Distributed Balancing under Interval Flow Constraints in Directed Communication Topologies

Christoforos N. Hadjicostis and Alejandro D. Domínguez-García

Abstract— **In this paper, we propose a distributed algorithm that relies on a strongly connected (but possibly directed) communication topology to achieve admissible and balanced flows in a given network. More specifically, we consider a flow network that is described by a digraph (physical topology), each edge of which can admit a flow within a certain interval. The paper proposes and analyzes a distributed iterative algorithm for computing *admissible* and *balanced* flows, i.e., flows that are within the given interval at each edge and balance the total in-flow and the total out-flow at each node. Unlike previous work that required a communication topology with bidirectional exchanges between pairs of nodes that are physically connected (i.e., nodes that share an edge in the physical topology), the distributed algorithm we propose only requires a communication topology that matches the physical topology (which is, in general, directed). The proposed algorithm allows the nodes to asymptotically (with geometric rate) compute a set of admissible and balanced flows, as long as such solution exists.**

I. Introduction

We consider a flow network (which we refer to as the physical digraph or topology) comprised of multiple nodes that are interconnected via some directed links through which a certain commodity can flow. We assume that the flow on each link is constrained to lie within an interval, with nonnegative end points that correspond to link lower and upper capacity limits. The objective is to find an admissible and balanced flow assignment, i.e., find flows on all the links that are within the corresponding capacity limits, such that the sum of in-flows is equal to the sum of out-flows. In this paper, we propose an algorithm that allows the nodes to compute a solution to this feasibility problem, in a distributed manner, utilizing a communication topology that matches the physical topology.

The proposed algorithm combines some of the features of the flow balancing algorithm in [1] (which achieves flow balancing, assuming *no* constraints on the flows and a communication topology that matches the possibly *directed* physical topology), and the algorithm in [2] (which achieves flow balancing, *with* constraints on the flows, but assuming that the communication topology matches the *undirected* graph that corresponds to the physical topology). Thus, the main difference with the work in [1] is that we allow constraints (lower and upper limits on the flows), whereas the main difference from [2] is that we do not require bidirectional communication between pairs of nodes that are physically connected. It will become evident in our development that combining the features of the two algorithms is non-trivial and comes at the cost of requiring the nodes to maintain additional variables and update them in a way that respects the communication constraints.

The problem of interest in this paper is a particular case of the standard network flow problem (see, e.g., [3]), where we are given a flow network with a cost associated to the flow on each link, and the objective is to minimize the total cost subject to the same constraints in the flow assignment problem described above. In this regard, it is common to assume that the individual costs are described by convex functions on the flow, which makes the optimization problem convex. Then, its solution can be obtained via the Lagrange dual, the formulation of which is well suited for algorithms that can be executed, in a distributed fashion, over a network that conforms to the same topology as that of the multi-node system (see, e.g., [4]); however, recovering the optimal primal solution from the dual one might not be straightforward [5]. The works in [1], [2], as well as this paper, propose distributed iterative algorithms that act directly on the *primal variables*, i.e., the flows, and are able to asymptotically obtain balanced flows.

The problem we deal with in this paper can also be viewed as the problem of weight balancing a given digraph. A weighted digraph is a directed graph in which each edge is associated with a real or integer value, called the edge weight. A weighted digraph is *weight-balanced* or *balanced* if, for each of its nodes, the sum of the weights of the edges outgoing from the node is equal to the sum of the weights of the edges incoming to the node. Flow/weight-balanced digraphs find numerous applications in distributed adaptive control or synchronization in complex networks, such as network adaptation strategies based on the use of continuous second order models [6], and distributed adaptive strategies to tune the coupling weights of a network based on local information of node dynamics [7]. Weight balancing can be associated with the matrix balancing problem in network optimization, with numerous applications, in economics, statistics and demography [8]. Weight balance is also closely related to weights that form a doubly stochastic matrix [9], which find applications in multicomponent systems (such as sensor networks), where one is interested in distributively averaging measurements at each component; see, for example, the discussions in [1] and [10].

C. N. Hadjicostis is with the Department of Electrical and Computer Engineering at the University of Cyprus, Nicosia, Cyprus, and also with the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign. E-mail: chadjic@ucy.ac.cy.

Alejandro D. Domínguez-García is with the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA. E-mail: aledan@illinois.edu.

Recently, quite a few works have appeared dealing with the problem of designing distributed algorithms for balancing a strongly connected digraph using real and integer weights, for the case when there are no constraints on the edge weights in terms of the nonnegative values they admit [1], [2], [10], [11], [9], [12]. Apart from [2] which considers interval constraints on the weights/flows but requires bidirectional communication (as described earlier), all of these works develop distributed algorithms that assume a communication topology that matches the physical topology. Thus, the algorithm proposed here can be viewed as a way for distributively achieving weight-balancing when there are lower and upper limit constraints on the edge weights, *and* the communication topology matches the physical topology.

II. Preliminaries

A. Graph-Theoretic Notions

A digraph (directed graph) of order n ($n \geq 2$), is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} - \{(v_j, v_j) \mid v_j \in \mathcal{V}\}$ is the set of edges. A directed edge from node v_i to node v_j is denoted by $(v_j, v_i) \in \mathcal{E}$. A digraph is called *strongly connected* if for each pair of vertices, $v_j, v_i \in \mathcal{V}$, $v_j \neq v_i$, there exists a directed *path* from v_i to v_j i.e., we can find a sequence of vertices $v_i \equiv v_{l_0}, v_{l_1}, \ldots, v_{l_t} \equiv v_j$ such that $(v_{l_{\tau+1}}, v_{l_\tau}) \in \mathcal{E}$ for $\tau = 0, 1, \ldots, t-1$. All nodes from which node v_j can be reached via a directed edge are said to be in-neighbors of node v_j and belong to the set $\mathcal{N}_j^- = \{v_i \in \mathcal{V} \mid (v_j, v_i) \in \mathcal{E}\}$. The cardinality of $\mathcal{N}_j^-$ is called the *in-degree* of v_j and is denoted by $\mathcal{D}_j^- = |\mathcal{N}_j^-|$. The nodes that can be reached from node v_j via a directed edge comprise its out-neighbors and are denoted by $\mathcal{N}_j^+ = \{v_l \in \mathcal{V} \mid (v_l, v_j) \in \mathcal{E}\}$. The cardinality of $\mathcal{N}_j^+$ is called the *out-degree* of v_j and is denoted by $\mathcal{D}_j^+ = |\mathcal{N}_j^+|$.

A distributed system the components of which can exchange a certain commodity via (possibly directed) links, can conveniently be captured by a digraph $\mathcal{G}_p = (\mathcal{V}, \mathcal{E}_p)$, which we will refer to as the *flow topology* or *physical digraph/topology*. Given $\mathcal{G}_p$, we can associate nonnegative flows (sometimes, also viewed as weights), $f_{ji} \in \mathbb{R}$ on each edge $(v_j, v_i) \in \mathcal{E}_p$. In this paper, the flow f_{ji} will be restricted to lie in a real interval $[l_{ji}, u_{ji}]$, where $0 \leq l_{ji} \leq f_{ji} \leq u_{ji}$. We will also use matrix notation to denote (respectively) the flow, lower limit, and upper limit matrices by the $n \times n$ matrices $F = [f_{ji}]$, $L = [l_{ji}]$, and $U = [u_{ji}]$, where $L(j,i) = l_{ji}$, $F(j,i) = f_{ji}$, and $U(j,i) = u_{ji}$ (and $f_{ji} = l_{ji} = u_{ji} = 0$ when $(v_j, v_i) \notin \mathcal{E}_p$). Following the notation introduced previously, we will denote the *physical* in-neighbors of node v_j by $\mathcal{N}_j^- = \{v_i \in \mathcal{V} \mid (v_j, v_i) \in \mathcal{E}_p\}$, and the *physical* in-degree of v_j by $\mathcal{D}_j^-$; similarly, we will denote the *physical* out-neighbors of node v_j by $\mathcal{N}_j^+ = \{v_l \in \mathcal{V} \mid (v_l, v_j) \in \mathcal{E}_p\}$, and the *physical* out-degree by $\mathcal{D}_j^+$.

For the purposes of developing our distributed algorithm, we will rely on a communication topology that is captured, in general, by a digraph $\mathcal{G}_c = (\mathcal{V}, \mathcal{E}_c)$ that describes how the nodes in the physical topology can communicate among themselves. Thus, at any particular instant during the execution of our algorithm, node v_j can receive information from its *communication* in-neighbors, denoted by $\mathcal{N}_{c,j}^- = \{v_i \in \mathcal{V} \mid (v_j, v_i) \in \mathcal{E}_c\}$, and can send information to its *communication* out-neighbors, denoted by $\mathcal{N}_{c,j}^+ = \{v_l \in \mathcal{V} \mid (v_l, v_j) \in \mathcal{E}_c\}$. In this paper, we are interested in the case when the communication topology matches exactly the physical topology (i.e., $\mathcal{E}_c = \mathcal{E}_p$). However, in our analysis we will also be interested in the case where $\mathcal{E}_c$ contains all the links in $\mathcal{E}_p$, as well as some of their reverse directions. Notice that earlier work in [2] considered the case where *all* links are bidirectional, i.e., $\mathcal{E}_c = \{(v_j, v_i), (v_i, v_j) \mid (v_j, v_i) \in \mathcal{E}_p\}$. This implies that all nodes that are neighbors in the physical topology can communicate in a bidirectional manner.

B. Problem Formulation

Given a physical digraph $\mathcal{G}_p = (\mathcal{V}, \mathcal{E}_p)$ of order n along with a flow assignment $F = [f_{ji}]$, we define the following.

Definition 1: The total *in-flow* of node v_j is defined as $f_j^- = \sum_{v_i \in \mathcal{N}_j^-} f_{ji}$, whereas the total *out-flow* of node v_j is defined as $f_j^+ = \sum_{v_l \in \mathcal{N}_j^+} f_{lj}$. The *flow balance* of node v_j is defined as $b_j = f_j^- - f_j^+$. The *total imbalance* (or *absolute imbalance*) of digraph $\mathcal{G}_p$ is defined as $\varepsilon = \sum_{j=1}^n |b_j|$.

Definition 2: A digraph $\mathcal{G}_p = (\mathcal{V}, \mathcal{E}_p)$ of order n, along with a flow assignment $F = [f_{ji}]$, is called *flow-balanced* or *weight-balanced* if its total imbalance (or absolute imbalance) is 0, i.e., $\varepsilon = \sum_{j=1}^n |b_j| = 0$.

Flow Assignment Problem: We are given a physical digraph $\mathcal{G}_p = (\mathcal{V}, \mathcal{E}_p)$, and a communication digraph $\mathcal{G}_c = (\mathcal{V}, \mathcal{E}_c)$, as well as lower and upper bounds l_{ji} and u_{ji} ($0 \leq l_{ji} \leq u_{ji}$) on each edge $(v_j, v_i) \in \mathcal{E}_p$. We want to develop a distributed iterative algorithm that respects the communication restrictions imposed by the communication digraph $\mathcal{G}_c$, and allows the nodes to iteratively adjust the flows on their outgoing edges, so that they asymptotically obtain a set of flows $\{f_{ji} \mid (v_j, v_i) \in \mathcal{E}_p\}$ that satisfy:

1) $0 \leq l_{ji} \leq f_{ji} \leq u_{ji}$ for each edge $(v_j, v_i) \in \mathcal{E}_p$;
2) $f_j^+ = f_j^-$ for every $v_j \in \mathcal{V}$.

A variety of centralized algorithms for obtaining such flows exist [13].

Theorem 1: (Circulation Theorem [3].) Consider a strongly connected digraph $\mathcal{G}_p = (\mathcal{V}, \mathcal{E}_p)$, with lower and upper bounds l_{ji} and u_{ji} ($0 \leq l_{ji} \leq u_{ji}$) on each edge $(v_j, v_i) \in \mathcal{E}_p$. The necessary and sufficient condition for the existence of a set of flows $\{f_{ji} \mid (v_j, v_i) \in \mathcal{E}_p\}$ that satisfy
1. *Interval constraints:* $0 \leq l_{ji} \leq f_{ji} \leq u_{ji}$, $\forall (v_j, v_i) \in \mathcal{E}_p$,
2. *Balance constraints:* $f_j^+ = f_j^-$, $\forall v_j \in \mathcal{V}$,
is the following: for each $\mathcal{S}$, $\mathcal{S} \subset \mathcal{V}$, we have

$$\sum_{(v_j, v_i) \in \mathcal{E}_{\mathcal{S}}^-} l_{ji} \leq \sum_{(v_l, v_j) \in \mathcal{E}_{\mathcal{S}}^+} u_{lj} \ , \tag{1}$$

where

$$\mathcal{E}_{\mathcal{S}}^- = \{(v_j, v_i) \in \mathcal{E}_p \mid v_j \in \mathcal{S},\ v_i \in \mathcal{V} - \mathcal{S}\} \ , \tag{2}$$

$$\mathcal{E}_{\mathcal{S}}^+ = \{(v_l, v_j) \in \mathcal{E}_p \mid v_j \in \mathcal{S},\ v_l \in \mathcal{V} - \mathcal{S}\} \ . \tag{3}$$

III. An Existing Distributed Flow Assignment Algorithm and its Limitations

The distributed algorithm developed in [2] is iterative and operates by having, at each iteration, nodes with positive imbalance attempt to change the flows on both their incoming and outgoing edges, so as to get closer to being balanced. It can be summarized as follows.

Initialization. At initialization, each node is aware of the feasible flow interval on each of its incoming and outgoing edges, i.e., node v_j is aware of l_{ji}, u_{ji} for each $v_i \in \mathcal{N}_j^-$ and l_{lj}, u_{lj} for each $v_l \in \mathcal{N}_j^+$. Furthermore, the flows are initialized at the middle of the feasible interval, i.e., $f_{ji}[0] = (l_{ji} + u_{ji})/2$. [This initialization is not critical and could be any value in the feasible flow interval $[l_{ji}, u_{ji}]$.]

Iteration. At each iteration $k \geq 0$, node v_j is aware of the flows on its incoming edges $\{f_{ji}[k] \mid v_i \in \mathcal{N}_j^-\}$ and outgoing edges $\{f_{lj}[k] \mid v_l \in \mathcal{N}_j^+\}$, and updates them using the following three steps:

[Step 1.] If node v_j has balance $b_j[k]$ that is negative or zero ($b_j[k] \leq 0$), then node v_j does not attempt to make any flow changes; however, if $b_j[k] > 0$, node v_j attempts to change the flows at its incoming edges $\{f_{ji}[k+1] \mid v_i \in \mathcal{N}_j^-\}$, and outgoing edges $\{f_{lj}[k+1] \mid v_l \in \mathcal{N}_j^+\}$ in a way that drives its balance $b_j[k+1]$ to zero (at least if no other changes are inflicted on the flows). Since node v_j is associated with $\mathcal{D}_j = \mathcal{D}_j^- + \mathcal{D}_j^+$ edges, it attempts to change each incoming flow by $-\frac{b_j[k]}{\mathcal{D}_j}$ and each outgoing flow by $+\frac{b_j[k]}{\mathcal{D}_j}$. Thus, from the perspective of node v_j, the desirable flows at the next iteration are given by

$$f_{ji}^{(j)}[k+1] = f_{ji}[k] - \frac{\tilde{b}_j[k]}{\mathcal{D}_j}\ , \quad v_i \in \mathcal{N}_j^-\ , \tag{4}$$

$$f_{lj}^{(j)}[k+1] = f_{lj}[k] + \frac{\tilde{b}_j[k]}{\mathcal{D}_j}\ , \quad v_l \in \mathcal{N}_j^+\ , \tag{5}$$

where $\tilde{b}_j[k] = b_j[k]$ if $b_j[k] > 0$ and zero otherwise.

[Step 2.] The interim value of the flow on each edge $(v_j, v_i) \in \mathcal{E}$ is taken to be

$$\begin{aligned} \tilde{f}_{ji}[k+1] &= \frac{1}{2}\left(f_{ji}^{(j)}[k] + f_{ji}^{(i)}[k]\right) \\ &= f_{ji}[k] + \frac{1}{2}\left(\frac{\tilde{b}_i[k]}{\mathcal{D}_i} - \frac{\tilde{b}_j[k]}{\mathcal{D}_j}\right)\ . \end{aligned} \tag{6}$$

[Step 3.] If the above value is in the interval $[l_{ji}, u_{ji}]$, then $f_{ji}[k+1] = \tilde{f}_{ji}[k+1]$; otherwise, if it is below l_{ji} (respectively, above u_{ji}), it is set to the lower bound l_{ji} (respectively, to the upper bound u_{ji}):

$$f_{ji}[k+1] = \begin{cases} \tilde{f}_{ji}[k+1], & \text{if } l_{ji} \leq \tilde{f}_{ji}[k+1] \leq u_{ji}\ , \\ u_{ji}, & \text{if } \tilde{f}_{ji}[k+1] > u_{ji}\ , \\ l_{ji}, & \text{if } \tilde{f}_{ji}[k+1] < l_{ji}\ . \end{cases} \tag{7}$$

Once the values $\{f_{ji}[k+1] \mid (v_j, v_i) \in \mathcal{E}\}$ are obtained, the iteration is repeated.

One of the keys in the successful operation of the algorithm in (4)–(7) is bidirectional communication between physically neighboring nodes. To see this, consider a variation of the algorithm in which, at each iteration, nodes with positive flow balance attempt to change the flows on their outgoing edges only (since nodes have no way of sending information to their in-neighbors, they do not attempt to change the flows on their incoming edges). Specifically, (4) in Step 1 of the above algorithm is not needed and $\mathcal{D}_j$ in (5) can be taken to be $\mathcal{D}_j^+$; perhaps more importantly, in Step 2, we can set $\tilde{f}_{ji}[k+1] = f_{ji}[k] + \frac{1}{2}\frac{\tilde{b}_i[k]}{\mathcal{D}_i^+}$, and go to Step 3. In fact, with these changes the algorithm resembles the distributed algorithm proposed in [1], which is able to achieve balance in a flow network using a communication topology that matches the physical topology, but without flow constraints on the edges. However, as illustrated in the following example, this variation of the algorithm in [2] fails to obtain a set of feasible and balanced flows.

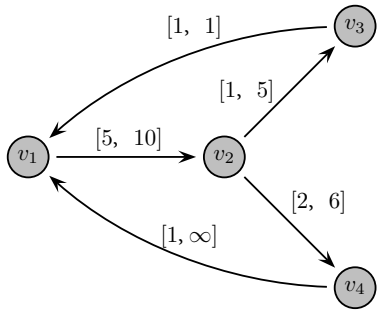

Fig. 1: Digraph used to demonstrate the ineffectiveness of the variant of the algorithm proposed in [2] for solving the flow assignment problem in a distributed fashion.

Example 1: Consider the digraph $\mathcal{G}_p = (\mathcal{V}, \mathcal{E}_p)$ shown in Figure 1 with four nodes ($\mathcal{V} = \{v_1, v_2, v_3, v_4\}$) and five edges, with lower and upper bounds on the edges as shown in the figure. It is easy to verify that the graph satisfies the circulation conditions in Theorem 1. In fact, if we execute the algorithm in [2] (assuming, of course, a bidirectional communication topology), we arrive at the following solution after approximately 200 iterations: $f_{21} = 5$, $f_{32} = 1$, $f_{42} = 4$, $f_{13} = 1$, $f_{14} = 4$. However, if we restrict ourselves to a communication topology that matches the physical topology (i.e., $\mathcal{G}_c = \mathcal{G}_p$) and execute the above variation of the algorithm (similar to the algorithm in [1] with Step 3 added in the end), we see that the algorithm fails to reach a balanced solution because it gets stuck (after a few steps) at a set of flows that does not achieve balance; these are: $f_{21} = 5$, $f_{32} = 2$, $f_{42} = 3$, $f_{13} = 1$, $f_{14} = 3$. ■

IV. Mixture of Bidirectional and Unidirectional Edges in the Communication Topology

We start our discussion with a rather stylized setting, where we make the following assumptions about the communication topology:

A1. Edges in the physical topology necessarily appear as

edges in the communication topology, i.e., $\mathcal{E}_p \subseteq \mathcal{E}_c$.

A2. If edge $(v_j, v_i) \in \mathcal{E}_c$, then either $(v_j, v_i) \in \mathcal{E}_p$ or $(v_i, v_j) \in \mathcal{E}_p$ (or both).

A3. Any edge (v_j, v_i) in the physical topology, the flow of which is restricted (i.e., $0 < l_{ji}$ and/or $u_{ji} < \infty$), necessarily appears in both directions in the communication topology.

The first two assumptions imply that the edges of the communication graph, $\mathcal{G}_c = (\mathcal{V}, \mathcal{E}_c)$, satisfy $\mathcal{E}_p \subseteq \mathcal{E}_c \subseteq \cup_{(v_j,v_i)\in\mathcal{E}_p}\{(v_j, v_i), (v_i, v_j)\}$. Moreover, the third assumption implies that, if edge $(v_j, v_i) \in \mathcal{E}_c$ but $(v_i, v_j) \notin \mathcal{E}_c$, then edge $(v_j, v_i) \in \mathcal{E}_p$ necessarily has $l_{ji} = 0$ and $u_{ji} = \infty$. Also, note that under Assumptions **A1**-**A3**, we have $\mathcal{N}_j^- \subseteq \mathcal{N}_{c,j}^-$ and $\mathcal{N}_j^+ \subseteq \mathcal{N}_{c,j}^+$ for every node $v_j \in \mathcal{V}$.

One natural thing to do is to try to combine the features of the algorithm in [1] (for edges with unrestricted flows under unidirectional communication), and the algorithm in [2] (for edges with restricted flows under bidirectional communication). Although there are many ways in which one can combine the features of the two algorithms, we describe below an algorithm that closely resembles the latter (which was described in detail in Section III). In particular, we assume that each node v_j is in charge of maintaining the flows on each of its outgoing edges (and also of informing its out-neighbors about any changes on the flows). If node v_j has a positive balance, it performs the update separately on each of its outgoing edges in a manner that depends on whether communication is bidirectional or not. In particular, we can partition the edges in $\mathcal{E}_p$ in two sets: the set of edges with bidirectional communication, denoted by

$$\mathcal{E}_{pb} = \{(v_j, v_i) \in \mathcal{E}_p \mid (v_j, v_i), (v_i, v_j) \in \mathcal{E}_c\} \ ,$$

and the set of edges with unidirectional communication, denoted by

$$\mathcal{E}_{pu} = \{(v_j, v_i) \in \mathcal{E}_p \mid (v_j, v_i) \in \mathcal{E}_c, (v_i, v_j) \notin \mathcal{E}_c\} \ .$$

The way node v_j updates the flows on its outgoing edges depends on whether the edge belongs in the set $\mathcal{E}_{pb}$ or $\mathcal{E}_{pu}$:

- **Bidirectional Communication:** For edge $(v_l, v_j) \in \mathcal{E}_{pb}$, node v_j updates f_{lj} as

$$f_{lj}[k+1] = \begin{cases} \tilde{f}_{lj}[k+1], & \text{if } l_{lj} \le \tilde{f}_{lj}[k+1] \le u_{lj} \ , \\ u_{lj}, & \text{if } \tilde{f}_{lj}[k+1] > u_{lj} \ , \\ l_{lj}, & \text{if } \tilde{f}_{lj}[k+1] < l_{lj} \ . \end{cases}$$

  where

$$\tilde{f}_{lj}[k+1] = f_{lj}[k] + \frac{1}{2}\left(\frac{\tilde{b}_j[k]}{\mathcal{D}_j} - \frac{\tilde{b}_l[k]}{\mathcal{D}_l}\right) \ ,$$

  with $\tilde{b}_j[k] = \max(b_j[k], 0)$, and $\tilde{b}_l[k] = \max(b_l[k], 0)$.
- **Unidirectional Communication:** For edge $(v_l, v_j) \in \mathcal{E}_{pu}$, node v_j updates f_{lj} as

$$f_{lj}[k+1] = f_{lj}[k] + \frac{1}{2}\frac{\tilde{b}_j[k]}{\mathcal{D}_j^+} \ ,$$

  with $\tilde{b}_j[k] = \max(b_j[k], 0)$.

Algorithm 1 below summarizes the proposed algorithm for dealing with both bidirectional and unidirectional edges in the communication graph. Algorithm 1 assumes that at initialization, each node is aware of the feasible flow interval on each of its incoming and outgoing edges, i.e., node v_j is aware of l_{ji}, u_{ji}, for each $v_i \in \mathcal{N}_j^-$, and l_{lj}, u_{lj}, for each $v_l \in \mathcal{N}_j^+$ (where the sets of in-neighbors $\mathcal{N}_j^-$ and out-neighbors $\mathcal{N}_j^+$, as well as $\mathcal{D}_j^-$, $\mathcal{D}_j^+$ and $\mathcal{D}_j$, of each node v_j are defined with respect to the physical topology $\mathcal{G}_p$, as in earlier sections). Furthermore, the flows are initialized at the lower limit of the feasible interval, i.e., $f_{ji}[0] = l_{ji}$ for each $(v_j, v_i) \in \mathcal{E}$. [This initialization is not critical and could be any value in the feasible flow interval $[l_{ji}, u_{ji}]$, as long as both node v_j and node v_i agree on the same value.] Subsequently, the nodes enter the iterative stage of the algorithm. At each iteration $k \ge 0$, node v_j is aware of the flows on its incoming edges $\{f_{ji}[k] \mid v_i \in \mathcal{N}_j^-\}$ and is in charge of updating its outgoing edges $\{f_{lj}[k] \mid v_l \in \mathcal{N}_j^+\}$. The update of the edge flow f_{lj} depends on whether the balance b_l of the corresponding out-neighbor is available to node v_j or not. Similarly, for each neighbor $v_i \in \mathcal{N}_j^-$, node v_j maintains the flow value f_{ji}, which it updates in the way its neighbor v_i would update, depending on whether v_j can send information to node v_i or not.

Theorem 2: Consider the setting described above where the physical topology is given by digraph $\mathcal{G}_p = (\mathcal{V}, \mathcal{E}_p)$ with interval constraints on (some of the) edges such that the circulation conditions in Theorem 1 hold. Suppose that Algorithm 1 is executed under a communication topology $\mathcal{G}_c = (\mathcal{V}, \mathcal{E}_c)$ that satisifes Assumptions **A1**-**A3**. If the digraph $\mathcal{G}_{pu} = (\mathcal{V}, \mathcal{E}_{pu})$ (which consists of the edges in $\mathcal{E}_p$ that do not have bidirectional communication capability) is strongly connected (or is a pure collection of strongly connected components, with no edges between these strongly connected components), then it holds that

$$\varepsilon[k+n] \le (1-c)\varepsilon[k] \ , \quad \forall k \ge 0 \ ,$$

where $n = |\mathcal{V}|$ is the number of nodes and $\varepsilon[k] \ge 0$ is the absolute imbalance of the network at iteration k (refer to Definition 1), with $c = \frac{1}{2n}\frac{1}{(2\mathcal{D}_{\max})^n}$, where $\mathcal{D}_{\max} = \max_{v_j\in\mathcal{V}} \mathcal{D}_j$. [Note that $\mathcal{D}_{\max}$ necessarily satisfies $1 \le \mathcal{D}_{\max} \le 2(n-1)$.]

Proof: The proof resembles the proof of Theorem 2 in [2]. In particular, it is easy to see that Propositions 1, 2 and 3 from [2], and most steps in that proof go through intact. Due to space limitations, we omit the proof. ■

V. Directed Communication Topology

In this section, we are interested in developing a distributed iterative algorithm for solving the constrained flow balancing problem in Section II-B under a communication topology that matches exactly the given physical topology, i.e., $\mathcal{G}_c = \mathcal{G}_p$. As before, we will have each node $v_j \in \mathcal{V}$ be in charge of assigning the flows $\{f_{lj} \mid v_l \in \mathcal{N}_j^+\}$ on its outgoing edges. However, node v_j will have to update flow f_{lj} without obtaining any information from out-neighbor v_l (at least not directly).

Algorithm 1: Distributed balancing under a communication topology that satisfies Assumptions **A1**-**A3**

Input: A strongly connected digraph $\mathcal{G}_p = (\mathcal{V}, \mathcal{E}_p)$ with $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}_p|$ edges (and no self-loops). Each node v_j is aware of lower and upper bounds on the flows ($l_{ji}, u_{ji}, \forall v_i \in \mathcal{N}_j^-$ and $l_{lj}, u_{lj}, \forall v_l \in \mathcal{N}_j^+$). A communication graph $\mathcal{G}_c = (\mathcal{V}, \mathcal{E}_c)$ (that satisfies Assumptions **A1**-**A3**) with $\mathcal{E}_p \subseteq \mathcal{E}_c \subseteq \cup_{(v_j,v_i)\in\mathcal{E}_p}\{(v_j, v_i), (v_i, v_j)\}$.

Initialization: Each node $v_j \in \mathcal{V}$ sets:

1) $f_{ji}[0] = l_{ji}, \forall v_i \in \mathcal{N}_j^-$

2) $f_{lj}[0] = l_{lj}, \forall v_l \in \mathcal{N}_j^+$

3) $\mathcal{D}_j = \mathcal{D}_j^- + \mathcal{D}_j^+$

Iteration: For $k = 0, 1, 2, \ldots$, each node $v_j \in \mathcal{V}$ does the following:

Step 1: It calculates $b_j[k] = \sum_{v_i\in\mathcal{N}_j^-} f_{ji}[k] - \sum_{v_l\in\mathcal{N}_j^+} f_{lj}[k]$ and sets $\tilde{b}_j[k] = \max(b_j[k], 0)$

Step 2: It transmits $\frac{\tilde{b}_j[k]}{\mathcal{D}_j}$ to $v_i \in \mathcal{N}_{c,j}^+$

Step 3: It receives $\frac{\tilde{b}_i[k]}{\mathcal{D}_i}$ from all $v_i \in \mathcal{N}_{c,j}^-$

Step 4: It calculates

$$\tilde{f}_{ji}[k+1] = \begin{cases} f_{ji}[k] + \frac{1}{2}\left(\frac{\tilde{b}_i[k]}{\mathcal{D}_i} - \frac{\tilde{b}_j[k]}{\mathcal{D}_j}\right), & \forall v_i \in \mathcal{N}_j^- \cap \mathcal{N}_{c,j}^+ \\ f_{ji}[k] + \frac{1}{2}\frac{\tilde{b}_i[k]}{\mathcal{D}_i}, & \forall v_i \in (\mathcal{N}_j^- \setminus \mathcal{N}_{c,j}^+) \end{cases}$$

$$\tilde{f}_{lj}[k+1] = \begin{cases} f_{lj}[k] + \frac{1}{2}\left(\frac{\tilde{b}_j[k]}{\mathcal{D}_j} - \frac{\tilde{b}_l[k]}{\mathcal{D}_l}\right), & \forall v_l \in \mathcal{N}_j^+ \cap \mathcal{N}_{c,j}^- \\ f_{lj}[k] + \frac{1}{2}\frac{\tilde{b}_j[k]}{\mathcal{D}_j}, & \forall v_l \in (\mathcal{N}_j^+ \setminus \mathcal{N}_{c,j}^-) \end{cases}$$

and sets

$$f_{ji}[k+1] = \begin{cases} \tilde{f}_{ji}[k+1], & \text{if } l_{ji} \le \tilde{f}_{ji}[k+1] \le u_{ji} \\ u_{ji}, & \text{if } \tilde{f}_{ji}[k+1] > u_{ji} \\ l_{ji}, & \text{if } \tilde{f}_{ji}[k+1] < l_{ji} \end{cases}$$

$$f_{lj}[k+1] = \begin{cases} \tilde{f}_{lj}[k+1], & \text{if } l_{lj} \le \tilde{f}_{lj}[k+1] \le u_{lj} \\ u_{lj}, & \text{if } \tilde{f}_{lj}[k+1] > u_{lj} \\ l_{lj}, & \text{if } \tilde{f}_{lj}[k+1] < l_{lj} \end{cases}$$

Let us assume that each node v_j has a unique identification (id). Furthermore, suppose (for convenience[1]) that each node is aware of the number of nodes n in the graph, and, of course, the id's of its in-neighbors and out-neighbors; however, it cannot send information to its in-neighbors (it can only receive information from them), and cannot receive information from its out-neighbors (it can only send information to them). In the algorithm we propose, each node v_j maintains additional variables in order to be able to determine how to increase/decrease the flow on each outgoing edge $(v_l, v_j) \in \mathcal{E}_p$. The main idea is best explained in terms of an extended digraph $\mathcal{G}_e = (\mathcal{V}_e, \mathcal{E}_e)$ that conforms to the constraints of the previous section (Assumptions **A1**-**A3**). More specifically, digraph $\mathcal{G}_e$ is constructed from $\mathcal{G}_p$ as follows (refer to Figure 2 for the extended digraph $\mathcal{G}_e$ that corresponds to the digraph in Figure 1):

A. The set of nodes $\mathcal{V}_e$ is given by $\mathcal{V}_e = \mathcal{V} \times \mathcal{V}$, where node (v_j, v_i) in $\mathcal{V}_e$ is denoted by $v_{j,i}$ for notational simplicity. From an algorithmic implementation point of view, nodes $\{v_{j,1}, v_{j,2}, ..., v_{j,n}\}$ are implemented by node v_j in the communication topology. In Figure 2, the flow updates related to the set of nodes surrounded by the dotted blue rectangle are executed by node v_1.

B. The set of edges $\mathcal{E}_e$ contains two types of edges and it is partitioned into two sets, $\mathcal{E}_{e_1}$ and $\mathcal{E}_{e_2}$ (so that $\mathcal{E}_e = \mathcal{E}_{e_1} \cup \mathcal{E}_{e_2}$ and $\mathcal{E}_{e_1} \cap \mathcal{E}_{e_2} = \emptyset$), as follows:

Type 1: Edge $(v_{l,j}, v_{i,j}) \in \mathcal{E}_{e_1}$ if $(v_l, v_i) \in \mathcal{E}_p$; such edges will receive flows that are unconstrained (they are drawn in black in Figure 2). Note that, for a fixed j, such edges form a digraph that resembles $\mathcal{G}_p$.

Type 2: Edge $(v_{j,l}, v_{j,j}) \in \mathcal{E}_{e_2}$ if $(v_l, v_j) \in \mathcal{E}_p$; such edges will receive flows that are constrained to be in the interval $[l_{(j,l),(j,j)}, u_{(j,l),(j,j)}] = [l_{lj}, u_{lj}]$ (they are drawn in red in Figure 2).

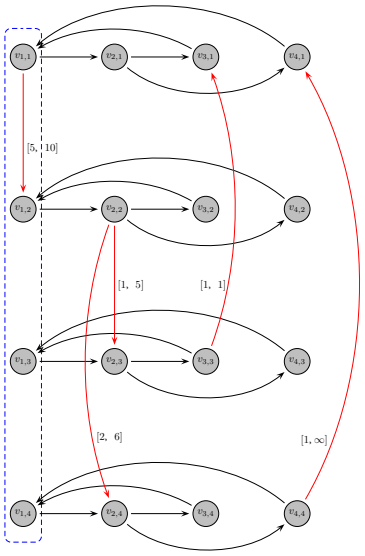

Fig. 2: Extended digraph for the graph in Figure 1.

It is not hard to check that digraph $\mathcal{G}_e$ will be strongly connected as long as the digraph $\mathcal{G}_p$ that describes the physical topology is strongly connected. For example, in the graph of Figure 2, we see that if we pick any two nodes on the same horizontal level, we can find a directed path that connects them (because the graph on each horizontal line is homomorphic to the physical topology, which is assumed strongly connected). Moreover, if we pick any two nodes on two different horizontal levels, we can find a directed path that connects these two levels (and, thus, these two nodes). This path will have to enter and exit each level at most once, but might have to traverse multiple nodes at each level. The reason is that, if we collapse all nodes on each horizontal

[1]It will become clear later that this assumption can easily be relaxed as nodes can simply track the (unique) id's of the nodes they receive information from.

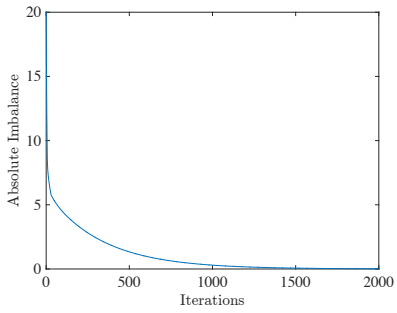

Fig. 3: Execution of the proposed algorithm on the digraph of Figure 1 (or on the extended graph in Figure 2), shown in terms of total (absolute) imbalance at each iteration.

level of the extended graph (all the nodes on each horizontal level are connected via directed paths), we recover a graph that is homomorphic to the physical topology $\mathcal{G}_p$ (and thus strongly connected).

Consider now executing Algorithm 1 under a physical topology that matches $\mathcal{G}_e$ and a communication topology in which edges with constraints have bidirectional communication capability and edges without constraints have unidirectional communication capability. Each unconstrained edge $(v_{l,j}, v_{i,j}) \in \mathcal{E}_{e_1}$ necessarily appears in a strongly connected component of the digraph of Type 1 (unconstrained) edges given by $\mathcal{G}_{e_1} = (\mathcal{V}, \mathcal{E}_{e_1})$: specifically, the induced subgraph of $\mathcal{G}_{e_1}$ that consists of the nodes $\{v_{1,j}, v_{2,j}, ..., v_{n,j}\}$ and (unconstrained) edges is homomorphic to the original graph and thus strongly connected (recall that the original graph $\mathcal{G}_p$ is assumed to be strongly connected). Thus, when we execute Algorithm 1 on the extended digraph under the communication topology described above, we know from Theorem 2 that we will reach a set of flows within the interval constraints and balanced, as long as $\mathcal{G}_e$ satisfies the circulation conditions in Theorem 1. The following lemma formally states that this is the case; its proof is omitted due to space limitations.

Lemma 1: Consider the physical topology $\mathcal{G}_p = (\mathcal{V}, \mathcal{E}_p)$ with interval constraints on the flow of each edge ($[l_{ji}, u_{ji}]$ with $0 \leq l_{ji} \leq u_{ji}$ for each edge $(v_j, v_i) \in \mathcal{E}_p$). Construct the extended digraph $\mathcal{G}_e$ based on $\mathcal{G}_p$ as described in the beginning of this section. The circulation conditions are satisfied on $\mathcal{G}_p$ if and only if they are satisfied on $\mathcal{G}_e$. Moreover, any set of flows $\{f^{(e)}_{j_1 j_2, i_1 i_2} \mid (v_{j_1,j_2}, v_{i_1,i_2}) \in \mathcal{E}_e\}$ that satisfy the interval and balance constraints on $\mathcal{G}_e$ can be used to find a set of flows $\{f^{(p)}_{ji} \mid (v_j, v_i) \in \mathcal{E}_p\}$ that satisfies the interval and balance constraints on $\mathcal{G}_p$. More specifically, the flow on each edge of the physical graph matches exactly the flow on the corresponding Type 2 edge of the extended graph, i.e., $f^{(p)}_{lj} = f^{(e)}_{jl,jj}$ for $(v_l, v_j) \in \mathcal{E}_p$.

Example 2: Let us revisit the digraph $\mathcal{G}_p = (\mathcal{V}, \mathcal{E}_p)$ shown in Figure 1. Figure 3 plots the absolute balance at each iteration during the execution of the proposed algorithm. Note that the plot includes the balances of all sixteen nodes in the extended digraph for this case. As we can see, the proposed algorithm is able to overcome the limitations imposed by the communication topology and reaches a set of flows (in the extended graph) that corresponds to a set of flows (in the physical graph) that satisfies the interval constraints and the balance constraints. This set of flows is given by $f_{21} = 5$, $f_{32} = 1$, $f_{42} = 4$, $f_{13} = 1$, $f_{14} = 4$ (in this case, they match exactly the flows obtained by the algorithm in [2] under a bidirectional communication topology though that will not necessarily be the case in general). Note that it takes significantly longer (in terms of number of iterations) to converge to this set of flows as the solution; this is due to the increased size of the extended graph. ■

VI. Concluding Remarks

In this paper, we introduced and analyzed a distributed algorithm for assigning balanced flows, within specified intervals, in a given digraph under a matching (possibly directed) communication topology. In the future, we plan to explore methodologies for allowing the nodes to distributively identify when such flow assignment is not feasible. We are also interested in reducing the time/space complexity of the proposed algorithm (e.g., is it possible for the red edges in Figure 2 be constructed in a different manner) and in generalizing it to more general topologies (that do not necessarily contain the physical topology).

References

[1] A. Rikos, T. Charalambous, and C. N. Hadjicostis, "Distributed weight balancing over digraphs," *IEEE Trans. on Control of Network Systems*, vol. 1, no. 2, pp. 190–201, May 2014.

[2] C. N. Hadjicostis and A. D. Domínguez-García, "Distributed balancing in digraphs under interval constraints," in *Proc. of IEEE Conf. on Decision and Control*, 2016, pp. 1769–1774.

[3] L. Ford and D. R. Fulkerson, *Flows in Networks*. Princeton University Press, 2010.

[4] D. P. Bertsekas, P. A. Hosein, and P. Tseng, "Relaxation methods for network flow problems with convex arc costs," *SIAM Journal of Control and Optimization*, vol. 25, no. 5, pp. 1219–1243, 1987.

[5] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY: Cambridge University Press, 2004.

[6] P. DeLellis, M. di Bernardo, F. Garofalo, and M. Porfiri, "Evolution of complex networks via edge snapping," *IEEE Trans. on Circuits and Systems*, vol. 57, no. 8, pp. 2132–2143, August 2010.

[7] W. Yu, P. DeLellis, G. Chen, M. di Bernardo, and J. Kurths, "Distributed adaptive control of synchronization in complex networks," *IEEE Trans. on Automatic Control*, vol. 57, no. 8, pp. 2153–2158, August 2012.

[8] M. H. Schneider and S. A. Zenios, "A comparative study of algorithms for matrix balancing," *Operations Research*, vol. 38, no. 3, pp. 439–455, 1990.

[9] A. D. Domínguez-García and C. N. Hadjicostis, "Distributed matrix scaling and application to average consensus in directed graphs," *IEEE Trans. on Automatic Control*, vol. 58, no. 3, pp. 667–681, March 2013.

[10] B. Gharesifard and J. Cortés, "Distributed strategies for generating weight-balanced and doubly stochastic digraphs," *European Journal of Control*, vol. 18, no. 6, pp. 539–557, 2012.

[11] C. N. Hadjicostis and A. I. Rikos, "Distributed strategies for balancing a weighted digraph," in *Proc. of Mediterranean Conf. on Control Automation*, 2012, pp. 1141–1146.

[12] A. Priolo, A. Gasparri, E. Montijano, and C. Sagues, "A decentralized algorithm for balancing a strongly connected weighted digraph," in *Proc. of American Control Conf.*, 2013, pp. 6547–6552.

[13] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice hall, 1993.