Preprints of the 2012 20th Mediterranean Conference on Control & Automation (MED), Barcelona, Spain, July 3-6, 2012

Distributed Strategies for Balancing a Weighted Digraph

Christoforos N. Hadjicostis and Apostolos Rikos

***Abstract*— A weighted digraph is balanced if, for each node, the sum of the weights of the edges outgoing from that node is equal to the sum of the weights of the edges incoming to that node. Weight-balanced digraphs play a key role in a number of applications, including cooperative control, distributed optimization, and distributed averaging problems. We address the weight-balance problem for a distributed system whose components (nodes) can exchange information via interconnection links (edges) that form an arbitrary, possibly directed, communication topology (digraph). We develop two iterative algorithms, a centralized one and a distributed one, both of which can be used to reach weight-balance, as long as the underlying communication topology forms a strongly connected digraph (or is a collection of strongly connected digraphs). The centralized algorithm is shown to reach weight-balance after a finite number of iterations (bounded by the number of nodes in the graph). The distributed algorithm operates by having each node adapt the weights on its outgoing edges and is shown to asymptotically lead to weight-balance. We also analyze the rate of convergence of the proposed distributed algorithm and obtain a (graph-dependent) worst-case bound for it. Finally, we provide examples to illustrate the operation, performance, and potential advantages of the proposed algorithms.**

I. Introduction and Background

The successful operation of a distributed system or network depends on a number of basic protocols to circulate and process data between its components. In distributed systems whose functionality does not simply consist of transmitting data, but also involves control and decision tasks (e.g., workload balancing across available computing resources and leader election), routing protocols may become unnecessary, insufficient, or difficult to maintain (e.g., in mobile environments). For this reason, the design of algorithms and protocols for distributed computation has attracted significant attention by the communication, control and computer science communities over the past few decades (e.g., [1]–[6], and references therein). In particular, given a distributed system whose components (nodes) can exchange information via interconnection links (edges) that form an arbitrary, possibly directed, communication topology (digraph), this work has revealed that the structure of this digraph (also called communication digraph) is one of the critical factors in our ability to perform a variety of distributed computational tasks.

Weight-balance is a property that is important in many well-studied cases of distributed computation, particularly when the components (nodes) want to distributively average their individual measurements (in this scenario, each node provides an initial value which could represent a local measurement of global quantity). This is a special case of the consensus problem which has received significant attention from the computer science community [1] and the control community (see [2], [8], [9]), due to its applicability to many topics such as multi-agent systems, cooperative control, and the modeling of flocking behavior in biological and physical systems (e.g., [2], [8], [10]). Specifically, one approach towards average consensus is to follow a linear iteration where (instead of routing the value of each node to all other nodes) each node repeatedly updates its value to be a weighted linear combination of its own value and the values of its neighboring nodes. The choice of the weights is important in how the interconnection behaves and whether average consensus is reached. For example, when the weights used in the linear iteration form a doubly stochastic matrix, asymptotic (average) consensus can be reached, that is, after running the linear iteration, the nodes asymptotically reach consensus to the average of their initial values [2], [10]–[12]. When the weights form a row stochastic matrix, consensus is reached but not necessarily to the average of their initial values [2].

The main contribution of this paper is to propose and analyze two algorithms (one centralized and one distribubed) that can be used in distributed systems with (possibly directed) interconnection topologies to achieve weight-balance. As explained in [13], [14], once weight-balance is achieved, the nodes can easily obtain weights that form a doubly stochastic matrix in a distributed manner (in order to use them to, for instance, asymptotically reach average consensus). We first analyze the centralized algorithm and then describe the distributed algorithm for weight-balance. In the process, we also obtain and compare the rates of convergence of both balancing algorithms, and establish worst-case bounds. We also present examples and simulation results.

II. Background and Notation

In this section, we establish some basic notions and notation that are needed for our development. In a distributed system whose components (nodes) can exchange information via interconnection links (edges), the exchange of information between components can be captured by digraph (directed graph). A digraph is defined as $G = (V, E)$ where

This material is based upon work supported in part by the European Community (EC) 7^{th} Framework Programme (FP7/2007-2013), under grant agreements INFSO-ICT-223844 and PIRG02-GA-2007-224877. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of EC.

The authors are with the Department of Electrical and Computer Engineering at the University of Cyprus, Nicosia, Cyprus. Corresponding author: Christoforos Hadjicostis, Department of Electrical and Computer Engineering, University of Cyprus, 75 Kallipoleos Avenue, P.O. Box 20537, 1678 Nicosia, Cyprus. Email: chadjic@ucy.ac.cy

978-1-4673-2529-5/12/$31.00 ©2012 IEEE

$V = (v_1, v_2, \cdots, v_n)$ is the vertex set and $E \subseteq V \times V$ is the edge set. We have $(v_j, v_i) \in E$ if there exists an edge from v_i to v_j (which captures the fact that v_j can receive information from v_i). By convention, the digraph contains no self-loops, i.e., $(v_i, v_i) \notin E$ for all $v_i \in V$. In this paper we do not necessarily require bi-directional communication links[1] but we do assume that the given digraph $G = (V, E)$ is strongly connected.[2] A strongly connected digraph is a directed graph in which there is a path from each vertex $v_i \in V$ to every other vertex $v_j \in V$, $v_i \neq v_j$ i.e., there exists a sequence of nodes (called a path) $v_i = v_{i_1}, v_{i_2}, \cdots, v_{i_t} = v_j$, such that $(v_{i_{e+1}}, v_{i_e}) \in E$, $e = 1, 2, \cdots, t-1$.

In a digraph $G = (V, E)$, the subset of nodes that can directly transmit information to node v_j is called the set of in-neighbors of v_j and is represented by $N_j^- = \{v_i \in V \mid (v_j, v_i) \in E\}$. Respectively, the subset of nodes that can receive information from v_j is called the set of out-neighbors of v_j and is represented by $N_j^+ = \{v_i \in V \mid (v_i, v_j) \in E\}$. The number of in-neighbors of v_j is called the in-degree of v_j and is denoted by $D_j^- = |N_j^-|$. The number of out-neighbors of v_j is called the out-degree of v_j and is denoted by $D_j^+ = |N_j^+|$. Any digraph can be equivalently represented by its adjacency matrix $A = [a_{ji}]$ of size $n \times n$ (where n is the number of nodes in the graph, i.e., $n = |V|$), and $a_{ji} = 1$ if $(v_j, v_i) \in E$ else $a_{ji} = 0$ (if $(v_j, v_i) \notin E$). Obviously, $D_j^- = \sum_{i=1}^n a_{ji}$ and $D_j^+ = \sum_{i=1}^n a_{ij}$.

III. Problem Formulation

Definition 1: A weighted digraph is a triplet $G_W = (V, E, W)$, where $G = (V, E)$ with $V = (v_1, v_2, \cdots, v_n)$, $E \subseteq V \times V$, is a digraph, and $W = [w_{ji}]$ is a nonnegative weight matrix, such that edge $(v_j, v_i) \in E$ is assigned a nonzero weight w_{ji} for all $(v_j, v_i) \in E$ (note that $w_{ji} = 0$ for all $(v_j, v_i) \notin E$).

Definition 2: Given a weighted digraph $G_W = (V, E, W)$, the total in- and out-weight of node v_j is defined respectively as

$$S_j^- = \sum_{i=1}^n w_{ji} \quad , \quad S_j^+ = \sum_{i=1}^n w_{ij} \ .$$

Definition 3: A weighted digraph $G_W(V, E, W)$ is called weight-balanced if the total in-weight equals the total out-weight for every node $v_j \in V$, i.e., $S_j^- = S_j^+$. Equivalently, if we let x_j be the imbalance of node v_j, defined as

$$x_j = S_j^- - S_j^+ \ ,$$

then a weighted digraph is weight-balanced if $x_j = 0, \forall v_j \in V$.

Given a digraph $G = (V, E)$, our goal will be to obtain a weighted digraph $G_W = (V, E, W)$ with nonnegative weights, i.e., we need to obtain a weight matrix $W = [w_{ji}]$ with weights that satisfy the following conditions:

1) $w_{ji} > 0$ for each edge $(v_j, v_i) \in E$;
2) $w_{ji} = 0$ if $(v_j, v_i) \notin E$;
3) $S_j^+ = S_j^-$ (i.e., $x_j = 0$) for every $v_j \in V$.

Balancing a weighted digraph can be accomplished via a variety of algorithms. We introduce and analyze two algorithms that, to the best of our knowledge, are novel: (a) a centralized algorithm, which serves as a benchmark for the performance of distributed algorithms, and (b) a distributed algorithm that appears to outperform existing algorithms suggested in the literature. Both algorithms achieve weight-balance as long as the underlying digraph is strongly connected or is a collection of strongly connected digraphs. As discussed in [7], [13], this is a necessary and sufficient condition for balancing to be possible.

IV. Centralized Algorithm for Weight-Balance

The centralized algorithm takes as input a strongly connected digraph $G = (V, E)$. It initializes the weights of all edges to unity and then iteratively performs the following steps until the graph is balanced:

1) Compute the weight imbalance of each node.
2) Pick one node with positive imbalance and one with negative imbalance (any two such nodes will do and, unless the graph is balanced, it is always possible to find two such nodes).
3) Find a path in the digraph from the node with positive imbalance to the node with negative imbalance (this is always possible as long as the graph is strongly connected).
4) Increase the weights of all the edges in the path by the value of the weight imbalance of the positively imbalanced node.

We discuss why the algorithm results in a weight-balanced graph (and how many steps it takes to do so), after we describe the algorithm more formally.

A. Description of the Centralized Algorithm

Input: A strongly connected digraph $G = (V, E)$ with $n = |V|$ nodes and $m = |E|$ edges.
Initialization: Set $k = 0$ and the edge weights as

$$w_{ji}[0] = \begin{cases} 0, & \text{if } (v_j, v_i) \notin E, \\ 1, & \text{if } (v_j, v_i) \in E. \end{cases}$$

Iteration:

1) For each node $v_j \in V$, compute its weight imbalance

$$x_j[k] = S_j^-[k] - S_j^+[k] = \sum_{i=1}^n w_{ji}[k] - \sum_{i=1}^n w_{ij}[k].$$

Exit if digraph is balanced (i.e., if $x_j[k] = 0$ for all $v_j \in V$).

2) Select one node v^+ with positive imbalance br^+ and one node v^- with negative imbalance br^- (e.g., select the node with the largest positive imbalance and the node with the largest absolute negative imbalance, respectively).

3) Find a non-cyclic path $v^+ = v_{j_1}, v_{j_2}, \cdots, v_{j_t} = v^-$ from v^+ to v^-.

[1]Note that the weight-balance problem is rather trivial in undirected graphs. Directed communication topologies can arise in a variety of distributed systems, e.g., due to nodes with diverse wireless transmission capabilities (because of varying power constraints or because of different interference conditions at each node).

[2]The requirement for a strongly connected digraph is a rather typical (and benign) assumption for many distributed algorithms, including iterative algorithms for average consensus [2], [10]–[12].

4) Increase the weight on each edge on the path, which connects v^+ to v^-, by br^+, i.e.

$$w_{j_{e+1},j_e}[k+1] = w_{j_{e+1},j_e}[k] + br^+$$

for $e = 1, 2, \cdots, t-1$. (Leave all other weights unchanged.)

5) Set $k = k+1$ and repeat the iteration (back to Step 1).

We first illustrate the centralized algorithm via an example. We then explain why it results in a weight-balanced digraph after a finite number of iterations (bounded by n in the worst-case).

Example 1: Consider the digraph $G = (V, E)$ in Figure 1, where $V = (v_1, v_2, \cdots, v_7)$, $E = (e_1, e_2, \cdots, e_{11})$, $E \subseteq V \times V$. The weight on each edge is initialized to $w_{ji}[0] = 1$ for $(v_j, v_i) \in E$ (otherwise $w_{ji}[0] = 0$). As a first step, we compute the weight imbalance $x_j[0] = S_j^-[0] - S_j^+[0]$ for each node (this is shown in Figure 1).

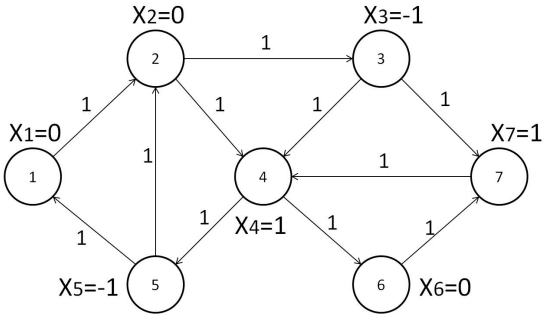

Fig. 1. Weighted digraph with initial weights and initial imbalance for each node.

Once we compute the imbalance of each node, the centralized algorithm selects (randomly or otherwise) one node with positive imbalance, say v_7, and one node with negative imbalance, say v_5. A path from node v_7 to node v_5 is selected (e.g., the path v_7, v_4, v_5) and the weights of all the edges in the path are increased by the value of the weight imbalance of the positively imbalanced node v_7 (namely, by the value of the weight imbalance $x_7 = 1$), as shown in Figure 2.

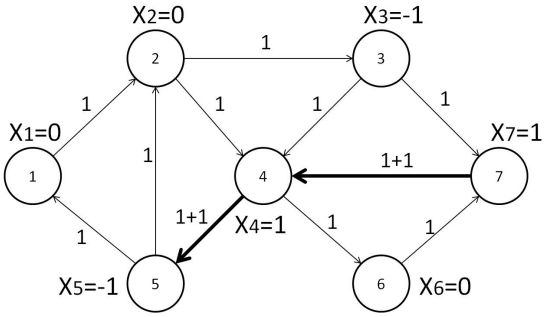

Fig. 2. Path selection in weighted digraph between a node with positive imbalance and a node with negative imbalance.

In the next iteration, after the increase of the weights of all the edges of the path, we recalculate the imbalance for each node v_j as $x_j[1] = S_j^-[1] - S_j^+[1]$ and perform the same steps as before; the process is repeated until the graph becomes

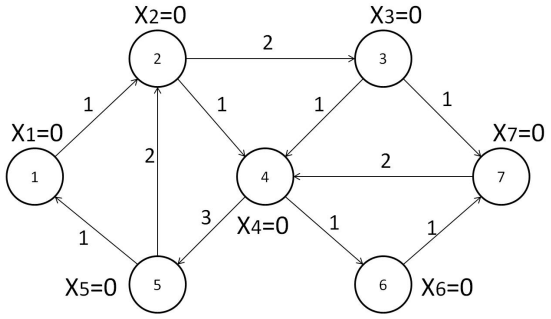

Fig. 3. Resulting weight-balanced graph (centralized algorithm).

weight-balanced. In this particular example, this occurs after two iterations and the final weights are shown in Figure 3. □

B. Analysis of Centralized Algorithm

Notice that each edge appears as the incoming edge of exactly one node and as the outgoing edge of exactly one (other) node. Thus, we immediately have that

$$\sum_{j=1}^{n} x_j[k] = 0 \ , \ \text{for all } k \ .$$

This means that if there is a node with positive imbalance at Step 2 of the iteration, then there has to be at least one node with negative imbalance (and vice-versa). Also, notice that the adjustment of weights in Step 3 of the iteration, achieves the following:

1) It balances the node v^+ that had positive imbalance.
2) It does not change the imbalance of all other intermediate nodes $v_{j_2}, \cdots, v_{j_{t-1}}$ in the path $v^+ = v_{j_1}, v_{j_2}, \cdots, v_{j_t} = v^-$ from v^+ to v^-.
3) It increases the imbalance of the node v^- that had negative imbalance.

Note that if a node starts balanced or becomes balanced during any iteration, it remains balanced for the remainder of the algorithm (because Steps 1, 2, and 3 do not affect the imbalance of intermediate nodes in the path that is selected, and also balanced nodes cannot be selected in Step 2 of the Centralized Algorithm). Furthermore, at each iteration, at least one node becomes balanced (namely, the node with positive imbalance that is picked at Step 1). Note that it is possible for two nodes to become balanced at each iteration (if the node with negative imbalance that is picked happens to also become balanced; in fact, this is the case at the last iteration). Thus, it is easy to see that the Centralized Algorithm takes at most $n-1$ iterations to reach a set of weights that forms a weight-balanced graph.

Another easily obtainable bound on the number of iterations is the following: if we think of the *absolute balance* of the graph at iteration k as $\varepsilon[k] = \sum_{j=1}^{n} |x_j[k]|$, then each iteration decreases this imbalance by at least 2 (i.e., $\sum_{j=1}^{n} |x_j[k+1]| \leq \sum_{j=1}^{n} |x_j[k]| - 2$) unless the graph is balanced. [Note that $\sum_{j=1}^{n} |x_j[0]|$ is necessarily an even number (because the sum of positive imbalances is equal to

the negative of the sum of the negative imbalances, and both of them are integer numbers). Also, each iteration decreases the sum of the positive imbalances by at least 1; thus, the absolute sum of the negative imbalances also has to decrease by at least 1 as well.]

The above discussion implies the proposition below.

Proposition 1: The number of iterations T required by the proposed centralized algorithm to balance a digraph $G = (V, E)$ satisfies $T \leq \min(n-1, \frac{1}{2}\sum_{j=1}^{n} |x_j[0]|)$.

V. Distributed Algorithm for Weight-Balance

We now introduce an algorithm in which the nodes distributively adjust the weights of their outgoing edges such that the digraph becomes weight-balanced. We assume that each node observes but cannot set the weights of its incoming edges. Given a strongly connected digraph $G = (V, E)$, the distributed algorithm has each node initialize the weights of all of its outgoing edges to unity. Then, it enters an iterative stage where each node performs the following steps:

1) It computes its weight imbalance.
2) If it has positive imbalance, it increases the weights of its outgoing edges by an equal amount so that it becomes weight-balanced (assuming no further changes by its in-neighbors on its incoming edges).

We discuss why the above distributed algorithm asymptotically obtains weights that balance the graph (and the rate with which it achieves this) after we describe the algorithm in more detail. For simplicity, we assume that during the execution of the distributed algorithm, the nodes update the weights on their outgoing edges in a synchronous[3] manner.

A. Description of the Distributed Algorithm

Input: A strongly connected digraph $G = (V, E)$ with $n = |V|$ nodes and $m = |E|$ edges.

Initialization: Set $k = 0$ and initialize the edge weights as

$$w_{ji}[0] = \begin{cases} 0, & \text{if } (v_j, v_i) \notin E, \\ 1, & \text{if } (v_j, v_i) \in E. \end{cases}$$

Iteration:

1) Each node $v_j \in V$ computes its weight imbalance $x_j[k]$.

2) Each node v_j^+ with positive imbalance br_j^+ increases by an equal amount the values $w_{ij}[k+1]$, $v_i \in N_j^+$, of the weights on its outgoing edges. Specifically, if node v_j^+ has positive imbalance br_j^+, then for every $v_i \in N_j^+$ we have

$$w_{ij}[k+1] = w_{ij}[k] + br_j^+ / D_j^+ \ , \ \ \forall v_j \in N_j^+ \ ,$$

where D_j^+ is the out-degree of node v_j.

3) Set $k = k+1$ and repeat the iteration (back to Step 1).

Example 2: Consider again the digraph $G = (V, E)$ in Figure 1. The weight on each edge is initialized to $w_{ji}[0] = 1$ for $(v_j, v_i) \in E$ (otherwise, $w_{ji}[0] = 0$). As a first step, each node computes its weight imbalance $x_j[0] = S_j^-[0] - S_j^+[0]$ (this is identical to what is shown in Figure 1 for each node). Once the nodes compute their imbalance, the distributed algorithm requires each node with positive imbalance to increase the value of the weights on its outgoing edges by an equal amount so that the total increase makes the node balanced (assuming the weights on its incoming edges do not change). In this case, the nodes that have positive imbalance (equal to 1) are nodes 4 and 7, which distribute equally their imbalance to their outgoing edges as shown in Figure 4.

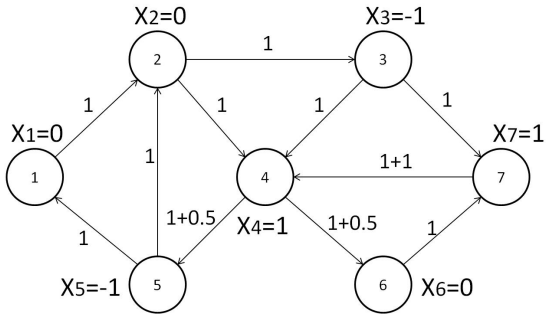

Fig. 4. Equal distribution of weight imbalance to outgoing edges by nodes with positive imbalance.

In the next iteration, after the equal weight distribution to the outgoing edges from each node with positive imbalance at $k = 0$, the nodes recalculate their imbalance as $x_j[1] = S_j^-[1] - S_j^+[1]$ and the process is repeated. The asymptotic weights obtained in this example are shown in the graph of Figure 5. □

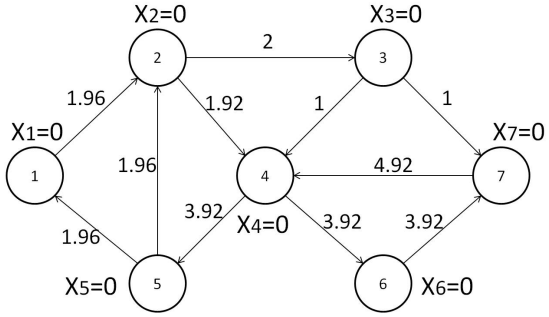

Fig. 5. Resulting weight-balanced graph (distributed algorithm).

B. Rate of Convergence of Distributed Algorithm

We now analyze the rate of convergence of the distributed algorithm by calculating an upper bound on the worst-case balancing scenario. Due to space considerations we only provide a sketch of the argument.

Consider a node v_j that at some point in the distributed algorithm, say at iteration k, has nonnegative imbalance br_j^+. This node tries to achieve zero imbalance (by increasing the weights on its outgoing edges by $\frac{br_j^+}{D_j^+}$ where D_j^+ is its out-degree). Note, however, that at the end of the iteration, node v_j might end up with positive imbalance because some of the weights of its incoming edges might increase (due to the actions of its in-neighbors). We conclude that nodes with nonnegative imbalance retain a nonnegative imbalance for

[3]Even though we do not discuss this issue in the paper, asynchronous operation is not a problem for the distributed algorithm.

the duration of the algorithm. On the contrary, nodes with negative imbalance (which do not take any action at Step 2 of the iteration) might have their imbalance change to nonnegative (due to the actions of their in-neighbors); however, once their imbalance becomes nonnegative, they have to remain at a nonnegative imbalance for the remaining iterations. Also note that (as argued earlier) $\sum_{j=1}^{n} x_j[k] = 0$, for all k; thus, at least one node will have a negative imbalance at any given iteration (unless the graph is balanced).

Given the above invariant property (i.e., the property that nodes with nonnegative imbalance, retain their nonnegative imbalance), we can think of the execution of the distributed algorithm as a flow of positive (excess) imbalance to nodes with negative imbalance, which essentially act as absorbing nodes (at least until their imbalance becomes nonnegative). Thus, depending on how many and which nodes have negative imbalance, this flow may be faster or slower. Sketching the proof, we argue that the worst-case occurs when there is a single node with negative imbalance (and all other nodes have nonnegative imbalance that they try to direct towards this single node with negative imbalance).

Consider the absolute balance $\varepsilon[k] = \sum_{j=1}^{n} |x_j[k]|$ of the weighted digraph at a given iteration k. We can obtain the total decrease of the absolute balance $\varepsilon[k+n]$ (after n iterations) by considering the worst-case path(s) from each node with nonnegative imbalance to every node with negative imbalance. For the worst-case scenario where node v_j is the only (thus last) node that has negative imbalance, this can be calculated based on the graph structure as follows. Define the matrix $W^{(j)}$ as

$$\begin{aligned} W^{(j)}(j,j) &= 1 , \\ W^{(j)}(i,j) &= 0 \text{ for } i \neq j , \\ W^{(j)}(i,l) &= \frac{1}{D_l^+} \text{ for } l \neq j , i \in N_l^+ , \\ W^{(j)}(i,l) &= 0 , \text{ for } l \neq j , i \notin N_l^+ . \end{aligned}$$

Note that $W^{(j)}$ essentially has node v_j acting as an absorbing node whereas all other nodes distribute equally their imbalance on their out-going neighbors. We can use $W^{(j)}$ to obtain a bound on $\varepsilon[k+n]/\varepsilon[k]$ as follows: we can calculate the total weight of all the paths from every node to node v_j by

- First calculating the matrix $W_{paths} = (W^{(j)})^n$.
- Selecting the minimum value $\rho_j = \min_i W_{paths}(i,j)$ of the entries of row j of matrix W_{paths} (let $i_{\min} = \text{argmin}_i W_{paths}(i,j)$).

The minimum value ρ_j is guaranteed to be nonzero (because the graph is connected). It captures the total weight of all paths from node $i_{\min}$ to node v_j where the weight of the path is the product of the weights on its edges (as defined in matrix $W^{(j)}$). This also captures how much excess imbalance node $v_{i_{\min}}$ can direct (in n or less iterations) to node v_j (which is the only one that has negative imbalance).

Clearly, if we find $\rho = \min_j \rho_j$, then we obtain a worst-case convergence rate for $\varepsilon[k]$. More specifically, we are guaranteed that $\varepsilon[k+n] = (1-\rho)\varepsilon[k]$. Note that ρ is guaranteed to be nonzero as long as the graph is connected. In practice, in obtaining the minimum ρ_j we only need to consider nodes v_j that start with negative imbalance (because those that start with nonnegative imbalance will never reach negative imbalance).

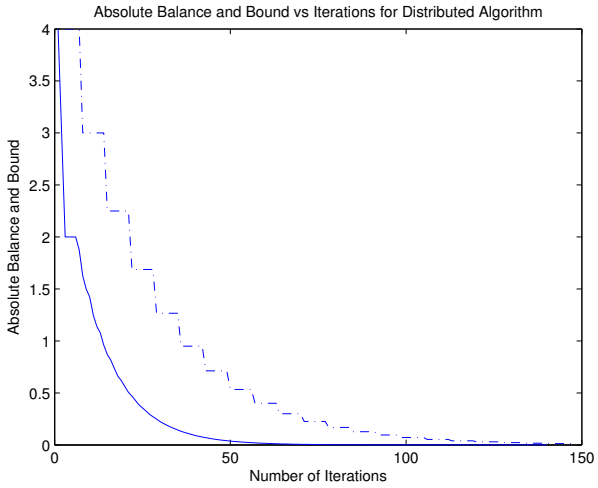

Fig. 6. Absolute balance and error bound for the distributed algorithm, plotted against the number of iterations for the digraph in Figure 1.

Example 3: In this example, we verify the above convergence rate bound for the digraph in Figure 1. In this particular case we calculate $\rho = 0.25$. In Figure 6 we plot the absolute balance $\varepsilon[k] = \sum_{j=1}^{n} |x_j[k]|$ as it evolves during the execution of the algorithm against the bound obtained in the above discussion (note that the bound remains flat for $n = 7$ iterations at a time). □

VI. Results and Comparisons

In this section we present simulation results and comparisons. In our plots we are typically concerned with the absolute balance $\varepsilon[k] = \sum_{j=1}^{n} |x_j[k]|$ as it evolves during the execution of the algorithms. We present numerical results for larger graphs (of sizes $n = 50, 200$) focusing on the distributed algorithm, which we compare against the *imbalance-correcting algorithm* in [14] in which every node v_j adds its entire weight imbalance x_j to one of its outgoing edges, namely the one with the lowest weight w. Specifically, the imbalance-correcting algorithm operates similarly to the proposed distributed algorithm with the only difference being that, at Step 2 of the iteration, each node with positive imbalance increases the weight of only one of its out-going edges, namely the one with the smallest weight, by the value of its imbalance.

Figure 7 shows what happens in the case of a randomly created graph of 50 nodes (left plot) and averaged over 100 randomly created graphs (right plot) of 50 nodes each. In the first case we plot the absolute balance $\varepsilon[k] = \sum_{j=1}^{50} |x_j[k]|$ and in the second case the average absolute balance $\varepsilon[k] = \frac{1}{100} \sum_{i=1}^{100} \sum_{j=1}^{50} |x_j^{(i)}[k]|$ over 100 graphs (indexed by $i = 1, 2, \cdots, 100$), as a function of the number of iterations k for the distributed (solid line), and the imbalance-correcting

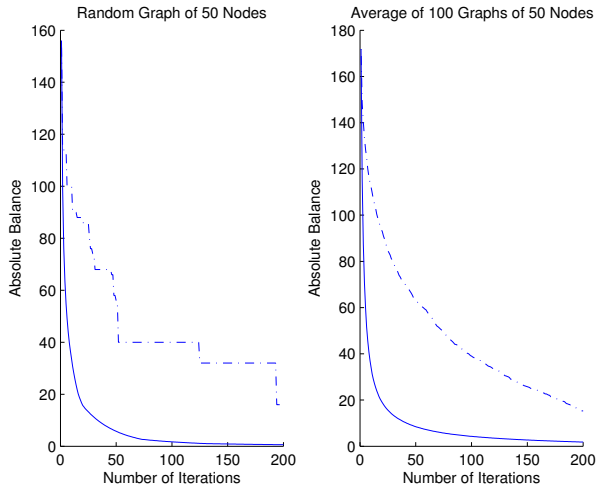

Fig. 7. Absolute balance plotted against the number of iterations for a random graph of 50 nodes (left plot) and averaged over 100 graphs of 50 nodes each (right plot) for the distributed (solid line) and the imbalance-correcting (dotted-dashed line) balancing algorithms.

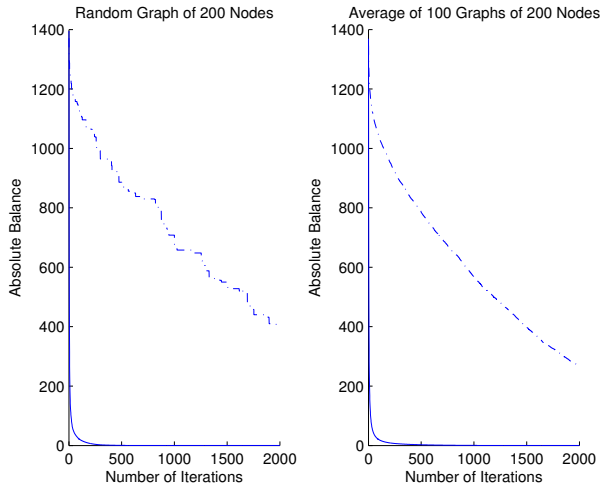

Fig. 8. Absolute balance plotted against the number of iterations for a random graph of 200 nodes (left plot) and averaged over 100 graphs of 200 nodes each (right plot) for the distributed (solid line) and the imbalance-correcting (dotted-dashed line) balancing algorithms.

algorithm (dashed-dotted line). The plots suggest that the distributed algorithm results in fast convergence whereas the imbalance-correcting algorithm leads to slower convergence. This is also the case in Figure 8 where we plot the absolute balances for graphs with 200 nodes. As we can see, the distributed algorithm appears to lead to faster convergence compared to the imbalance-correcting algorithm, especially as the number of nodes increases. Note, however, that the imbalance-correcting algorithm is an algorithm that is guaranteed to complete in a finite number of steps (as shown in [14]). The asymptotic nature of the proposed algorithm would normally be a disadvantage (as one would typically first need to obtain the proper weights before executing any —average or other— consensus algorithm). Nevertheless, there are a number of applications where the consensus iteration can be run in parallel with the weight update, enabling the nodes to simultaneously reach proper weights and (average or other) consensus.

VII. Conclusions

We have developed two iterative algorithms for balancing a weighted digraph. The first algorithm is centralized and is shown to balance the graph after a finite number of iterations (bounded by the numbers of nodes in the graph). The second algorithm is distributed and allows the nodes to asymptotically reach weight-balance, at a rate that we bounded explicitly based on the graph structure. Simulations indicate that the distributed algorithm, though asymptotic in nature, performs very well compared to existing algorithms. In the future, we plan to study hybrid approaches in which the nodes distributively run the proposed algorithm for some iterations and then switch to a variant of the imbalance-correcting algorithm for the remaining iterations.

VIII. Acknowledgements

The authors would like to thank Themistoklis Charalambous and Shreyas Sundaram for helpful discussions.

References

[1] N. Lynch, *Distributed Algorithms*. San Mateo, CA: Morgan Kaufmann Publishers, 1996.

[2] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[3] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, Oct. 2003.

[4] M. Rabbat and R. D. Nowak, "Distributed optimization in sensor networks," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN)*, pp. 20–27, 2004.

[5] A. Giridhar and P. R. Kumar, "Computing and communicating functions over sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 755–764, Apr. 2005.

[6] J. Hromkovic, R. Klasing, A. Pelc, P. Ruzicka, and W. Unger, *Dissemination of Information in Communication Networks*. Springer-Verlag, 2005.

[7] L. Hooi-Tong, "On a class of directed graphs — with an application to traffic-flow problems," *Operations Research*, vol. 18, no. 1, pp. 87–94, Feb. 1970.

[8] A. Jadbabaie, J. Lin, and A. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, June 2003.

[9] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, no. 1, pp. 65–78, Sep. 2004.

[10] W. Ren and R. W. Beard, *Distributed Consensus in Multi-vehicle Cooperative Control: Theory and Applications*. Communication and Control Engineering Series, Springer Verlag, New York, 2007.

[11] J. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, 1984.

[12] J. Cortés, "Distributed algorithms for reaching consensus on general functions," *Automatica*, vol. 44, no. 3, pp. 726–737, Mar. 2008.

[13] B. Gharesifard and J. Cortés, "When does a digraph admit a doubly stochastic adjacency matrix?" in *Proceedings of American Control Conference*, pp. 2440–2445, July 2010.

[14] B. Gharesifard and J. Cortés, "Distributed strategies for making a digraph weight-balanced," in *Proceedings of 47th Annual Allerton Conference on Communication, Control, and Computing*, pp. 771–777, Sept. 2009.