{\delta\}$

(move one random task from one node to the other one)

return Sets $\mathcal{K}_i^+$ and $\mathcal{K}_j^+$.

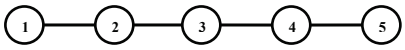

Figure 1: Graph considered in Example 3.2.

We point out that the current literature on quantized consensus algorithms does not usually consider a stop criterion on the edge selection process.

Example 3.2. *We now propose a simple example to corroborate the description of Algorithm 1 and illustrate a limitation of a previously developed algorithm in [2]. Let us consider the line network in Figure 3.2 where* $\gamma_1 = 101$, $\gamma_2 = 102$, $\gamma_3 = 103$, $\gamma_4 = 104$, $\gamma_5 = 105$. *Tasks with cost* $c_1 = c_2 = \dots c_{15} = 10$ *are assigned such that*

$$\mathcal{K}_1(0) = \{10\}, \quad \mathcal{K}_2(0) = \{10, 10\}, \quad \mathcal{K}_3(0) = \{10, 10, 10\},$$
$$\mathcal{K}_4(0) = \{10, 10, 10, 10\}, \quad \mathcal{K}_5(0) = \{10, 10, 10, 10, 10\}.$$

It holds $c_{ave} = 30$, $c_{max} = 10$, $\gamma_{ave} = 103$, $\gamma_{min} = 101$. *It can be seen that each node is optimally balanced with its neighbors, i.e., by solving a local optimization problem involving two nodes at a time such as in the Discrete Consensus algorithm proposed in [2], it is not possible to improve the maximum execution time. On the other hand, it can be seen that the assignment is neither optimal nor it belongs to set* $\mathcal{Y}_\varepsilon$ *for an arbitrary small* ε *because, as it will be explained in Section 5 (Proposition 5.1), it holds that a lower bound for the objective value at the optimal solution is* $F(Y^*) \geq \frac{c_{ave}}{\gamma_{ave}}$, *and* $F(Y) = 0.476 > \frac{c_{ave}}{\gamma_{ave}} + \frac{c_{max}}{\gamma_{min}} = 0.39$. *Furthermore, since the nodes have different execution speed the "swap" mechanism, popular in quantized consensus algorithms and employed also in [2] can not be implemented because a swap of tasks among two nodes with different speed which are locally optimally balanced implies an increment of the maximum execution time among the two nodes.*

We now examine the evolution of Algorithm 1 starting from the above task assignment. We choose $\varepsilon = 10^{-5}$. *Table 3.2 summarizes the most significant steps of the algorithm. More precisely, for increasing values of time* $t_0 < t_1 < t_2 \dots$, *it specifies the selected edge, and for any node* $i = 1, \dots, 5$ *it points out: the current task assignment* $\mathcal{K}_i$, *the current execution time* $x_i(t)$, *and its current estimate of average load* $\hat{c}_{ave,i}$, *average execution speed* $\hat{\gamma}_{ave,i}$ *and average execution time* $\hat{x}_i$. *We can see that at time* t_0 *node* 5 *holds the maximum execution time* $x_5 = 0.4761$ *and all estimation variables are initialized with the local values of speed and load, respectively, of each node according to step* 1 *of Algorithm 1.*

At t_1, *edge* $(2, 3)$ *is selected according to step* 3 *of the algorithm, and estimation variables* $\hat{c}_{ave,2}$, $\hat{c}_{ave,3}$ $\gamma_{ave,2}$ $\gamma_{ave,3}$ *are averaged and updated according to step* 4. *The balancing rule is then executed at step* 5 *following Algorithm 2. However, since the two nodes are locally optimally balanced, no better local task assignment is found and thus the tasks are not moved.*

At time t_2 *edge* $(4, 5)$ *is selected, the estimation variables are updated but again no better task assignment is found.*

The algorithm continues to iterate until the estimation variables reduce their error, thus at time t_{11} edge $(4,5)$ is randomly selected again and despite the balancing rule can not find a better task assignment, it triggers the "if" condition at step 4 of Algorithm 2 which is executed because $x_5 = 0.4761 > \hat{x}_{ave,5} + \frac{c_{max}}{\gamma_{min}} + \frac{\varepsilon}{2} = 0.4749$. As a result, one task is moved from node 5 to node 4 even if this leads to an increment in the maximum execution time that becomes equal to $x_4 = 0.4807$. This value remains unaltered until time t_{13} when edge $(3,4)$ is selected. In particular at time t_{13} a task assignment inside set $\mathcal{Y}_\varepsilon$ is found. This shows the effectiveness of the proposed algorithm: it allows to worsen the performance function in order to overcome local minima. Since the initial blocking condition is now overcome, Algorithm 1 can now eventually further improve the maximum execution time, up to the optimal solution but, since this is a greedy approach, optimality can not be guaranteed and indeed we see that at time t_{200}, while the execution time of some nodes has decreased, the maximum execution time remains the same.

4. Convergence Properties of local estimation variables

In this section we discuss the convergence properties of local estimation variables $\hat{c}_{ave,i}(t)$ and $\hat{\gamma}_{ave,i}(t)$ updated according to Algorithm 1.

Their evolution follows the *gossip averaging algorithm* in [5]. Under Assumption 3.1 it has been proved that they asymptotically converge to consensus on the average of the initial values, i.e., for $i = 1, \ldots, n$:

$$\lim_{t\to\infty} \hat{c}_{ave,i}(t) = \frac{1}{n}\sum_{i=1}^{n} \hat{c}_{ave,i}(0) = \frac{1}{n}\sum_{j=1}^{K} c_j = c_{ave},$$
$$\lim_{t\to\infty} \hat{\gamma}_{ave,i}(t) = \frac{1}{n}\sum_{i=1}^{n} \hat{\gamma}_{ave,i}(0) = \frac{1}{n}\sum_{i=1}^{n} \gamma_i = \gamma_{ave}.$$

It can be shown that the iterated pairwise averages in eq. (4) have the following monotonicity property: if at a given time t it holds

$$\max_{i\in\mathcal{V}} |\hat{c}_{ave,i}(t) - c_{ave}| \leq \Delta$$

and

$$\max_{i\in\mathcal{V}} |\hat{\gamma}_{ave,i}(t) - \gamma_{ave}| \leq \Delta,$$

for some $\Delta \in \mathbb{R}^+$, then for all $t' \geq t$ it holds

$$\max_{i\in\mathcal{V}} |\hat{c}_{ave,i}(t') - c_{ave}| \leq \Delta$$

and

$$\max_{i\in\mathcal{V}} |\hat{\gamma}_{ave,i}(t') - \gamma_{ave}| \leq \Delta.$$

A simple proof of these facts can be sketched as follows: First, notice that as shown in [5], the average value of the full set of estimation variables, i.e.,

c_{ave} and γ_{ave}, is constant for the considered gossip algorithm. Now consider the first quantity of interest $\max_{i\in\mathcal{V}} |\hat{c}_{ave,i}(t) - c_{ave}|$. When an edge is selected and the estimation variables are updated, the following cases may occur: 1) A node which holds the maximum value of the estimation in the network is not selected for the update; 2) A node which holds the maximum value of the estimation is selected but it is not unique; 3) A node which holds the maximum value is selected and it is unique. Cases 1 and 2 result in no change in the quantity $\max_{i\in\mathcal{V}} |\hat{c}_{ave,i}(t) - c_{ave}|$ because at least one node which holds the maximum value did not update its estimation. In case 3 it holds that the selected node has to average its estimation with a node with smaller value and thus the quantity $\max_{i\in\mathcal{V}} |\hat{c}_{ave,i}(t) - c_{ave}|$ decreases. Furthermore, the maximum value in the network can not be lower than the average, i.e., c_{ave}. This proves that the considered quantity is non-increasing.

The same reasoning applies to show the monotonicity of the estimated value of γ_{ave}.

On the basis of the above results we can state the following monotonicity property for the variable $\hat{x}_{ave,i} = \hat{c}_{ave,i}/\hat{\gamma}_{ave,i}$.

Proposition 4.1. *If at time t it holds*

$$\max_{i\in\mathcal{V}} |\hat{c}_{ave,i}(t) - c_{ave}| \leq \Delta \quad \textit{and} \quad \max_{i\in\mathcal{V}} |\hat{\gamma}_{ave,i}(t) - \gamma_{ave}| \leq \Delta, \tag{5}$$

for some $\Delta \in \mathbb{R}^+$, then for any $t' \geq t$ it holds

$$\max_{i\in\mathcal{V}} \left| \hat{x}_{ave,i}(t') - \frac{c_{ave}}{\gamma_{ave}} \right| \leq \Delta \frac{\gamma_{ave} + c_{ave}}{\gamma_{min}\gamma_{ave}}. \tag{6}$$

Proof: Let $c_{err,i}(t) = \hat{c}_{ave,i}(t) - c_{ave}$, $\gamma_{err,i}(t) = \hat{\gamma}_{ave,i}(t) - \gamma_{ave}$ and $x_{err,i}(t) = \hat{x}_{ave,i}(t) - \frac{c_{ave}}{\gamma_{ave}}$. It follows that

$$x_{err,i}(t) = \frac{c_{err,i}(t) + c_{ave}}{\gamma_{err,i}(t) + \gamma_{ave}} - \frac{c_{ave}}{\gamma_{ave}} = \frac{c_{err,i}(t)\gamma_{ave} - \gamma_{err,i}(t)c_{ave}}{\gamma_{err,i}(t)\gamma_{ave} + \gamma_{ave}^2}$$

By assumption, $|c_{err,i}(t)| \leq \Delta$ and $|\gamma_{err,i}(t)| \leq \Delta$. Since $\gamma_{err,i}(t) \geq \gamma_{min} - \gamma_{ave}$, then

$$|x_{err,i}(t)| \leq \Delta \frac{\gamma_{ave} + c_{ave}}{\gamma_{min}\gamma_{ave}}.$$

□

5. Convergence Properties of Algorithm 1

In this section we characterize the convergence properties of Algorithm 1. Our objective is to prove in Theorem 5.7 that almost surely, i.e., with unitary probability, there exists a finite time after which the task assignment computed by our method belongs to the target set $\mathcal{Y}_\varepsilon$ in eq. (3). Set $\mathcal{Y}_\varepsilon$ characterizes the

set of assignments which achieve the proposed performance guarantee, i.e., a maximum execution time in the network which differs from the optimal one by at most a small constant that does not depend on the size of the network or on its topology. To do so, we will exploit the results of Section 4. We first compute a lower bound on the optimal execution time (Proposition 5.1) as function of the average load and execution speed. Then, we show that Algorithm 1 is able to estimate the average execution time with bounded error in finite time. In particular, we show in Corollary 5.5 that if the error in the estimation variables is small enough, then a task assignment belonging to the target set $\mathcal{Y}_\varepsilon$ is reached in a finite number of iterations. The invariance of the target $\mathcal{Y}_\varepsilon$, proven by Proposition 5.6, completes the set of results needed to prove Theorem 5.7.

Now, we provide a lower bound to the optimal value $F(Y^*)$ of the execution time.

Proposition 5.1. *A lower bound on the optimal value of the objective function of Problem* (2) *is:*

$$\frac{c_{ave}}{\gamma_{ave}} \leq F(Y^*). \tag{7}$$

Proof: Consider a relaxed optimization problem where tasks are infinitely divisible so that each node has the same execution time x_{opt}. Then $\sum_{j \in \mathcal{K}_i} c_j = x_{opt}\gamma_i$ for all $i \in \mathcal{V}$. Therefore, summing up on all nodes it holds

$$\sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{K}_i} c_j = \sum_{i \in \mathcal{V}} x_{opt}\gamma_i,$$

thus

$$x_{opt} = \frac{\sum_{j=1}^{K} c_j}{\sum_{i=1}^{n} \gamma_i}.$$

By multiplying and dividing by n we can write equivalently $x_{opt} = \frac{c_{ave}}{\gamma_{ave}}$. This proves the statement being obviously $x_{opt} \leq F(Y^*)$ because of the discrete nature of tasks. □

By Proposition 5.1, it follows that the target set $\mathcal{Y}_\varepsilon$ in eq. (3) can rewritten as:

$$\mathcal{Y}_\varepsilon = \left\{ Y \;\mid\; F(Y) \leq \frac{c_{ave}}{\gamma_{ave}} + \frac{c_{max}}{\gamma_{min}} + \varepsilon \right\}. \tag{8}$$

It is obvious that $\mathcal{Y}_\varepsilon \subseteq \mathcal{Y}_{\varepsilon'}$ for $\varepsilon \leq \varepsilon'$.

In the following we first prove a series of results from which it derives that Algorithm 1 almost surely converges in finite time to set $\mathcal{Y}_\varepsilon$.

The next proposition shows that any improvement of the objective function $F(Y)$ is lower bounded by a positive constant.

Proposition 5.2. *Given a task assignment Y, let Y' be a new task assignment determined by Algorithm 1 in one iteration. If $F(Y') < F(Y)$, then $F(Y') \leq F(Y) - \frac{1}{\varrho}$, where ϱ is the least common multiplier (lcm) among γ_i for $i = 1, \ldots, n$.*

Proof: Obviously the set of possible task assignments Y is finite, therefore the set of values taken by $F(Y)$ is finite as well. It follows that the minimum difference between different values of $F(Y)$ is lower bounded by a constant.

To compute this constant, we first observe that $\varrho x_i = \frac{\varrho}{\gamma_i} c^T y_i$ is an integer for all $i = 1, \ldots, n$. Therefore, for any i, j, if $x_i > x_j$, the following inequality holds

$$\frac{\varrho c^T y_i}{\gamma_i} - \frac{\varrho c^T y_j}{\gamma_j} \geq 1,$$

or equivalently, $x_j \leq x_i - \frac{1}{\varrho}$.

□

Given a task assignment Y we now introduce a new performance index $J(Y) = (F(Y), n_{\max}(Y))$ consisting of two terms. The first term $F(Y)$ is the network execution time, while the second one $n_{\max}(Y)$ denotes the cardinality of the set of nodes that have maximal execution time given Y. We impose a lexicographic ordering on the performance index, i.e., $J(Y') < J(Y)$ if either $F(Y') < F(Y)$ or $F(Y') = F(Y)$ and $n_{\max}(Y') < n_{\max}(Y)$. This lexicographic ordering is exploited to prove that the maximum execution time is decreasing if a sufficient number of iterations of Algorithm 1 is executed.

Proposition 5.3. *Given a task assignment $Y \notin \mathcal{Y}_\varepsilon$ there exists a new assignment Y' with $J(Y') < J(Y)$ that is identical to Y except for the transfer of one task from a node i with maximal execution time to another node k.*

Proof: If $Y \notin \mathcal{Y}_\varepsilon$ then by eq. (8)

$$\max_{i \in \mathcal{V}} x_i > \frac{c_{ave}}{\gamma_{ave}} + \frac{c_{max}}{\gamma_{min}} + \varepsilon.$$

This implies that there exists at least one node k such that $x_k < \frac{c_{ave}}{\gamma_{ave}}$. Therefore, if the number of nodes with maximum execution time is greater than one, we can move one task from one of such nodes and put it in node k to lower this number by one. If only one node holds the maximum execution time then moving a task from such a node to node k reduces the maximum execution time, thus proving the statement. □

Note that in Proposition 5.3 the new configuration Y' may not be reachable from Y in a single gossip iteration because the node(s) with maximum load and node k with $x_k < \frac{c_{ave}}{\gamma_{ave}}$ might not be connected by an edge. We can finally state the following results.

Proposition 5.4. *Consider at time t a task assignment $Y \notin \mathcal{Y}_\varepsilon$ and local estimation errors*

$$\max_{i \in \mathcal{V}} |\hat{c}_{ave,i}(t) - c_{ave}| \leq \frac{\varepsilon}{2} \frac{\gamma_{ave}\gamma_{min}}{\gamma_{ave} + c_{ave}},$$

and

$$\max_{i \in \mathcal{V}} |\hat{\gamma}_{ave,i}(t) - \gamma_{ave}| \leq \frac{\varepsilon}{2} \frac{\gamma_{ave}\gamma_{min}}{\gamma_{ave} + c_{ave}}.$$

Algorithm 1 can always reach an assignment Y' with $J(Y') < J(Y)$ in a finite number of iterations $\alpha < 2d$, where d is the network diameter.

Proof: We observe that by Proposition 4.1 the conditions on the local estimation errors imply that for all $t' \geq t$ it holds

$$\max_{i \in \mathcal{V}} |\hat{x}_{ave,i}(t) - c_{ave}/\gamma_{ave}| \leq \varepsilon/2. \tag{9}$$

Now, let node i be a node with maximal execution time $x_i = F(Y)$. By Proposition 5.3 there exists in this node a task δ that if transferred to another node k would reduce the performance index J. Consider now the shortest path from i to k and let it be $i, j, l, \ldots, k$. We now show that by a suitable edge selection, load δ is transferred to node k and either for all other nodes along the path the maximum execution time is less than $F(Y)$ or we reach before an assignment Y' with $J(Y') < J(Y)$.

Assume edge (i, j) is selected. Two cases are possible.
1a) If the Balancing Rule leads to a new assignment Y' with maximum local execution time smaller than $F(Y)$ then $J(Y') < J(Y)$ and the statement holds.
1b) If no better balancing is reached, by the assumption that $Y \notin \mathcal{Y}_\varepsilon$ and the condition in eq. (9), it holds

$$\begin{aligned} \max(x_i, x_j) &= F(Y) > \frac{c_{ave}}{\gamma_{ave}} + \frac{c_{max}}{\gamma_{min}} + \varepsilon \\ &\geq \hat{x}_{ave,i} + \frac{c_{max}}{\gamma_{min}} + \frac{\varepsilon}{2} \end{aligned}$$

hence step 4 in Algorithm 2 will be executed and load δ may be transferred to node j thus reaching an assignment Y' where: i) $x_i < F(Y)$; ii) node j has maximal execution time $x_j = F(Y') \geq F(Y)$; iii) load δ is one step closer to node k.

Assume edge (j, l) is selected. Three cases are possible.
2a) If the Balancing Rule leads to a new assignment Y'' with maximum local execution time smaller that $F(Y)$ then $J(Y'') < J(Y)$ and the statement holds.
2b) If no better balancing is possible by the previous argument it is possible to reach a configuration Y'' where: i) $x_i < F(Y)$; ii) node l has maximal execution time $x_l = F(Y'') \geq F(Y)$; iii) load δ is assigned to node l.
2c) If the Balancing Rule leads to a new assignment Y'' with maximum local execution time greater than $F(Y)$, then by reselecting edge (j, l) since no further balancing is possible we can show again that by transferring a load from node j to l it is possible to reach a configuration Y'' where: i) $x_j < F(Y)$; ii) node l has maximal execution time $x_l = F(Y'') \geq F(Y)$; iii) load δ is assigned to node l.

By repeating the argument we can be sure to reach a new assignment with an improved performance index. The bound on the number of iterations follows from the necessity in case 2c) to select twice the same edge. □

Based on Proposition 5.3 and Proposition 5.2, the next result trivially follows.

Corollary 5.5. *The task assignment set $\mathcal{Y}_\varepsilon$ is always reachable by executing a finite number of iterations of Algorithm 1.* ◇

Finally we prove that set $\mathcal{Y}_\varepsilon$ is invariant if the local estimate variables are sufficiently precise.

Proposition 5.6. *Consider at time t a task assignment $Y \in \mathcal{Y}_\varepsilon$ and local estimation errors*

$$\max_{i\in\mathcal{V}} |\hat{c}_{ave,i}(t) - c_{ave}| \leq \frac{\varepsilon}{2}\frac{\gamma_{ave}\gamma_{min}}{\gamma_{ave}+c_{ave}},$$

and

$$\max_{i\in\mathcal{V}} |\hat{\gamma}_{ave,i}(t) - \gamma_{ave}| \leq \frac{\varepsilon}{2}\frac{\gamma_{ave}\gamma_{min}}{\gamma_{ave}+c_{ave}}.$$

Any new assignment Y' determined by Algorithm 1 is such that $Y' \in \mathcal{Y}_\varepsilon$.

Proof: We first observe that due to Proposition 4.1 it holds

$$\max_{i\in\mathcal{V}} \left|\hat{x}_{ave,i}(t') - \frac{c_{ave}}{\gamma_{ave}}\right| \leq \frac{\varepsilon}{2}.$$

Thus, step 4 of the Balancing Rule is never executed. In fact for any two nodes i and j it holds:

$$\max(x_i, x_j) \quad = F(Y) \leq \frac{c_{ave}}{\gamma_{ave}} + \frac{c_{max}}{\gamma_{min}} + \varepsilon \leq \hat{x}_{ave,i} + \frac{c_{max}}{\gamma_{min}} + \frac{\varepsilon}{2}$$

□

We can finally characterize the convergence property of Algorithm 1 as follows.

Theorem 5.7. *The task assignment Y updated iteratively according to Algorithm 1 with a given $\varepsilon > 0$ converges almost surely in finite time to set $\mathcal{Y}_\varepsilon$ defined in* (3)*, i.e.,*

$$Pr\left(\exists\tau : Y(\tau') \in \mathcal{Y}_\varepsilon), \quad \forall\tau' \geq \tau\right) = 1.$$

Proof: The claim follows from the following three facts.

1) Given the convergence property of the local estimated variables discussed in Section 4, starting from any task configuration Y at time t, there exists a finite time $t' \geq t$ such that $\max_{i\in\mathcal{V}} \left|\hat{x}_{ave,i}(t') - \frac{c_{ave}}{\gamma_{ave}}\right| \leq \frac{\varepsilon}{2}$.

2) Starting from any task configuration Y, set $\mathcal{Y}_\varepsilon$ is reachable in a finite number of iterations by Corollary 5.5.

3) The target set $\mathcal{Y}_\varepsilon$ is invariant by Proposition 5.6. □

6. Edge selection process and self-triggered stop criterion

We point out that once set $\mathcal{Y}_\varepsilon$ is reached, task exchanges among nodes terminate. However the nodes are not aware of this and continue to communicate to attempt local improvements. In this section we provide a distributed edge selection process with an embedded self-triggered stop criterion which halts any inter-agent communication once the convergence set has been reached.

Next, we recall a definition of edge selection process which can be found in [27].

Definition 6.1 (Edge Selection Process). *An edge selection process* $\mathfrak{e} : \mathbb{R}^+ \times E \to \{0, 1\}$ *maps each time instant* $t \in \mathbb{R}^+$ *and each edge* $(i, j) \in E$ *to a binary value: if* $\mathfrak{e}(t, (i, j)) = 1$ *then edge* (i, j) *is active at time* t*, not active otherwise.*

We consider the case, as in [5], that the probability of selection of each edge follows an exponential distribution of parameter $\lambda = 1$ and thus the number of selections of the same edge in a given time interval is modeled by an independent Poisson process. The expected rate of edge triggering is therefore equal to λ for each edge. This implies that given an arbitrary time t, the probability of choosing a particular edge in the network among the others has uniform distribution.

Such clocks can be easily implemented by associating a uniform edge triggering probability at each time measurement.

We now recall a result proposed in [5] regarding the convergence time of gossip based distributed averaging, which we exploit in Algorithm 1 to estimate the parameters c_{ave} and γ_{ave}. Such result holds under the above assumption on the edge selection process. Let $\hat{\gamma}_{ave}(t) \in \mathbb{R}^n$ and $\hat{c}_{ave}(t) \in \mathbb{R}^n$ be the vectors whose i-th elements are respectively $\hat{\gamma}_{ave,i}(t)$ and $\hat{c}_{ave,i}(t)$, and $\mathbf{1}_n$ be the n-th element vector with unitary entries. In [5] the authors defined the ξ-convergence time, where $\xi \in (0, 1)$ as:

$$T_{ave}(\xi) = \sup_{\hat{\gamma}_{ave}(0)} \inf \left\{ t : Pr\left(\frac{\|\hat{\gamma}_{ave}(t) - \gamma_{ave}\mathbf{1}_n\|_2}{\|\hat{\gamma}_{ave}(0)\|_2} \geq \xi \right) \leq \xi \right\}. \tag{10}$$

In simple words, $T_{ave}(\xi)$ is the smallest time it takes for $\hat{\gamma}_{ave}(t)$ to get within ξ of $\gamma_{ave}\mathbf{1}_n$ with high probability, regardless of the initial value of $\hat{\gamma}_{ave}(0)$ [5]. The same holds for vector $\hat{c}_{ave}(t)$. Let matrix $W \in \mathbb{R}^{n \times n}$ be defined as

$$W = I - \frac{1}{2n} D + \frac{P + P^T}{2n},$$

where $P \in \mathbb{R}^{n \times n}$ is a matrix whose P_{ij} element represents the probability of selecting edge (i, j) at time t by the gossip algorithm and D is a diagonal matrix whose elements are $D_i = \sum_{j=1,\ldots,n} [P_{ij} + P_{ji}]$. Finally, let $\lambda_2(W)$ denote the second largest eigenvalue of matrix W. In [5], it is proven that the convergence time in eq. (10) as function of parameter ξ and W is upper bounded by

$$T_{ave}(\xi) \leq \frac{3log\left(\xi^{-1}\right)}{log\left(\lambda_2(W)^{-1}\right)}, \tag{11}$$

Thus, for a given graph $\mathcal{G}$ where edges are selected with uniform probability with corresponding matrix W, and the probability of selection of each edge follows an exponential distribution of parameter $\lambda = 1$, $T_{ave}(\xi)$ represents the time it takes to achieve a relative error on the estimation variables $\frac{\|\hat{\gamma}_{ave}(t) - \gamma_{ave}\mathbf{1}_n\|_2}{\|\hat{\gamma}_{ave}(0)\|_2} < \xi$ with probability greater than $1 - \xi$. The same holds with $\hat{c}_{ave}(t)$.

We are now ready to propose an edge selection process with an embedded self-triggered stop criterion for Algorithm 1. We assume that each node has

a clock which ticks as a Poisson process with rate $\lambda = 1$. The basic idea behind the proposed stop criterion consists in exploiting an upper bound on the convergence time of the distributed estimation process as performance certificate for the quality of the local estimation. Then, the proposed mechanism stops the communications requests to neighbors of a given node i if node i is locally balanced with its neighbors and if its own execution time is above a threshold computed by exploiting the upper bound to $T_{ave}(\xi)$. If a neighbor of node i requests a state update to i, and this request leads to a state update, then node i resumes its communication requests with the neighbors until the stop condition is triggered again. In Theorem 6.3 it is proven that the edge selections in the whole network stop with a given probability which can be made arbitrarily high after a finite number of iterations.

Definition 6.2 (Edge selection process of node i with embedded stop criterion). *Whenever the clock of node i ticks at time t, it executes the next operations:*

1. *If $t \leq T_{ave}(\xi)$ or $x_i(t) > \hat{x}_{ave,i}(t) + \frac{c_{max}}{\gamma_{min}} + \frac{\varepsilon}{2}$ then choose uniformly at random a node from the set of neighbors $\mathcal{N}_i$ and execute an iteration of Algorithm 1;*
2. *Else stop the clock until a neighbor requests a state update;*
3. *Endif.*

■

We point out that an upper bound on $T_{ave}(\xi)$ used in the stop criterion can be computed by eq.(11) ([5]).

The next theorem characterizes the convergence properties of Algorithm 1 in the case in which the edge selection process and stop criterion in Definition 6.2 is executed.

Theorem 6.3. *Consider a network which executes Algorithm 1. Let*

$$\xi < \min\left\{\frac{1}{c_{max}}, \frac{1}{\gamma_{max}}\right\} \frac{\varepsilon}{2\sqrt{n}} \frac{\gamma_{min}\gamma_{ave}}{\gamma_{ave} + c_{ave}}.$$

If edges are selected according to the strategy in Definition 6.2, then with probability $1 - \xi$ there exists a finite time such that no edges are selected anymore, interagent communications come to a halt and $Y \in \mathcal{Y}_\varepsilon$.

Proof: For $t \in (0, T_{ave}(\xi)]$ the edge selection process in Definition 6.2 is such that Assumption 3.1 holds and thus the convergence result of Theorem 5.7 holds.

Now, if $t > T_{ave}(\xi)$, with probability $1 - \xi$ it holds that the relative errors on the estimation variables satisfy

$$\|\hat{\gamma}_{ave}(t) - \gamma_{ave}\mathbf{1}_n\|_2 \leq \|\hat{\gamma}_{ave}(0)\|_2 \, \xi,$$

and

$$\|\hat{c}_{ave}(t) - c_{ave}\mathbf{1}_n\|_2 \leq \|\hat{c}_{ave}(0)\|_2 \, \xi.$$

Since $\|\hat{\gamma}_{ave}(0)\|_2 \leq \sqrt{n}\gamma_{max}$ and

$$\max_{i\in\mathcal{V}} |\hat{\gamma}_{ave,i}(t) - \gamma_{ave}| \leq \|\hat{\gamma}_{ave}(t) - \gamma_{ave}\mathbf{1}_n\|_2 \,,$$

it holds

$$\max_{i\in\mathcal{V}} |\hat{\gamma}_{ave,i}(t) - \gamma_{ave}| \leq \sqrt{n}\gamma_{max}\xi,$$

and analogously

$$\max_{i\in\mathcal{V}} |\hat{c}_{ave,i}(t) - c_{ave}| \leq \sqrt{n}c_{max}\xi.$$

Therefore, if

$$\sqrt{n}\gamma_{max}\xi \leq \frac{\varepsilon}{2}\frac{\gamma_{min}\gamma_{ave}}{\gamma_{ave} + c_{ave}},$$

and

$$\sqrt{n}c_{max}\xi \leq \frac{\varepsilon}{2}\frac{\gamma_{min}\gamma_{ave}}{\gamma_{ave} + c_{ave}},$$

or equivalently

$$\xi < \min\left\{\frac{1}{c_{max}}, \frac{1}{\gamma_{max}}\right\}\frac{\varepsilon}{2\sqrt{n}}\frac{\gamma_{min}\gamma_{ave}}{\gamma_{ave} + c_{ave}},$$

then for all $t \geq T_{ave}(\xi)$ by Proposition 4.1

$$\max_{i\in\mathcal{V}} \left|\hat{x}_{ave,i}(t) - \frac{c_{ave}}{\gamma_{ave}}\right| \leq \frac{\varepsilon}{2}. \tag{12}$$

Therefore, if

$$\xi < \min\left\{\frac{1}{c_{max}}, \frac{1}{\gamma_{max}}\right\}\frac{\varepsilon}{2\sqrt{n}}\frac{\gamma_{min}\gamma_{ave}}{\gamma_{ave} + c_{ave}},$$

then with probability greater than $1 - \xi$, for all $t \geq T_{ave}(\xi)$ the condition required by Propositions 5.4 holds.

Thus, whenever the internal clock of an agent triggers the selection of an edge, according to Definition 6.2, if $t > T_{ave}(\xi)$ and $x_i \leq \hat{x}_{ave,i} + \frac{c_{max}}{\gamma_{min}} + \frac{\varepsilon}{2}$ then no edge is selected thus preventing any interagent communication.

As long as there exist a node with load $x_i > \hat{x}_{ave,i} + \frac{c_{max}}{\gamma_{min}} + \frac{\varepsilon}{2}$, thus $Y \notin \mathcal{Y}_\varepsilon$, edges incident on such node continue to be selected with strictly positive probability.

Furthermore, we can repeat the reasoning in the proofs of Proposition 5.3, 5.4, 5.6 since the edge selection process in Definition 6.2 does not violate the assumption that the probability of selecting edges incident on the nodes holding the maximum value in the network is strictly positive.

Thus, in a finite time τ almost surely $Y(\tau) \in \mathcal{Y}_\varepsilon$. Then, for all $t \geq \tau$, $Y \in \mathcal{Y}_\varepsilon$, and step 1 Definition 6.2 is never executed. Therefore, no edges are selected anymore effectively halting the execution of Algorithm 1 and thus proving the statement. □

Remark 6.4. *If we wish to arbitrarily increase the probability that after a time at most equal to $T_{ave}(\xi)$ the inequality (12) holds, we can choose $\bar{T}_{ave}(\xi, k) = kT_{ave}(\xi)$, by keeping the bound on the desired error constant the probability increases to $(1-\xi)(1+\sum_{s=1}^{k}\xi^s)$, which tends to 1 as $k \to \infty$.*

7. Numerical simulations

In this section we corroborate the theoretical characterization of the convergence properties of Algorithm 1 with numerical simulations. First, we compare the proposed algorithm with the algorithm proposed in [2]. We considered a network represented by a line graph composed by 30 nodes, each with an execution speed chosen uniformly at random in the interval $[1, 3]$. We considered a set of $K = 180$ tasks to be distributed among the nodes, each with an integer cost chosen uniformly at random in the interval $[1, 10]$.

We simulated the Discrete Consensus Algorithm (DCA) in [2] and the Heterogeneous Discrete Consensus (HDC) algorithm proposed in this paper with the same set of random initial conditions and with the same sequence of random edge selections. In these simulations we chose parameter $\varepsilon = 10^{-3}$.

In Figure 2 it is shown the evolution of the maximum execution time during the execution of the DCA. It can be seen that the execution time is non-increasing but since the network does not satisfy the condition of fully connected "swap domains" ([2]) it can not be guaranteed that the final task assignment is close to the optimal solution. On the contrary, in the chosen example the worst case performance may differ from the optimal value of the execution time by a quantity proportional to the number of nodes.

In Figure 3 it is shown the evolution of the maximum execution time during the execution of the HDC algorithm. It can be seen that the execution time does not monotonically decrease because to overcome blocking configurations of tasks some of them are moved at random. Once the estimation error is sufficiently small in each node then after a sufficiently long time a task assignment in set $\mathcal{Y}_\varepsilon$ is reached and the local interactions stop.

In Figure 4 we show a direct comparison between the simulations of the evolution of the maximum execution time during the execution of the DCA and HDC algorithm. It can be seen that Algorithm 1 outperforms the algorithm proposed in [2].

In Figure 5 we show the evolution of variables $\hat{x}_{ave,i}$, which evolve according to the gossip algorithm presented in [5]. It can be noticed that despite Algorithm 1 involving quantized and randomized dynamics, its simulated convergence rate does not appear to be significantly different from the simple averaging gossip algorithm in [5].

Finally, to corroborate our theoretical results in Section 5 we propose a set of numerical simulations to evaluate the expected convergence time of the proposed algorithm. In particular, in Figure 6 it is shown how the average convergence time of 10 simulations varies with respect to the number of nodes

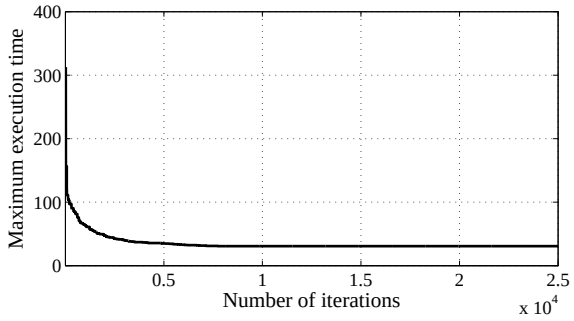

Figure 2: Evolution of the network maximum execution time according to the Discrete Consensus Algorithm in [2] in a line network of 30 nodes.

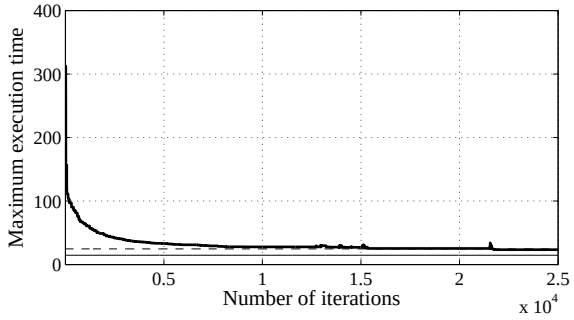

Figure 3: Evolution of the network maximum execution time according to Algorithm 1 in a line network of 30 nodes. The dashed line represents $F(Y) = \frac{c_{ave}}{\gamma_{ave}} + \frac{c_{max}}{\gamma_{min}}$ while the continuous thin line represents the lower bound to the optimal execution time.

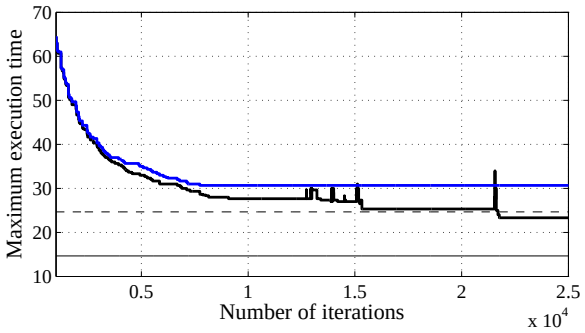

Figure 4: Comparison of the evolution of the network maximum execution time according to Algorithm 1 (thick black line) and the Discrete Consensus Algorithm in [2] (blue line) in a line network of 30 nodes. The dashed line represents $F(Y) = \frac{c_{ave}}{\gamma_{ave}} + \frac{c_{max}}{\gamma_{min}}$ while the continuous thin line represents the lower bound to the optimal execution time.

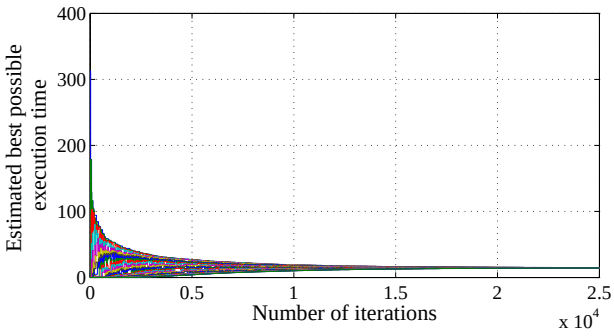

Figure 5: Evolution of the estimated lower bound on the optimal execution time $F(Y^*)$ computed as $\frac{\hat{c}_{ave,i}}{\tilde{\gamma}_{ave,i}}$ by each node.

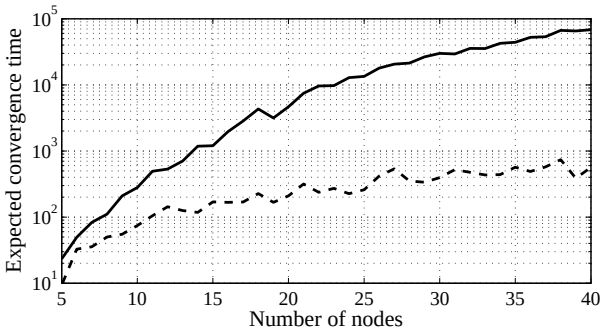

Figure 6: Expected convergence time for line graphs (continuous line) and random graphs (dashed line) with increasing numbers of nodes.

in a semi-logarithmic chart in line graphs (continuous line) and random graphs (dashed line). To allow fair comparisons we kept the average number of tasks constant and thus selected at each simulation a total number of tasks equal to $K = 6n$. It can be seen that the convergence time grows polynomially with respect to the number of nodes. Random graphs are generated with a probability of edge existence among pairs of nodes equal to $p = \frac{log(n)}{n}$. This probability of edge existence is chosen to generate graphs that with high probability have similar diameter. Comparing the simulations in Figure 6 it can be seen that the convergence time in random graphs is much smaller than in line graphs. A theoretical study of the convergence time will be carried out in future work.

8. Conclusions

In this paper we proposed a novel algorithm, the Heterogeneous Discrete Consensus (HDC) algorithm, which optimizes with guaranteed performance the execution time of a set of tasks by a network of nodes with heterogeneous execution speed exploiting only asynchronous and pairwise local state updates, i.e., gossip-based. The proposed algorithm extends the state of the art in that it guarantees the achievement of an assignment whose objective function value differs from the optimal one only by a constant function of the maximum task cost and minimum task execution speed. Therefore, the proposed distributed algorithm scales well with network size and is suitable to solve task assignment problems in large networks. We characterized the convergence properties of the algorithm and proved an absolute performance guarantee on the final computed task assignment. We proposed a distributed edge selection process with an embedded stop criterion which allows to halt also interagent communications once a task assignment with desired performance has been achieved. We discussed numerical simulations to further validate the proposed algorithm.

Future work will involve a theoretical characterization of the convergence time of the proposed algorithm.

References

[1] M. Franceschelli, A. Giua, C. Seatzu, Distributed task assignment based on gossip with guaranteed performance on heterogeneous networks, in: Analysis and Design of Hybrid Systems ADHS, 2015, pp. 218–223.

[2] M. Franceschelli, A. Giua, C. Seatzu, A gossip-based algorithm for discrete consensus over heterogeneous networks, IEEE Transactions on Automatic Control, 55 (5) (2010) 1244 –1249.

[3] A. Kashyap, T. Başar, R. Srikant, Quantized consensus, Automatica 43 (7) (2007) 1192–1203.

[4] C. Kai, H. Ishii, Quantized consensus and averaging on gossip digraphs, IEEE Transactions on Automatic Control 56 (9) (2011) 2087–2100.

[5] S. Boyd, A. Ghosh, B. Prabhakar, D. Shah, Randomized gossip algorithms, IEEE Transactions on Information Theory 52 (6) (2006) 2508–2530.

[6] C. Ravazzi, P. Frasca, R. Tempo, H. Ishii, Ergodic randomized algorithms and dynamics over networks, Control of Network Systems, IEEE Transactions on 2 (1) (2015) 78–87.

[7] A. Dimakis, S. Kar, J. Moura, M. Rabbat, A. Scaglione, Gossip algorithms for distributed signal processing, Proceedings of the IEEE 98 (11) (2010) 1847–1864.

[8] S. Riazi, C. Seatzu, O. Wigstrom, B. Lennartson, Benders/gossip methods for heterogeneous multi-vehicle routing problems, in: IEEE 18th Conference on Emerging Technologies Factory Automation (ETFA), 2013, pp. 1–6.

[9] S. Riazi, P. Chehrazi, O. Wigström, K. Bengtsson, B. Lennartson, A gossip algorithm for home healthcare scheduling and routing problems, Vol. 19, 2014, pp. 10754–10759.

[10] M. Zhu, S. Martínez, On the convergence time of asynchronous distributed quantized averaging algorithms, IEEE Transactions on Automatic Control 56 (2011) 386–390.

[11] S. Etesami, T. Başar, Convergence time for unbiased quantized consensus, in: IEEE 52nd Annual Conference on Decision and Control, 2013, pp. 6190–6195.

[12] T. Başar, S. Etesami, A. Olshevsky, Fast convergence of quantized consensus using metropolis chains, in: IEEE 53rd Annual Conference on Decision and Control, 2014, pp. 1330–1334.

[13] R. Carli, F. Fagnani, P. Frasca, S. Zampieri, Gossip consensus algorithms via quantized communication, Automatica 46 (1) (2010) 70–80.

[14] J. Lavaei, R. Murray, Quantized consensus by means of gossip algorithm, IEEE Transactions on Automatic Control 57 (1) (2012) 19 –32.

[15] M. Fanti, A. M. Mangini, W. Ukovich, A quantized consensus algorithm for distributed task assignment., in: IEEE 51st Annual Conference on Decision and Control, 2012, pp. 2040–2045.

[16] M. Fanti, M. Franceschelli, A. Mangini, G. Pedroncelli, W. Ukovich, Discrete consensus in networks with constrained capacity, in: IEEE 52nd Annual Conference on Decision and Control, 2013, pp. 2012–2017.

[17] E. Gravelle, S. Martinez, Quantized distributed load balancing with capacity constraints, in: IEEE 53rd Annual Conference on Decision and Control, 2014, pp. 3866–3871.

[18] M. Franceschelli, A. Giua, C. Seatzu, Quantized consensus in Hamiltonian graphs, Automatica 47 (11) (2011) 2495–2503.

[19] M. Franceschelli, A. Giua, C. Seatzu, Fast discrete consensus based on gossip for makespan minimization in networked systems, Automatica 56 (2015) 60–69.

[20] M. Franceschelli, D. Rosa, C. Seatzu, F. Bullo, Gossip algorithms for heterogeneous multi-vehicle routing problems, Nonlinear Analysis: Hybrid Systems 10 (1) (2013) 156–174.

[21] S. Chopra, M. Egerstedt, Spatio-temporal multi-robot routing, Automatica 60 (2015) 173 – 181.

[22] S. Chopra, M. Egerstedt, Heterogeneous multi-robot routing, in: American Control Conference, 2014, pp. 5390–5395.

[23] M. R. Garey, D. S. Johnson, Computers and intractability: a guide to the theory of np-completeness, WH Freeman & Co., San Francisco.

[24] S. M. Johnson, Optimal two- and three-stage production schedules with setup times included, Naval Research Logistics Quarterly 1 (1) (1954) 61–68.

[25] L. Babel, H. Kellerer, V. Kotov, The k-partitioning problem, Mathematical Methods of Operations Research 47 (1) (1998) 59–82.

[26] J. He, P. Cheng, L. Shi, J. Chen, Y. Sun, Time synchronization in wsns: A maximum-value-based consensus approach, IEEE Transactions on Automatic Control 59 (3) (2014) 660–675.

[27] M. Franceschelli, A. Giua, C. Seatzu, Fast discrete consensus based on gossip for makespan minimization in networked systems, Automatica 56 (2015) 60–69.

Table 1: Evolution of the task assignment in Example 3.2

Selected edge	Node 1 $\gamma_1 = 101$	Node 2 $\gamma_2 = 102$	Node 3 $\gamma_3 = 103$	Node 4 $\gamma_4 = 104$	Node 5 $\gamma_5 = 105$
t_0	$\mathcal{K}_1 = \{10\}$ $x_1 = 0.0990$ $\hat{c}_{ave,1} = 10$ $\hat{\gamma}_{ave,1} = 101$ $\hat{x}_{ave,1} = 0.0099$	$\mathcal{K}_2 = \{10, 10\}$ $x_2 = 0.196$ $\hat{c}_{ave,2} = 20$ $\hat{\gamma}_{ave,2} = 102$ $\hat{x}_{ave,2} = 0.0196$	$\mathcal{K}_3 = \{10, 10, 10\}$ $x_3 = 0.2912$ $\hat{c}_{ave,3} = 30$ $\hat{\gamma}_{ave,3} = 103$ $\hat{x}_{ave,3} = 0.0291$	$\mathcal{K}_4 = \{10, 10, 10, 10\}$ $x_4 = 0.3846$ $\hat{c}_{ave,4} = 40$ $\hat{\gamma}_{ave,4} = 104$ $\hat{x}_{ave,4} = 0.0385$	$\mathcal{K}_5 = \{10, 10, 10, 10, 10\}$ $x_5 = 0.4761$ $\hat{c}_{ave,5} = 50$ $\hat{\gamma}_{ave,5} = 105$ $\hat{x}_{ave,5} = 0.0476$
t_1 $(2, 3)$	$\mathcal{K}_1 = \{10\}$ $x_1 = 0.0990$ $\hat{c}_{ave,1} = 10$ $\hat{\gamma}_{ave,1} = 101$ $\hat{x}_{ave,1} = 0.0099$	$\mathcal{K}_2 = \{10, 10\}$ $x_2 = 0.196$ $\hat{c}_{ave,2} = 25$ $\hat{\gamma}_{ave,2} = 102.5$ $\hat{x}_{ave,2} = 0.0244$	$\mathcal{K}_3 = \{10, 10, 10\}$ $x_3 = 0.2912$ $\hat{c}_{ave,3} = 35$ $\hat{\gamma}_{ave,3} = 102.5$ $\hat{x}_{ave,3} = 0.0244$	$\mathcal{K}_4 = \{10, 10, 10, 10\}$ $x_4 = 0.3846$ $\hat{c}_{ave,4} = 40$ $\hat{\gamma}_{ave,4} = 104$ $\hat{x}_{ave,4} = 0.0385$	$\mathcal{K}_5 = \{10, 10, 10, 10, 10\}$ $x_5 = 0.4761$ $\hat{c}_{ave,5} = 50$ $\hat{\gamma}_{ave,5} = 105$ $\hat{x}_{ave,5} = 0.0476$
t_2 $(4, 5)$	$\mathcal{K}_1 = \{10\}$ $x_1 = 0.0990$ $\hat{c}_{ave,1} = 10$ $\hat{\gamma}_{ave,1} = 101$ $\hat{x}_{ave,1} = 0.0099$	$\mathcal{K}_2 = \{10, 10\}$ $x_2 = 0.196$ $\hat{c}_{ave,2} = 25$ $\hat{\gamma}_{ave,2} = 102.5$ $\hat{x}_{ave,2} = 0.0244$	$\mathcal{K}_3 = \{10, 10, 10\}$ $x_3 = 0.2912$ $\hat{c}_{ave,3} = 35$ $\hat{\gamma}_{ave,3} = 102.5$ $\hat{x}_{ave,3} = 0.0244$	$\mathcal{K}_4 = \{10, 10, 10, 10\}$ $x_4 = 0.3846$ $\hat{c}_{ave,4} = 45$ $\hat{\gamma}_{ave,4} = 104.5$ $\hat{x}_{ave,4} = 0.0431$	$\mathcal{K}_5 = \{10, 10, 10, 10, 10\}$ $x_5 = 0.4761$ $\hat{c}_{ave,5} = 45$ $\hat{\gamma}_{ave,5} = 104.5$ $\hat{x}_{ave,5} = 0.0431$
⋮	⋮	⋮	⋮	⋮	⋮
t_{10} $(4, 5)$	$\mathcal{K}_1 = \{10\}$ $x_1 = 0.0990$ $\hat{c}_{ave,1} = 17.5$ $\hat{\gamma}_{ave,1} = 101.75$ $\hat{x}_{ave,1} = 0.0172$	$\mathcal{K}_2 = \{10, 10\}$ $x_2 = 0.196$ $\hat{c}_{ave,2} = 21.25$ $\hat{\gamma}_{ave,2} = 102.12$ $\hat{x}_{ave,2} = 0.0208$	$\mathcal{K}_3 = \{10, 10, 10\}$ $x_3 = 0.2912$ $\hat{c}_{ave,3} = 33.125$ $\hat{\gamma}_{ave,3} = 103.312$ $\hat{x}_{ave,3} = 0.0321$	$\mathcal{K}_4 = \{10, 10, 10, 10\}$ $x_4 = 0.3846$ $\hat{c}_{ave,4} = 39.063$ $\hat{\gamma}_{ave,4} = 103.906$ $\hat{x}_{ave,4} = 0.0376$	$\mathcal{K}_5 = \{10, 10, 10, 10, 10\}$ $x_5 = 0.4761$ $\hat{c}_{ave,5} = 39.063$ $\hat{\gamma}_{ave,5} = 103.906$ $\hat{x}_{ave,5} = 0.0376$
t_{11} $(4, 5)$	$\mathcal{K}_1 = \{10\}$ $x_1 = 0.099$ $\hat{c}_{ave,1} = 17.5$ $\hat{\gamma}_{ave,1} = 101.75$ $\hat{x}_{ave,1} = 0.0172$	$\mathcal{K}_2 = \{10, 10\}$ $x_2 = 0.196$ $\hat{c}_{ave,2} = 21.25$ $\hat{\gamma}_{ave,2} = 102.125$ $\hat{x}_{ave,2} = 0.0208$	$\mathcal{K}_3 = \{10, 10, 10\}$ $x_3 = 0.2912$ $\hat{c}_{ave,3} = 33.125$ $\hat{\gamma}_{ave,3} = 103.312$ $\hat{x}_{ave,3} = 0.0321$	$\mathcal{K}_4 = \{10, 10, 10, 10\}$ $x_4 = 0.4807$ $\hat{c}_{ave,4} = 39.063$ $\hat{\gamma}_{ave,4} = 103.906$ $\hat{x}_{ave,4} = 0.0376$	$\mathcal{K}_5 = \{10, 10, 10, 10, 10\}$ $x_5 = 0.3809$ $\hat{c}_{ave,5} = 39.063$ $\hat{\gamma}_{ave,5} = 103.906$ $\hat{x}_{ave,5} = 0.0376$
t_{12} $(1, 2)$	$\mathcal{K}_1 = \{10\}$ $x_1 = 0.099$ $\hat{c}_{ave,1} = 19.375$ $\hat{\gamma}_{ave,1} = 101.937$ $\hat{x}_{ave,1} = 0.019$	$\mathcal{K}_2 = \{10, 10\}$ $x_2 = 0.196$ $\hat{c}_{ave,2} = 19.375$ $\hat{\gamma}_{ave,2} = 101.937$ $\hat{x}_{ave,2} = 0.019$	$\mathcal{K}_3 = \{10, 10, 10\}$ $x_3 = 0.2912$ $\hat{c}_{ave,3} = 33.125$ $\hat{\gamma}_{ave,3} = 103.312$ $\hat{x}_{ave,3} = 0.0321$	$\mathcal{K}_4 = \{10, 10, 10, 10\}$ $x_4 = 0.4807$ $\hat{c}_{ave,4} = 39.063$ $\hat{\gamma}_{ave,4} = 103.906$ $\hat{x}_{ave,4} = 0.0376$	$\mathcal{K}_5 = \{10, 10, 10, 10\}$ $x_5 = 0.3809$ $\hat{c}_{ave,5} = 39.063$ $\hat{\gamma}_{ave,5} = 103.906$ $\hat{x}_{ave,5} = 0.0376$
t_{13} $(3, 4)$	$\mathcal{K}_1 = \{10\}$ $x_1 = 0.099$ $\hat{c}_{ave,1} = 19.375$ $\hat{\gamma}_{ave,1} = 101.937$ $\hat{x}_{ave,1} = 0.019$	$\mathcal{K}_2 = \{10, 10\}$ $x_2 = 0.196$ $\hat{c}_{ave,2} = 19.375$ $\hat{\gamma}_{ave,2} = 101.937$ $\hat{x}_{ave,2} = 0.019$	$\mathcal{K}_3 = \{10, 10, 10, 10\}$ $x_3 = 0.3883$ $\hat{c}_{ave,3} = 36.094$ $\hat{\gamma}_{ave,3} = 103.609$ $\hat{x}_{ave,3} = 0.0348$	$\mathcal{K}_4 = \{10, 10, 10, 10\}$ $x_4 = 0.3846$ $\hat{c}_{ave,4} = 36.094$ $\hat{\gamma}_{ave,4} = 103.609$ $\hat{x}_{ave,4} = 0.0348$	$\mathcal{K}_5 = \{10, 10, 10, 10\}$ $x_5 = 0.3809$ $\hat{c}_{ave,5} = 39.063$ $\hat{\gamma}_{ave,5} = 103.906$ $\hat{x}_{ave,5} = 0.0376$
⋮	⋮	⋮	⋮	⋮	⋮
t_{200} $(3, 4)$	$\mathcal{K}_1 = \{10, 10\}$ $x_1 = 0.198$ $\hat{c}_{ave,1} = 30$ $\hat{\gamma}_{ave,1} = 103$ $\hat{x}_{ave,1} = 0.2913$	$\mathcal{K}_2 = \{10, 10\}$ $x_2 = 0.196$ $\hat{c}_{ave,2} = 30$ $\hat{\gamma}_{ave,2} = 103$ $\hat{x}_{ave,2} = 0.2913$	$\mathcal{K}_3 = \{10, 10, 10\}$ $x_3 = 0.2913$ $\hat{c}_{ave,3} = 30$ $\hat{\gamma}_{ave,3} = 103$ $\hat{x}_{ave,3} = 0.2913$	$\mathcal{K}_4 = \{10, 10, 10, 10\}$ $x_4 = 0.3846$ $\hat{c}_{ave,4} = 30$ $\hat{\gamma}_{ave,4} = 103$ $\hat{x}_{ave,4} = 0.2913$	$\mathcal{K}_5 = \{10, 10, 10, 10, 10\}$ $x_5 = 0.3809$ $\hat{c}_{ave,5} = 30$ $\hat{\gamma}_{ave,5} = 103$ $\hat{x}_{ave,5} = 0.2913$
⋮	⋮	⋮	⋮	⋮	⋮