Finite-Time Distributed Flow Balancing

Christoforos N. Hadjicostis, Alejandro D. Domínguez-García, and Apostolos I. Rikos

***Abstract*— 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, with nonnegative end points that correspond to lower and upper flow limits. The paper proposes and analyzes a distributed iterative algorithm for computing, *in finite time*, admissible and balanced flows, i.e., flows that are within the given intervals at each edge and balance the total in-flow with the total out-flow at each node. The algorithm assumes a communication topology that allows bidirectional exchanges between pairs of nodes that are physically connected (i.e., nodes that share a directed edge in the physical topology). If the given initial flows and flow limits are commensurable (i.e., integer multiples of a given constant), then the proposed distributed algorithm operates exclusively with flows that are commensurable and is shown to complete in a finite number of steps (assuming a solution set of admissible and balanced flows exists). When no upper limits are imposed on the flows, a variation of the proposed algorithm is shown to complete in finite time even when initial flows and lower limits are arbitrary nonnegative real values (not necessarily commensurable).**

***Index Terms*— Flow Balancing, Distributed Algorithms, Finite Time Algorithms.**

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 at each node. In the network flow literature, this problem is referred to as the feasible circulation problem (see, e.g., [1], [2]). The feasible circulation problem has been studied quite extensively with most algorithmic approaches assuming implicitly the existence of a centralized processor with access to all data defining the problem; recent work has focused on developing distributed algorithms (see Chapter 5 of [3]).

In this paper, we propose an algorithm that allows the nodes to compute, in a distributed manner, a solution to the feasible circulation problem. The distributed algorithm operates in an iterative fashion and assumes that nodes that are physically connected (i.e., nodes that are connected via an edge in the flow network) can communicate in a bidirectional manner. The proposed distributed algorithm can be shown to converge as long as a feasible solution exists. The algorithm combines some of the features of the distributed flow balancing algorithm in [4], which achieves flow balancing in finite time (assuming integer flows and integer lower and upper flow limits) and the asymptotic distributed flow balancing algorithm in [5], [6] (which allows flows and lower/upper flow limits to be real-valued).

The second contribution of this work is that it establishes conditions under which the proposed algorithm completes in finite time. More specifically, when the initial flows and flow limits are commensurable, i.e., they all take values that are integer multiples of a given constant c, then the proposed algorithm operates exclusively with flows that are commensurable and its proof of termination in finite time shares some of the features of the proof of termination for the algorithm in [4]. Even when the initial flows and the lower flow limits are not commensurable (but under the additional assumption that there are no upper limits on the flows), a variation of the proposed algorithm can be shown to also complete in finite time. In either case, the proposed algorithm (or its variation) generates/utilizes a finite set of flow values and terminates in finite time. In that sense, these algorithms could also be seen as finite-time distributed algorithms for obtaining balanced flows with quantized values [7].

The feasible circulation problem addressed in this paper is a special case of the feasible distribution problem, which is also closely related to the max-flow min-cut problem (see, e.g., [1], [2]); thus, the algorithms utilized to solve the feasible circulation problem relate to the algorithms used to solve the max-flow min-cut problem, the implementations of which are typically centralized. One such algorithm is the *feasible distribution algorithm* (see, [2]), which is discussed in more detail at the end of Section II. It will become clear from our developments later on, that the algorithm proposed in this paper can be seen as a distributed alternative algorithm to this (centralized) feasible distribution algorithm in [2]. The problem we deal with in this paper can also be viewed as the problem of weight balancing a given digraph in a distributed manner, a topic that has attracted recent attention [8], [9], [10], [11], [6].

II. Preliminaries

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

C. N. Hadjicostis and A. I. Rikos are with the Department of Electrical and Computer Engineering at the University of Cyprus, Nicosia, Cyprus. E-mails: {chadjic, rikos.apostolos}@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.

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 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 $f_{ji} \in \mathbb{R}$ 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 define (respectively) the flow, lower limit, and upper limit $(n \times n)$-dimensional matrices as follows: $F = [f_{ji}]$, $L = [l_{ji}]$, 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 of v_j by $\mathcal{D}_j^+$. We will also use $\mathcal{D}_j = \mathcal{D}_j^- + \mathcal{D}_j^+$ to denote the *total degree* of node v_j.

For the purposes of developing our distributed flow balancing algorithm, we will rely on a communication topology that is captured by a digraph $\mathcal{G}_c = (\mathcal{V}, \mathcal{E}_c)$ that includes bidirectional edges between all nodes that are neighbors in the physical topology. In other words,

$$\mathcal{E}_c = \{(v_j, v_i), (v_i, v_j) \mid (v_j, v_i) \in \mathcal{E}_p\} \ . \qquad (1)$$

This implies that all nodes that are neighbors in the physical topology can communicate in a bidirectional manner.

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 quantities.

Definition 1: The total *in-flow* of node v_j is denoted by f_j^-, and is defined as $f_j^- = \sum_{v_i \in \mathcal{N}_j^-} f_{ji}$, whereas the total *out-flow* of node v_j is denoted by f_j^+, and is defined as $f_j^+ = \sum_{v_l \in \mathcal{N}_j^+} f_{lj}$. The *flow balance* or *balance* of node v_j is denoted by b_j and is defined as $b_j = f_j^- - f_j^+$.

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 if its *total imbalance* (or *absolute imbalance*) is 0, i.e., $\varepsilon = \sum_{j=1}^n |b_j| = 0$.

The distributed algorithm we will develop is iterative, and we will use k to index the iterations. Thus, we will use $f_{ji}[k]$ to denote the estimate, at iteration k, of the flow value on the link $(v_j, v_i) \in \mathcal{E}_p$; $f_j^+[k]$ to denote the estimate, at iteration k, of the total out-flow of node v_j at iteration k; and so forth.

Flow Assignment Problem: We are given a physical (strongly connected) digraph $\mathcal{G}_p = (\mathcal{V}, \mathcal{E}_p)$, and a communication digraph $\mathcal{G}_c = (\mathcal{V}, \mathcal{E}_c)$ (with $\mathcal{E}_c$ as defined in (1)), 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 obtain a set of flows $\{f_{ji} \mid (v_j, v_i) \in \mathcal{E}_p\}$ that satisfy the following:

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 $v_j \in \mathcal{V}$.

Theorem 1: (Circulation Theorem [1]) 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

$$\sum_{(v_j,v_i) \in \mathcal{E}_\mathcal{S}^-} l_{ji} \leq \sum_{(v_l,v_j) \in \mathcal{E}_\mathcal{S}^+} u_{lj} \ , \qquad (2)$$

where

$$\mathcal{E}_\mathcal{S}^- = \{(v_j, v_i) \in \mathcal{E}_p \mid v_j \in \mathcal{S}, \ v_i \in \mathcal{V} - \mathcal{S}\} \ , \qquad (3)$$

$$\mathcal{E}_\mathcal{S}^+ = \{(v_l, v_j) \in \mathcal{E}_p \mid v_j \in \mathcal{S}, \ v_l \in \mathcal{V} - \mathcal{S}\} \ . \qquad (4)$$

Assuming the above circulation conditions hold, a variety of centralized algorithms for obtaining such flows exist (see, e.g., [1], [2]). As mentioned in the introduction, one possibility is the feasible distribution algorithm [2], which is a centralized algorithm that iteratively attempts to find a path between the set of positively-balanced nodes and the set of negatively-balanced nodes. If such a path exists, then the flow on each link is increased or decreased by a fixed amount that can be computed from the actual flows, the link lower and upper capacity limits, and the imbalance due to the first and last nodes in the path. This operation reduces the imbalance contribution due to the first and last nodes, while keeping the balance unchanged for all other nodes, clearly reducing the total imbalance from one iteration to the next. If a solution exists, the algorithm achieves balance in finite time.

III. Distributed Flow Assignment Algorithm

The distributed algorithm we develop is iterative and operates by having, at each iteration, nodes with positive balance attempt to change the flows on one of their (incoming or outgoing) edges, so as to get balanced (or get closer to being balanced). More specifically, a node v_j with *positive* balance b_j attempts to increase by b_j the flow of one of its outgoing edges or decrease by b_j the flow of one of its incoming edges, so as to become balanced. One potential problem in this process is that flows on different edges are restricted to lie in certain intervals (thus, the $\pm b_j$ attempted change on the flow may not be allowed in full). In addition, since the flow on each edge affects the balance of *two* nodes (both of which may happen to simultaneously attempt to adjust the edge flow), the nodes need to coordinate with the corresponding neighbor in order to reach an agreement on the flow for that particular edge. This can always be done because we assume bidirectional communication. For simplicity, we assume that the (outgoing and incoming) edges of each node are assigned an order, and that each time a node needs to adjust its flows, it chooses one of its incident edges in a round-robin fashion. The proposed algorithm 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 $[l_{ji}, u_{ji}]$.]

Each node also assigns an order in the set $\{0, 1, 2, ..., (\mathcal{D}_j - 1)\}$ to each of its outgoing and its incoming edges (where $\mathcal{D}_j = \mathcal{D}_j^- + \mathcal{D}_j^+$ is the total degree of node v_j). It also initializes an internal index variable FTC_j (next *flow to change* by node v_j), which indicates the order of the edge whose flow will be changed next, to be $FTC_j = 0$.

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 flow at one of its incoming edges $\{f_{ji}[k+1] \mid v_i \in \mathcal{N}_j^-\}$, or one of its 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). More specifically, node v_j chooses to change the flow associated with the edge that has order FTC_j. There are two cases to consider:

(i) If this edge is an incoming edge (v_j, v_i), then node v_j attempts to change the flow on this edge as $f_{ji}^{(j)}[k+1] = f_{ji}[k] - b_j[k]$; we will denote this change as

$$\Delta f_{ji}^{(j)} = -b_j[k] \; . \tag{5}$$

(ii) If this edge is an outgoing edge (v_l, v_j), then node v_j attempts to change the flow on this edge as $f_{lj}^{(j)}[k+1] = f_{lj}[k] + b_j[k]$; we will denote this change as

$$\Delta f_{lj}^{(j)} = +b_j[k] \; . \tag{6}$$

Note that for all other edges $(v_j, v_{i'})$, $v_{i'} \in \mathcal{N}_j^-$, and $(v_{l'}, v_j)$, $v_{l'} \in \mathcal{N}_j^+$, node v_j does not attempt any change, i.e., $\Delta f_{ji'}^{(j)}[k] = 0$ and $\Delta f_{l'j}^{(j)}[k] = 0$.

Also, once a change is imposed to the edge associated with FTC_j, node v_j updates its index for the next flow to change to $FTC_j = (FTC_j + 1) \mod \mathcal{D}_j$. This ensures that each node considers all of its edges, in a round-robin fashion, each time it has a positive balance. [The next time node v_j needs to change the flow of an incoming/outgoing edge, it will continue from the edge it stopped the previous time.]

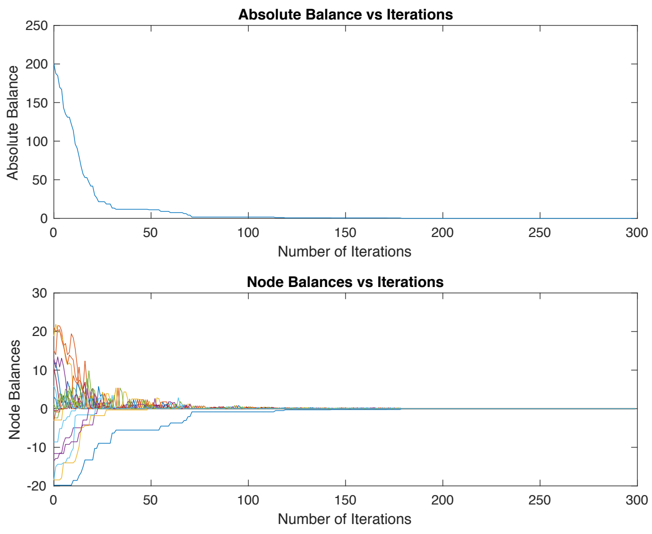

Fig. 1: Execution of Algorithm 1 on a random digraph with 20 nodes and commensurable flows and flow limits.

[Step 2.] For each edge $(v_j, v_i) \in \mathcal{E}$, both nodes v_j and v_i calculate the interim value of the flow as

$$\tilde{f}_{ji}[k+1] = f_{ji}[k] + \Delta f_{ji}^{(j)}[k] + \Delta f_{ji}^{(i)}[k] \; , \tag{7}$$

where $\Delta f_{ji}^{(j)}[k] \leq 0$ and $\Delta f_{ji}^{(i)}[k] \geq 0$ (it is possible that for many edges $(v_j, v_i) \in \mathcal{E}_p$, we will have $\Delta f_{ji}^{(j)}[k] = 0$ and/or $\Delta f_{ji}^{(i)}[k] = 0$).

[Step 3.] If value $\tilde{f}_{ji}[k+1]$ 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}):

$$f_{ji}[k+1] = \begin{cases} \tilde{f}_{ji}[k+1], & \text{if } l_{ji} \leq \tilde{f}_{ji}[k+1] \leq u_{ji} \; , \\ u_{ji}, & \text{if } \tilde{f}_{ji}[k+1] > u_{ji} \; , \\ l_{ji}, & \text{if } \tilde{f}_{ji}[k+1] < l_{ji} \; . \end{cases} \tag{8}$$

Once the values $\{f_{ji}[k+1] \mid (v_j, v_i) \in \mathcal{E}\}$ are obtained, the iteration is repeated. The pseudocode for the algorithm described by iterations (5)–(8) is provided as Algorithm 1.

An example of the operation of the proposed algorithm is shown in Fig. 1. We randomly generated a digraph of 20 nodes and selected random lower and upper flow limits for each edge so that they have *commensurable*[1] values. We verified in the end that the lower and upper limits satisfy the circulation conditions in Theorem 1, which also implies that the graph is strongly connected. We observe that the algorithm allows the nodes to reach a set of admissible and balanced flows after a finite number of iterations.

[1]All values for lower and upper limits were selected to be positive integer multiples of a small positive constant $c = 11/512$.

Algorithm 1: Finite-Time Distributed Flow Balancing

```
Input: $l_{ji}, u_{ji}, \forall v_i \in \mathcal{N}_j^-$ and $l_{lj}, u_{lj}, \forall v_l \in \mathcal{N}_j^+$
Output: $f_{ji}, \forall v_i \in \mathcal{N}_j^-$ and $f_{lj}, \forall v_l \in \mathcal{N}_j^+$
Each node $v_j \in \mathcal{V}$ separately does the following.
Initialization:
1. Set $f_{ji}[0] = \frac{l_{ji}+u_{ji}}{2}$, for all $(v_j, v_i) \in \mathcal{E}_p$
2. Assign a unique order in $\{0, 1, 2, ..., \mathcal{D}_j - 1\}$ to each
   incoming and each outgoing edge
3. Set $FTC_j = 0$
Iteration: foreach iteration, $k = 0, 1, ...,$ do
    Determine: $f_j^+[k]$, $f_j^-[k]$, $b_j[k]$
    Set:
    $\Delta f_{ji}^{(j)}[k] = 0, \forall v_i \in \mathcal{N}_j^-$; $\Delta f_{lj}^{(j)}[k] = 0, \forall v_l \in \mathcal{N}_j^+$
    If $b_j[k] > 0$, then
      1. Select the edge $(v_j, v_i)$ (or $(v_l, v_j)$) associated
         with $FCT_j$ and set
           $\Delta f_{ji}^{(j)}[k] = -b_j[k]$ (or $\Delta f_{lj}^{(j)}[k] = +b_j[k]$)
      2. Set $FTC_j = (FTC_j + 1) \mod \mathcal{D}_j$
    Transmit:
    $\Delta f_{ji}^{(j)}$ to in-neighbor $v_i$ for all $v_i \in \mathcal{N}_j^-$
    $\Delta f_{lj}^{(j)}$ to out-neighbor $v_l$ for all $v_l \in \mathcal{N}_j^+$
    Receive:
    $\Delta f_{ji}^{(i)}$ from in-neighbor $v_i$ for all $v_i \in \mathcal{N}_j^-$
    $\Delta f_{lj}^{(l)}$ from out-neighbor $v_l$ for all $v_l \in \mathcal{N}_j^+$
    Set: for all $v_i \in \mathcal{N}_j^-$ and for all $v_l \in \mathcal{N}_j^+$
    $\tilde{f}_{ji}[k+1] = f_{ji}[k] + \Delta f_{ji}^{(j)}[k] + \Delta f_{ji}^{(i)}[k]$
    $\tilde{f}_{lj}[k+1] = f_{lj}[k] + \Delta f_{lj}^{(l)}[k] + \Delta f_{lj}^{(j)}[k]$
    $f_{ji}[k+1] = \begin{cases} \tilde{f}_{ji}[k+1], & \text{if } l_{ji} \le \tilde{f}_{ji}[k+1] \le u_{ji} \\ u_{ji}, & \text{if } \tilde{f}_{ji}[k+1] > u_{ji} \\ l_{ji}, & \text{if } \tilde{f}_{ji}[k+1] < l_{ji} \end{cases}$
    $f_{lj}[k+1] = \begin{cases} \tilde{f}_{lj}[k+1], & \text{if } l_{lj} \le \tilde{f}_{lj}[k+1] \le u_{lj} \\ u_{lj}, & \text{if } \tilde{f}_{lj}[k+1] > u_{lj} \\ l_{lj}, & \text{if } \tilde{f}_{lj}[k+1] < l_{lj} \end{cases}$
```

IV. Termination of Proposed Algorithm

A. Commensurable Flow Limits

Consider the Flow Assignment Problem described in Section II, where we additionally assume that all the flow limits are commensurable, i.e., (nonnegative) integer multiples of a constant c. More formally, for each edge $(v_j, v_i) \in \mathcal{E}_p$, we have $l_{ji} = l'_{ji}c$ and $u_{ji} = u'_{ji}c$, where c is a real number and l'_{ji} is a nonnegative integer, and u'_{ji} is a positive integer.

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 the edges such that the circulation conditions in Theorem 1 hold. Algorithm 1 reaches a set of feasible and balanced flows after a finite number of iterations.

Proof: The proof establishes a connection with the proof of Proposition 3 in [4]. In particular, since flow limits (lower and upper) are commensurable, it is easy to see that the initial flows $f_{ji}[0] = \frac{l_{ji}+u_{ji}}{2} = (l'_{ji} + u'_{ji})\frac{c}{2}$, as well as the initial balances at each node, are integer multiples of $\frac{c}{2}$. Moreover, since the positive or negative changes inflicted on flows at each iteration are determined by the addition and/or subtraction of certain positive balances (all of which are integer multiples of $\frac{c}{2}$), subsequent flows and balances will also be integer multiples of $\frac{c}{2}$. Even when the limits are imposed (because the flows exceed the upper bound or are below the lower bound, see (8)), the resulting flows will be integer multiples of $\frac{c}{2}$ because the projection operation in (8) will set the flows to the upper or lower capacity limit values, which are also a multiple of $\frac{c}{2}$.

If we divide the initial flows and the flow limits by $\frac{c}{2}$ we obtain a new instance of the problem where the initial flows and flow limits have integer values, given by $\overline{f}_{ji}[0] = \frac{f_{ji}[0]}{c/2} = l'_{ji} + u'_{ji}$, $\overline{l}_{ji} = \frac{l_{ji}}{c/2} = 2l'_{ji}$, $\overline{u}_{ji} = \frac{u_{ji}}{c/2} = 2u'_{ji}$, for each edge $(v_j, v_i) \in \mathcal{E}_p$. It is not hard to see that this instance of the problem will satisfy the circulation conditions of Theorem 1 if and only if the original instance of the problem satisfies the circulation conditions. Moreover, the execution of Algorithm 1 on the new instance of the problem generates/utilizes only integer flows.

The operation of Algorithm 1 on the new instance of the problem resembles some of the features of Algorithm 1 in [4]. The main difference from [4] is that the algorithm here (when applied on the new instance of the problem) will have node v_j with positive balance $\overline{b}_j$ (that is also an integer) attempt to increase (decrease) the flow of exactly one outgoing (incoming) edge by $\overline{b}_j$; this edge is chosen by node v_j in a round-robin fashion. [In contrast, Algorithm 1 in [4] may increase (decrease) the flow of multiple outgoing (incoming) edges, by going through all edges in a round-robin fashion and attempting to increase (decrease) by 1 the corresponding flow, until its balance becomes zero (assuming no other flow changes on its edges).] Despite this difference, the proof for the termination of Algorithm 1 in [4] in finite time can be adjusted to prove that Algorithm 1 in this paper (applied on the modified instance of the problem) terminates after a finite number of steps. In particular, Proposition 1 from [4] holds intact and is reproduced below for completeness.

Proposition 1: Consider the problem formulation described in Section II and assume some flow assignment $f_{ji}[k]$ for all edges $(v_j, v_i) \in \mathcal{E}_p$. The following hold:

1) For any subset of nodes $\mathcal{S} \subset \mathcal{V}$, let $\mathcal{E}_\mathcal{S}^-$ and $\mathcal{E}_\mathcal{S}^+$ be defined by (3) and (4) respectively. Then,
$$\sum_{v_j \in \mathcal{S}} b_j[k] = \sum_{(v_j,v_i) \in \mathcal{E}_\mathcal{S}^-} f_{ji}[k] - \sum_{(v_l,v_j) \in \mathcal{E}_\mathcal{S}^+} f_{lj}[k] \ ;$$
2) $\sum_{j=1}^{n} b_j[k] = 0$;
3) $\varepsilon[k] = 2\sum_{v_j \in \mathcal{V}^-[k]} |b_j[k]|$ where $\mathcal{V}^-[k] = \{v_j \in \mathcal{V} \mid b_j[k] < 0\}$.

Notice that in Algorithm 1 only nodes with positive balance attempt to make changes, causing a change that (i) cannot make their own balance negative (but could possibly drive it to zero) and (ii) necessarily increases the balance of one of their out-neighbors (or in-neighbors) by increasing (or decreasing) the flow in one of their outgoing (or incoming) edges. These two facts can be used to establish (employing similar arguments as in [4]) the following important property: *nodes with nonnegative balance retain nonnegative balance (whereas nodes with negative balance could obtain at some iteration nonnegative balance, and remain with nonnegative balance for all subsequent iterations).*

From the third statement of Proposition 1, we have $\varepsilon[k+1] = 2\sum_{v_j \in \mathcal{V}^-[k+1]} |b_j[k+1]|$ and $\varepsilon[k] = 2\sum_{v_j \in \mathcal{V}^-[k]} |b_j[k]|$. Moreover, when a node with positive balance attempts to make a change in the flow values, the induced increase in the absolute value of the balance of one of the out-neighbors (or in-neighbors) cannot be more than the decrease in the absolute value of the balance of the node that initiates the change. Thus, using these two observations, we can establish that

$$0 \leq \varepsilon[k+1] \leq \varepsilon[k] \ , \ \ \forall k \geq 0 \ , \tag{9}$$

where $\varepsilon[k] \geq 0$ is the total imbalance of the network at iteration k (in Definition 2). To see this, consider a node $v_j \in \mathcal{V}^-[k]$ with balance $b_j[k] < 0$. We have two cases to consider (i) all neighbors of node v_j have negative or zero balance; and (ii) at least one neighbor of node v_j has positive balance. In both cases, during the execution of Algorithm 1, node v_j will not attempt to make any flow changes on its edges. In the first case, the flows on the edges associated with node v_j, and thus the balance of node v_j, will not change (i.e., $b_j[k+1] = b_j[k] < 0$). In the second case, we have $b_i[k] \geq 0$ or $b_l[k] \geq 0$, for some $v_i \in \mathcal{N}_j^-$ or $v_l \in \mathcal{N}_j^+$, which may lead to an increase of the balance of node v_j, i.e., $b_j[k+1] \geq b_j[k]$. In fact, we will have that either $b_j[k+1] \geq 0$ (i.e., $v_j \notin \mathcal{V}^-[k+1]$) or $b_j[k+1] < 0$ and $|b_j[k+1]| \leq |b_j[k]|$. Since $\varepsilon[k+1] = 2\sum_{v_j \in \mathcal{V}^-[k+1]} |b_j[k+1]|$, the inequality in (9) holds.

To establish that Algorithm 1 applied on the modified instance of the problem terminates after a finite number of steps, we can use a contradiction argument similar to the one in [4] (not developed here due to space limitations).

Finally, the last step needed is to argue that the execution of Algorithm 1 on the original version of the problem follows identical steps as its execution on its modified instance (with values that are scaled by $\frac{c}{2}$ compared to the values of the modified instance of the problem, i.e., for each iteration k, we have $f_{ji}[k] = \overline{f}_{ji}[k]\frac{c}{2}$ for each $(v_j, v_i) \in \mathcal{E}_p$). [Note that if the nodes know the value of constant c, then it is easy to run the modified instance of the balancing problem; this is harder to do if the value of c is not known.] ■

B. Non-Commensurable Initial Flows with No Flow Limits

In this section, we argue that even when the initial flows and lower flow limits are arbitrary nonnegative real numbers (not necessarily commensurable) but there are no upper flow limits, a variation of Algorithm 1 terminates after a finite number of steps. The variation of the algorithm is as follows:
1.) Since there are no upper flow limits, we can set the initial flows to an arbitrary f_{ji}, $f_{ji} \geq l_{ji}$, for all $(v_j, v_i) \in \mathcal{E}$.
2.) Each node v_j with positive balance $b_j[k]$ at iteration k, only attempts to change (increase) the flow on one of its outgoing edges (i.e., it does not attempt to decrease the flows on its outgoing edges). This can always be done, regardless of which outgoing edge is selected, because there are no upper limits on the flows. In order to select which flow to increase, node v_j assigns at initialization a unique order in $\{0, 1, 2, ..., \mathcal{D}_j^+ - 1\}$ to each outgoing edge and uses variable FTC_j to select a different outgoing edge each time it becomes positively balanced (FTC_j is updated as $FTC_j = (FTC_j + 1) \mod \mathcal{D}_j^+$, so that outgoing edges are selected in a round-robin fashion).

It should be clear that the second and third conditions in (8) will never be invoked when executing the above variation of Algorithm 1 (initial flows are set to be greater than or equal to lower limits and cannot be decreased, plus there are no upper bound restrictions). Due to space limitations, we only provide a sketch of the proof of the following theorem.

Theorem 3: Consider the setting described above where the physical topology is given by a strongly connected digraph $\mathcal{G}_p = (\mathcal{V}, \mathcal{E}_p)$ with a positive lower bound l_{ji} and no upper bound ($u_{ji} = \infty$) on each edge $(v_j, v_i) \in \mathcal{E}$. The variation of Algorithm 1 reaches a set of feasible and balanced flows after a finite number of iterations.

Proof: (Sketch) Note that when there are positive lower limits and no upper limits on flows, the circulation conditions are satisfied as long as the graph is strongly connected. Since the flow values at each edge are set to f_{ji}, $f_{ji} \geq l_{ji}$, the initial balances at each node v_j are of the form

$$b_j[0] = \sum_{v_i \in \mathcal{N}_j^-} f_{ji} - \sum_{v_l \in \mathcal{N}_j^+} f_{lj} \ . \tag{10}$$

Notice that each f_{ji} appears in two such balances, once in the balance of node v_j with a positive sign, and once in the balance of node v_i with a negative sign.

The first part of the proof is to establish that the absolute (total) imbalance of the network (i.e., the sum of the absolute values of the node balances) cannot increase during the execution of Algorithm 1. This can be established following similar steps as in the proofs for the termination/convergence of the algorithms in [4], [5]. For this reason, we only provide some brief commentary about this first part of the proof. Intuitively, a node v_j with a positive balance $b_j[k]$ takes steps to make its balance $b_j[k+1] = 0$ by increasing one of its outgoing edges. If node v_j succeeds (i.e., if no other changes are inflicted on the flows of its incoming edges), then there will be a gain (decrease) in the absolute imbalance $\varepsilon[k+1]$ in terms of the value contributed by $b_j[k+1]$ (which will be zero). However, this decrease could be annihilated (in the worst case) by an equal increase in the balance of the out-neighbor that was affected by the change that node v_j attempted. Again, using the fact that nodes can only increase the balance of their out-neighbors, we can establish (see, for example, [4]) that nodes with nonnegative balance retain nonnegative balance (whereas nodes with negative balance could obtain at some iteration nonnegative balance, and remain with nonnegative balance for all subsequent iterations). One can then argue that since (i) nodes with negative balance do not attempt to make any changes, and (ii) nodes with positive balance increase the balances of out-neighbors in a round-robin fashion, eventually the absolute imbalance will decrease because the change will reach a node with negative balance. Unless the circulation conditions of Theorem 1 do *not* hold, one can show that after a finite number of steps, the absolute imbalance will decrease.

The second part of the proof is to establish that the values of the node balances, at any iteration k, during the execution of the proposed algorithm can be written as

$$b_j[k] = \sum_{(v_\ell, v_i) \in \mathcal{E}} c_{\ell i}^{(j)}[k] f_{\ell i} \; , \tag{11}$$

where the coefficients $c_{\ell i}^{(j)}[k]$ belong in the set $\{-1, 0, 1\}$. In fact, among all parameters associated with $f_{\ell i}$ at iteration k, namely the parameters

$$\{ c_{\ell i}^{(j)}[k] \mid v_j \in \mathcal{V} \} \; , \tag{12}$$

we have that either (i) all of them are zero, or (ii) exactly one of them has value $+1$ and exactly one of them has value -1 (and all other values are zero).

We provide some intuition about why (11) holds at the end of the proof. Assuming, for now, that (11) is true, the third part of the proof relies on the fact that node balances can only take a finite set of values (clearly, each node balance can take at most $3^{|\mathcal{E}|}$ values). Since balances can only take a finite set of values, the sum of the absolute balances can also only take a finite set values. Moreover, during the execution of Algorithm 1, we have from the first part of the proof that the absolute imbalance (i.e., the sum of the absolute balances over all nodes) cannot increase and, in fact, it will necessarily decrease after a finite number of iterations. Since the absolute imbalance can only take a finite set of values, we conclude that Algorithm 1 will complete after a finite number of iterations.

Before closing, we provide some intuition as to why (11) holds. Clearly, (11) holds for $k = 0$ (see (10)). We can use induction to argue that it will also hold for node balances at any k. To see this, notice that according to Algorithm 1, each node v_j with positive $b_j[k]$ balance at iteration k selects one outgoing edge and increases its flow by $b_j[k]$. If no other changes are imposed on the edges of node v_j, its balance at the next iteration will be zero and effectively all of its positive balance will be added to its out-neighbor. One can think of this process as a transfer of the coefficients $c_{\ell i}^{(j)}[k]$ in (11) (many of which may be zero) to that neighbor of node v_j. These coefficients will be removed from the balance of v_j and will be added to the balance of the neighbor of node v_j, possibly causing the annihilation of some of these coefficients (the "-1" coefficient for some flow $f_{\ell i}$ may meet the "$+1$" coefficient for the same flow, which will cause the overall coefficient for $f_{\ell i}$ to be zero). Note that it is not possible to get a new coefficient that is different from -1, 0, or $+1$ because, at any given iteration k, the condition stated after (12) holds. The reason is that each flow $f_{\ell i}$ appears with positive sign in the balance of node v_ℓ and with negative sign in the balance of node v_i; the only times these coefficients can be transferred to other nodes is when node v_ℓ obtains positive balance for the first time (coefficient "$+1$") and when node v_i obtains positive balance for the first time (coefficient "-1"). When other nodes become positively balanced for the first time, they will attempt to transfer their imbalance by transferring the accumulated imbalance due to (i) their own initial flows (generating in the process $+1$ coefficients for their in-flows and -1 coefficients for their out-flows) and (ii) any imbalance that has been transferred to them up to that point (generated by other nodes the first time these other nodes obtained negative balance). Note that when nodes subsequently become positively balanced again, this imbalance is due to transfer of imbalances from other neighboring nodes; no new coefficients are generated. ■

V. Concluding Remarks

In this paper, we introduced and analyzed a distributed algorithm for obtaining an admissible and balanced flow assignment. The proposed algorithm completes in finite time when initial flows and flow limits are commensurable values; a variation of the algorithm also completes if finite time if there are no upper limits (without any commensurability requirements). In the future, we plan to investigate whether the ideas in [12] can be used to extend Algorithm 1 to communication topologies that match the physical topology or even arbitrary communication topologies.

References

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

[2] R. T. Rockafellar, *Network Flows and Monotropic Optimization*. Belmont, Massachusetts: Athena Scientific, 1998.

[3] C. N. Hadjicostis, A. D. Domínguez-García, and T. Charalambous, "Distributed averaging and balancing in network systems, with applications to coordination and control," *Foundations and Trends® in Systems and Control*, vol. 5, no. 3–4, 2018.

[4] A. I. Rikos and C. N. Hadjicostis, "Distributed balancing with constrained integer weights," *IEEE Trans. on Automatic Control*, vol. 64, no. 6, pp. 2553–2558, 2018.

[5] C. N. Hadjicostis and A. D. Domínguez-García, "Distributed balancing of commodity networks under flow interval constraints," *IEEE Trans. on Automatic Control*, vol. 64, no. 1, pp. 51–65, 2019.

[6] ——, "Distributed balancing in digraphs under interval constraints," in *Proc. of IEEE Conf. on Decision and Control*, 2016, pp. 1769–1774.

[7] C.-S. Lee, N. Michelusi, and G. Scutari, "Distributed quantized weight-balancing and average consensus over digraphs," in *Proc. of IEEE Conf. on Decision and Control*, 2018, pp. 5857–5862.

[8] 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.

[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] 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.

[11] 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.

[12] C. N. Hadjicostis and A. D. Domínguez-García, "Distributed balancing under interval flow constraints in directed communication topologies," in *Proc. of IEEE Conf. on Decision and Control*, 2017, pp. 1070–1075.