Hamiltonian Quantized Gossip

Mauro Franceschelli, Alessandro Giua, Carla Seatzu

Abstract

The main contribution of this paper is an algorithm to solve the quantized consensus problem over networks represented by Hamiltonian graphs, i.e., graphs containing a Hamiltonian cycle. The algorithm is proved to converge almost surely to a finite set containing the optimal solution. A worst case study of the average 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, "Hamiltonian Quantized Gossip," *3rd IEEE Multi-conference on Systems and Control*, (St. Petersburg, Russia), July 2009.

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@diee.unica.it, {giua,seatzu}@diee.unica.it.

I. Introduction

In this paper we consider a problem of quantized consensus over an Hamiltonian graph, using a gossip algorithm.

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 [7], [15], [20], [21], as a practical implementation of the continuous one [1], [14], [16], [17], [18]. Such problem has relevant applications such as sensor networks, task assignment and token distribution over networks (a simplified load balancing problem) [6], [9], [11], [13]. 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 [19] 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 [2], [21] 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.

A couple of years ago in [15] it was originally proposed an algorithm used for load balancing to solve the distributed average problem with uniformly quantized measurements. Such an algorithm guarantees that almost surely the state of all the agents 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., the algorithm iterates even if the convergence set $\mathcal{S}$ is reached.

In this paper we propose an algorithm to solve the quantized distributed average problem using gossip algorithms [3]. Our algorithm can also be successfully applied to the token distribution problem, i.e., the problem of evenly distribute a set of tokens among the agents [15]. We investigated the extension of this problem to the distribution of arbitrary sized tokens in [7].

Our algorithm presents three main advantages with respect to other applications and approaches in the literature. The first main advantage is that of having a decentralized stopping criterion. Secondly, the average convergence time is significantly reduced with respect to [7], [15]. Finally, following previous work by the authors in [7] we removed the assumption that all tokens have the same size. Note that in the particular case of tokens with the same size our convergence set coincides with the convergence set in [15].

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 [20], [21]. However, uniform quantization is surely easier to implement and less cost consuming than the other types of quantization. Moreover, in [20], [21] a convergence set is not defined, and the convergence properties are given in terms of probability. Furthermore such approaches have proven to be successful for sensor networks while the use of uniform quantization extends the efficiency of the algorithm to several other applications, e.g., tasks assignment for teams of mobile robots and token distribution over networks.

Finally, our algorithm is based on gossip, i.e., only adjacent nodes 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 synchronization or information exchange may occur between distant agents.

A. Algorithm Applications

• *Tokens distribution over networks.* Computer networks for distributed computing greatly benefit from load balancing algorithms to exploit the full potential of their architectures. The token distribution problem is a static variant of the load balancing problem [5], [8], [10], [11], [13], [12], [23] where K indivisible tokens are to be uniformly distributed over N parallel processors. The algorithm proposed in this paper is asynchronous and decentralized and as such it has minimum overhead.

Furthermore, it can achieve a globally balanced state with tokens of various size with a finite number of total transfers and finite maximum discrepancy (the maximum difference between the load of two nodes) that is a function only of the token sizes and thus it is optimal for unitary sized tokens.

• *Decentralized tasks assignment.* Within this framework, our algorithm well applies in the case of multi agent systems when the task's cost is independent from the agents. With this assumption the agents can locally exchange tasks with their neighbors and have a decentralized criterion to know when a balanced assignment has been achieved without knowing the full state of the net.

• *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.

II. Background

In this section we briefly recall some results we recently presented in [7].

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 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, $j = 1, \dots, K$. Notice that assuming unitary size for all tokens is equivalent to the problem of quantized consensus with integer state variables [15].

Our goal is that of achieving a globally balanced state, starting from any initial condition, such that the total size of tokens in each node is as close as possible, in the least-square sense, to the average size of tokens in the network, namely to

$$c_{ave} = \frac{1}{n} \sum_{j=1}^{K} c_j. \tag{1}$$

In the token distribution problem no token enters nor leaves the network thus the total amount of tokens is preserved during the iterations. In the following we will refer to the total size of the tokens in the generic node as the load of such 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

$$y_{i,j} = \begin{cases} 1 & \text{if the } j\text{-th token is assigned to node } i \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

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

$$V_1(Y) = \sum_{i=1}^{n} \left(c^T y_i - c_{ave}\right)^2, \tag{3}$$

is minimum, where

$$Y(t) = [y_1(t)\ y_2(t)\ \ldots\ y_n(t)] \tag{4}$$

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

$$c_{\max} = \max_{j=1,\ldots,K} c_j \tag{5}$$

the maximum 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.

***Algorithm** 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 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.

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.

***Definition** 1:* [7] [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 $\mathcal{I}_r = \mathcal{K}_r(t)$, while we say that a *partial swap* occurs if either $\mathcal{I}_i \subsetneq \mathcal{K}_i(t)$ or $\mathcal{I}_r \subsetneq \mathcal{K}_r(t)$. ■

Usually, see e.g. [15], [7], the rule to execute swap is not specified, but are given the specifications that any desirable rule must have to preserve the convergence properties of the algorithm. Such rule will be obviously different depending on the application, i.e., sensor networks, load balancing, token distribution, task assignment, but its main feature will be the same.

III. Hamiltonian Quantized Gossip Algorithm

In the literature it is commonly assumed that swaps are executed following a random process. In this section we show how it is possible to improve the efficiency of gossip based algorithms introducing appropriate criteria to execute swaps.

The 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 an Hamiltonian cycle.

***Definition** 2:* A *Hamiltonian cycle* is a cycle in an undirected graph that visits each vertex exactly once and returns to the starting vertex. ■

We observe that finding an Hamiltonian cycle in a graph is an NP-complete problem. On the contrary, many simple algorithms can be formulated to design a network such that a Hamiltonian cycle is embedded in it by construction.

For sake of simplicity in the following we assume that the Hamiltonian cycle $\mathcal{H} = (V, H)$ embedded in $\mathcal{G}$ is oriented in the *clockwise* direction.

In such Hamiltonian cycle we select an edge e_{ae} and call it *absorbing edge*.

The agents need not to know the network topology nor the number of agents. The agents only know who is the next and previous agent on the directed hamiltonian cycle and whether one of their incident edges is the absorbing edge. Notice that the network can be *arbitrary* connected as long as it contains an hamiltonian cycle.

In the following we denote the total amount of load in the generic node i at time t as $x_i(t) = c^T y_i(t)$ and we define the optimal assignment of tokens between two different nodes as the one that minimizes the following quantity:

$$\bar{x}_i, \bar{x}_r : |\bar{x}_i - \bar{x}_r| \le |x_i(t+1) - x_r(t+1)|$$

$\forall x_i(t+1), x_r(t+1)$ such that $y_i(t+1), y_r(t+1)$ contain exactly the tasks contained in the nodes at time t, namely the ones in $\mathcal{K}_i(t) \cup \mathcal{K}_r(t)$.

***Algorithm** 2 (Hamiltonian Quantized Gossip Algorithm):*

1) Let $t = 0$.
2) Select an edge $e_{i,r}$ 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 $|\bar{x}_i - \bar{x}_r| < |x_i(t) - x_r(t)|$,

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

else if $e_{i,r} \neq H$ or $e_{i,r} \equiv e_{ae}$ then

$$\begin{aligned} y_i(t+1) &= y_i(t), \\ y_r(t+1) &= y_r(t); \end{aligned}$$

c) if $e_{i,r} \in H \setminus \{e_{ae}\}$ (assume with no loss of generality that $e_{i,r}$ is oriented *clockwise*),

i) if $x_r(t) > x_i(t)$ then
execute a swap such that

$$x_i(t+1) > x_r(t+1).$$

else

$$\begin{aligned} y_i(t+1) &= y_i(t), \\ y_r(t+1) &= y_r(t); \end{aligned}$$

4) if $x_i(t) = x_r(t)$,

$$\begin{aligned} y_i(t+1) &= y_i(t), \\ y_r(t+1) &= y_r(t); \end{aligned}$$

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

■

A. Explanation of the algorithm behavior

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). In any case, no reordering is performed if the edge is not in the Hamiltonian cycle or if it coincides with the absorbing edge. On the contrary, if the edge belongs to the Hamiltonian cycle but does not coincide with the absorbing state, then the largest load is moved in the clockwise direction and the smallest one in the anti-clockwise direction, both in the case of a better balancing and in the case of no improvement.

Finally, if the two loads are equal, they are not moved.

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 greatly reduces the converge time; then, it makes finite the total number of tokens exchange between the nodes to achieve the global token 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).

[1]We point out that other algorithms in the literature [15] 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 improvement also allows to save power by avoiding averaging indefinitely after having reached a satisfactory agreement.

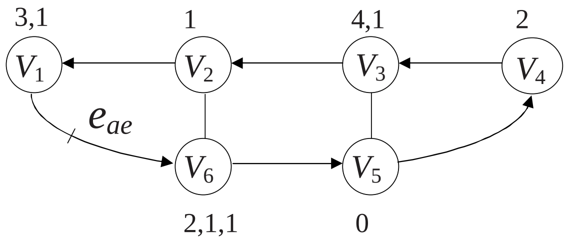

Fig. 1. The network considered in Subsection III-B: initial token distribution.

Time	Edge\Node	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
THE EVOLUTION OF THE NETWORK IN SUBSECTION III-B.

B. A numerical example

Let us consider the network in Fig. 1. It consists of six nodes whose connections allow the existence of a Hamiltonian cycle. Assume that it contains 9 tokens with the following size: $c_1 = 3$, $c_2 = 1$, $c_3 = 1$, $c_4 = 4$, $c_5 = 1$, $c_6 = 2$, $c_7 = 2$, $c_8 = 1$, $c_9 = 1$.

Let the initial token distribution be that represented in Fig. 1, where the integer numbers upon nodes denote the size of tokens in their inside.

Let $e_{ae} = \{1, 6\}$ be the absorbing edge.

We now run Algorithm 2 and select edges randomly. In Table III-B the evolution of the network is shown: it can be seen that when Algorithm 2 can not locally balance the loads, it moves the largest in the anti-clockwise direction and the smallest in the clockwise direction. This behavior makes the largest loads filter toward node V_1 while the smallest toward V_6.

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 ten local updates the network is in a globally balanced configuration, due to the token quantization a better distribution is not reachable.

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, thus improving respect to other randomized algorithms which keep on swapping loads even after the best load configuration achievable is reached.

IV. Convergence properties of HQG Algorithm

The convergence properties of Algorithm 2 are stated by the following theorem. In particular, Theorem 1 claims that using Algorithm 2 the net distribution will almost surely converge to a given set $\mathcal{Y}$ defined as in the following equation (6).

Theorem 1: Let us consider

$$\begin{aligned} \mathcal{Y} = \{Y = [\vec{y}_1\ \vec{y}_2\ \cdots\ \vec{y}_n] \ \mid \ & |c^T\vec{y}_i - c^T\vec{y}_r| \leq c_{\max}, \\ & \forall\ i, r \in \{1, \ldots, n\}\}. \end{aligned} \tag{6}$$

Let $Y(t)$ be the matrix that summarizes the token distribution resulting from Algorithm 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: In the following we define $x_i(t) = c^T y_i(t)$ for $i = 1, \ldots, n$. We define a Lyapunov-like function

$$V(t) = [V_1(t), V_2(t)] \tag{7}$$

consisting of two terms. The first one is:

$$V_1(Y(t)) = \sum_{i=1}^{n} (x_i(t) - c_{ave})^2 \tag{8}$$

The second one is a measure of the ordering of the loads:

$$V_2(t) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} f(x_i(t) - x_j(t)) \tag{9}$$

where

$$f(t) = \begin{cases} 1 & if\ x_i(t) - x_j(t) > 0 \\ 0 & if\ x_i(t) - x_j(t) \leq 0. \end{cases}$$

Note that here we are assuming that $e_{ae} = \{n, 1\}$ and nodes are labeled as in Fig. 2, namely following the cycle in the clockwise direction starting from 1, we meet nodes with an increasing index. 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 $V_1(t+1) \leq V_1(t)$.

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

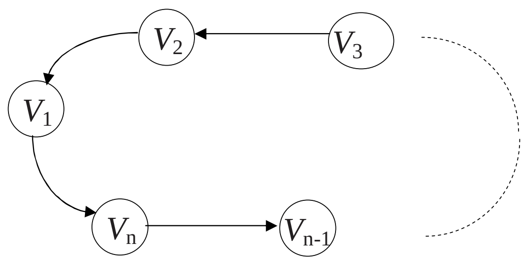

Fig. 2. The oriented Hamiltonian cycle considered in the proof of Theorem 1 and Proposition 3.

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 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

$$(x_i(t+1) - c_{ave})^2 + (x_r(t+1) - c_{ave})^2 <$$
$$(x_i(t) - c_{ave})^2 + (x_r(t) - c_{ave})^2$$

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.,

$$|x_i(t+1) - x_r(t+1)| \leq \max_{j \in \mathcal{K}_i(t) \cup \mathcal{K}_r(t)} c_j \leq c_{\max}. \qquad (10)$$

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) $V_2(t)$ 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 do the fact that in such 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 instant 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 exist an edge with strictly positive probability p that if selected makes $V(t+1) < V(t)$. The probability that such 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 edge is selected goes to 1 as t goes to infinity, thus proving the statement. □

A. Convergence time

In this section we discuss the expected convergence time of Algorithm 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:

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

Thus, $T_{con}(Y)$ represents the number of steps required at a certain execution of Algorithm 2 to reach the convergence set $\widetilde{\mathcal{Y}}$ starting from a given token distribution.

We denote as $\mathcal{E}[T_{con}(Y)]$ the *expected convergence time*.

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 2 to reach the set $\widetilde{\mathcal{Y}}$, starting from a given configuration.
- $T_{\max}$ is the maximum average time between two consecutive improvements of $V_1(Y)$ in any realization of Algorithm 2, starting from a given configuration.

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

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 [15] 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 1: [15] For the Hamiltonian Quantized Gossip 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 2: 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

$$T_{\max} = \frac{n(n-1)}{2}. \tag{11}$$

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 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 [7], [15].

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

Proposition 3: Let us consider a net with an Hamiltonian cycle. Let n be the number of nodes, and N be the number of arcs of the net.

It holds

$$T_{\max} \leq N(n-2). \tag{12}$$

Proof: We preliminary observe that, due to the gossip nature of Algorithm 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 $T_{\max}$.

Such upper bound is computed determining 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.

[3]The problem of random walk and average meeting times has been extensively studied in different applications [4], [22].

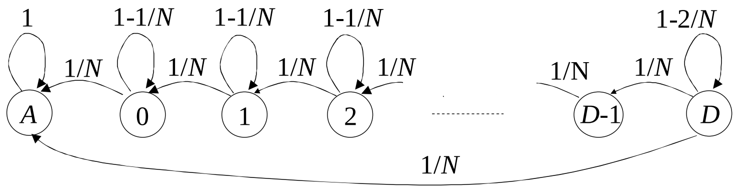

Fig. 3. The Markov chain associated to a net containing an Hamiltonian cycle.

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

The distance between the two agents is equal to the length of the *clockwise* path going from the first agent to the second one. 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 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.

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 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:

$$(I - P')\tau = \mathbf{1} \tag{13}$$

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; 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 (13), we found out that the maximum average hitting time of the absorbing state occurs when the distance

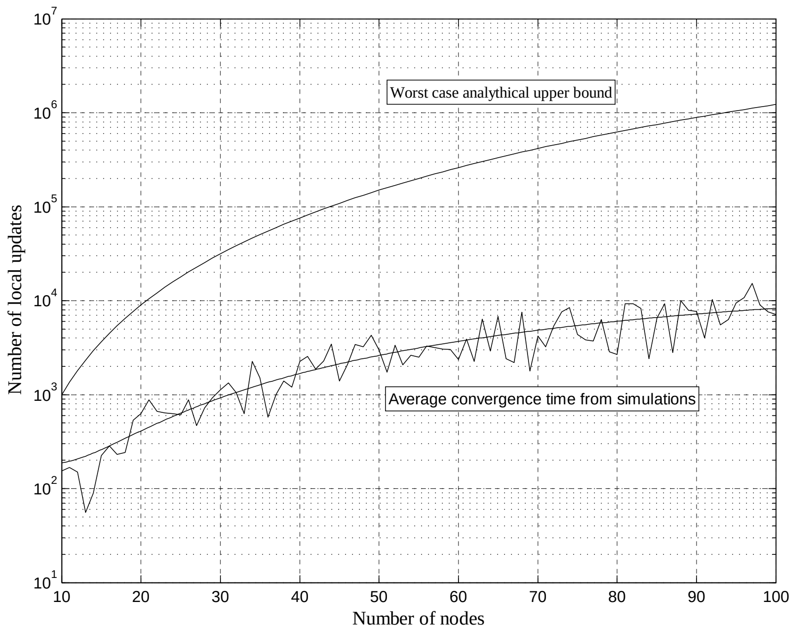

Fig. 4. Comparison between simulation results and the worst case analytical average converge time.

between the two nodes is equal to $D-1$. In particular, it holds $\tau(D-1) = N(n-2)/2$ that proves the statement. $\square$

Proposition 4: An upper bound to the average convergence time of Algorithm 2 is

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

Proof: Follows from Propositions 1 and 3 and the fact that, by definition, it is $\mathcal{E}[T_{con}(Y)] \leq N_{\max} \cdot T_{\max}$. $\square$

The above result is an improvement respect to [7], [15] where the convergence time for certain topologies (e.g. path networks) is $\mathcal{O}(n^4)$, while it is $\mathcal{O}(n^3)$ only for fully connected networks.

In Figure 4 is shown the average converge time for a ring network of n nodes with $n = 10, \ldots, 100$ with random initial loads ranging from 0 to 10. For each network size the average convergence time is taken over 100 realizations of the experiment. In such Figure is also shown a comparison with the previously computed upper bound to the average convergence time, it is evident that such 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.

B. Algorithm extension for convergence in finite time

The effectiveness of Algorithm 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

2 converges in finite time, as will be shown in the following. Furthermore if Algorithm 2 is applied to networks whose edge selection process is deterministic, it still preserve its convergence properties while other algorithms as the one in [15] may cycle indefinitely without reaching the consensus set of final configurations. Obviously Algorithm 2 prevents the existence of such cycles due to the deterministic swap rule. In particular, the following result holds.

Proposition *5:* If there exists a period of time T_h such that each edge along the Hamiltonian cycle is selected at least once, the convergence time of Algorithm 2 is

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

Proof: By Proposition 1 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 3), 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 $\widetilde{\mathcal{Y}}$. □

The above proposition states a finite time bound on the convergence time of Algorithm 2.

We note that to make Proposition 5 useful in practical cases, namely if we want to use it as a criterion to know when $\widetilde{\mathcal{Y}}$ is reached for sure, then a slight overhead needs to be added to Algorithm 2 to evaluate the maximum 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 Gossip 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 known by the authors [7], [15] with the feature of an embedded stopping criterion that will block the algorithm once quantized consensus has been achieved.

In this case we 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 thus ensuring a finite and known amount of total transfers for load balancing applications.

References

[1] M. Mesbahi A. Rahmani, M. Ji and M. Egerstedt. Controllability of multi-agent systems from a graph-theoretic perspective. *SIAM Journal on Control and Optimization*, 48:162–186, 2009.

[2] T. C. Aysal, M. J.Coates, and M. G. Rabbat. Distributed average consensus using probabilistic quantization. *Statistical Signal Processing, 2007. SSP '07. IEEE/SP 14th Workshop on*, pages 640–644, Aug. 2007.

[3] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Randomized gossip algorithms. *IEEE/ACM Trans. on Networking*, 14:2508–2530, 2006.

[4] N. H. Bshouty, L. Higham, and J. Warpechowska-Gruca. Meeting times of random walks on graphs. *Inf. Process. Lett.*, 69(5):259–265, 1999.

[5] 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. of the 7th Euromicro Workshop on Parallel and Distributed Processing*, pages 170–177, Funchal, Portugal, February 1999.

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

[7] 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, December 2007.

[8] 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.

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

[10] F. Meyer Auf Der Heide, B. Oesterdiekhoff, and R. Wanka. Strongly adaptive token distribution. In *Proc. of the 20th International Colloquium on Automata, Languages and Programming*, pages 398–409, 1993.

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

[12] 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*, 64:591–605, 2004.

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

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

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

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

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

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

[19] A. Speranzon S. Zampieri R. Carli, F. Fagnani. Communication constraints in the average consensus problem. *Automatica*, 44.

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

[21] M. J. Coates T. C. Aysal and M. G. Rabbat. Distributed average consensus with dithered quantization. *IEEE Transactions on Signal Processing*, 56(10), OCT 2008.

[22] P. Tetali and P. Winkler. On a random walk problem arising in self-stabilizing token management. In *PODC '91: Proc. of the tenth annual ACM symposium on Principles of distributed computing*, pages 273–280, New York, NY, USA, 1991. ACM.

[23] G. Turner and H. Schroder. Token distribution on reconfigurable d-dimensional meshes. In *Proc. of the First IEEE International Conference on Algorithms and Architectures for Parallel Processing*, pages 335–344, 1995.