Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Fast Discrete Consensus Based on Gossip for Makespan Optimization in Networked Systems

Mauro Franceschelli[†], Alessandro Giua[†‡], Carla Seatzu[†].

[†] *DIEE, University of Cagliari, Cagliari, Italy*
[‡] *LSIS, University of Aix-Marseille, Marseille, France*

Abstract: In this paper we propose a novel algorithm to solve the discrete consensus problem, i.e., the problem of distributing evenly a set of tokens of arbitrary weight among the nodes of a networked system. Tokens are tasks to be executed by the nodes and the proposed distributed algorithm optimizes monotonically the makespan of the assigned tasks. The algorithm is based on *gossip*-like asynchronous local interactions between the nodes. The convergence time of the proposed algorithm is superior with respect to the state of the art and grows at worst quadratically with respect to the number of nodes.

1. INTRODUCTION

The problem of *quantized consensus* consists in the design of a decentralized algorithm to steer a set of quantized state variables toward a common value. One of the early formulations of the quantized consensus problem was in Kashyap et al. (2007) where the issue of quantization to implement consensus algorithms was brought to attention and a solution inspired by distributed load balancing of quantized indivisible tasks was proposed inspired by similar approaches by Cybenko (1989); Ghosh and Muthukrishnan (1996); Herlihy and Tirthapura (2005); Houle et al. (1999). Since then, several approaches were developed to study the issues of quantization for the consensus problem. See, e.g., Cai and Ishii (2011); Carli et al. (2010); J. Lavaei and Murray (2012); Zhu and Martínez (2011); Franceschelli et al. (2010, 2011).

In Franceschelli et al. (2010) it is proposed a generalization of the quantized consensus problem called *discrete consensus*, i.e., the problem of distributing evenly a set of indivisible tokens of different weight. Thus, quantized consensus is a special case of discrete consensus where all tokens are of equal weight. An extended version of the discrete consensus algorithm characterized by an improved convergence time is proposed in Franceschelli et al. (2011) where the existence of a known Hamiltonian cycle in the network is required. The key idea of these algorithms consists in swapping the current task assignment between nodes which can not further optimize locally their current task assignment. This simple operation has shown to bring great benefits to the performance of these algorithms.

⋆ M. Franceschelli, A. Giua, C. Seatzu are with the Dept. of Electrical and Electronic Engineering, University of Cagliari, Piazza D'Armi, 09123 Cagliari, Italy. Email: {mauro.franceschelli,giua,seatzu}@diee.unica.it. A. Giua is also with LSIS, University of Aix-Marseille, Marseille, France. The research leading to these results has received funding from the European Union Seventh Framework Programme [FP7/2007-2013] under grant agreement n 257462 HYCON2 Network of excellence and by Region Sardinia, LR 7/2007 (call 2010) under project SIAR (CRP-24709).

In Fanti et al. (2012) the problem of distributed task assignment is formalized as a distributed consensus algorithm. The authors consider tasks of different cost and type which can be executed only by subsets of nodes in the network.

In Cai and Ishii (2011) a quantized consensus algorithm for networks described by directed graphs is proposed.

In Carli et al. (2010) the quantized consensus problem is formulated with nodes with continuous states but capable of transmitting only a finite number of symbols. The authors study algorithms in the framework of randomized gossip algorithms developed by Boyd et al. (2006) applying deterministic and probabilistic uniform quantizers with a combination of three local state update rules defined as partially quantized, totally quantized and compensating.

In this paper we propose a novel decentralized algorithm based on gossip to solve the *discrete consensus* problem. The proposed algorithm converges in linear time with respect to the number of nodes for graphs of given diameter.

This paper extends our results in Franceschelli et al. (2011) in that existence of an Hamiltonian cycle in the network is no longer required.

Several approaches to the study of the convergence time of quantized consensus problems investigated by Kashyap et al. (2007); J. Lavaei and Murray (2012); Zhu and Martínez (2011) are based on the computation of the average meeting time between two random walks in the graph that represents the network topology. In this paper, we propose an alternative approach to analyze the convergence time based on the computation of the expected time it takes for an edge selection process to select all edges in the graph. This analysis allows to include in a simple way the inherent parallelism of asynchronous state updates in the computation of the convergence time.

Summarizing, the contributions of this paper are the following.

- A novel algorithm to solve the discrete and quantized consensus algorithm is proposed.
- The algorithm converges almost surely in finite time, the expected convergence time grows at best linearly and at worst quadratically with respect to the number of nodes depending on the graph topology.
- With respect to Franceschelli et al. (2011), we both improve the convergence time and remove the assumption that there exists a known Hamiltonian cycle in the network.
- Simulations are provided to compare the theoretical upper bound to the convergence time with the simulated convergence time for networks with increasing number of nodes.

2. PROBLEM STATEMENT

Consider a network of n nodes whose connections 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 that represent the existence of a communication link. We consider K indivisible tokens to be assigned to the nodes, and a weight $c_j \in \mathbb{N}^+$, $j = 1, \dots, K$, is associated with each token. 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 token is assigned to node i, $y_{i,j} = 0$ otherwise. Finally, $c_{\max} = \max_{j=1,\dots,K} c_j$. To each node $i \in \mathcal{V}$ is allocated a load $x_i = c^T y_i$ consisting in the sum of the weight of tokens assigned to node i. Denoting $Y(t) = [y_1(t)\ y_2(t)\ \dots\ y_n(t)]$ the state of the network at time t, we want to achieve a network state that belongs to the set

$$\mathcal{Y} = \{Y = [y_1\ y_2\ \dots\ y_n] \ \mid \ \left|c^T y_i - c^T y_j\right| \leq c_{\max}, \ \forall i, j \in \mathcal{V}\} \quad (1)$$

under the constraint of constant total load. Let $x = [x_1\ x_2\ \dots\ x_n]^T = Y^T c$, it holds $\mathbf{1}^T x(t) = \mathbf{1}^T x(0)$ for any $t \geq 0$, where $x(0)$ represents the initial load configuration. We say that *discrete consensus* is achieved when the state of all nodes in the network satisfy condition (1).

3. PRELIMINARIES

In this section we introduce some notation and some local interaction rules that are exploited by the Fast Discrete Consensus Algorithm 1 whose statement and characterization of convergence properties are the main contribution of this paper.

We denote an arbitrary node as the *sink* node and without loss of generality we assume that it is the node with id $i = 1$. In this paper we consider the following working assumpion.

***Assumption** 3.1. Any node $i \in \mathcal{V}$ knows its* distance h_i *from the sink node, i.e., the length of the shortest path from node i to node* 1. ■

Note that $h_1 = 0$ and $h_i > 0$ for $i > 1$. We point out that the estimation of the information required by Assumption 3.1 without any a priori knowledge on the network topology and with a sink node with label unknown to all the other agents can be performed easily in a distributed and asynchronous way with simple message passing between the nodes. Due to space constraints we do not discuss this known approach in this paper, it will be presented in a future work.

***Definition** 3.2.* Given a graph $\mathcal{G} = (\mathcal{V}, E)$ we call *depth of the graph with respect to the sink node*, or shortly *depth*, the length of the greatest *distance* h_i from any node i and the sink node, i.e., $d(\mathcal{G}) = \max_{i \in \mathcal{V}} h_i$. ■

Obviously, for any choice of the sink node the depth is always smaller than or equal to the diameter of $\mathcal{G}$. To each node $i \in \mathcal{V}$ at any time instant t we associate two state variables: $x_i(t)$ is equal to the load associated with the i-th node at time t; $z_i(t)$ is equal to the local estimation of the sink load at time t of node i. Clearly, the sink node always estimates correctly its own load and it holds $z_1(t) = x_1(t)$ for all $t \geq 0$.

Let $\mathcal{K}_i(t)$ be the set of indices of tokens assigned to node i at time t. We denote $\hat{\mathcal{K}}_{ij}(t) = \mathcal{K}_i(t) \cup \mathcal{K}_j(t)$ the set of tokens present in nodes i and j at time t. We define $\hat{c}(t) = c \uparrow \hat{\mathcal{K}}_{ij}(t)$ the projection of c on $\hat{\mathcal{K}}_{ij}(t)$, namely a vector whose elements are the weights of the tokens present in nodes i and j at time t. Using the same notation we define two binary vectors $\hat{y}_i(t) = y_i(t) \uparrow \hat{\mathcal{K}}_{ij}(t)$ and $\hat{y}_j(t) = y_j(t) \uparrow \hat{\mathcal{K}}_{ij}(t)$, in other words each vector has a number of elements equal to the number of tokens locally present in the nodes.

We now introduce a local state update rule to average the load of two nodes incident on the selected edge. The rule computes a new token assignment $\mathcal{K}_i(t_{k+1})$, $\mathcal{K}_j(t_{k+1})$ given the token assignment $\mathcal{K}_i(t_k)$, $\mathcal{K}_j(t_k)$ at $t = t_k$. This local state update rule between nodes i and j involves a heuristic that was presented in Franceschelli et al. (2007). This heuristic can be summarized as follows.

***Rule** 3.3.* (**Balancing rule**).

(1) Let $\mathcal{K} = \mathcal{K}_i \cup \mathcal{K}_j$.
(2) Let $\mathcal{K}'_i := \emptyset$ and $\mathcal{K}'_j := \emptyset$ *(we define new temporary sets, both initialized to the empty set, including the indices of tokens in the selected nodes).*
(3) **While** $\mathcal{K} \neq \emptyset$, **do**
- let $\delta := \text{argmax}_{j \in \mathcal{K}} c_j$;
- **if** $\sum_{r \in \mathcal{K}'_i} c_r \leq \sum_{r \in \mathcal{K}'_j} c_r$, then let $\mathcal{K}'_i := \mathcal{K}'_i \cup \{\delta\}$, $\mathcal{K}'_j := \mathcal{K}'_j$; *(if the load of the i-th node is smaller than or equal to that of the j-th node, then assign the current token to node i)*
  else let $\mathcal{K}'_i := \mathcal{K}'_i$, $\mathcal{K}'_j := \mathcal{K}'_j \cup \{\delta\}$.
- $\mathcal{K} := \mathcal{K} \setminus \{\delta\}$

endwhile

(4) **if** $\left|\sum_{j \in \mathcal{K}'_i} c_j - \sum_{j \in \mathcal{K}'_j} c_j\right| \leq \left|\sum_{r \in \mathcal{K}_i} c_r - \sum_{r \in \mathcal{K}_j} c_r\right|$,
and $\mathcal{K}'_i \neq \mathcal{K}_i(t_k)$ or $\mathcal{K}'_j \neq \mathcal{K}_j$ *(we find a solution that is neither worse nor equal to the previous one)* **then** let $\mathcal{K}_i := \mathcal{K}'_i$, $\mathcal{K}_j := \mathcal{K}'_j$. ■

This balancing rule can be completed in $|\mathcal{K}|$ steps, thus with linear complexity with respect to the number of tokens contained in node i and j. The resulting load configuration reduces the maximum load by making the load difference surely less than or equal to $|c_{max} - c_{min}|$

as in Franceschelli et al. (2007). It can be shown that the proposed heuristic provides an assignment with absolute performance guarantee of error less than c_{max} with respect to the optimal solution of the corresponding optimization problem.

Let us now introduce an elaborate local state update rule which exploits Rule 3.3 to locally balance the tokens between pairs of nodes. In the following we denote $x_i(t_k) = c^T y_i(t_k)$ the sum of the costs of tasks assigned to node i at time t_k.

***Rule** 3.4.* (**Local State Update Rule**).

(1) Let $r = \arg\min_{i,j}\{h_i, h_j\}$ and $s \in \{i, j\}$ with $s \neq r$.
(2) **If** $|x_i(t_k) - x_j(t_k)| > c_{max}$ **then** apply the **Balancing Rule** 3.3.
endif.
Assign the largest load to the node closest to the sink according to

$$\begin{cases} p = \arg\max_{i,j}\{x_i, x_j\}; \\ q = \arg\min_{i,j}\{x_i, x_j\}; \\ \mathcal{K}_r = \mathcal{K}_p \\ \mathcal{K}_s = \mathcal{K}_q \end{cases} \tag{2}$$

(3) **If** $r = 1$ (sink node) **then**

$$\begin{cases} z_r(t_{k+1}) = x_r, \\ z_s(t_{k+1}) = x_r. \end{cases} \tag{3}$$

else

$$\begin{cases} z_r(t_{k+1}) = z_r(t_k), \\ z_s(t_{k+1}) = z_r(t_k). \end{cases} \tag{4}$$

endif.

■

In simple words, Step 2 balances the load of the two given nodes, when possible, and assigns the largest load to the node r closest to the sink node. In Step 3 the node s furthest from the sink node updates its estimate by adopting the estimate of node r closest to it. Node r keeps its estimate of the sink load unaltered. If the sink node itself is involved, then both nodes update their estimation with the actual load of the sink node.

Next we define a further local state update rule that consists only in swapping the current task assignment between two nodes.

***Rule** 3.5.* (**Swap**). Let $\mathcal{K}_i(t_k)$ and $\mathcal{K}_j(t_k)$ be the set of tasks assigned to nodes i and j and time t_k. A *swap* is a local state update that exchanges both sets of tasks as follows

$$\begin{cases} \mathcal{K}_i(t_{k+1}) = \mathcal{K}_j(t_k) \\ \mathcal{K}_j(t_{k+1}) = \mathcal{K}_i(t_k) \end{cases} \tag{5}$$

■

The aim of the *swap* operation is to ensure that the case when no node can balance its load with any of its neighbors may happen only if the network has reached the *discrete consensus* state.

4. PROPOSED ALGORITHM

In this section we state the main contribution of this paper, namely the *Fast Discrete Consensus* (FDC) algorithm.

***Algorithm** 1.* (Fast Discrete Consensus).

(1) Let $k = 0$, $t_k = 0$, $x_i(t_k) = c^T y_i(t_k)$, $z_1(t_k) = x_1(t_k)$ and $z_i(t_k) = 0$ for $i = 2, \ldots, n$.
(2) Select an edge (i, j) according to an edge selection process.
(3) Apply the **Local State Update Rule** 3.4.
(4) Let s be the index of the node furthest from the sink node.
If $z_s - x_s > c_{max}$, **then** execute a **Swap** of the updated states according to Rule 3.5
(5) Let $k = k + 1$ and go back to Step 2.

■

Due to space constraints we do not discuss a stop criterion for Algorithm 1. However, it can be shown that after *discrete consensus* has been achieved there exists a maximum interval of time after which no more load exchanges between nodes may happen.

We now briefly explain the main steps of Algorithm 1. At Step 3 the local state update rule balances the loads of the nodes whenever possible and makes sure that the largest load is assigned to the node closest to the sink. Moreover it updates the estimation of the sink load.

Then, at Step 4 if the node furthest from the sink has a load sufficiently small such that a balancing is possible with the sink node according to its estimation, then loads are swaped between the nodes. This is to ensure that loads that can be balanced with the sink node are moved closer to it at each iteration.

Note that the above algorithm iterates on the number of selected edges. This implies that several edges may perform the state update rule simultaneously with the only exception of two edges incident on the same node. Moreover, Algorithm 1 exploits only locally available information at each step.

Next we show an example of execution of Algorithm 1 to further clarify its functioning.

***Example** 4.1.* Consider the network of 4 nodes depicted in Fig. 1. Nodes are labeled from 1 to 4. The sink node is node 1. Thus, node 2 has distance 1 with respect to the sink node, node 3 has distance 2 and node 4 has distance 3. In this example for sake of clarity we consider tokens of unitary weight $c_j = 1$ for $j = 1, \ldots, K$, thus representing a standard quantized consensus problem. The initial state of the network is at $t = t_0$: $x_1(t_0) = 2$, $x_2(t_0) = 8$, $x_3(t_0) = 4$, $x_4(t_0) = 3$. All variables containing the estimation of the sink load are initialized to zero except for the sink node, thus $z_1(t_0) = 2$, $z_2(t_0) = z_3(t_0) = z_4(t_0) = 0$. In this example we consider an arbitrary edge selection sequence for sake of clarity.

(1) At t_0 edge $(1, 2)$ is selected. Since $|x_1 - x_2| > c_{max}$, the nodes balance their load leading to $x_1 = x_2 = 5$. Then they update their estimations of the sink load that are set to $z_1 = z_2 = 5$.
(2) At t_1 edge $(2, 3)$ is selected. Since $|x_2 - x_3| \leq c_{max}$ and the largest load is already in the node closest to the sink, then the nodes update only the sink load estimate to $z_2 = 5$ and $z_3 = 5$.

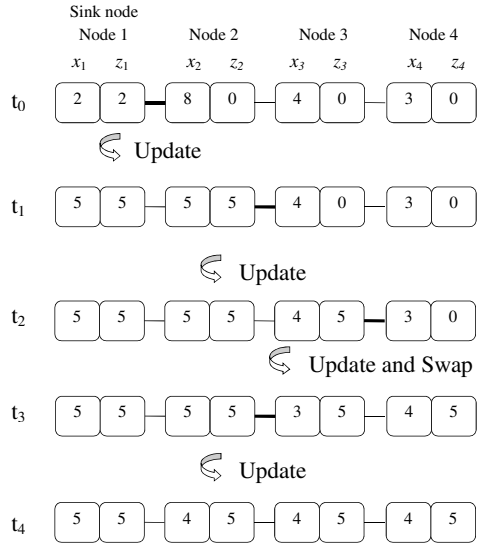

Fig. 1. Example of execution of Algorithm 1 described in Example 4.1.

(3) At t_2 edge $(3,4)$ is selected. Since $|x_3 - x_4| \leq c_{max}$ no balancing occurs but estimations are updated as $z_3 = 5$ and $z_4 = 5$. Furthermore, since $z_4 - x_4 > c_{max}$ a swap is executed.
(4) At t_3 edge $(2,3)$ is selected. Since $|x_2 - x_3| > c_{max}$ the nodes balance their load.
(5) At t_4 the discrete consensus condition (1) is reached and the state of the network can not change anymore by the execution of Algorithm 1.

5. CONVERGENCE PROPERTIES

In this section we study the convergence properties of Algorithm 1 in the case in which nodes interact according to an edge selection process. In particular, we now formally define a *stochastically persistent* edge selection process.

***Definition** 5.1.* (Edge Selection Process). An edge selection process $\mathfrak{e} : \mathbb{R}^+ \times E \to \{0,1\}$ is a function $\mathfrak{e}$ that associates with each time instant $t \in \mathbb{R}^+$ and each edge $(i,j) \in E$ a binary value: if $\mathfrak{e}(t,(i,j)) = 1$ then edge (i,j) is active at time t, not active otherwise.

We denote as

$$\hat{E}(t, t+T) = \{(i,j) \in E : \mathfrak{e}(\tau,(i,j)) = 1 \text{ for some } \tau \in [t, t+T]\}$$

the set of edges selected at least once during the time interval $[t, t+T]$. ■

We now recall a definition that has been used in Franceschelli et al. (2013) to prove almost sure finite time stability in gossip based vehicle routing problems.

***Definition** 5.2.* (Stochastic persistence). An edge selection process $\mathfrak{e} : \mathbb{R}^+ \times E \to \{0,1\}$ is said to be *stochastically persistent* if $\forall t \geq 0$ there exists a finite $T > 0$ and a probability $p \in (0,1)$ such that

$$Pr(\hat{E}(t, t+T) \equiv E) \geq p \tag{6}$$

where $Pr(\cdot)$ denotes a probability. ■

Stochastic persistence implies that, if we consider a finite but sufficiently large time interval, then each edge has a probability greater than or equal to a finite value p of being selected during such an interval. For instance, any Markov process is stochastically persistent.

***Definition** 5.3.* Given a stochastically persistent edge selection process we define a continuous random variable τ that represents the smallest interval of time in which $\hat{E}(t, t+\tau) \equiv E$, independently from the initial time t. ■

The expected value $E[\tau]$ of the random variable τ can be computed directly from the Definition 5.2 of stochastically persistent edge selection process as $E[\tau] \leq \sum_{i=1}^{\infty} iTp(1-p)^{i-1}$.

Let us now introduce two preliminary results. First, we consider the case in which the load of the sink node does not change for some time. Proposition 5.4 characterizes in what interval of time the generic node estimates correctly the load of the sink node under such assumption.

***Proposition** 5.4.* Consider a stochastically persistent edge selection process $\mathfrak{e}$. Let τ_i, $i = 1, \ldots, d(\mathcal{G})$, be $d(\mathcal{G})$ realizations of the continuous random variable τ defined as in Definition 5.3, which represents the time such that $\hat{E}(T_{i-1}, T_i) \equiv E$, where $T_i = T_{i-1} + \tau_i$ for $i = 1, \ldots, d(\mathcal{G})$. Let $T_{d(\mathcal{G})} = \sum_{i=1}^{d(\mathcal{G})} \tau_i$. If $x_1(t) = x_1$ for $t \in [t_1, t_2]$ with $t_2 - t_1 \geq T_{d(\mathcal{G})}$, then

$$z_i(t) = x_1 \qquad \forall t \in \left[t_1 + T_{d(\mathcal{G})}, t_2\right], \quad \forall i \in \mathcal{V}.$$

Proof: Due to space constraints we do not include a complete proof in this preliminary manuscript. □

The following proposition guarantees that if the number of nodes whose load is maximum in the network remains constant for a sufficiently long time, then the sink node within a given time interval is guaranteed to hold the maximum load in the network.

***Proposition** 5.5.* Consider a stochastically persistent edge selection process $\mathfrak{e}$. Let τ_i, $i = 1, \ldots, 2d(\mathcal{G})$, be $2d(\mathcal{G})$ occurrences of the continuous random variable τ defined as in Definition 5.3, namely such that $\hat{E}(T_{i-1}, T_i) \equiv E$, where $T_i = T_{i-1} + \tau_i$. Let $T_{2d(\mathcal{G})} = \sum_{i=1}^{2d(\mathcal{G})} \tau_i$. If in a time interval $[t_1, t_2]$ with $t_2 - t_1 \geq T_{2d(\mathcal{G})}$, both the maximum load and the number of nodes with the maximum load remain constant, then

$$x_1(t) = \max_{i \in \mathcal{V}} x_i(t) \qquad \forall t \in \left[t_1 + T_{2d(\mathcal{G})}, t_2\right].$$

Proof: Due to space constraints we do not include a complete proof in this preliminary manuscript. □

We now prove that, in the case of a stochastically persistent edge selection process, Algorithm 1 converges almost surely in finite time to discrete consensus.

***Theorem** 5.6.* Consider a network $\mathcal{G}$ of n nodes that executes Algorithm 1. If the edge selection process $\mathfrak{e}$ is stochastically persistent, then

$$Pr\left(\exists T_{conv} \in \mathbb{R}^+ \ : \ \forall t \geq T_{conv}, \ \ Y(t) \in \mathcal{Y}\right) = 1$$

where $\mathcal{Y}$ is defined as (1) and $Pr(\cdot)$ denotes a probability.

Proof: We define a Lyapunov-like function

$$V(t) = \|c^T Y(t)\|_\infty = \|x(t)\|_\infty \tag{7}$$

The proof is based on three main arguments.

(1) First, we prove that $\forall t_k \geq 0$ it holds $V(t_{k+1}) \leq V(t_k)$.

If there is only one node with the maximum load and such a node is one of the two selected nodes, as soon as the local state update rule in Definition 3.4 is applied, it holds $V(t_{k+1}) \leq V(t_k)$.

If none of the selected nodes has the maximum load, it holds $V(t_{k+1}) = V(t_k)$. The same occurs when one of the selected nodes has the maximum load but there also exist other nodes with the maximum load.

(2) Whenever $V(t_{k+1}) < V(t_k)$, the local maximum must decrease of at least 1 being $c_j \in \mathbb{N}$ for all $j = 1, \ldots, K$, thus $V(t_{k+1}) \leq V(t_k) - 1$.

(3) We now prove that if at a given time instant t_k it is

$$\max_{i,j \in \mathcal{V}} |x_i(t_k) - x_j(t_k)| > c_{max} \tag{8}$$

then, after a number of iterations of Algorithm 1 at most equal to $\Delta = 4(n-1)d(\mathcal{G})T$ it is $V(t_k + \Delta) < V(t_k)$, where $T > 0$ is the length of a finite time interval such that $\hat{E}(t, t+T) \equiv E \ \forall t \geq 0$.

By Proposition 5.5 if the maximum load and the number of nodes with the maximum load remain constant, then if the edge selection process is stochastically persistent there exists $T_{d(\mathcal{G})} = \sum_{i=1}^{d(\mathcal{G})} \tau_i$ such that after $2T_{d(\mathcal{G})}$ units of time the sink node has the maximum load.

Moreover, by Proposition 5.5 if the maximum load and the number of nodes with the maximum load remain constant, then the value of the sink node remains constant as well. By Proposition 5.4 after a further time equal to $T_{d(\mathcal{G})}$ all nodes have an exact estimation of the sink load. If inequality (8) holds, then there exists at least one node whose load is smaller than the maximum load of a quantity larger than $c_{\max}$. After further $T_{d(\mathcal{G})}$ units of time the execution of swaps reduce surely by at least one unit the distance between a load that can balance with the sink node and the sink node itself. The distance of such a node is at most equal to the depth $d(\mathcal{G})$, then after further $T_{d(\mathcal{G})}$ units of time such a node balances with the sink node, thus reducing the number of nodes verifying inequality (8). The same reasoning can be repeated until there exists some node verifying inequality (8). Since such a number may be at most equal to $n-1$, then inequality (8) cannot hold for a time larger than $(n-1)T_{4d(\mathcal{G})}$.

According to Definition 5.2 $\forall t \geq 0$ there exists a finite $T > 0$ and a probability $p \in (0, 1)$ such that

$$Pr(\hat{E}(t, t+T) \equiv E) \geq p. \tag{9}$$

therefore, the probability that the event $\hat{E}(t, t+T) \equiv E$ occurred in disjoint time intervals at least $(n-1)4d(\mathcal{G})$ times in an interval of time of length kT is at least

$$Pr(\ \hat{E}(t_i, t_i + T) \equiv E, i = 1, \ldots (n-1)4d(\mathcal{G}), \text{ with } t_{(n-1)4d(\mathcal{G})} \leq (n-1)T_{4d(\mathcal{G})}) \geq p^{(n-1)4d(\mathcal{G})}. \tag{10}$$

Therefore, since the probability of occurrence of the above event over a finite interval of time is strictly positive, it directly follows that

$$\lim_{i \to \infty} Pr(\hat{E}(t_i, t_i+T) \equiv E, \text{ at least } (n-1)4d(\mathcal{G}) \text{ times}) = 1.$$

Since the condition $|x_i(t) - x_j(t)| \leq c_{max} \ \forall i, j \in \mathcal{V}$ is achieved with a finite number of occurrences of the event $\hat{E}(t_i, t_i + T) \equiv E$ it follows that with probability one (almost surely), $\exists T_{conv} \in \mathbb{R}^+$ such that $\forall t \geq T_{conv}$, and $\forall i, j \in \mathcal{V}$, it is $|x_i(t) - x_j(t)| \leq c_{max}$, thus proving the statement of the Theorem. □

6. EXPECTED CONVERGENCE TIME

In this section we determine the expected convergence time of Algorithm 1 when edges are selected according to a stochastically persistent process.

***Proposition** 6.1.* Let τ be the continuous random variable introduced in Definition 5.3. If the edge selection process $\mathfrak{e}$ is stochastically persistent, then

$$E[T_{conv}] \leq 4(M-m)(n-1)d(\mathcal{G})E[\tau], \tag{11}$$

where

$$M = \|x(0)\|_\infty, \quad m = \frac{\mathbf{1}^T x(0)}{n}, \tag{12}$$

and $E[\cdot]$ denotes the expected value.

Proof:

The proof is carried out in three steps.

(a) Using the same arguments as in item 3 of Theorem 5.6 the time between two consecutive improvements of $V(t)$ is at most equal to $T_{conv} \leq \sum_{i=1}^{4N(n-1)d(\mathcal{G})} \tau$, where N is the maximum number of improvements of $V(t)$ defined as in (7), needed by any realization of Algorithm 1 to reach the set $\mathcal{Y}$, starting from any initial token assignment.

(b) We prove that $N \leq M - m$.
By definition $M - m$ is the difference between the maximum load and the average load at the initial configuration. Therefore, since the smallest decrement of $V(t)$ is equal to one, the maximum number of decrements of $V(t)$ is less or equal to $M - m$.

(c) Finally, if edges are selected by a independent and identical stochastic processes, the expected convergence time depends on the expected value of T_{conv} which can be bounded as $E\left[T_{d(\mathcal{G})}\right] \leq 4(M-m)(n-1)d(\mathcal{G})E[\tau]$. □

Since Algorithm 1 consists only of local and asynchronous state updates between nodes, we now consider the case where there exists an independent stochastic process for each edge in the network that in parallel governs the activation of each edge.

***Proposition** 6.2.* Let τ_e be a random variable that represents the time interval between two consecutive selections of a given edge by its own stochastic process. If edges are selected according to independent and identical *stochastically persistent* processes, then

$$E[T_{conv}] \leq 4(M-m)(n-1)d(\mathcal{G})E[\tau_e], \tag{13}$$

where

$$M = \|x(0)\|_\infty, \quad m = \frac{\mathbf{1}^T x(0)}{n}, \tag{14}$$

and $E[\cdot]$ denotes the expected value.

Proof: The proof follows from Proposition 6.1 and the fact that we are considering independent and identical *stochastically persistent* processes to activate each edge. This implies that the average time for the selection of all edges is equal to the average interval of time for activation of any given edge, i.e., the expected value of random variable τ_e corresponds to the expected value of random variable τ in Definition 5.3. □

We now consider as an example the case where edges are chosen according to Poisson stochastic processes with parameter λ_e, which is a stochastically persistent process.

The probability that a given edge $e \in E$ is selected k times in the unit of time is

$$P[k] = e^{-\lambda_e} \frac{\lambda_e^k}{k!}, \quad k = 0, 1, \ldots.$$

As a consequence the continuous random variable τ_e defining the waiting time among two consecutive selections of the same edge follows an exponential distribution and its expected value is $E[\tau_e] = \frac{1}{\lambda_e}$.

Summarizing, the expected convergence time of Algorithm 1 if edges are selected according to a Poisson process with parameter λ_e is upper bounded by

$$E[T_{conv}] \leq 4(M-m)(n-1)\frac{d(\mathcal{G})}{\lambda_e} \approx \mathcal{O}(n)\, d(\mathcal{G}). \quad (15)$$

7. NUMERICAL SIMULATIONS

A series of numerical simulations have been carried out to compare the convergence time of Algorithm 1 resulting from numerical simulations with the upper bound defined in equation (13). Results of such a comparison are shown in Fig. 2 where the continuous blue line represents the average value of the convergence time over 100 executions of Algorithm 1 over networks randomly generated with an increasing number of nodes and a constant value of the depth equal to 4. The dashed black curve represents the upper bound defined in equation (13). As it can be noted the upper bound is much larger than the convergence time resulting from simulations and their difference increases as the number of nodes increases. The considered convergence time is computed as the number of times that all edges are selected at least once, in particular we choose $T = 1$. This means that if the network has a large number of nodes, then more parallel state updates may to occur during the same time period T. This is in line with the fact that Algorithm 1 iterates over the number of selected edges, thus allowing parallel executions of the state update rules.

8. CONCLUSIONS

In this paper we have presented a novel decentralized algorithm that solves the discrete and quantized consensus problem. It has been shown that the proposed algorithm has improved convergence time with respect to other algorithms in the literature. Simulations have been provided to compare the theoretical upper bound to the expected convergence time with the simulated convergence time for networks with increasing number of nodes.

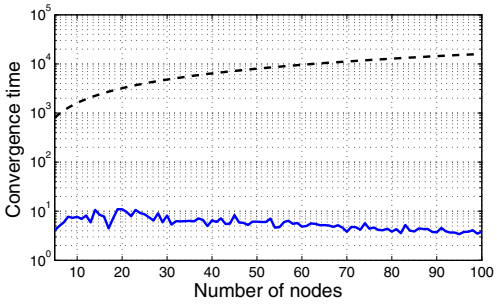

Fig. 2. Comparison between simulated convergence time and the upper bound defined in equation (13).

REFERENCES

S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE Trans. on Information Theory*, 52(6):2508–2530, 2006.

Kai Cai and H. Ishii. Quantized consensus and averaging on gossip digraphs. *IEEE Transactions on Automatic Control,*, 56(9):2087 –2100, 2011.

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

G. Cybenko. Dynamic load balancing for distributed memory multiprocessors. *J. of Parallel and Distributed Computing*, 7:279–301, 1989.

M.P. Fanti, A.M. Mangini, and W. Ukovich. A quantized consensus algorithm for distributed task assignment. *Proceedings of the IEEE Conference on Decision and Control*, pages 2040–2045, 2012.

M. Franceschelli, A. Giua, and C. Seatzu. Load balancing on networks with gossip-based distributed algorithms. *Proceedings of the IEEE Conference on Decision and Control*, pages 500–505, 2007.

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

M. Franceschelli, A. Giua, and C. Seatzu. Quantized consensus in Hamiltonian graphs. *Automatica*, 47(11): 2495 – 2503, 2011.

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

B. Ghosh and S. Muthukrishnan. Dynamic load balancing by random matchings. *J. of Computer and Systems Sciences*, 53(3):357–370, 1996.

M. Herlihy and S. Tirthapura. Self-stabilizing smoothing and balancing networks. *Distributed Computing*, 2005.

M.E. Houle, E. Tempero, and G. Turner. Optimal dimension exchange token distribution on complete binary trees. *Theoretical Computer Science*, 220:363–377, 1999.

R.M. J. Lavaei and Murray. Quantized consensus by means of gossip algorithm. *IEEE Transactions on Automatic Control*, 57(1):19 –32, 2012.

A. Kashyap, T. Başar, and R. Srikant. Quantized consensus. *Automatica*, 43(7):1192–1203, 2007.

M. Zhu and S. Martínez. On the convergence time of asynchronous distributed quantized averaging algorithms. *IEEE Transactions on Automatic Control*, 56:386–390, 2011.