Quantized Consensus in Hamiltonian graphs

Mauro Franceschelli, Alessandro Giua, Carla Seatzu

Abstract

The main contribution of this paper is an algorithm to solve an extended version of the quantized consensus problem over networks represented by Hamiltonian graphs, i.e., graphs containing a Hamiltonian cycle, which we assume to be known in advance. Given a network of agents, we assume that a certain number of tokens should be assigned to the agents, so that the total number of tokens weighted by their sizes is the same for all the agents. The algorithm is proved to converge almost surely to a finite set containing the optimal solution. A worst case study of the expected convergence time is carried out, thus proving the efficiency of the algorithm with respect to other solutions recently presented in the literature. Moreover, the algorithm has a decentralized stop criterion once the convergence set is reached.

Published as:

Mauro Franceschelli, Alessandro Giua, Carla Seatzu, "Quantized Consensus in Hamiltonian graphs," *Automatica*, 2011. Published on-line with *doi:10.1016/j.automatica.2011.08.032*.

M. Franceschelli, A. Giua and 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.

I. INTRODUCTION

In this paper we consider a problem of quantized consensus over a Hamiltonian graph, using a gossip algorithm. The limitation of the proposed algorithm is that a Hamiltonian cycle in the network must be known in advance.

Recently a fair effort has been devoted to the problem of quantized consensus, i.e., the consensus problem over a network of agents with quantized state variables [2], [9], [16], [26], as a practical implementation of the continuous one [3], [25], [31], [33], [34]. Such problem has relevant applications such as sensor networks, task assignment and token distribution over networks (a simplified load balancing problem) [11], [21], [22], [24]. In the case of sensor networks, the quantized distributed average problem arises from the fact that sensor measurements are inevitably quantized given the finite amount of bits used to represent variables and the finite amount of bandwidth of the communication links between the nodes. Some approaches [8] deal with quantization by adding a quantization noise in the communication links to model such effect and study the resulting convergence properties without modifying the algorithms. Other approaches propose probabilistic quantization [1], [2] to ensure that after a certain amount of time each node has exactly the same value, even though it might be slightly different from the actual initial average of the measurements.

Some years ago in [26] it was originally proposed an algorithm to solve the distributed average problem with uniformly quantized measurements. Such an algorithm guarantees that almost surely the state of all the agents (x_i, $i = 1, \ldots, n$) will reach a value that is either equal to the floor of the average of the net (L), or the ceil ($L + 1$), i.e., it ensures that the net will almost surely reach the convergence set

$$\mathcal{S} \triangleq \{\mathbf{x} : \{x_i\}_1^N \in \{L, L+1\}, L = \lfloor N^{-1} \sum_{i=1}^{N} x_i \rfloor\}.$$

However, a stopping criterion is missing, i.e., load transfers may occur even if the convergence set $\mathcal{S}$ is reached.

Several works followed the pioneering work in [26]. In [7] three quantized consensus algorithms are proposed which achieve a comparable performance respect to the one in [26]. In [37] quantized consensus over random and switching graphs is addressed and polynomial upper-bounds to the convergence time are provided. In [28], [29] quantized gossip algorithms are

investigated in the case of edges with different weights corresponding to different probabilities of being chosen.

In this paper we propose an algorithm to solve the quantized distributed average problem using a gossip algorithm [5]. Our algorithm can be applied to the token distribution problem, i.e., the problem of evenly distribute a set of tokens among the agents [26]. We investigated the extension of this problem to the distribution of tokens of arbitrary size [16]. Our algorithm presents two main advantages with respect to other applications and approaches in the literature:

1) A decentralized stopping criterion.
2) An expected convergence time reduced with respect to [16], [26].

Moreover, let us observe that in our approach tokens may have different size. However, in the particular case of tokens with the same size our convergence set coincides with the convergence set in [26], defined as *quantized consensus*.

Our work has three main differences with respect to [7], [28], [29], [37]. First, we consider tokens with arbitrary and possibly different size or cost as in [15], [16]. Second, we consider Hamiltonian graphs, i.e., graphs in which a Hamiltonian cycle exists. Third, we propose a novel interaction rule to be applied when no averaging due to quantization issues can be applied, that improves the convergence time of the algorithm by reducing the average meeting time of two random walks in graph. Since the convergence time of all the quantized gossip and consensus algorithms proposed in [7], [28], [29], [37] depend upon the average meeting time of two random walks in a graph, we propose as future work to improve the convergence times of such algorithms with the ideas proposed in this paper.

We remark that the issue of providing a stop criterion has already been solved by other authors using non uniform quantization, e.g., probabilistic or logarithmic quantization [2], [9]. However, uniform quantization is surely easier to implement and less cost consuming than the other types of quantization. Moreover, in [2], [9] a convergence set is not defined, and the convergence properties are given in terms of probability.

Finally, our algorithm is based on gossip, i.e., only adjacent nodes asynchronously exchange information to achieve a global objective. In particular, one edge is selected at each iteration, and only the nodes incident on this edge may communicate and redistribute their tokens. Thus, no time synchronization is required nor information exchange between distant agents may occur. This clearly reduces significantly the implementation complexity and cost of the procedure. Note that

parallel communications between disjoint sets of nodes are allowed as in [21]. Nevertheless the convergence time is expressed as total number of updates to allow a straightforward comparison to other gossip algorithms.

This paper is an extended version of [16], [17]. We provide both a convergence proof for the case in which edges are selected at random and a proof for the case in which there exists a periodic interval of time in which each link is selected at least once.

A. Algorithm Applications

The proposed Hamiltonian Quantized Consensus (HQC) algorithm may be applied in several application domains. The most significant ones are discussed in the following items.

- *Token distribution over networks.* The token distribution problem is a static variant of the load balancing problem [10], [20], [22], [23], [24], [32], [36] where K indivisible tokens of possibly different size should be uniformly distributed over N parallel processors.
- *Sensor networks.* The case in which tokens are indivisible and of unitary size is equivalent to the case in which a network of agents need to agree on the average of integer state variables.
- *Token Ring/IEEE 802.5 networks.* Our proposed algorithm well applies to all those application domains where the communication architecture is based on a Token Ring network which has an embedded Hamiltonian cycle.

B. Paper content

The paper is structured as follows. In Section II we provide some background on quantized consensus algorithms. In Section III we propose the *Hamiltonian Quantized Consensus Algorithm*, whose convergence properties are discussed in Section IV. Conclusions are finally drawn in Section V.

II. Background

Let us consider a network of n agents whose connections can be described by an undirected connected graph $\mathcal{G} = (V, E)$, where V is the set of nodes (agents) and E is the set of edges.

Assume that K indivisible tokens should be assigned to the nodes, where the size of the generic j-th token is denoted as c_j, $j = 1, \ldots, K$. Notice that assuming unitary size for all tokens is equivalent to the problem of quantized consensus with integer state variables [26].

Our goal is that of achieving a globally balanced state, starting from any initial condition, such that the total number of tokens weighted by their sizes in each node is as close as possible, in the least-square sense, to the best possible token distribution

$$\bar{c} = \frac{1}{n} \sum_{j=1}^{K} c_j.$$

In the token distribution problem no token enters nor leaves the network thus the total amount of tokens is preserved during the iterations. This assumption is helpful in abstracting the convergence properties of the network that depend on the topology and on the actual token distribution. In the following we will refer to the total size of the tokens in the generic node as the load of such a node.

We define a cost 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

$$y_{i,j} = \begin{cases} 1 & \text{if the } j\text{-th token is assigned to node } i \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

In the following, given a generic node i, we denote $\mathcal{K}_i(t)$ the set of indices of tokens assigned to i at time t, where $\sum_{j \in \mathcal{K}_i} c_j = c^T y_i$.

The optimal token distribution corresponds to any distribution such that the following performance index

$$V_1(Y) = \sum_{i=1}^{n} \left(c^T y_i - \bar{c} \right)^2, \tag{2}$$

is minimum, where $Y(t) = [y_1(t)\ y_2(t)\ \dots\ y_n(t)]$ denotes the state of the network at time t and Y^* (resp., V_1^*) is the optimal token distribution (resp. optimal value of the performance index). Finally, we denote

$$c_{\max} = \max_{j=1,\dots,K} c_j \qquad c_{\min} = \min_{j=1,\dots,K} c_j \tag{3}$$

respectively the maximum and the minimum size of tokens in the network.

An interesting class of decentralized algorithms for load balancing or averaging networks is given by *gossip-based* algorithms that can be summarized as follows [16], [26].

***Algorithm** 2.1 (Quantized Gossip Algorithm):*

1) Let $t = 0$.
2) Select an edge $e_{i,r}$.

3) Perform a local balancing between nodes i and r using a suitable rule such that the difference between their loads is reduced.

   If such a balancing is not possible execute a *swap* among the loads in i and r.

4) Let $t := t + 1$ and goto Step 2. ■

A *swap* is an operation between two communicating nodes that, while not reducing nor increasing their load difference, it modifies the token distribution.

***Definition** 2.2:* [16] [Swap] Let us consider two nodes i and r incident on the same edge and let $\mathcal{I}_i \subseteq \mathcal{K}_i(t)$ and $\mathcal{I}_r \subseteq \mathcal{K}_r(t)$ be two subsets of their tokens.

We call *swap* the operation that moves the tokens in $\mathcal{I}_i$ to r, and the tokens in $\mathcal{I}_r$ to i at time $t+1$, reaching the distribution

$$\mathcal{K}_i(t+1) = \mathcal{I}_r \cup (\mathcal{K}_i(t) \setminus \mathcal{I}_i),$$
$$\mathcal{K}_r(t+1) = \mathcal{I}_i \cup (\mathcal{K}_r(t) \setminus \mathcal{I}_r)$$

provided the absolute value of the load difference between the two nodes does not change. In particular, we say that a *total swap* occurs if $\mathcal{I}_i = \mathcal{K}_i(t)$ and $\mathcal{I}_r = \mathcal{K}_r(t)$. ■

In the following section we provide an algorithm that is still based on the notion of swap. However, the main difference with respect to Algorithm 2.1 is that in Algorithm 2.1 swaps are executed following a random process, while in the proposed algorithm we exploit the existence of a Hamiltonian cycle in the graph so that they can be executed following an appropriate criterion. As discussed in detail in the rest of the paper, this leads to two main advantages. First, if the average out-degree of the nodes is not high, it results in a smaller convergence time. Secondly, our algorithm has a stopping criterion, while Algorithm 2.1 indefinitely iterates even if no further improvement can be obtained.

III. Quantized Consensus Algorithm for Hamiltonian graphs

Our idea is based on the notion of Hamiltonian cycle, and our assumption is that the considered nets are represented by Hamiltonian graphs, i.e., they have a Hamiltonian cycle.

***Definition** 3.1:* A *Hamiltonian cycle* is a cycle in an undirected graph that visits each vertex exactly once and returns to the starting vertex. ■

Given a network represented by graph $\mathcal{G} = \{V, E\}$ we label the nodes $V = 1, \ldots, n$ along the Hamiltonian cycle, which is assumed to be known, in increasing order such that node i is connected to node $i+1$ and node n is connected to node 1. According to this, we define the set

of edges belonging to the Hamiltonian cycle as $\mathcal{H} = \{e_{i,i+1} = \{V_i, V_{i+1}\}, \quad i = 1, \ldots, n-1\} \cup \{e_{n,1}\}$. It follows that if $\mathcal{G}$ is Hamiltonian then $\mathcal{H} \subseteq E$.

In such a Hamiltonian cycle we label edge $e_{n,1}$ as e_{ae} and call it *absorbing edge*.

In the literature the question of how common Hamiltonian cycles are in arbitrary graphs is still an open issue even if many results exist in this framework. In particular it is known that if the number of nodes and arcs is sufficiently high then almost surely a Hamiltonian cycle exists [14], [27].

Finding a Hamiltonian cycle in a graph is an NP-complete problem [19]. On the other hand, many algorithms can be formulated to design a network such that a Hamiltonian cycle is embedded in it by construction [12] or to find it in a distributed way [4], [30]. Furthermore there exist communication architectures where a Hamiltonian cycle is embedded in their structure. A famous example of such a communication architecture is the Token Ring network [13].

Note that the proposed algorithm is "distributed". Indeed the agents need not to know the network topology nor the number of agents. The agents only know who are the next and previous agents on the directed Hamiltonian cycle and whether one of their incident edges is the absorbing edge. The assignment of increasing integer numbers as labels to the nodes is an arbitrary choice we have done for simplicity of presentation.

Notice that the network can be *arbitrarily* connected as long as it contains a Hamiltonian cycle.

In the following we denote the total amount of load in the generic node i at time t as $x_i(t) = c^T y_i(t)$. The optimal assignment of tokens $\bar{y}_i, \bar{y}_r$ at time t between two different nodes with respect to (2) is the one that minimizes the following quantity:

$$(\bar{y}_i, \bar{y}_r) = \arg\min_{y_i, y_r} |x_i(t) - x_r(t)|$$

given the set of tasks $\mathcal{K}_i(t) \cup \mathcal{K}_r(t)$.

The following algorithm assumes that a Hamiltonian cycle is determined before its initialization.

***Algorithm** 3.2 (HQC):*

1) Let $t = 0$.
2) An edge $e_{i,r}$ is selected at random.
3) If $x_i(t) \neq x_r(t)$ *(the load balancing among the two nodes may potentially be improved)*

a) Let $\bar{x}_i$, $\bar{x}_r$ and respectively $\bar{y}_i$, $\bar{y}_r$, be the optimal assignment of tokens with indices in $\mathcal{K}_i(t) \cup \mathcal{K}_r(t)$

b) If $|\bar{x}_i - \bar{x}_r| < |x_i(t) - x_r(t)|$, then

$$y_i(t+1) = \bar{y}_i,$$
$$y_r(t+1) = \bar{y}_r;$$

and goto step 6.

4) If $x_i(t) = x_r(t)$ or $e_{i,r} \notin \mathcal{H}$ or $e_{i,r} \equiv e_{ae}$ then

$$y_i(t+1) = y_i(t),$$
$$y_r(t+1) = y_r(t);$$

else if $e_{i,r} \in \mathcal{H} \setminus \{e_{ae}\}$ and $x_r(t) \equiv x_{i+1}(t) > x_i(t)$ then execute a swap such that

$$x_i(t+1) > x_{i+1}(t+1),$$

5) Let $t = t + 1$ and go back to Step 2.

■

A. Explanation of the algorithm

In simple words, at each time t an edge is arbitrary selected. If the two nodes incident on the edge have different loads we look for a better load balancing (that may potentially occur only if their loads differ of more than one unit). If the edge belongs to the Hamiltonian cycle but it is not the absorbing edge, then the larger loads are moved toward nodes with smaller index and the smaller loads to nodes with higher index. Thus, the largest and smallest loads eventually meet at the absorbing edge where they can eventually be balanced.

***Remark** 3.3:* We point out that in general if the tokens are not of unitary size it is not guaranteed that the final load configuration is optimal. The following Theorem 4.1 characterizes the convergence properties of the algorithm and shows that disregarding the network topology, the number of tokens and the number of nodes, the maximum distance of the final tokens distribution from the optimal one depends only on the token sizes. ■

As it will be formally proved in the following section, while preserving the asynchrony of the local updates, the simple notion of a "preferred" direction produces several important advantages. Firstly, it reduces the convergence time; then, it makes finite the total number of tokens exchanges

between the nodes to achieve the global tokens distribution; finally, it makes the algorithm stop once a balanced state is reached[1] to allow a change of mode of operation (e.g., take a new measurement in the case of a sensor network or proceed with task execution in the case of multi agent systems).

***Remark** 3.4:* Algorithm 3.2 does not contain an explicit stopping criterion. What happens in practice is that, after a certain number of iterations, no load can be further balanced nor swapped. However, the communication among nodes continues indefinitely.

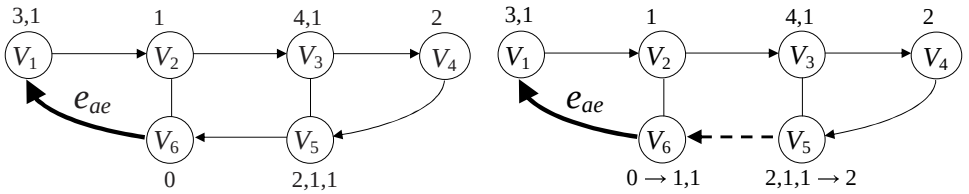
To impose a stopping criterion on communications, we may assume that the edge selection is implemented in a distributed fashion as follows: any node may asynchronously start a communication request with one of its neighbors. After a node has already tested all its possible communications and no balancing or swap was possible, it will enter a "sleeping" state in which it will wait for communication requests but it will not start any new communication. If a sleeping node receives a communication request and as a result its load changes, then it leaves the sleeping state. This ensures that once the network reaches a configuration from which no evolution is possible, each node, after having tested all its link, will reach a sleeping state and all communications will eventually stop. ■

B. A numerical example

Let us consider the network in Fig. 1(a). It consists of six nodes whose connections allow the existence of a Hamiltonian cycle. By assumption arcs are undirected. The direction given to the edges in the Hamiltonian cycle is only introduced to better explain the steps of the algorithm. Assume that the initial token distribution is that in Fig. 1(a): here the integer numbers upon nodes denote the size of tokens in their inside. Finally, $e_{ae} = e_{6,1}$ is the absorbing edge.

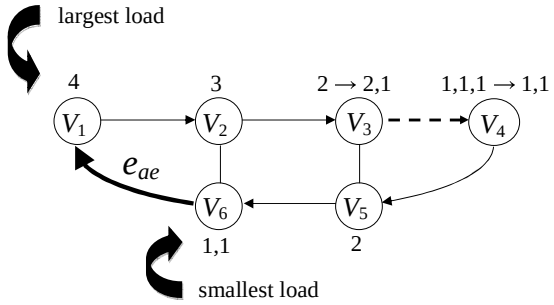
We now run Algorithm 3.2. In Table III-B the evolution of the network is shown. As it can be seen, when Algorithm 3.2 can not locally balance the loads, it moves the largest load toward nodes with smaller index and the smallest one to nodes with higher index. This behavior makes the largest load move toward node V_1 and the smallest one to V_6. In Fig. 1(b) is shown the token

[1]We point out that some algorithms in the literature [26] achieve quantized consensus asymptotically, without actually terminating. This is a relevant issue in the case of load balancing and tasks assignment. In wireless sensor networks such an improvement also allows to save power by avoiding averaging indefinitely after having reached a satisfactory agreement.



(a) Initial token distribution at $t = 0$.

(b) Token distribution at $t = 1$



(c) Final token distribution at $t = 10$

Fig. 1. The network considered in Subsection III-B.

distribution at time $t = 1$. Here the thick dashed edge denotes the selected edge. In Fig. 1(c) is shown the final token distribution reached at time $t = 10$.

Let us finally observe that all the updates are decentralized and asynchronous, i.e., the order in which edges are selected is not relevant to the algorithm convergence properties. After $t = 10$ local updates of the network is in a globally balanced configuration: due to the token quantization a better distribution is not reachable.

Moreover, starting from the last configuration no further load transfer is allowed because every node is locally balanced with its neighbors and the loads are in descending order starting from node V_1 to node V_6. This is a great advantage with respect to other randomized algorithms which keep on swapping loads even after the best load configuration achievable is reached [16], [26].

IV. Convergence properties of HQC Algorithm

The convergence properties of Algorithm 3.2 are stated by the following theorem. In particular, Theorem 4.1 claims that using Algorithm 3.2 the net distribution will almost surely converge to a given set $\mathcal{Y}$ defined as in the following equation (4).

		Nodes					
Time	Edge	V_1	V_2	V_3	V_4	V_5	V_6
0		$3,1$	1	$4,1$	2	$2,1,1$	0
1	$e_{5,6}$	$3,1$	1	$4,1$	2	2	$1,1$
2	$e_{3,5}$	$3,1$	1	4	2	$2,1$	$1,1$
3	$e_{2,3}$	$3,1$	4	1	2	$2,1$	$1,1$
4	$e_{1,6}$	3	4	1	2	$2,1$	$1,1,1$
5	$e_{4,5}$	3	4	1	$2,1$	2	$1,1,1$
6	$e_{1,2}$	4	3	1	$2,1$	2	$1,1,1$
7	$e_{3,4}$	4	3	2	$1,1$	2	$1,1,1$
8	$e_{5,6}$	4	3	2	$1,1$	$2,1$	$1,1$
9	$e_{4,5}$	4	3	2	$1,1,1$	2	$1,1$
10	$e_{3,4}$	4	3	$2,1$	$1,1$	2	$1,1$

TABLE I

RESULTS OF THE NUMERICAL EXAMPLE IN SUBSECTION III-B.

***Theorem** 4.1:* Let us consider

$$\mathcal{Y} = \{Y = [y_1\ y_2\ \cdots\ y_n] \ \mid \ |c^T y_i - c^T y_r| \leq c_{\max}, \quad \forall\ i, r \in \{1, \ldots, n\}\}. \tag{4}$$

Let $Y(t)$ be the matrix that summarizes the token distribution resulting from Algorithm 3.2 at the generic time t. It holds

$$\lim_{t \to \infty} \Pi(Y(t) \in \mathcal{Y}) = 1$$

where $\Pi(Y(t) \in \mathcal{Y})$ denotes the probability that $Y(t) \in \mathcal{Y}$.

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:

$$V_1(Y(t)) = \sum_{i=1}^{n} (x_i(t) - \bar{c})^2 \tag{6}$$

where $x_i(t) = c^T y_i(t)$ for $i = 1, \ldots, n$. The second one is a measure of the ordering of the loads:

$$V_2(t) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} f(x_i(t) - x_j(t)) \tag{7}$$

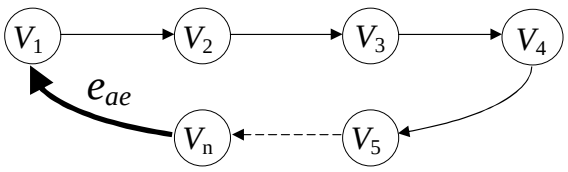


Fig. 2. The oriented Hamiltonian cycle considered in the proof of Theorem 4.1 and Proposition 4.9.

where $f(x_i(t) - x_j(t)) = \max\ (sign(x_i(t) - x_j(t)),\ 0))$.

Note that here we are assuming that $e_{ae} = e_{n,1}$ and nodes are labeled as in Fig. 2. Therefore, $V_2(t)$ denotes the number of couples of nodes that are not ordered[2] at time t.

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) - $V_1(t)$ is a non increasing function of t. In fact, at any time t it holds $V_1(t+1) \leq V_1(t)$.

The case $V_1(t+1) = V_1(t)$ holds during a token exchange when the resulting load difference between the nodes is not reduced. In such a case the loads at the nodes may either swap or not, thus not increasing nor decreasing the value of the Lyapunov function.

The case of $V_1(t+1) < V_1(t)$ holds when a new load balancing occurs. Assume that a combination of tokens with total cost q with $0 < q < |x_i(t) - x_r(t)|$ is moved from i to r at the generic time t such that $|x_i(t+1) - x_r(t+1)| < |x_i(t) - x_r(t)|$. It is easy to verify, by simple computations, that $(x_i(t+1) - \bar{c})^2 + (x_r(t+1) - \bar{c})^2 < (x_i(t) - \bar{c})^2 + (x_r(t) - \bar{c})^2$ which implies $V_1(t+1) < V_1(t)$. We also observe that if two nodes (e.g., i and r) communicate at time t, the resulting difference among their loads at time $t+1$ is surely less or equal to the largest cost of tokens in the nodes at time t, i.e.,

$$|x_i(t+1) - x_r(t+1)| \leq \max_{j \in \mathcal{K}_i(t) \cup \mathcal{K}_r(t)} c_j \leq c_{\max}. \tag{8}$$

This is due to the fact that if the load difference between two nodes is greater than $c_{\max}$, it is always possible to move at least one token with $c \leq c_{\max}$ to the less loaded node to reduce the load difference.

[2]According to Algorithm 3.2 and the notation in Fig. 2 a couple of nodes $\{i, j\}$ is said to be ordered if for $i < j$, it is $x_i < x_j$.

(2) - $V_2(t)$ is a positive non-increasing function of t if $V_1(t+1) = V_1(t)$. Function $V_2(t)$ is positive because it is the summation of positive quantities. Moreover, $V_2(t+1) = V_2(t)$ anytime an edge connecting two nodes already ordered along the Hamiltonian cycle is chosen, or alternatively when the absorbing edge is chosen. This is due to the fact that in such a case the ordering of loads does not change. While $V_2(t+1) < V_2(t)$ anytime the loads of two nodes are reordered along the Hamiltonian cycle and the load difference between the loads is not reduced. This follows from the fact that if the loads of nodes i and j are not ordered at time t, i.e., for $i < j$, $x_i(t) < x_j(t)$, we have that $f(x_i(t) - x_j(t)) = 1$. If the edge connecting them is selected and they are ordered, then at time $t+1$ it is $f(x_i(t+1) - x_j(t+1)) = 0$. Furthermore since the nodes are directly connected, their ordering does not affect the value of f for other couples of nodes. If a ordering happens, then $V_2(t+1) = V_2(t) - 1$. Finally, if at time t all the loads are ordered along the Hamiltonian cycle it is easy to verify that $V_2(t) = 0$.

(3) - If the Lyapunov-like function $V(t)$ has not reached its minimum at a given time t, then there exists an edge along the Hamiltonian cycle with strictly positive probability to be chosen such that $V(t+1) < V(t)$.

(a) If an edge is selected and the load difference between two nodes is reduced then $V_1(t+1) < V_1(t)$.

(b) If there does not exist an edge such that the load difference between the two nodes is reduced, we can always select an edge such that the loads are reordered if $V_2(t) \neq 0$, then $V_2(t+1) < V_2(t)$.

(c) If $V_2(t) = 0$ then the nodes connected by the absorbing edge contain the maximum and minimum load in the network. If their difference is greater than c_{max} then we can select the absorbing edge and have $V_1(t+1) < V_1(t)$.

(d) If $V_2(t) = 0$ and the load difference between the nodes connected by the absorbing edge is less or equal than c_{max} then $Y(t) \in \mathcal{Y}$.

Finally, at each instant of time, we proved that there exists an edge with strictly positive probability p that if selected makes $V(t+1) < V(t)$. The probability that such an edge is selected at least once in t time steps is $P(t) = 1 - (1-p)^t$. Thus since we assume p to be strictly positive, the probability that such an edge is selected goes to 1 as t goes to infinity, thus proving the statement.

□

***Remark** 4.2:* We remark that such a theorem states the convergence toward a balanced situation in which the load difference between any couple of nodes in the network is at most c_{max}. However, in principle any load balancing rule can be designed to have a greater threshold to trigger the local balancing mechanism, for instance one in which the load difference between the two nodes is $\gamma > c_{max}$. In such a case the theorem gives a design criterion for such threshold since it states that the local threshold used for the balancing mechanism will hold globally by bounding the maximum load difference between any two nodes. ■

A characterization of the maximum distance of the final set of token distribution using Algorithm 3.2 from the optimal one is given by the following proposition.

***Proposition** 4.3:* Let us consider the optimal token distribution problem, and let the set $\mathcal{Y}$ be defined as in equation (4). Let $V_1(Y) = \sum_{i=1}^{n}(c^T y_i - \bar{c})$, where $Y \equiv Y(t)$ results from the application of Algorithm 3.2 for a sufficiently long time t.

The following inequalities hold for any $Y \in \mathcal{Y}$:

$$0 \leq V_1^* \leq V_1(Y) \leq \alpha \tag{9}$$

where

$$\alpha = \begin{cases} \dfrac{n c_{\max}^2}{4} & \text{if } n \text{ is even,} \\ \left\lfloor \dfrac{n}{2} \right\rfloor \left\lceil \dfrac{n}{2} \right\rceil \dfrac{c_{\max}^2}{n} & \text{if } n \text{ is odd.} \end{cases} \tag{10}$$

Proof. The first two inequalities are trivial. To prove the last inequality we look at the worst case, i.e., the token distribution in $\mathcal{Y}$ that has the highest value of $V_1(Y)$.

If n is even, the worst case corresponds to a balancing where half of the nodes have a load k and the remaining half have a load $k + c_{\max}$. In this case $\bar{c} = k + 0.5c_{\max}$, and the first value of bound can be computed.

If n is odd, the worst case corresponds to a configuration where $\lfloor n/2 \rfloor$ of the nodes have a load k and the remaining $\lceil n/2 \rceil$ have a load $k + c_{\max}$. Now $c_{ave} = k + \lceil n/2 \rceil c_{\max}/n$, which gives the other value of the bound. □

The above results enable us to characterize some cases in which Algorithm 3.2 provides the optimal solution to the token distribution problem.

***Proposition** 4.4:* Let $c_{\min}$ and $c_{\max}$ be defined as in (3).

If $c_{\min} = c_{\max} = c$, then all load distributions that belong to a set of final distributions (4) are optimal, hence Algorithm 3.2 provides a token distribution for which $V_1(Y)$ is minimum and

thus it is an optimal distribution.

Proof. If $c_{\min} = c_{\max}$ the set of final distributions is

$$\mathcal{Y} = \{[y_1 \ \cdots \ y_n] \ \mid (\forall \ i) \ c^T y_i \in \{\lfloor \tfrac{K \cdot c}{n} \rfloor, \lfloor \tfrac{K \cdot c}{n} \rfloor + c\}\}. \tag{11}$$

We can normalize the weight c so that it is unitary. With this formulation the problem corresponds to that of quantized consensus, and the set $\mathcal{Y}$ coincides with the set of the *quantized-consensus distributions* defined in [26] and shown to be optimal. □

We now prove that Algorithm 3.2 always reaches a blocking configuration.

***Proposition** 4.5:* Given a Hamiltonian graph $\mathcal{G}$, if the network evolves according to Algorithm 3.2, then

$$\forall \ Y(0), \ \exists t' : \forall t \geq t', Y(t) \equiv Y(t') \in \mathcal{Y}.$$

Proof. Due to Theorem 4.1 $\exists t'$ such that $\forall \ t \geq t', \ Y(t) \in \mathcal{Y}$. Let us consider the Lyapunov-like function (5): $V(t) = [V_1(t), V_2(t)]$. It can be shown that if $V_2(t) = 0$ then the loads are ordered such that $x_i \geq x_{i+1}$ for $i = 1, \ldots, n-1$. If at time t' the loads are ordered and $V_1(t')$ has reached a local minimum, then according to Algorithm 3.2 no token exchange is performed since no balancing is feasible and no swap is allowed. Then it follows that $Y(t' + \Delta t) \equiv Y(t') \quad \forall \Delta t \geq 0$. □

A. Convergence time

In this section we discuss the expected convergence time of Algorithm 3.2, and provide an upper bound for arbitrary Hamiltonian graphs.

We assume that edges are selected with uniform probability, so the probability to select the generic edge $e_{i,j}$ at time t is equal to $p = 1/N$ where N is the number of edges in the network.

The *convergence time* is a random variable defined for a given initial load configuration $Y(0) = Y$ as:

$$T_{con}(Y) = \inf \{t \ \mid \ \forall \ t' \geq t, \ Y(t') \in \mathcal{Y}\}.$$

Thus, $T_{con}(Y)$ represents the number of steps required at a certain execution of Algorithm 3.2 to reach the convergence set $\mathcal{Y}$ starting from a given token distribution.

Now, let us provide some further definitions that will occur in the following.

- $N_{\max}$ is the maximum number of improvements of $V_1(Y)$ needed by any realization of Algorithm 3.2 to reach the set $\mathcal{Y}$, starting from a given configuration.

- $T_{\max}$ is the maximum average time between two consecutive improvements of $V_1(Y)$ in any realization of Algorithm 3.2, starting from a given configuration.

From the previous definitions, it is possible to give an upper bound on the expected convergence time.

***Proposition** 4.6:* Let $\mathcal{E}[T_{con}(Y)]$ be the *expected convergence time*. It holds $\mathcal{E}[T_{con}(Y)] \leq N_{\max} \cdot T_{\max}$. ■

Notice that the term *maximum average time* in the above definition is intended as in the following.

The average time between two consecutive improvements is a function of the load distribution: an unbalanced distribution has a short average time between two consecutive improvements, while a nearly balanced distribution has a long average time. In our definition we consider the longest possible average time between two improvements and take it as an upper bound to the average time between two consecutive improvements.

In [26] an upper bound on $N_{\max}$ is given when $c_{max} = 1$. In our case the result still holds since it is based on the fact that the improvement of the performance index is lower bounded by $V_1(Y(t+1)) \leq V_1(Y(t)) - 2$ since the minimum token exchange allowed decreases the load difference between two nodes of at least 1. Finally, the initial value of $V_1(Y(0))$ can be upper bounded by a function of the maximum and the minimum amount of load in the generic node.

***Proposition** 4.7:* [26] For the Hamiltonian Quantized Consensus it holds:

$$N_{\max} = \frac{(M-m)n}{4}$$

where $M = \max_i c^T y_i$ and $m = \min_i c^T y_i$.

We now focus on $T_{\max}$. As shown in the following proposition, it is easy to compute in the case of fully connected networks.

***Proposition** 4.8:* Let us consider a fully connected network, namely a net such that $E = \{V \times V\}$. 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 edges of the net, then the probability of selecting the only edge whose incident nodes may balance their load is equal to $p = 1/N$, while

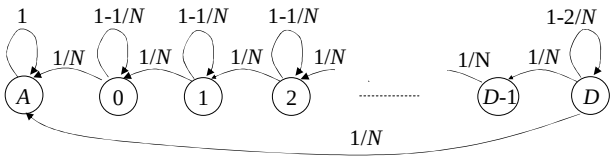


Fig. 3. The Markov chain associated to a net containing a Hamiltonian cycle.

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 edges is $N = n(n-1)/2$ and so $T_{\max} = n(n-1)/2$. $\square$

Notice that the previous proposition holds for various gossip based algorithms [16], [26].

We now show that $T_{\max}$ for Hamiltonian graphs is of the same order with respect to the number of nodes as for fully connected topologies when using the Hamiltonian Quantized Consensus Algorithm.

***Proposition** 4.9:* Let us consider a net with a Hamiltonian cycle. Let n be the number of nodes, and N be the number of arcs of the net. It holds

$$T_{\max} \leq N(n-2). \tag{13}$$

Proof. We first observe that, due to the gossip nature of Algorithm 3.2 and to the 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 the Hamiltonian cycle in opposite directions[3]. 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 $T_{\max}$.

To compute such an upper bound we determine the average meeting time of the largest and smallest load walking on the graph along the Hamiltonian cycle in the worst case. To this aim we define the discrete Markov chain in Fig. 3 whose states (apart from the first one, named A) characterize the distance between the two agents.

For simplicity of explanation we assume that the first agent is the one corresponding to the largest load.

[3]The problem of random walk and average meeting times has been extensively studied in different applications [6], [35].

The distance between the two agents is equal to the length of the path going from the first agent to the second one in the direction of nodes with increasing index. In other words, the distance between the two agents is equal to the minimum number of movements they need to perform, following the rule at Step 3 of Algorithm 3.2, to meet each other.

Now, if a net has n nodes, then the Hamiltonian cycle has n edges, and the maximum distance among the two agents is equal to $D = n - 1$, while their minimum distance is equal to 1. Note that both these conditions correspond to the case in which the two agents are in nodes incident on the same edge. However, the first case occurs when such an edge is directed from the second agent to the first one, while the second case happens when the edge is directed from the first agent to the second one. As an example, if the Hamiltonian cycle is that reported in Fig. 2, if the first agent is in V_1 and the second one is V_n, then their distance is null; if the first agent is in V_n and the second one in V_1, then their distance is equal to D. The absorbing state (node A in Fig. 3) corresponds to the case in which the agents are in nodes incident on the same edge and this edge is selected. Thus, the absorbing state may only be reached from nodes 1 and D, and the probability that this occurs is in both cases equal to $1/N$.

Moreover, given the rule of step 3 of Algorithm 3.2, the distance among two nodes with load difference greater than c_{max} may only decrease, regardless their initial position. In particular, the probability of going from node i to node $i - 1$, with $i = D, D - 1, \ldots, 1$, is equal to $2/N$, because two are the edges whose selection leads to a unitary reduction of the distance among the agents. Finally, we consider the linear system:

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

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. Solving analytically the linear system (14), we found out that $\tau(i) = iN$ for $i = 1, \ldots, D - 1$, and $\tau(D) = N(n - 1)/2$. Thus the maximum average hitting time of the absorbing state occurs when the distance between the two nodes is

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

equal to $D-1$ if $n \geq 3$. In particular, it holds $\tau(D-1) = N(n-2)$ that proves the statement. □

***Proposition** 4.10:* An upper bound to the aexpected convergence time of Algorithm 3.2 is

$$\mathcal{E}[T_{con}(Y)] \;\leq\; \frac{(M-m)n}{4} \cdot N(n-2) \;=\; \mathcal{O}(n^2 N).$$

Proof. The statement follows from Propositions 4.7 and 4.9 and Fact 4.6. □

***Proposition** 4.11:* If a net is fully connected, an upper bound to the expected convergence time of Algorithm 3.2 is

$$\mathcal{E}[T_{con}(Y)] \;\leq\; \frac{(M-m)n}{4} \cdot \frac{n(n-1)}{2} \;=\; \mathcal{O}(n^3).$$

Proof. Follows from Propositions 4.7 and 4.8 and Fact 4.6. □

The above propositions enable us to conclude that Algorithm 3.2 leads to a significant improvement respect to [16], [26] in terms of convergence time for networks with low average out-degree (e.g. path networks). Indeed for such networks an upper bound for the expected convergence time can be found to be $\mathcal{O}(n^4)$ for ring networks using the approaches in [16], [26]. In particular in [18] the computation of an upper bound to the expected convergence time is carried out for the so-called "generalized ring topology" that consists in several ring networks connected together. In case of a single ring the result of Proposition 4.5 still holds and we can state the following proposition:

***Proposition** 4.12:* For a ring network, an upper bound to the expected convergence time of the algorithms in [16], [26], [18] is

$$\mathcal{E}[T_{con}(Y)] \;\leq\; \frac{(M-m)n}{4} \cdot \frac{n^2(n+16)}{16} \;=\; \mathcal{O}(n^4).$$

Proof. Follows from the upper bound on T_{max} given in Proposition 4.5 in [18] assuming a single ring ($s = 1$) and from Propositions 4.6 and 4.7. □

By Proposition 4.9, in the case of Hamiltonian networks with a number of edges $\mathcal{O}(n)$, such as ring networks, the expected convergence time of Algorithm 3.2 is at most $\mathcal{O}(n^3)$. On the contrary, if we consider fully connected networks, the expected convergence time is still $\mathcal{O}(n^3)$ and the advantage of Algorithm 3.2 is basically that of providing a stopping criterion.

In Figure 4 is shown the expected convergence time for a ring network of n nodes with $n = 10, \ldots, 100$ and random initial loads ranging from 0 to 10. For each network size the expected convergence time is taken over 100 realizations of the experiment. In such a figure is

also shown a comparison with the previously computed upper bound to the expected convergence time, it is evident that such a bound is not strict, i.e., the actual performance of the algorithm is considerably better than the worst case analysis prediction. Furthermore we point out that the convergence time is given in number of local updates, not time, thus disregarding the effects of parallel communications for analysis purposes.

***Remark** 4.13:* Note that in [37] it is proposed an algorithm for quantized consensus named "synchronous quantized averaging on fixed graphs", similar to the one proposed in [26]. It differs from other algorithms in the literature in that a token is used to select which nodes perform an update and at each instant of (discrete) time the node that owns the token performs an update with a neighbor and pass the ownership to it. The analysis of the convergence time with this assumption is shown to be $\mathcal{O}(n^2)$ for complete graphs, $\mathcal{O}(n^3)$ for line networks and $\mathcal{O}(n^4)$ for arbitrary connected graphs. Such tighter upper bounds were developed by the authors by exploiting the token mechanism to synchronize the agents. This method improves the bound on the convergence time but prevent parallel updates in the network. In our case, this assumption would lower the expected converge time by $\mathcal{O}(n)$ but would violate the assumption of asynchronous communications. ■

B. Algorithm extension for convergence in finite time

The effectiveness of Algorithm 3.2 is even more evident if a periodic interval of time T_h exists such that each edge in the Hamiltonian cycle is selected at least once. In such a case Algorithm 3.2 converges in finite time, as will be shown in the following. Furthermore if Algorithm 3.2 is applied to networks whose edge selection process is deterministic, it still preserves its convergence properties while other algorithms as the one in [26] may cycle indefinitely without reaching the consensus set of final configurations. Obviously Algorithm 3.2 prevents the existence of such cycles due to the deterministic swap rule. In particular, the following result holds.

***Proposition** 4.14:* If there exists a period of time T_h such that each edge along the Hamiltonian cycle is selected at least once, then a deterministic upper-bound to the convergence time of Algorithm 3.2 is

$$\max(T_{con}(Y)) \leq (n-1)^2 \cdot (M-m) \cdot T_h = \mathcal{O}(n^2).$$

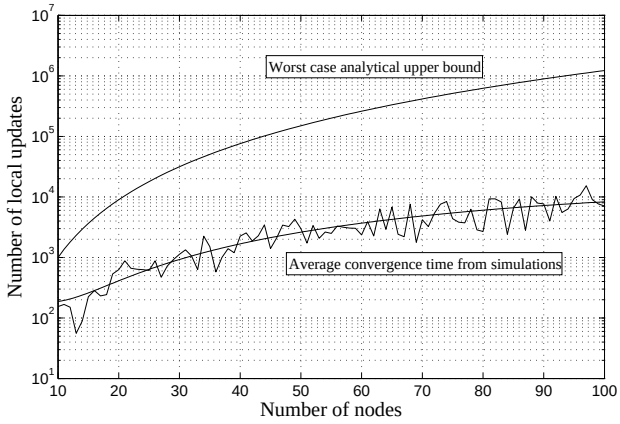


Fig. 4. Comparison between simulation results and the worst case analytical expected convergence time.

Proof. By Proposition 4.7 the maximum number of balancing between two consecutive improvements of $V(Y)$ is at most equal to $\frac{(M-m)n}{4}$. Now, if each edge of the Hamiltonian cycle is selected at least once during T_h, being the maximum distance between the two nodes with the smallest and highest load in the network equal to $n-1$ (see the proof of Proposition 4.9), then at each interval T_h their distance is surely reduced by at least 1 and they meet after at most $(n-1)T_h$ units of time. Then, $\frac{(M-m)n}{4}(n-1)\cdot T_h$ is the maximum number of time units required to reach the convergence set $\mathcal{Y}$. □

We note that to make Proposition 4.14 useful in practical cases, namely if we want to use it as a criterion to know when $\mathcal{Y}$ is reached for sure, then a slight overhead needs to be added to Algorithm 3.2 to evaluate the difference $M-m$ of the initial load. This can be done in a decentralized way with a consensus-like algorithm (namely consensus on $\max_i\ x_i(0)$).

V. Conclusions

In this paper we proposed a new algorithm, the Hamiltonian Quantized Consensus Algorithm, that solves the quantized distributed average problem and the token distribution problem on Hamiltonian graphs with a grater efficiency respect to other gossip algorithms based on uniform quantization [16], [26] provided that Hamiltonian cycle is known in advance. A feature of the proposed algorithm is an embedded stopping criterion that will block the algorithm once quantized consensus has been achieved. We have also shown that, if there exists a periodic

interval of time where each edge along the Hamiltonian cycle is selected at least once, a finite time convergence bound can be given. Future work will involve the design of algorithms for more general graph structures such as tree.

REFERENCES

[1] T.C. Aysal, M.J. Coates, and M.G. Rabbat. Distributed average consensus using probabilistic quantization. *IEEE/SP 14th Workshop on Statistical Signal Processing*, pages 640–644, August 2007.

[2] T.C. Aysal, M.J. Coates, and M.G. Rabbat. Distributed average consensus with dithered quantization. *IEEE Trans. on Signal Processing*, 56(10, Part 1):4905–4918, 2008.

[3] D. Bauso, L. Giarré, and R. Pesenti. Non-linear protocols for optimal distributed consensus in networks of dynamic agents. *Systems and Control Letters*, 55(11):918–928, 2006.

[4] B. Bollobás, T.I. Fenner, and A.M. Frieze. An algorithm for finding Hamilton paths and cycles in random graphs. *Combinatorica*, 7(4):327–341, 1987.

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

[6] N.H. Bshouty, L. Higham, and J. Warpechowska-Gruca. Meeting times of random walks on graphs. *Information Processing Letters*, 69(5):259–265, 1999.

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

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

[9] R. Carli and S. Zampieri. Efficient quantization in the average consensus problem. *Advances in Control Theory and Applications*, 353:31–49, 2007.

[10] A. Cortes, A. Ripoll, M.A. Senar, P. Pons, and E. Luque. On the performance of nearest-neighbors load balancing algorithms in parallel systems. In *Proc. 7th Euromicro Workshop on Parallel and Distributed Processing*, pages 170–177, Funchal, Portugal, february 1999.

[11] G. Cybenko. Dynamic load balancing for distributed memory multiprocessors. *J. of Parallel and Distributed Computing*, 7(2):279–301, 1989.

[12] K. Day and A. Tripathi. Embedding of cycles in arrangement graphs. *IEEE Trans. on Computers*, 42(8):1002–1006, 1993.

[13] R.C. Dixon, N.C. Strole, and J.D. Markov. A token-ring network for local data communications. *IBM Systems J.*, 22(1-2):47–62, 1983.

[14] T.I. Fenner and A.M. Frieze. On the existence of hamiltonian cycles in a class of random graphs. *Discrete Mathematics*, 45(2-3):301–305, 1983.

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

[16] 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*, pages 500–505, New Orleans, Louisiana USA, december 2007.

[17] M. Franceschelli, A. Giua, and C. Seatzu. Hamiltonian quantized gossip. In *Proc. 2009 IEEE Multi-conference on Systems and Control*, pages 648–654, St. Petersburg, Russia, july 2009.

[18] M. Franceschelli, A. Giua, and C. Seatzu. A Gossip-Based Algorithm for Discrete Consensus Over Heterogeneous Networks. *Automatic Control, IEEE Transactions on*, 55(5):1244–1249, 2010.

[19] M.R. Garey and D.S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-completeness*. W. H. Freeman and Co., New York, NY, USA, 1990.

[20] B. Ghosh, F.T. Leighton, B.M. Maggs, S. Muthukrishnan, C.G. Plaxton, R. Rajaraman, A.W. Richa, R.E. Tarjan, and D. Zuckerman. Tight analyses of two local load balancing algorithms. *SIAM J. on Computing*, 29(1):29–64, 2000.

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

[22] M. Herlihy and S. Tirthapura. Self-stabilizing smoothing and balancing networks. *Distributed Computing*, 18(5):345–357, 2006.

[23] M.E. Houle, A. Symvonis, and D.R. Wood. Dimension exchange algorithms for token distribution on tree-connected architectures. *J. of Parallel and Distributed Computing*, 64(5):591–605, 2004.

[24] M.E. Houle, E. Tempero, and G. Turner. Optimal dimension exchange token distribution on complete binary trees. *Theoretical Computer Science*, 220(2):363–377, 1999.

[25] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48:988 –1001, 2003.

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

[27] J. Komlós and E. Szemerédi. Limit distribution for the existence of Hamiltonian cycles in a random graph. *Discrete mathematics*, 43(1):55–63, 1983.

[28] J. Lavaei and R.M. Murray. On quantized consensus by means of gossip algorithm–Part I: convergence proof. In *American Control Conference, St. Louis, MO, USA*, pages 394–401, 2009.

[29] J. Lavaei and R.M. Murray. On quantized consensus by means of gossip algorithm–Part II: convergence time. In *American Control Conference, St. Louis, MO, USA*, pages 2958–2965, 2009.

[30] E. Levy, G. Louchard, and J. Petit. A distributed algorithm to find hamiltonian cycles in random graphs. In *Combinatorial and Algorithmic Aspects of Networking*, pages 63–74.

[31] X. Lin and S. Boyd. Fast linear iterations for distributed averaging. *Systems and Control Letters*, 53(1):65–78, 2004.

[32] F. Meyer Auf Der Heide, B. Oesterdiekhoff, and R. Wanka. Strongly adaptive token distribution. In *Lecture Notes in Computer Science*, volume 700, pages 398–409, 1993.

[33] R. Olfati-Saber. Flocking for multi-agent dynamic system: Algorithms and theory. *IEEE Transactions on Automatic Control*, 51:401–420, 2006.

[34] R. Olfati-Saber and R.M. Murray. Consensus problems in networks of agents with switching topology and time-delays. *IEEE Trans. on Automatic Control*, 49(9):1520–1533, 2004.

[35] P. Tetali and P. Winkler. On a random walk problem arising in self-stabilizing token management. In *PODC '91: Proc. 10th Annual ACM Symposium on Principles of Distributed Computing*, pages 273–280, New York, NY, USA, 1991.

[36] G. Turner and H. Schroder. Token distribution on reconfigurable d-dimensional meshes. In *Proc. 1st IEEE Int. Conf. on Algorithms and Architectures for Parallel Processing*, volume 1, pages 335–344, 1995.

[37] M. Zhu and S. Martinez. On the convergence time of distributed quantized averaging algorithms. In *47th IEEE Conf. on Decision and Control, Cancun, Mexico*, pages 3971–3976, 2008.