Distributed strategies for making a digraph weight-balanced

Bahman Gharesifard and Jorge Cortés

***Abstract*— A digraph is weight-balanced if, at each node, the sum of the weights of the incoming edges (in-degree) equals the sum of the weights of the outgoing edges (out-degree). Weight-balanced digraphs play an important role in a variety of cooperative control problems, including formation control, distributed averaging and optimization. We call a digraph weight-balanceable if it admits an edge weight assignment that makes it weight-balanced. It is known that semiconnectedness is a necessary and sufficient condition for a digraph to be weight-balanceable. However, to our knowledge, the available approaches to compute the appropriate set of weights are centralized. In this paper, we propose a distributed algorithm running synchronously on a directed communication network that allows individual agents to balance their in- and out-degrees. We also develop a systematic centralized algorithm for constructing a weight-balanced digraph and compute its time complexity. Finally, we modify the distributed procedure to design an algorithm which is distributed over the mirror digraph and has a time complexity much smaller than the centralized algorithm.**

I. Introduction

Weight-balanced digraphs have been shown to play a crucial role in the distributed coordination of networks of dynamic agents. This class of digraphs is an integral part in deriving a Lyapunov function for convergence analysis of average-consensus [1], [2] and consensus on general functions [3]. Furthermore, weight-balanced digraphs appear in the design of stable flocking algorithms for agents with significant inertial effects, where the weight-balanced assumption allows decoupling the centroid dynamic from the internal group formation [4]. In [5] a traffic-flow problem is introduced with n junction and m one-way streets with the goal of ensuring a smooth traffic flow. It is shown that the problem can be reduced to computing weights on the edges of the associated digraph that makes the digraph weight-balanced, in the sense that the sum of the in- and the out-degrees are equal at each junction. Furthermore, necessary and sufficient conditions are given for a digraph to be weight-balanced and a centralized algorithm is presented for computing the weight on each edge. It is thus an important question to design distributed algorithms that allow agents to balance their in- and out-degrees so that the overall interaction digraph is weight-balanced.

The main contribution of this paper is a synchronized distributed algorithm on a directed communication network in which each agent balances its in- and out-degrees. In this algorithm, each individual agent sends a message to one of its out-neighbors and receives messages from its in-neighbors. Our next step is to systematize the centralized algorithm of [5] using the fundamental cycle matrix. We compute the time complexity of this centralized algorithm and show that it does not scale well with the size of the network. Finally, we introduce a modified version of the weight-balance distributed algorithm, distributed over the mirror digraph, and we characterize its time complexity. We show that the convergence in this algorithm is much faster than that of the centralized one. We conclude with some remarks and ideas for future work.

Bahman Gharesifard and Jorge Cortés are with the Department of Mechanical and Aerospace Engineering, University of California San Diego, {bgharesifard,cortes}@ucsd.edu

II. Problem Statement

We adapt some basic notions from [6]. Let $\mathsf{V} \subset \mathbb{R}^n$ be a subspace and $\alpha = (\alpha_1, \dots, \alpha_n) \in \mathsf{V}$. We denote $(|\alpha_1|, \dots, |\alpha_n|)$ by $|\alpha|$. We denote a digraph by $G = (V, E)$, where V is a finite set, called the vertex set, and $E \subseteq V \times V$, called the edge set. For a digraph with an edge $(u, v) \in E$, u is called the *in-neighbor* of v and v is called the *out-neighbor* of u. We denote the set of in-neighbors and out-neighbors of v, respectively, with $\mathcal{N}_G^{\text{in}}$ and $\mathcal{N}_G^{\text{out}}$. The *in-degree* and *out-degree* of v are the cardinality of $\mathcal{N}_G^{\text{in}}$ and $\mathcal{N}_G^{\text{out}}$, respectively. A digraph is called *topologically balanced* if it has the same in- and out- degrees. A *directed path* in a digraph is an ordered sequence of vertices so that any two consecutive vertices in the sequence are an edge of the digraph. A *cycle* in a digraph is a directed path that starts and ends at the same vertex and has no other repeated vertex.

A *weighted digraph* is a triplet $G = (V, E, A)$, where the pair (V, E) is a digraph and $A \in \mathbb{R}_{\geq 0}^{n \times n}$, called the *adjacency matrix*, where n is the number of elements of V. The adjacency matrix has the property that, for all $i, j \in \{1, \dots, n\}$, the entry $a_{ij} > 0$ if $(v_i, v_j) \in E$ and $a_{ij} = 0$ otherwise. For a weighted digraph the out-degree and in-degree are, respectively, defined by

$$d_{\text{out}}(v_i) = \sum_{j=1}^{n} a_{ij}, \qquad d_{\text{in}}(v_i) = \sum_{j=1}^{n} a_{ji}.$$

A weighted digraph $G = (V, E, A)$ is called *weight-balanced* if $d_{\text{out}}(v_i) = d_{\text{in}}(v_i)$, for all $v_i \in V$. A digraph $G = (V, E)$ is called *strongly semiconnected* if the existence of a path from v_i and v_j implies the existence of a path from v_j to v_i, for all $v_i, v_j \in V$. The following two theorems establish a constructive *centralized* approach for determining whether a digraph is weight-balanceable.

Theorem 2.1 ([5]): A digraph $G = (V, E)$ is weight-balanced if and only if the edge set E can be decomposed into k subsets $E_1, \dots, E_k$ such that the following statements hold

1) $E = E_1 \cup E_2 \cup \ldots \cup E_k$ and
2) every subgraph $G = (V, E_i)$, for $i = \{1, \ldots, k\}$, is a weight-balanced digraph.

The following theorem reveals the importance of cycles in the weight-balanced digraphs.

Theorem 2.2 ([5]): Let $G = (V, E)$ be a directed digraph. The following statements are equivalent.

1) Every element of E lies in a cycle.
2) G is weight-balanced.
3) G is strongly semiconnected.

Although the approach taken in [5] is constructive, it relies on computing the cycles no systematic algorithm is proposed for computing the cycles.

III. Weight-balanced distributed algorithm

Consider a network of robotic agents with a strongly semiconnected graph topology $G = (V, E)$. In the following, we introduce an algorithm in which the agents synchronously compute the weights on each edge such that the digraph is weight-balanced.

Informal description

1) Each agent can send messages to its out-neighbors and receive messages from its in-neighbors. Thus each agent can compute its in- and out-degrees.
2) For each agent, if the in-degree is more than the out-degree, the agent changes the weight on one of the out-edges with the minimum weight such that she is balanced.
3) Each agent updates the in- and out- degrees in the next round and repeats the above process.

Note that this algorithm updates the weights synchronously. In following, we give a formal description of the distributed algorithm presented above. Furthermore, we demonstrate that the algorithm converges to a weight-balanced digraph.

Formal description

Suppose that a communication network is given by $G = (V, E, A)$, where (V, E) is strongly semiconnected. Let $X \subset \mathbb{R}^{n\times n}$ be a subspace generated by all possible adjacency matrices associated to (V, E), thus $A \in X$. We define an evolution by (X, f), where f is defined as follows. For $i \in \{1, \ldots, n\}$, let

$$a_i^* = \min_{k=1,k\neq i}^{n} \{a_{ik} \mid a_{ik} \neq 0\},$$

$$j_i^* = \min_{j=1,j\neq i}^{n} \{j \in \{1, \ldots, n\} \mid a_{ij} = a_i^*\},$$

We define f as following

$$f(a_{ij}) = \begin{cases} a_{ij}, & \sum_{k=1}^n a_{ik} \geq \sum_{k=1}^n a_{ki}, \\ & \forall j, \\ a_{ij}, & \sum_{k=1}^n a_{ik} < \sum_{k=1}^n a_{ki}, \\ & j \neq j_i^*, \\ a_{ij} + \omega(i) & \sum_{k=1}^n a_{ik} < \sum_{k=1}^n a_{ki}, \\ & j = j_i^*, \end{cases} \quad (1)$$

where

$$\omega(i) = \sum_{k=1}^n a_{ki} - \sum_{k=1}^n a_{ik}.$$

Note that function f is continuous on X with the subspace topology induced form $\mathbb{R}^{n\times n}$: for any $A \in X$, each nonzero entry of A can be modified within a sufficiently small neighborhood such that the image of this neighborhood is in a neighborhood of $f(A)$. Furthermore, $A^* \in X$ is an equilibrium point for the dynamical system (X, f) if and only if A is an adjacency matrix associated to a weight-balanced digraph. Such an equilibrium point exists since the digraphs assumed in this paper are all strongly semiconnected, see Theorem 2.2. Let V be a function from X to $\mathbb{R}$ defined through

$$V(A) = \sum_{i=1}^n |\sum_{j=1}^n a_{ij} - \sum_{j=1}^n a_{ji}|. \quad (2)$$

This function is continuous on X, since one can modify the columns of $A \in X$ within a sufficiently small neighborhood $U \subset X$ such that the $V(U)$ is in a neighborhood of $V(A)$. Note that if A is an equilibrium point for (X, f) then $V(A) = 0$. The following theorem contains the main result of this paper.

Theorem 3.1: Suppose that a robotic network is given by $G = (V, E, A)$, where (V, E) is strongly semiconnected. Let $X \subset \mathbb{R}^{n\times n}$ be a subspace generated by all the possible adjacency matrices associated to (V, E) and let (X, f) be the evolution defined by Equation (1). Then each evolution with initial condition in $W(A) = \{B \in X \mid 0 \leq V(B) \leq V(A)\}$ approaches a set of the form $V^{-1}(0) \cap S$, where S is the set of all weight-balanced assignments in $W(A)$. Furthermore, if $A \in \mathbb{Z}_{\geq 0}^{n\times n}$, the weight-balanced distributed algorithm converges in finite time to a weight-balanced digraph.

Proof: The set $W(A)$ is closed in X, since $W(A)$ is a level set of the continuous function V. Moreover, this set is positively invariant for (X, f) by definition of f. If an agent v_i modifies one of its out-edges say by $\epsilon \in \mathbb{R}_{>0}$ in order to balance itself, $|d_{\text{out}}(v_i) - d_{\text{out}}(v_i)|$ decreases by ϵ. Moreover, the in-degree of one of the out-neighbors of v_i, say v_j, where $j \neq i$, increases by ϵ. This increases $|d_{\text{out}}(v_j) - d_{\text{out}}(v_j)|$ by at most ϵ. Since the function V measures the sum of $|d_{\text{out}}(v_j) - d_{\text{out}}(v_j)|$ for all $j \in \{1, \ldots, n\}$, V is non-increasing along f on $W(A)$. Finally, all evolutions of (X, f) are bounded in $W(A)$, f and V are continuous, and $W(A)$ is closed. Thus the proof follows by the LaSalle invariance principle [7].

For the second part of the proof, we assign to each vertex v_i a weight

$$\omega(v_i) = d_{\text{in}}(v_i) - d_{\text{out}}(v_i). \quad (3)$$

Note that a weighted digraph $G = (V, E, A)$ is weight-balanced if and only if $\omega(v_i) = 0$, for all $v_i \in V$. We start the proof by making an observation. It is clear, from the definition of the algorithm, that at each time $t \in \mathbb{Z}_{>0}$ of the algorithm, the agents with non-positive weights are inactive. Furthermore, each agent with positive weight will change the weight on one of its out-edges, thus changing the in-degree of one of its out-neighbors. As a result, one can, equivalently, say that the agents with positive weight will

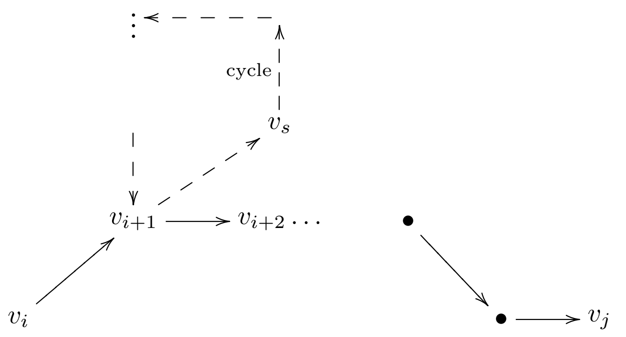


Fig. 1. Sending a message between v_i and v_j, possibly after repeating some cycles.

send their weight $\omega(v_i)$ via an edge, with minimum weight, to an out-neighbor v_j and each time that the edge e_{ij} is used for sending a message $\omega(v_i)$, the weight on this edge increases by $\omega(v_i)$. Each agent adds the received message with its previous weight to compute its new weight. Since the digraph is strongly semiconnected

$$\sum_{i=1}^{n} \omega(v_i) = 0,$$

where n is the number of agents. The weight-balanced distributed algorithm does not terminate until there is no agent with positive weight. Suppose that the agent v_i has the weight $\omega(v_i) > 0$. Without loss of generality, we consider a digraph for which there exists an agent v_j with $\omega(v_j) = -\omega(v_i)$ and $\omega(v_k) = 0$, for $k \in \{1, \ldots, n\}$ and $k \neq i, j$. Note that the algorithm does not terminate till the message $\omega(v_i)$ reaches v_j. Suppose that this message should be carried via a simple path shown in Figure 1 to get to v_j. We must show that this is possible in finite time. Suppose that, at time $t \in \mathbb{Z}_{>0}$, the agent v_i sends the message $\omega(v_i)$ to v_{i+1}. It is enough to show that the agent v_{i+1} can send the message to v_{i+2} in finite time. Without loss of generality, assume that the agent v_{i+1} has two out-neighbors v_{i+2} and $v_s \neq v_i$, where v_s is not in the same cycle with v_j. In the next iteration, v_{i+1} chooses the out-neighbor with the edge with minimum weight for transmitting $\omega(v_i)$. Suppose that v_{i+1} chooses v_s, i.e. $a(v_{i+1}, v_{i+2})\{t\} > a(v_{i+1}, v_s)\{t\}$, where $t \in \mathbb{Z}_{>0}$ indicates the time. Since we assumed that the weights on all the agents except v_i and v_j are zero, the message $\omega(v_i)$ will come back to v_{i+1} after a finite time, possibly after going through some cycles. After that, v_{i+1} will choose v_s again if $a(v_{i+1}, v_{i+2})\{t\} > a(v_{i+1}, v_s)\{t\} + \omega(v_i)$ and will choose v_{i+2} otherwise. Thus, at most, after a finite time $T \in \mathbb{Z}_{>0}$, we have $a(v_{i+1}, v_{i+2})\{T\} < a(v_{i+1}, v_s)\{T\}$ and the agent v_{i+1} will choose v_{i+2}; thus the claim follows. ■

Remark 3.2 (Convergence rate): Obtaining the convergence rate of the distributed weight-balanced algorithm is a hard combinatorial problem. We have rough upper bounds for this convergence rate and we postpone the details to future work. However, we characterize the time complexity of a modified version of this algorithm in Section V. •

Example 3.3: (Execution of weight-balanced distributed algorithm): Consider the digraph G shown in Figure 2. This algorithm converges to a weight-balanced digraph in 6 iterations as demonstrated in Figure 3. In Example 4.8, we show that, for this example, the distributed approach is more efficient than the centralized one. •

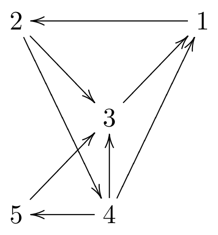


Fig. 2. The digraph of Example 3.3.

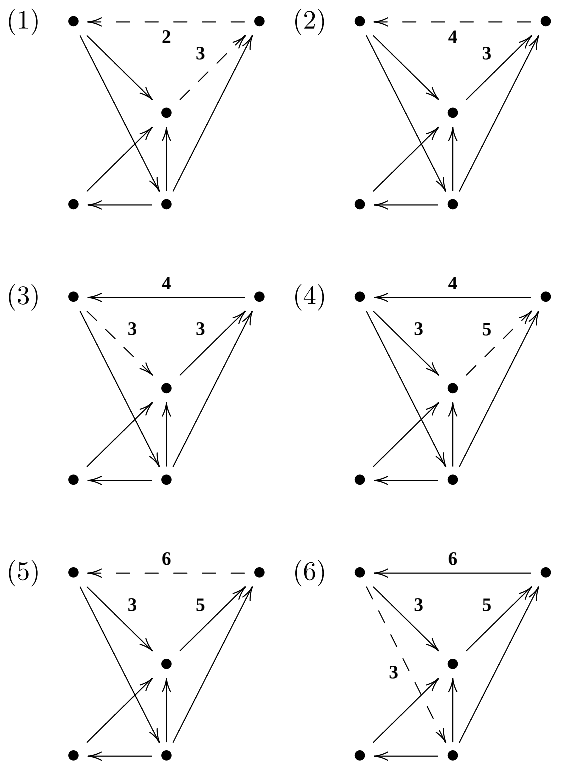


Fig. 3. Iterations of the weight-balanced distributed algorithm for the digraph of Example 3.3. In each iteration, the edges which are used for sending messages are shown with dash lines. Note that the Lyapunov function reads $V(0) = 6$, $V(1) = V(2) = \ldots = V(5) = 4$, and $V(6) = 0$.

IV. Weight-balanced centralized algorithm

In this section we propose a centralized algorithm for constructing a weight-balanced digraph which is essentially similar to the centralized algorithm proposed in [5]. The main advantageous of this algorithm is that the algorithm only uses the so-called *fundamental cycle matrix* of a digraph. This gives a more systematic approach for constructing a weight-balanced digraph. Moreover, we characterize the time complexity of this algorithm. In this section we assume that all the digraphs are strongly semiconnected.

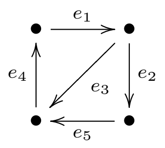


Fig. 4. Composition of semi-cycles.

A. Fundamental cycle matrix

Let $G=(V,E)$ be a digraph with n vertices and m edges. We denote the ordered set $E^- \subset V\times V$ to be the set of all ordered pairs obtained by changing the orders of E, i.e. if $(u,v)\in E$ then $(v,u)\in E^-$, where $u,v\in V$. We call the digraph $\bar{G}=(V,E\oplus E^-)$ *the mirror of* G and a cycle of $\bar{G}$ a *semi-cycle* of G. Note that a semi-cycle is a directed path. We have the following definition.

Definition 4.1: Let $G=(V,E)$ be a digraph. The *cycle matrix* $\mathrm{C}(G)$ is an $l\times m$ matrix, where l and m are, respectively, the number of cycles and edges in the digraph and for $i\in\{1,\dots,l\}$ and $j\in\{1,\dots,m\}$,

$$\mathrm{C}(G)_{ij}=\begin{cases}1, & \text{if the } i\text{th semi-cycle includes}\\ & \text{edge } j \text{ with the same direction,}\\ -1, & \text{if the } i\text{th semi-cycle includes}\\ & \text{edge } j \text{ with the opposite direction,}\\ 0, & \text{otherwise.}\end{cases}$$

Note that a row α_i of $\mathrm{C}(G)$, where $i\in\{1,\dots,n\}$, refers to a cycle of the digraph G if and only if $\alpha_{ij}\geq 0$ for all $j\in\{1,\dots,m\}$ or $\alpha_{ij}\leq 0$ for all $j\in\{1,\dots,m\}$. For a digraph $G=(V,E)$, we say that an edge $(u,v)\in E$ is *incident away from* u and *incident toward* v. This motivates the following definition.

Definition 4.2: Let $G=(V,E)$ be a digraph. The *incident matrix* $\mathrm{I}(G)$ is an $n\times m$ matrix, where n and m are, respectively, the number of vertices and edges of G and for $i\in\{1,\dots,n\}$ and $j\in\{1,\dots,m\}$,

$$\mathrm{I}(G)_{ij}=\begin{cases}1, & \text{if the } j\text{th edge is incident away from } i,\\ -1, & \text{if the } j\text{th edge is incident toward } i,\\ 0, & \text{otherwise.}\end{cases}$$

The following theorem reveals the relationship between the cycle matrix and the incident matrix, see [8] for details.

Theorem 4.3: For any digraph $G=(V,E)$ we have $\mathrm{C}(G)\mathrm{I}(G)^{\mathrm{T}}=0$.

Consider the digraph $G=(V,E)$ in Figure 4, where the edges are labeled with the set $\{1,\dots,5\}$. This digraph consists of three semi-cycles given by

$$\begin{aligned}\alpha &= \begin{pmatrix}1 & 0 & 1 & 1 & 0\end{pmatrix},\\ \beta &= \begin{pmatrix}0 & 1 & -1 & 0 & 1\end{pmatrix},\\ \gamma &= \begin{pmatrix}1 & 1 & 0 & 1 & 1\end{pmatrix}.\end{aligned}$$

If we *sum* the first two semi-cycles, the two edges with different orientations cancel out each other, and thus the third semi-cycle will be obtained. We emphasize that two semi-cycles can be summed only if the common edges have different orientations. This suggest investigating the minimum number of rows of the cycle matrix which can generate all the rows. If we reduce the cycle matrix by selecting the independent rows, the resulting matrix is called the *fundamental cycle matrix* and we denote it by $\mathrm{F}(G)$.

In following, we give a formal definition for the fundamental cycle matrix. Furthermore, we study an algorithm, adapted from [8], for computing this matrix. Recall the definition of a breadth-first spanning tree $\mathrm{BFS}(G,v)$ of a digraph G rooted at v, see [6]. We start by the following definition.

Definition 4.4: Let $G=(V,E)$ be a digraph and let $\mathrm{BFS}(G,v)$ be a breadth-first spanning tree. An edge $(u,w)\in E$, $u,w\in V$, is called a *chord* if it is not an edge of $\mathrm{BFS}(G,v)$. We denote by $E_{\mathrm{c}}^{\mathrm{BFS}}(G,v)\subset E$ the set of all chords of G with respect to $\mathrm{BFS}(G,v)$.

Note that adding a chord to a breadth-first spanning tree defines a row of $\mathrm{C}(G)$, up to a sign, which consists of the chord and some edges of the breadth-first spanning tree. We call such a semi-cycle *fundamental cycle*. It is easy to verify that a digraph with n vertices and m edges has $m-n+1$ chords; thus there are $m-n+1$ fundamental cycles. We have the following definition.

Definition 4.5: Let $G=(V,E)$ be a digraph. A *fundamental cycle matrix* $\mathrm{F}(G)$ is an $(m-n+1)\times m$ matrix, where n and m are, respectively, the number of vertices and edges of G and for $i\in\{1,\dots,m-n+1\}$ and $j\in\{1,\dots,m\}$,

$$\mathrm{F}(G)_{ij}=\begin{cases}1, & \text{if the } i\text{th fundamental cycle includes}\\ & \text{edge } j \text{ with the same orientation,}\\ -1, & \text{if the } i\text{th fundamental cycle includes}\\ & \text{edge } j \text{ with the opposite orientation,}\\ 0, & \text{otherwise.}\end{cases}$$

Fundamental cycle centralized algorithm Given a digraph $G=(V,E)$,

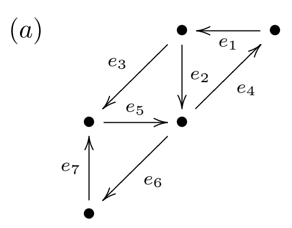
1) compute a breadth-first spanning tree $\mathrm{BFS}(G,v)$, rooted at an arbitrary vertex $v\in V$;
2) compute the set of all chords $E_{\mathrm{c}}^{\mathrm{BFS}}(G,v)$ with respect to the $\mathrm{BFS}(G,v)$;
3) compute a fundamental cycle matrix $\mathrm{F}(G)$ by adding each chord to the breadth-first spanning $\mathrm{BFS}(G,v)$ and recording the resulting semi-cycle of G.

B. Weight-balanced centralized algorithm via fundamental cycle matrix

Let $G=(V,E)$ be a strongly semiconnected digraph. We have the required tools to propose a centralized algorithm for computing an adjacency matrix A such that $G=(V,E,A)$ is weight-balanced. This algorithm is basically similar to the one of [5]. However, the current algorithm suggests a concrete procedure for computing the cycles via a fundamental cycle matrix.

Weight-balanced centralized algorithm. Given a digraph $G=(V,E)$,

1) compute a fundamental cycle matrix $\mathrm{F}(G)$ using the fundamental cycle centralized algorithm;
2) use this fundamental cycle matrix to compute the cycle matrix $\mathrm{C}(G)$;



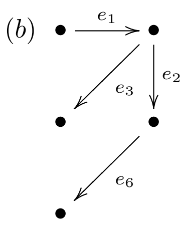


Fig. 5. Digraph of Example 4.7, (a), and a breadth-first spanning tree that contains the edge e_1, (b).

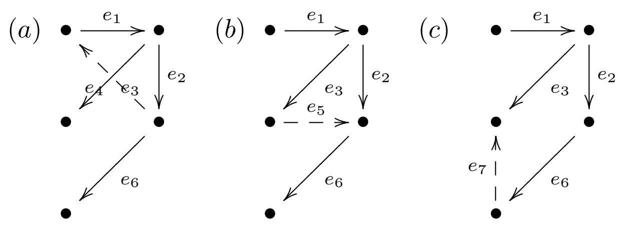


Fig. 6. Adding chords to the breadth-first spanning tree in order to find a fundamental cycle matrix.

3) identify the rows α_i of $\mathrm{C}(G)$ for which $\alpha_{ij} \geq 0$ for all $j \in \{1, \dots, m\}$ or $\alpha_{ij} \leq 0$ for all $j \in \{1, \dots, m\}$ and denote the set of all such α_is by $\mathrm{S}(G)$;
4) suppose that $\mathrm{S}(G)$ has $p \leq l$ elements, where l is the number of rows of $\mathrm{C}(G)$. Then $W = \sum_{i=1}^{p} |\alpha_i|$, where $\alpha_i \in \mathrm{S}(G)$, gives the weights on each edge of G that makes G weight-balanced;
5) use W to compute an adjacency matrix A such that $G = (V, E, A)$ is weight-balanced.

We now compute the time complexity of this algorithm.

Proposition 4.6: Let G be a strongly semiconnected digraph with n vertices. The weight-balanced centralized algorithm has the time complexity $O(2^{n^2})$.

Proof: Note that the time complexity of constructing a breadth-first spanning tree for G is $O(n)$. If the digraph G has m edges, then there are $m - n + 1$ fundamental cycles. In order to construct the cycles of the digraph, one needs to find all the linear combination of such fundamental cycles and detect $\mathrm{S}(G)$. We compute the time complexity of such computation, which we denote by $\mathcal{T}_{\text{w-cen}}(G)$, explicitly. Suppose that $\{\alpha_1, \alpha_2, \dots, \alpha_{m-n+1}\}$ are the fundamental cycles of G. Then the number of cycles that are constructed with two of these fundamental cycles, not necessarily independent, is at most $\binom{m-n+1}{2}$ (note that some the fundamental cycles can not be added). Similarly, the number of cycles, not necessarily independent, which are constructed by i fundamental cycles is $\binom{m-n+1}{i}$. Thus we have

$$\mathcal{T}_{\text{w-cen}}(G) = \mathcal{T}_{\text{BFS}}(G) + \sum_{i=2}^{m-n+1} \binom{m-n+1}{i},$$

where we denoted by $\mathcal{T}_{\text{BFS}}(G)$ the time complexity of computing a first-breadth spanning tree for G. Note that we have

$$\sum_{i=2}^{m-n+1} \binom{m-n+1}{i} = 2^{m-n+1} - (m-n+1) + 1.$$

Since we do not allow repeated edges between two vertices, $m \leq \frac{n(n+1)}{2}$ and the claim follows. ■

Proposition 4.6 shows that the centralized algorithm can be computationally very time consuming.

Example 4.7: (Execution of weight-balanced centralized algorithm): Consider the digraph shown in Figure 5–(a), where we labeled the edges. A breadth-first spanning tree that contains edge e_1 is shown in Figure 5–(b). Note that edges e_4, e_5 and e_7 are the chords for this example, see Figure 6. Thus the fundamental cycle matrix is an element of $\mathbb{R}^{3\times 7}$. We compute the following fundamental cycles

$$\begin{aligned} \alpha &= \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}, \\ \beta &= \begin{pmatrix} 0 & -1 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}, \\ \gamma &= \begin{pmatrix} 0 & 1 & -1 & 0 & 0 & 1 & 1 \end{pmatrix}. \end{aligned}$$

Thus the fundamental cycle matrix is

$$\mathrm{F}(G) = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

One can compute, using the fundamental cycles, the following rows of $\mathrm{C}(G)$ which belong to $\mathrm{S}(G)$

$$\begin{aligned} \alpha &= \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}, \\ \alpha + \beta &= \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}, \\ \beta + \gamma &= \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}. \end{aligned}$$

Thus the weight on each edge which makes the digraph G weight-balanced is given by

$$W = \begin{pmatrix} 2 & 1 & 1 & 2 & 2 & 1 & 1 \end{pmatrix}.$$

One can use this weights to compute the adjacency matrix that makes G weight-balanced. •

Example 4.8 (Example 3.3 Cont.): Consider the digraph of Example 3.3. This digraph has three chords and thus, using the centralized algorithm and after computing a breadth-first spanning tree, one needs to check nine different combinations of the fundamental cycles. Thus, for this example, the distributed algorithm converges faster than the centralized one. •

V. Weight-balanced modified distributed algorithm

We showed in Proposition 4.6 that the weight-balanced centralized algorithm is computationally complex. In this section we modify the weight-balanced distributed algorithm to an algorithm, distributed over the mirror of the digraph, which converges quickly to a weight-balanced digraph. The importance of this algorithm is that it can be utilized to construct a weight-balanced digraph without constructing the cycles. Furthermore, as we will see in Proposition 5.2, this algorithm is much faster than the weight-balanced centralized algorithm.

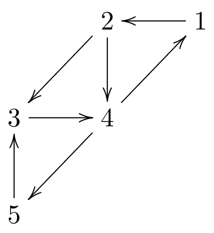


Fig. 7. The digraph of Example 4.7. In this figure the agents are labeled.

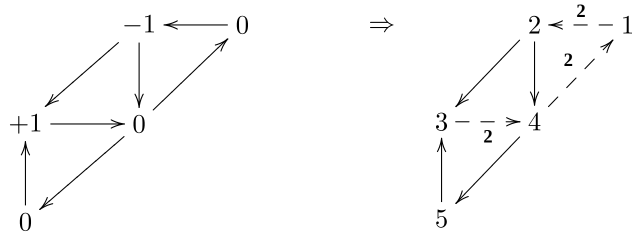


Fig. 8. The weight-balanced modified algorithm execution for Example 4.7. The dash lines show the edges which has been used to send messages.

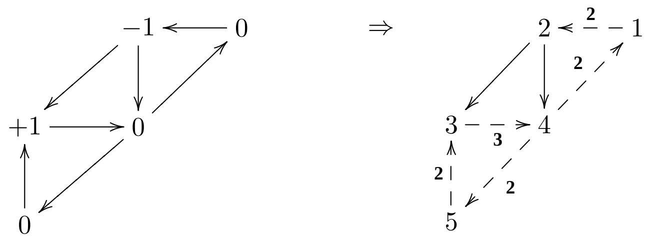


Fig. 9. The second possible weight-balanced modified algorithm execution for Example 4.7. The dash lines show the edges which has been used to send messages.

Weight-balanced modified distributed algorithm. Let $G = (V, E, A)$ be a strongly semiconnected digraph, where $V = \{v_1, \dots, v_n\}$.

1) Each agent v_i can observe the weights of its out-neighbors (the algorithm is assumed to be distributed over the mirror of the digraph G).
2) At each iteration, agents are able to receive a message from one of their in-neighbors.
3) Any agent $v_i \in V$ with positive weight passes its weight to only one the out-neighbor with minimum weight via an edge.
4) If agent v_i uses the edge $(v_i, v_j) \in E$ to send its weight to an out-neighbor, $a_{ij} = a_{ij} + \omega(v_i)$.
5) *Multiple-messages rule*: If an agent receives more than one message from its in-neighbors, it adds the received messages to its weight.
6) *Fair-decision rule*: If the agent v_i has more than one out-neighbors with the exact same weight, it randomly chooses one. However, the next time v_i needs to choose between its out-neighbors with the same weight, it will choose a new out-neighbor.

In Theorem 5.1 we show that this algorithm converges in finite time to a weight-balanced digraph. We first execute this algorithm for Example 4.7. Recall that in this example, after computing the breadth-first spanning tree, one needs to examine nine different combinations of the fundamental cycles in order to find a weight-balanced digraph. In Figure 7, we labeled each agent with the set $\{1, \dots, 5\}$, where we assumed that the initial weight on each edge is one. In Figure 8, we show that the weight-balanced modified algorithm converges in three iterations to a weight-balanced digraph.

Note that agent 4 has two out-neighbors with the same weights, namely 1 and 5. In Figure 9, we show the execution of the algorithm in case agent 4 passes the weight to agent 5; this shows the importance of the fair-decision rule in the body of the weight-balanced modified algorithm. If after the $4th$ iteration of the algorithm the agent 4 would keep passing its weight to agent 5, then the algorithm would never converge to a weight-balanced digraph.

Theorem 5.1: Suppose $G = (V, E)$ is a strongly semiconnected digraph. Then the weight-balanced modified algorithm converges to a weight-balanced digraph in finite time.

Proof: Assume that an agent v_i, where $v_i \in V$, has a positive weight in the kth iteration, where $k \in \mathbb{Z}_{>0}$. Thus in the $(k+1)$th iteration, agent v_i passes its weight to its neighbors with the least weight. Furthermore, agents with negative weights are not active, in the sense that they do not pass their weight to any out-neighbor; thus these agents act like stationary sinks. Note that for a strongly semiconnected digraph $\sum_{j=1}^{n} \omega(v_j) = 0$, where $\omega(v_i)$ is defined in Equation (3). Since the digraph is strongly semiconnected, the agents with positive weights pass their weights until the weight reaches an agent with negative weight. The fair-decision rule prevents any agent $v_i \in V$ from passing $\omega(v_i)$ in a cycle. Since the digraph has finite number of edges, after repeating the algorithm for a finite number of iterations and at most by passing the weights in some cycles, see Proposition 5.2, the positive weights will reach the agents with negative weights and thus the digraph will become weight-balanced. ■

We investigate the rate of convergence of the weight-balanced modified distributed algorithm introduced above. We make the following observation. Suppose that $G = (V, E)$ is a digraph and an agent v_i, where $v_i \in V$, has a weight $\omega(v_i) = r > 0$. Then, by the weight-balanced modified-distributed algorithm, the agent passes the weight to an out-neighbor and the process continues until this weight reaches an agent with negative weight. Without loss of generality, assume that this weight gets passed through a path to reach an agent v_j with weight $\omega(v_j) = -r$, see Figure 1. After the next iteration, the next agent might pass the weight to the next agent in the path directly, or after sending it by mistake to a cycle. In the later case, the agent v_{i+1} will receive the weight r again and by the fair-decision rule, this time it will pick a different out-neighbor. Denote the set of all cycles of G by $\mathrm{cyc}(G)$ and let

$$C_{\max}(G) = \max\{\mathrm{diam}(G_c) \mid G_c \in \mathrm{cyc}(G)\},$$
$$d_{\text{out}}^{\max}(G) = \max\{d_{\text{out}}(v_k) \mid v_k \in G\}.$$

We have the following proposition.

Proposition 5.2: Let $G = (V, E)$ be a digraph. The time complexity of the weight-balanced-game algorithm $\mathcal{T}_{\text{w-mod-dis}}(G) \leq d_{\text{out}}^{\text{max}}(G)\text{diam}(G)C_{\max}(G)$.

Proof: The maximum distance between any two agents is $\text{diam}(G)$. Suppose a message needs to be sent through a path of length $\text{diam}(G)$. Suppose that all the agents in this path have the maximum number of out-neighbors and furthermore, suppose that all the agents try all their other out-neighbors (via cycles) before finding the correct out-neighbor. Then, by the above observation, they need $d_{\text{out}}^{\text{max}}\text{diam}(G)C_{\max}$ iterations to execute the task. ■

The following is an immediate corollary of this theorem.

Corollary 5.3: The time complexity of the weight-balanced modified algorithm is $O(n^3)$.

VI. Conclusions

In this paper, we have proposed three different algorithms for constructing a weight-balanced digraph from a strongly semiconnected digraph:

1) the *weight-balanced distributed algorithm*, running synchronously, is developed for computing a weight-balanced communication network for a group of agents. We have established the finite-time convergence of this algorithm via the discrete-time LaSalle invariance principle. Its convergence is illustrated in different examples. The algorithm is of importance as weight-balanced graphs play an important role in variety of cooperative algorithms.
2) the *weight-balanced centralized algorithm* essentially systematizes the centralized approach in [5] for constructing a weight-balanced digraph. We compute the time complexity of this algorithm and show that it does not scale well with the size of the network.
3) the *weight-balanced modified distributed algorithm*, distributed over the mirror of the original digraph, is developed using the weight-balanced distributed algorithm. We have also established the finite-time convergence of this algorithm and characterized its time complexity, which is substantially better than that of the centralized algorithm.

Future work will include determining the time complexity of the weight-balanced distributed algorithm and investigating the connection of the proposed algorithms with the stability of stochastic systems.

Acknowledgments

This research was partially supported by NSF Award CCF-0917166.

References

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

[2] R. Olfati-Saber and R. M. Murray, “Consensus problems in networks of agents with switching topology and time-delays,” *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

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

[4] D. Lee and M. W. Spong, “Stable folcking of multiple interial agents on balanced digraphs,” *IEEE Transaction on Automatic Control*, vol. 52, no. 8, pp. 1469–1475, 1984.

[5] 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, 1970.

[6] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks*. Applied Mathematics Series, Princeton University Press, 2009. Electronically available at http://coordinationbook.info.

[7] J. P. LaSalle, “Some extensions of Liapunov’s second method,” *IRE Trans. Circuit Theory*, vol. CT-7, pp. 520–527, 1960.

[8] D. F. Robinson and L. R. Foulds, *Digraphs: Theory and Techniques*. Gordon and Breach Science Publishers, 1980.