A gossip-based algorithm for discrete consensus over heterogeneous networks

Mauro Franceschelli, Alessandro Giua, Carla Seatzu

Abstract

Quantized consensus assumes that the state of each node may only take nonnegative integer values. Reaching consensus under quantization is equivalent to determining a balanced assignment of identical tasks to nodes. In this paper we generalize this problem in two ways and denote the resulting framework *discrete consensus*. First, we consider tasks that are not identical: each one is characterized by its own weight. Secondly, we assume that nodes are not identical as well. As an example, in the case of task assignment, that we consider as a reference problem in this framework, nodes may have different speeds and should be assigned a total weight proportional to their speed.

We provide a gossip-based distributed algorithm that aims to minimize the maximum execution time over nodes, whose convergence to a bounded set is guaranteed. We show that the convergence time of the proposed algorithm relies ultimately on the average meeting time between two agents performing a random walk on a graph.

Published as:

M. Franceschelli, A. Giua, C. Seatzu ”A gossip-based algorithm for discrete consensus over heterogeneous networks” *IEEE Tran. on Automatic Control*, Vol. 55, No. 5, pp. 1244-1249, May 2010.

This work has been partially supported by the European Community’s Seventh Framework Programme under project DISC (Grant Agreement n. INFSO-ICT-224498)

M. Franceschelli, A. Giua and C. Seatzu are with Dip. di Ingegneria Elettrica ed Elettronica, Università di Cagliari, Italy, {mauro.franceschelli,giua,seatzu}@diee.unica.it

I. Introduction

The study of consensus networks has recently stirred much interest in the control community with a particular focus on the deep connection between consensus and algebraic graph theory [1]–[6]. In several applicative domains related to consensus the assumption that the state of each node is a continuous variable is clearly an oversimplified assumption, and it is necessary to explicitly take into account its discrete nature. This defines a new framework called *quantized consensus* [7]–[9] where the state only takes nonnegative integer values. It has been observed in [8] that reaching a consensus under this quantization constraint is equivalent to determining a balanced assignment of identical tasks to nodes. We generalize this approach, assuming that the tasks to be assigned to nodes may not be identical. Thus we assume that the state of each node is not described by an integer number, but by a collection of objects each one with its own integer weight. We call this framework *discrete consensus*.

One of the most important classes of problems that can be formulated in terms of discrete consensus is given by *task assignment*. Tasks may be generic objects, e.g., spatial locations, network resources, or classifications. In particular, recent advances in communication and computation have allowed efficient solutions to the problem of task assignment [10]. Both on-line [11], [12] and off-line [13] approaches have been proposed: on-line approaches keep into account the variations of the task environment, while off-line approaches assume that the task environment keeps constant thus a solution can be computed in advance. Processor assignment and computation of an optimal execution sequence for multiversion software is an example of this kind of problems [14].

In this paper, that is a journal version of [15], we will focus on a problem of discrete consensus over heterogeneous networks. To provide a more intuitive interpretation to the considered physical variables, our problem formulation will be given in terms of task assignment although the approach is general. We assume that a given set of tasks, that may have different weight, should be assigned to agents (nodes). Networks are denoted as heterogeneous because nodes may have different speeds. The consensus problem for this type of nets, as far as we know, has not received much attention in the control literature.

Our goal is that of determining, using *consensus* algorithms based on *gossip* [8], [16], the solution that minimizes the maximum execution time over nodes. It is based on the recent work by Kashyap *et al.* [8] and on our previous results in [17] where homogeneous networks have been considered.

Note that due to the discrete nature of tasks and to the assumption that tasks may have arbitrary weights, the optimality of the solution is not guaranteed. As discussed in [17] this is not related to our particular approach but is intrinsic in the nature of gossip, that implements at each step a pairwise optimization, and does not always yield an optimal solution. However, we prove that there exists a bounded set that contains the optimal solution that is always reachable and we study the convergence properties and the convergence time to this bounded set.

As mentioned in the literature [8], [17], in the case of discrete consensus to ensure good convergence properties it is necessary to enrich the gossip algorithm with an appropriate *swapping rule*. Whenever a balancing between two nodes is not possible, the swap "shakes" the network configuration 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.

When swaps are performed, the maximum average convergence time depends on the average meeting time of agents performing a random walk in a graph. We show that this can always be computed numerically, modeling the swap process with a Markov Chain with a single absorbing state that represents the meeting of the agents. We also discuss two particular net structures (fully connected networks and networks with ring topology) for which it is possible to compute the average meeting time analytically.

II. Problem statement

We consider a heterogeneous network of n nodes whose connections can be described by an undirected connected graph $\mathcal{G} = (V, E)$, where $V = \{1, 2, \ldots, n\}$ is the set of nodes and $E \subseteq V \times V$ is the set of edges.

We assume that K indivisible tasks should be assigned to the nodes, and an integer weight c_j, $j = 1, \ldots, K$, is associated to each task. We define a weight vector $c \in \mathbb{N}^K$ whose j-th component is equal to c_j, and 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.

To each node $i \in V$ is allocated a *load* $x_i = c^T y_i$ consisting in the sum of the costs of tasks assigned to node i that must be processed.

The *speed factor*, denoted γ_i, represents the amount of load that can be processed in a time unit by node i. In the following we denote $\gamma_{\min}$ the smallest speed in the network (clearly $\gamma_{\min} > 0$), and $c_{\max}$ the maximum weight of tasks in the network.

The task assignment we are looking for is the one that minimizes the *maximum execution time*, starting from any initial condition. Namely, if we define the load and speed vectors $x = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^T$, $\gamma = \begin{bmatrix} \gamma_1 & \gamma_2 & \dots & \gamma_n \end{bmatrix}^T$ and $\Gamma = \text{diag}(\gamma)$, we would like to minimize the following objective function:

$$f(x) = \max_{i=1,\dots,n} \frac{x_i}{\gamma_i} = \|\Gamma^{-1}x\|_\infty \tag{1}$$

under the assumption that the total load remains constant, namely $\mathbf{1}^T x = \mathbf{1}^T x(0)$, where $x(0)$ represents the initial load configuration.

Denoting $Y(t) = [y_1(t)\ y_2(t)\ \dots\ y_n(t)]$ the state of the network at time t, a centralized optimal solution to this problem can be determined solving the following integer programming problem with binary variables:

$$\begin{cases} \min V &= \|c^T Y \Gamma^{-1}\|_\infty \\ s.t. & Y\mathbf{1} = \mathbf{1} \\ & y_{i,j} \in \{0,1\} \quad \forall\ i = 1,\dots,n;\ \ j = 1,\dots,K. \end{cases} \tag{2}$$

We denote Y^* (resp., V^*) the optimal solution (resp., the optimal value of the performance index) of Problem (2).

III. Gossip algorithm

A. A distributed algorithm

We first define a task exchange process between two adjacent nodes that, while not changing the value of the objective function, modifies the load configuration.

***Definition** 3.1 (Swap):* Let us consider two nodes i and r incident on the same edge. Let $\mathcal{K}_i(t)$, resp. $\mathcal{K}_r(t)$, be the set of tasks contained in node i, resp. r, at time t.

Let us call *swap* the operation that moves the tasks in $\mathcal{K}_i(t)$ to r, and the tasks in $\mathcal{K}_r(t)$ to i at time $t+1$, reaching the distribution $\mathcal{K}_i(t+1) = \mathcal{K}_r(t)$ and $\mathcal{K}_r(t+1) = \mathcal{K}_i(t)$, provided that the objective function locally defined for the two nodes does not change, i.e.,

$$\max\left\{\sum_{j\in\mathcal{K}_{i(t+1)}}\left(\frac{c_j}{\gamma_i}\right),\sum_{j\in\mathcal{K}_{r(t+1)}}\left(\frac{c_j}{\gamma_r}\right)\right\}=$$
$$\max\left\{\sum_{j\in\mathcal{K}_{i(t)}}\left(\frac{c_j}{\gamma_i}\right),\sum_{j\in\mathcal{K}_{r(t)}}\left(\frac{c_j}{\gamma_r}\right)\right\}.$$

■

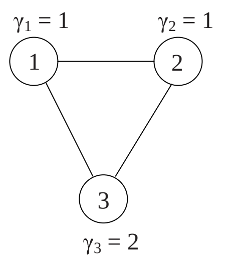
We denote $\hat{\mathcal{K}}_{ir}(t)=\mathcal{K}_i(t)\cup\mathcal{K}_r(t)$ the set of tasks present in nodes i and r at time t. We define $\hat{c}=c\uparrow\ \hat{\mathcal{K}}_{ir}(t)$ the projection of c on $\hat{\mathcal{K}}_{ir}(t)$, namely a vector whose elements are the weights of the tasks present in nodes i and r at time t. Using the same notation we define two binary vectors $\hat{y}_i=y_i\uparrow\ \hat{\mathcal{K}}_{ir}(t)$ and $\hat{y}_r=y_r\uparrow\ \hat{\mathcal{K}}_{ir}(t)$, in other words each vector has a number of elements equal to the number of tasks locally present in the nodes.

***Algorithm** 1 (Gossip Algorithm with discrete tasks):*

1) Let $t=0$.
2) Select an edge $\{i,r\}$ at random.
3) Solve the integer programming problem (IPP):

$$\left\{\begin{array}{lcl} k^* & = & \min k \\ s.t. & & \dfrac{\hat{c}^T\hat{y}_i}{\gamma_i}\le k \\ & & \dfrac{\hat{c}^T(\mathbf{1}-\hat{y}_i)}{\gamma_r}\le k \\ & & k\in\mathbb{R}_+\cup\{0\}, \\ & & \hat{y}_i\in\{0,1\}^{|\hat{\mathcal{K}}_{ir}(t)|}\end{array}\right. \tag{3}$$

4) **If** $k^*<\max\left\{\dfrac{\hat{c}^T\hat{y}_i(t)}{\gamma_i},\dfrac{\hat{c}^T(\mathbf{1}-\hat{y}_i(t))}{\gamma_r}\right\}$ **then**
   let $\hat{y}_i(t+1)=\hat{y}_i$ and $\hat{y}_r(t+1)=\mathbf{1}-\hat{y}_i$,
   else execute a swap if possible.
5) Let $t=t+1$ and goto step 2. ■



t	edge	node 1	node 2	node 3	$V(Y)$
0		4, 10		1, 2, 2, 3, 3, 5, 6, 7	14.5
1	$\{1,3\}$	4, 10		1, 2, 2, 3, 3, 5, 6, 7	14.5
2	$\{2,3\}$	4, 10	2,7	1,2,3,3,5,6	14
3	$\{1,3\}$	5, 6	2, 7	1, 2, 3, 3, 4, 10	11.5
4	$\{2,3\}$	5, 6	1, 2, 7	2, 3, 3, 4, 10	11

Fig. 1. The network discussed in Example 3.2 and the results of Algorithm 1.

In practice IPP (3) provides the task assignment that minimizes the execution time at the two nodes. If the resulting assignment is better than the previous one, tasks are assigned accordingly, otherwise a swap is executed if possible.

The swap allows to overcome several blocking conditions: anytime the network reaches a local minimum of the objective function the swap "shakes" the network ensuring convergence within some precise bounds (see Theorem 3.7).

***Example** 3.2:* Let us consider the fully connected[1] net in Fig. 1 composed by 3 nodes with speeds $\gamma_1 = \gamma_2 = 1$ and $\gamma_3 = 2$. Assume that it contains 10 tasks whose weights are equal to $c_1 = 1$, $c_2 = c_3 = 2$, $c_4 = c_5 = 3$, $c_6 = 4$, $c_7 = 5$, $c_8 = 6$, $c_9 = 7$ and $c_{10} = 10$ and let the initial configuration be $\mathcal{K}_1(0) = \{6, 10\}$, $\mathcal{K}_2(0) = \emptyset$, $\mathcal{K}_3(0) = \{1, 2, 3, 4, 5, 7, 8, 9\}$. Using Algorithm 1, we obtain the optimal task assignment in four steps, as summarized in the Fig. 1. In particular, the first line of the table summarizes the initial assignment to which it corresponds a value of the objective function that is equal to 14.5 (see the last column). The other lines point out the task assignment for $t = 1, 2, 3, 4$: in the second column we point out the selected edge, columns 3 to 5 point out the task assignment in the three nodes; finally, the last column points out the corresponding value of the objective function. ■

B. Convergence properties

The convergence properties of Algorithm 1 depend on the possibility of performing swaps.

[1]A network is *fully connected* if there is an arc from each node to any other one.

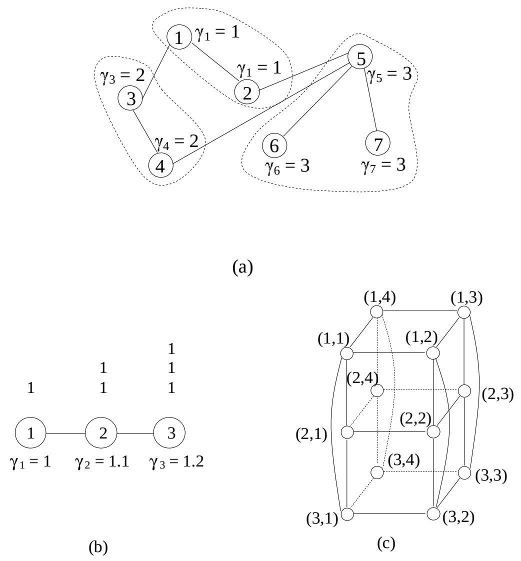


Fig. 2. (a) The network discussed in Example 3.4: (b) Network in example 3.9 (c) A net with a generalized ring topology where $s = 3$ and $k = 4$.

***Definition** 3.3 (Swap domain):* We call swap domain $G_\gamma \subseteq G$ a connected subgraph induced by nodes with the same speed. ■

***Example** 3.4:* Let us consider the network in Fig. 2.a that has seven nodes with three different speeds. This network can be partitioned in three different subgraphs G_1, G_2 and G_3 induced respectively by nodes $\{1, 2\}$, $\{3, 4\}$ and $\{5, 6, 7\}$. In this case each swap domain is connected to each other. ■

Each swap domain identifies a set of nodes where swaps may always happen. On the contrary swaps between adjacent nodes of different domains may either be possible or not, depending on the particular tasks and on the speed of nodes. As an example, if we consider the network in Example 3.2 a swap among nodes 1 and 3, that belong to different swap domains, is not allowed at time $t = 1$ because it would lead to an increasing of the maximum execution time at the two nodes. The following example shows a scenario in which a swap among nodes belonging to different swap domains is admissible.

***Example** 3.5:* Let us assume that two nodes 1 and 2 have speed equal to $\gamma_1 = 10$ and $\gamma_2 = 11$, respectively. Moreover, let us assume that node 1 only contains one task of weight 10, while the second node contains two tasks both of weight equal to 5. The corresponding maximum execution time is equal to 1. Now, no tasks exchange may occur that leads to a better tasks assignment, while a swap may happen keeping unaltered the maximum execution time at the two nodes. ■

It is relevant to note that the definition of "swap domain" is embedded in the graph topology: the nodes don't need to know in which domain they are or even that any domain exists.

***Definition** 3.6:* We call *final set*

$$\widetilde{\mathcal{Y}} = \{Y = [y_1\ y_2\ \cdots\ y_n] \mid \left|\frac{c^T y_i}{\gamma_i} - \frac{c^T y_r}{\gamma_r}\right| \le \frac{c_{\max}}{\gamma_{\min}}, \quad \forall\ i, r \in \{1, \ldots, n\}\} \tag{4}$$

i.e., the set of configurations such that, for any couple of nodes $i, r \in V$, the difference among their execution times is at most equal to the ratio $c_{\max}/\gamma_{\min}$. ■

***Theorem** 3.7:* Let $Y(t)$ be the matrix that summarizes the task assignment resulting from Algorithm 1 at the generic time t. If each swap domain is connected to each other, it holds $\lim_{t\to\infty} \Pr\left(Y(t) \in \widetilde{\mathcal{Y}}\right) = 1$ where $\Pr(Y(t) \in \widetilde{\mathcal{Y}})$ denotes the probability that $Y(t) \in \widetilde{\mathcal{Y}}$.

Proof: We define a Lyapunov-like function

$$V(t) = [V_1(t), V_2(t)] \tag{5}$$

consisting of two terms. The first one is equal to the objective function of (2), namely $V_1(t) = \|c^T Y(t)\Gamma^{-1}\|_\infty$. The second one is a measure of the number of nodes whose execution time is equal to $\|c^T Y(t)\Gamma^{-1}\|_\infty$, i.e., $V_2(t) = \left|\arg \max_{i=1,\ldots,n} \frac{c^T y_i(t)}{\gamma_i}\right|$.

Note that we impose a lexicographic ordering on the performance index, i.e., $V = \bar{V}$ if $V_1 = \bar{V}_1$ and $V_2 = \bar{V}_2$; $V < \bar{V}$ if $V_1 < \bar{V}_1$ or $V_1 = \bar{V}_1$ and $V_2 < \bar{V}_2$.

The proof is based on three arguments.

(1) We first prove that $V(t)$ is a non increasing function of t.

This is trivially true when a swap is executed, since in such a case $V(t+1) = V(t)$.

Consider the case in which the selected nodes i and r balance their load. It holds

$$\max\left\{\frac{c^T y_i(t+1)}{\gamma_i}, \frac{c^T y_r(t+1)}{\gamma_r}\right\} < \max\left\{\frac{c^T y_i(t)}{\gamma_i}, \frac{c^T y_r(t)}{\gamma_r}\right\},$$

hence three different cases may happen.

(a) One of the selected nodes is the only node in the network such that its execution time is equal to $\|c^T Y \Gamma^{-1}\|_\infty$. In such a case $V_1(t+1) < V_1(t)$ hence $V(t+1) < V(t)$.

(b) One of selected nodes is such that its execution time is equal to $\|c^T Y(t) \Gamma^{-1}\|_\infty$ but there exists at least one other node in the network with the same execution time. In such a case $V_1(t+1) = V_1(t)$ and $V_2(t+1) = V_2(t) - 1$, hence $V(t+1) < V(t)$.

(c) The execution time of both the selected nodes is smaller than $\|c^T Y(t) \Gamma^{-1}\|_\infty$. In such a case $V(t+1) = V(t)$.

(2) Secondly, we observe that, if the current configuration is outside the final set $\widetilde{\mathcal{Y}}$, then there exists at least one node whose execution time is equal to $\|c^T Y(t) \Gamma^{-1}\|_\infty$ that could balance his load with (at least) one other node if they were incident on the same arc: this would reduce function $V(t)$ (see cases (a) and (b) of the previous item).

To prove this we observe that if the current configuration is outside the final set $\widetilde{\mathcal{Y}}$, then there exists (at least) one couple of nodes i and r such that

$$\frac{c^T y_i(t)}{\gamma_i} - \frac{c^T y_r(t)}{\gamma_r} > \frac{c_{\max}}{\min\{\gamma_i, \gamma_r\}} \tag{6}$$

where $\dfrac{c^T y_i(t)}{\gamma_i}$ is equal to the maximum execution time. If we move a task $c_j \leq c_{\max}$ from node i to node r we have: $c^T y_i(t+1) = c^T y_i(t) - c_j$, and $c^T y_r(t+1) = c^T y_r(t) + c_j$. Now

$$\frac{c^T y_i(t+1)}{\gamma_i} = \frac{c^T y_i(t) - c_j}{\gamma_i} < \frac{c^T y_i(t)}{\gamma_i} \tag{7}$$

and

$$\frac{c^T y_r(t)}{\gamma_r} + \frac{c_j}{\gamma_r} \leq \frac{c^T y_r(t)}{\gamma_r} + \frac{c_{\max}}{\min\{\gamma_i, \gamma_r\}} < \frac{c^T y_i(t)}{\gamma_i}$$

where the second inequality follows from assumption (6); thus

$$\frac{c^T y_r(t+1)}{\gamma_r} = \frac{c^T y_r(t) + c_j}{\gamma_r} < \frac{c^T y_i(t)}{\gamma_i}. \tag{8}$$

By (7) and (8) it follows that

$$\max\left\{\frac{c^T y_i(t+1)}{\gamma_i}, \frac{c^T y_r(t+1)}{\gamma_r}\right\} < \max\left\{\frac{c^T y_i(t)}{\gamma_i}, \frac{c^T y_r(t)}{\gamma_r}\right\}.$$

(3) Finally, we observe that being each swap domain connected to each other, there exists a series of swaps that lead to a configuration in which the loads of the two nodes identified in the previous item are adjacent and the arc between them is selected. This happens with probability 1 as t goes to infinity. □

Remark *3.8:* Theorem 3.7 characterizes the convergence properties of Algorithm 1 in terms of a finite set $\widetilde{\mathcal{Y}}$. This obviously does not imply that an optimal task assignment is achieved. As shown in [17] this is not a limitation of the particular algorithm. To reach consensus an optimization involving more than two nodes at the same time may be necessary. ■

Finally the following example shows that if each swap domain is not connected to each other, then the convergence set $\widetilde{\mathcal{Y}}$ may not be reached by Algorithm 1.

Example *3.9:* Let us consider the network in Fig. 2.b where $\gamma_1 = 1$, $\gamma_2 = 1.1$, $\gamma_3 = 1.2$. Tasks with cost $c_1 = c_2 = \ldots c_6 = 1$ are assigned such that

$$\mathcal{K}_1(0) = \{1\}, \quad \mathcal{K}_2(0) = \{1, 1\}, \quad \mathcal{K}_3(0) = \{1, 1, 1\}.$$

It can be seen that the network is in a blocking configuration where no task exchange may happen according to Algorithm 1. As a result the convergence set $\widetilde{\mathcal{Y}}$ is not reached being

$$\left|\frac{c^T y_1}{\gamma_1} - \frac{c^T y_3}{\gamma_3}\right| = \left|1 - \frac{3}{1.2}\right| = 1.5 > \frac{c_{\max}}{\gamma_{\min}} = 1.$$

On the other hand if node 3 and node 1 were connected, then node 3 could exchange one task with node 1 and achieve a configuration included in $\widetilde{\mathcal{Y}}$. ■

IV. Convergence time of Algorithm 1

The *convergence time* is a random variable defined for a given initial task assignment $Y(0) = Y$ as: $T_{conv}(Y) = \inf\,\{t \mid \forall\, t' \geq t,\ Y(t') \in \widetilde{\mathcal{Y}}\}$. Thus, $T_{conv}(Y)$ represents the number of steps required at a certain execution of Algorithm 1 to reach the convergence set $\widetilde{\mathcal{Y}}$ starting from a given tasks distribution. Let us firstly introduce the following notation.

- $N_{\max}$ is the maximum number of improvements of $V(t)$ defined as in (5), needed by any realization of Algorithm 1 to reach the set $\widetilde{\mathcal{Y}}$, starting from a given configuration.
- $T_{\max}$ is the maximum average time between two consecutive improvements of $V(t)$ defined as in (5), needed by any realization of Algorithm 1, starting from a given configuration.

Using the previous notation, it follows that the *expected convergence time* is

$$\mathcal{E}[T_{conv}(Y)] \leq N_{\max} \cdot T_{\max}. \tag{9}$$

The following proposition provides a topology independent upper bound on $N_{\max}$.

***Proposition** 4.1:* Let us consider a net with n nodes and let γ be the corresponding speed vector. Let $x(0)$ be the vector representative of the initial amount of load at nodes. It holds:

$$N_{max} \leq (n-1) \cdot \varrho \cdot (M - m) \tag{10}$$

where

$$M = \|\Gamma^{-1} x(0)\|_\infty, \qquad m = \frac{\sum_{i=1}^{n} x_i(0)}{\sum_{i=1}^{n} \gamma_i} = \frac{\mathbf{1}^T x(0)}{\mathbf{1}^T \Gamma \mathbf{1}}, \qquad \varrho = \max_{\{i,r\} \in E} \text{lcm}\{\gamma_i, \gamma_r\}, \tag{11}$$

and lcm denotes the least common multiple.

Proof: By definition the maximum number of improvements of $V_1 = f$ needed by any realization of Algorithm 1 to reach the set $\widetilde{\mathcal{Y}}$ is smaller or equal to the ratio between the global improvement of f needed before reaching the convergence set $\widetilde{\mathcal{Y}}$ starting from $x(0)$, and its minimum admissible improvement.

By Step 5 of Algorithm 1 the task assignment is updated if and only if leads to an improvement of the objective function, otherwise a swap is executed. Thus, the largest value of $f(x)$ occurs at the initial configuration and is equal to $M = f(x(0)) = \|\Gamma^{-1} x(0)\|_\infty$.

The minimum value of $f(x)$ corresponds to the case of perfect task assignment, that in general is not achievable in the discrete case. However, a lower estimate of it is given by its optimal value in the case of infinitely divisible tasks, namely by $f(x^*)$ where $x^* = \alpha\gamma$ and $\alpha = \dfrac{\mathbf{1}x(0)}{\mathbf{1}^T\Gamma\mathbf{1}}$. Thus, if we define $m = f(x^*) = \alpha$, then for any task assignment x it holds $m \leq f(x)$.

We also observe that the minimum load exchange is equal to 1 since all tasks have an integer weight.

Now, if we consider the generic edge $\{i, r\}$, we know that the minimum improvement of f that we may obtain when balancing this edge is equal to $1/\text{lcm}\{\gamma_i, \gamma_r\}$. As a consequence the minimum improvement of f at a generic step of Algorithm 1 is equal to $1/\varrho = 1/\max_{\{i,r\}\in E}\ \text{lcm}\{\gamma_i, \gamma_r\}$, where E is the set of edges.

Thus, we may conclude that the largest number of improvements of f before reaching the convergence set $\widetilde{\mathcal{Y}}$ starting from $x(0)$ is at most equal to $\varrho \cdot (M - m)$.

Finally, in the worst case $n-1$ consecutive balancing may occur before having an improvement of f, namely $n-1$ consecutive reductions of V_2 may occur before having a reduction of $V_1 = f$. In particular, this case may happen if $n-1$ nodes have the same execution time that is equal to the maximum one. In this case, a first balancing may occur between the only "different" node and any of the other ones. Then, a new balancing may occur between any of the remaining $n-2$ nodes with the maximum execution time and one with a smaller execution time, and so on. □

We now focus on $T_{\max}$. Evaluating $T_{\max}$, and hence the average convergence time (9), is in general a difficult issue because it is strictly related to the particular topology of the net.

In the following we consider two cases: *fully connected* networks and *generalized ring topology* nets. Similar approaches based on Markov chains can always be used to evaluate numerically an upper bound on $T_{\max}$ for a particular net example.

A. Fully connected networks

***Proposition** 4.2:* Let us consider a fully connected network, and let n be the number of nodes. It holds

$$T_{\max} = \frac{n(n-1)}{2}. \tag{12}$$

Proof: The maximum average time between two consecutive balancing occurs when only one balancing is possible. Thus, if N is the number of arcs of the net, then the probability of selecting the only arc whose incident nodes may balance their load is equal to $p = 1/N$, while the average time needed to select it is equal to N. Since the network is fully connected, if n is the number of nodes, the number of arcs is $N = n(n-1)/2$ and so $T_{\max} = n(n-1)/2$. □

***Proposition** 4.3:* If a net is fully connected, the average convergence time of Algorithm 1 is

$$\mathcal{E}[T_{conv}(Y)] \leq \varrho \cdot (M - m) \cdot \frac{n(n-1)^2}{2} = \mathcal{O}(n^3).$$

Proof: Follows from equation (9) and Propositions 4.1 and 4.2. □

B. Generalized ring topology

***Definition** 4.4 (Generalized ring topology):* A graph $\mathcal{G} = (E, V)$ has a generalized ring topology if it satisfies the following assumptions.

• It is composed by s rings, each one with k nodes. The generic j-th ring R_j is a graph $R_j = (V_j, E_j)$ with $V_j = \{1, \ldots, k\}$ and $E_j = \{\{i, r\} \in E \mid r = i{+}1, \; \forall i = 1, \ldots, k{-}1\} \cup \{k, 1\}$.

• The same speed is associated to all nodes in the same ring, while nodes of different rings have different speeds. Thus each ring defines a different swap domain.

• Let (i, j), with $i = 1, \ldots, k$ and $j = 1, \ldots, s$, be the i-th node of ring R_j. Let $\Sigma_i = \{(i, j) \in V, \; j = 1, \ldots, s\}$ be the set of the nodes of *index* i in all rings. All nodes in Σ_i are fully connected, i.e., for all $i = 1, \ldots, k$, there exists an edge in E that connects each node in Σ_i with any other node in Σ_i. ■

An example of a net with a generalized ring topology is reported in Fig. 2.c: here $s = 3$, $k = 4$.

Note that such a topology well fits with our problem for two main reasons. Firstly, it is scalable both in the number of nodes in the rings and in the number of rings (namely in the number of swap domains). Secondly, the diameter of the net, namely the maximum distance among nodes that may balance, increases with the number of nodes in the ring.

***Proposition** 4.5:* Let us consider a net with a generalized ring topology. Let s be the number of rings and $n = k \cdot s$ be the total number of nodes in the net. It holds

$$T_{\max} \leq \frac{n^2(s+1)}{32 \cdot s} \cdot \left(\frac{n}{s} + 16\right) = \frac{k^2 s(s+1)}{32} \cdot (k + 16)\,. \tag{13}$$

Proof: We first observe that, due to the gossip nature of Algorithm 1 and to the random rule used to select the edges, the problem of evaluating an upper bound on $T_{\max}$ can be formulated as the problem of finding the average meeting time of two agents walking on a graph executing a random walk. In fact, the average meeting time of the two agents may be thought as the average time of selecting an edge whose incident nodes may balance their load. Note that in general more than two edges may balance their load, thus assuming that only two agents are walking on the graph provides us an upper bound on the value of $T_{\max}$. In particular, the worst case in terms of meeting time occurs when the two agents are on different rings.

In the following we compute the average meeting time using discrete Markov chains assuming that the two agents walk on different rings (worst case). For the sake of simplicity, we assume that the number of nodes k in each ring is even[2].

We call distance between two agents in nodes (i, j) and (i', j'), with $j \neq j'$, $d_{i,i'} = 1 + \min\{|i - i'|, k - |i - i'|\}$, namely the number of arcs in the shortest path connecting node i with node i'. In simple words the above distance is equal to the distance between the two agents, computed as if they were in the same ring, plus 1 due to the fact that they are on different rings. This is consistent with the assumption that, in a generalized ring topology net, any node with a given index in a certain ring is connected to all the other nodes having the same index in different rings. Therefore nodes with a unitary distance are nodes within the same section Σ. Under the assumption that k is even, the maximum distance between the two agents is equal to $D = k/2 + 1$.

The Markov chain relative to a net with an even value of k is shown in Fig. 3, thus it is a particular birth-death process. Each node (apart from the first one, named A) is characterized by an integer number that denotes the distance between the two nodes. Let us now discuss the weight of the arcs in the Markov chain.

— The weight of the arcs going from nodes i to $i + 1$, and viceversa, for $i = 2, \ldots, D - 1$ is equal to $2/N$ where $N = ks(s + 1)/2$ is the number of arcs[3]. This follows from the fact that if a net has N arcs the probability of selecting a generic edge is equal to $1/N$; moreover, if the

[2]The case of rings with an odd number of nodes k is upper bounded by the case of rings with $k + 1$ nodes.

[3]The number of arcs of a ring topology net is equal to k times the number of arcs of each section Σ, plus k times the number of arcs of each ring. Being each Σ a fully connected graph with s nodes, its number of arcs is equal to $s(s - 1)/2$. Therefore, $N = ks(s + 1)/2 + ks = ks(s + 1)/2$.

distance between the two agents is $i = 1, \ldots, D-1$, two are the edges whose selection leads to an increasing or decreasing of their distance. The same reasoning explains the weight of the arc going from $D-1$ to D and the weight of the arc going from 2 to 1.

— If the distance between the two agents is unitary (the state of the Markov chain is 1) being by assumption the two agents on different rings, it means that they are on the same section. Two different cases may occur: either we select an edge that leads to a distance equal to 2, or the edge incident on the nodes containing the agents is selected. The first case occurs with a probability equal to $4/N$; the second case occurs with a probability equal to $1/N$ and leads to the absorbing state A.

— Now, assume that the distance between the agents is equal to D. Since by assumption the two agents walk on different rings, in such a case the selection of 4 different arcs may lead to a decreasing of their distance. Therefore the arc of the Markov chain going from node D to node $D-1$ has a weight equal to $4/N$.

— Finally, the weights of all self-loops are due to the fact that the sum of the weights of arcs exiting a node is equal to 1 in a discrete Markov chain.

Given the Markov chain in Fig. 3 it is easy to compute the average hitting time of the absorbing state from any admissible distance. This can be done solving analytically the following linear system of equations:

$$(I - P')\ \tau = \mathbf{1} \tag{14}$$

where I is the D-dimensional identity matrix; P' has been obtained by the probability matrix P of the Markov chain in Fig. 3 removing the row and the column relative to the absorbing state[4]; τ is the D-dimensional vector of unknowns: its i-th component $\tau(i)$ is equal to the hitting time of the absorbing state starting from an initial distance equal to i, for $i = 1, \ldots, D$; finally, $\mathbf{1}$ is the D-dimensional column vector of ones. We found out that the worst case in terms of hitting time occurs when the two agents are at their maximum distance, i.e., for $i = D$. In particular it is $\tau(D) = \frac{n^2(s+1)}{32 \cdot s} \cdot \left(\frac{n}{s} + 16\right) = \dfrac{k^2 s(s+1)}{32} \cdot (k+16)$ where the last equality follows from the fact that $n = ks$. This proves the statement being $T_{\max} \leq \tau(D)$. $\square$

[4]It obviously holds that the hitting time of the absorbing state is null from the absorbing state itself.

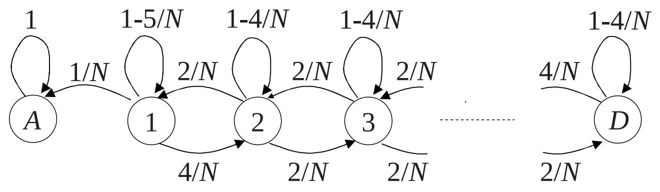


Fig. 3. The Markov chain associated to a generalized ring topology net with an even value of k.

***Proposition** 4.6:* If a net has a generalized ring topology, then the average convergence time of Algorithm 1 in terms of the number of nodes n is

$$\mathcal{E}[T_{conv}(Y)] \leq \\ \varrho \cdot (M-m) \cdot \frac{n^2(s+1)}{32 \cdot s} \cdot \left(\frac{n}{s}+16\right) \cdot (n-1) = \mathcal{O}(n^4)$$

or, in terms of the net parameters k and s

$$\mathcal{E}[T_{conv}(Y)] \leq \\ \varrho \cdot (M-m) \cdot \frac{k^2 s(s+1)}{32} \cdot (k+16) \cdot (k\ s-1) = \mathcal{O}(k^4 s^3).$$

Proof: Follows from equation (9) and Propositions 4.1 and 4.5.

□

V. Conclusions and future work

In this paper we introduced a framework denoted as *discrete consensus*, that is a generalization of *quantized consensus*. We assumed that a set of tasks of different weight should be assigned to nodes with different speeds with the aim of minimizing the maximum execution time. A solution based on gossip has been proposed and convergence properties have been examined in detail.

Future research in this topic will focus on determining appropriate rules to execute swaps to improve the convergence properties of Algorithm 1 and provide a stop criterion when the optimality set is reached. Finally, we plan to study the effects of delays in communication that often occurs in real applications.

References

[1] R. Carli, F. Fagnani, A. Speranzon, and S. Zampieri, "Communication constraints in the average consensus problem," *Automatica*, vol. 44 (3), pp. 671–684, 2008.

[2] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Trans. Automatic Control*, vol. 48, pp. 988 –1001, 2003.

[3] M. Ji, G. Ferrari-Trecate, M. Egerstedt, and A. Buffa, "Containment control in mobile networks," *IEEE Trans. on Automatic Control*, vol. 53, pp. 65–78, 2008.

[4] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. on Automatic Control*, vol. 49, pp. 1520–1533, 2004.

[5] W. Ren and R. Beard, *Distributed consensus in multi-vehicle cooperative control. Theory and applications*. Springer Verlag, 2008.

[6] G. Xie and L. Wang, "Consensus control for a class of networks of dynamic agents," *Int. Journal of Robust and Nonlinear Control*, vol. 17 (10-11), pp. 941–959, 2006.

[7] R. Carli and S. Zampieri, "Efficient quantization in the average consensus problem," *Advances in Control Theory and Applications*, pp. 31–49, 2007.

[8] A. Kashyap, T. Başar, and R. Srikant, "Quantized consensus," *Automatica*, vol. 43,7, pp. 1192–1203, 2007.

[9] T. Aysal, M. Coates, and M. Rabbat, "Distributed average consensus with dithered quantization," *IEEE Trans. on Signal Processing*, vol. 56, no. 10, pp. 4905–4918, 2008.

[10] N. Michael, M. Zavlanos, V. Kumar, and G. Pappas, "Distributed multi-robot task assignment and formation control," in *Proc. 2008 IEEE International Conference on Robotics and Automation*, Pasadena, California, May 2008.

[11] K. Lerman, C. Jones, A. Galstyan, and M. Mataric, "Analysis of dynamic task allocation in multi-robot systems," *The Int. Journal of Robotics Research*, vol. 25 (3), pp. 225–242, 2006.

[12] M. Zavlanos and G. Pappas, "Dynamic assignment in distributed motion planning with local coordination," *IEEE Trans. on Robotics*, vol. 24 (1), pp. 232–242, 2008.

[13] M. Ji, S. Azuma, and M. Egerstedt, "Role-assignment in multi-agent coordination," *Int. Journal of Assistive Robotics and Mechatronics,*, vol. 7 (1), pp. 32–40, 2006.

[14] Y.-W. Leung, "Processor assignment and execution sequence for multiversion software," *IEEE Tran. on Computers*, vol. 14 (12), pp. 1371–1377, 1997.

[15] M. Franceschelli, A. Giua, and C. Seatzu, "Load balancing over heterogeneous networks with gossip-based algorithms," in *2009 American Control Conference*, St. Louis, Missouri, USA, Jun. 2009.

[16] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip algorithms: design, analysis and applications," in *Proc. IEEE Infocom 2005*, Miami, USA, Mar. 2005.

[17] M. Franceschelli, A. Giua, and C. Seatzu, "Load balancing on networks with gossip based distributed algorithms," in *Proc. 46th IEEE Conf. on Decision and Control*, New Orleans, LA, USA, Dec. 2007.