Distributed Algorithms for Control of Demand Response and Distributed Energy Resources

Alejandro D. Domínguez-García and Christoforos N. Hadjicostis

Abstract— **This paper proposes distributed algorithms for control and coordination of loads and distributed energy resources (DERs) in distribution networks. These algorithms are relevant for load curtailment control in demand response programs, and also for coordination of DERs for provision of ancillary services. Both the distributed load-curtailment and DER coordination problems can be cast as distributed resource allocation problems with constraints on resource capacity. We focus on linear iterative algorithms in which each resource j maintains a set of values that is updated to be a weighted linear combination of the resource's own previous set of values and the previous sets of values of its neighboring resources. This set of values can be used by each node to determine its own contribution to load curtailment or to resource request.**

I. Introduction and Motivation

Driven by the US-DoE SmartGrid initiative and its European counterpart, electrical energy systems are undergoing radical transformations in functionality in a quest to increase efficiency and reliability. These transformations are not only in the bulk power transmission system, but also in distribution systems, and are enabled by the integration of new technologies, such as: i) advanced communication and control; ii) integration of distributed energy resources (DERs), e.g., photovoltaics (PV); and iii) new storage-capable loads, e.g., plug-in hybrid electric vehicles (PHEVs).

Focusing on the distribution level, proper coordination and control of loads and DERs, for both generation and storage, provides more flexibility in the provision of ancillary services, which can result in enhanced efficiency and reliability. Load control is currently achieved through demand response programs in which participants, i.e., demand response resources (DRRs), sign a contract with an aggregating entity—the demand response provider—so as their load can be curtailed by the aggregator in response to market prices or in order to ensure system reliable operation, in exchange for lower electricity prices. DER control is envisioned to be achieved through similar aggregating entities that will gather together and coordinate a set of DERs to provide specific services to the grid, in exchange for monetary benefits.

In the above scenario, there are several ways in which demand response providers and DER aggregators can coordinate and control DRRs and DERs respectively[1]. In this paper, we pursue a distributed control strategy, in which the aggregator initially relays requests to a limited number of resources that it can directly communicate with. Then, through a distributed algorithm, the initial requests are disseminated to all other available resources. This dissemination process relies on a distributed linear-iterative algorithm, where each resource can exchange information with a number of other close-by resources, and subsequently makes a control decision based on locally available information.

The class of algorithms discussed in this paper is similar in spirit to distributed linear-iterative algorithms for consensus problems (see, e.g., [1], [2]); however the end goal here is very different. In consensus problems, the objective is to have a set of nodes reach agreement on a value that is function of their initial conditions. In our setup, the objective is for the nodes to converge to a value (not necessarily the same for everyone) that lies within an interval defined by upper and lower node-capacity limits, while the sum of the values is equal to the amount of resource requested by the aggregator. Another difference is the communication modality as we allow for asymmetric exchange of information, whereas in most consensus works, except for a few instances (see, e.g., [3], [4]), symmetric information exchange is assumed.

The remainder of this paper is organized as follows. Section II provides background on graph theory and poses the distributed resource coordination problem. Section III describes several algorithms that solve the distributed resource coordination problem. The performance of the algorithms is compared in Section IV through a numerical example. Concluding remarks are presented in Section V.

II. Problem Formulation and Motivation

Information exchange between resources is described by a directed graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{1, 2, \ldots, n\}$ is the vertex set (each vertex corresponds to a resource), and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, where $(j, i) \in \mathcal{E}$ if node j can receive information from node i. We assume no self-loops in $\mathcal{G}$ (i.e., $(j, j) \notin \mathcal{E}$ for all $j \in \mathcal{V}$). All nodes that

The work of A. D. Domínguez-García was supported in part by the National Science Foundation (NSF) under grant ECCS-CAR-0954420. The work of C. N. Hadjicostis was supported in part by the European Community (EC) 7^{th} Framework Programme (FP7/2007-2013), under grants INFSO-ICT-223844 and PIRG02-GA-2007-224877. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of NSF or EC.

A. 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.

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, Urbana, IL 61801, USA. E-mail: chadjic@UCY.AC.CY.

[1]Throughout the paper, we will interchangeably use the term *aggregator* to refer to a demand response provider or to a DER aggregator, and will also interchangeably use the term *resource* to refer to a DRR or DER.

can transmit information to node j are called its neighbors, and are represented by the set $\mathcal{N}_j = \{i \in \mathcal{V} : (j,i) \in \mathcal{E}\}$. The number of neighbors of j is called the in-degree of j and is denoted by $\mathcal{D}_j^-$. The number of nodes that have j as neighbor is called the out-degree of j and is denoted by $\mathcal{D}_j^+$.

Let $\pi_j[k]$ be the amount of resource requested from node j at the k round of information exchange between nodes. Let $0 < \pi_j^{min} < \pi_j^{max}$, for $j = 1, 2, \ldots, n$, be the minimum and maximum capacity that node j can provide, and define the corresponding maximum (minimum) resource capacity vector as $\pi^{max} = [\pi_1^{max},\ \pi_2^{max}, \ldots, \pi_n^{max}]'$ ($\pi^{min} = [\pi_1^{min},\ \pi_2^{min}, \ldots, \pi_n^{min}]'$). Let ρ_d, $\sum_{j=1}^n \pi_j^{min} \leq \rho_d \leq \sum_{j=1}^n \pi_j^{max}$, be the total amount of resource requested by the aggregator. The objective is to design a distributed iterative algorithm such that, at step k, each node j updates its resource request based on i) its current request, and ii) the current request of neighboring nodes that communicate with j, so that the limits $\lim_{k\to\infty} \pi_j[k] = \pi_j^{ss},\ j = 1, 2, \ldots, n$ exist and satisfy

$$\pi_j^{min} \leq \pi_j^{ss} \leq \pi_j^{max},\ \forall j, \quad \text{and} \quad \sum_{j=1}^n \pi_j^{ss} = \rho_d. \quad (1)$$

In the algorithms proposed in this paper, the graph describing the information exchange is assumed to be strongly connected, and each node j updates its resource request $\pi_j[k]$ as a function of some $\mu_j[k] = \left[\hat{\mu}_j[k], \check{\mu}_j[k]\right]'$ that each node j obtains recursively as a linear combination of its previous $\mu_j[k]$ and the previous $\{\mu_i[k] \mid i \in \mathcal{N}_j\}$ of its neighbors; specifically,

$$\mu_j[k+1] = p_{jj}[k]\mu_j[k] + \sum_{i\in\mathcal{N}_j} p_{ji}[k]\mu_i[k], \quad (2)$$

where the $p_{ji}[k]$'s are a set of time-varying weights chosen such that for every k, $\sum_{i=1}^n p_{ij}[k] = 1,\ \forall j$; and $p_{ji}[k] > 0$ if $i \in \mathcal{N}_j$ and $p_{ji}[k] = 0$ otherwise. Additionally, there is at least node i for which $p_{jj}[k] > 0$. Each node obtains the value of $\pi_j[k]$ from $\mu_j[k]$ via

$$\pi_j[k] = f_j(\mu_j[k]), \quad (3)$$

for some function $f_j : \mathbb{R}^{+2} \mapsto \mathbb{R}^+$ to be described.

In the proofs of the algorithms discussed in the subsequent, the following lemma from [5] plays a fundamental role.

Lemma 1: Let P be a primitive column stochastic matrix with diagonal entries $p_{jj},\ j = 1, 2, \ldots, n$, and let $v = \pi^{ss} = [\pi_1^{ss}, \pi_2^{ss}, \ldots, \pi_n^{ss}]' > 0$ be the unique solution to $v = Pv$, normalized so that $\sum_{j=1}^n \pi_j^{ss} = \rho$. Let $\Delta = \text{diag}\big(\delta_1, \delta_2, \ldots, \delta_n\big)$ be a diagonal matrix with $0 < \delta_j \leq \frac{1}{1-p_{jj}},\ \forall j = 1, 2, \ldots, n$, with at least one $i \in \{1, 2, \ldots, n\}$ such that $0 < \delta_i < \frac{1}{1-p_{ii}}$, and define $\hat{P} = P\Delta + (I - \Delta)$, where I is the identity matrix. Then, (i) $\hat{P}$ is a primitive column stochastic matrix, and (ii) if $\hat{v} = \hat{\pi}^{ss} = [\hat{\pi}_1^{ss},\ \hat{\pi}_2^{ss}, \ldots, \hat{\pi}_n^{ss}]' > 0$ is the unique solution to $\hat{v} = \hat{P}\hat{v}$, normalized so that $\sum_{j=1}^n \hat{\pi}_j^{ss} = \rho$, we have

$$\delta_j \hat{\pi}_j^{ss} = \alpha \pi_j^{ss}, \forall j = 1, 2, \ldots, n,$$

for some constant $\alpha > 0$.

III. Algorithms for Distributed Allocation

We formulate three algorithms that solve the distributed resource allocation problem. The first one uses constant weights, while the other two update their weights based on the algorithm progress. It is assumed that the aggregator (who knows the amount of resource ρ_d that needs to be collectively provided by the nodes) can communicate with $l \geq 1$ nodes, and initially sends out to each one of them a command demanding ρ_d/l units of resource. Then, $\hat{\mu}_j[0]$ is set to $x_j = \rho_d/l$ if j is a neighbor of the aggregator and $x_j = 0$ otherwise. Additionally, each node sets $\check{\mu}_j[0] = \pi_j^{max} - \pi_j^{min} > 0$.

A. Algorithm 1

This algorithm uses constant weights for iteration (2). Specifically, each node j sets its weights to be $p_{jj} = 1/(1 + \mathcal{D}_j^+)$ and $p_{ij} = 1/(1 + \mathcal{D}_j^+)$ for all i that have j as a neighbor, i.e., all i such that $j \in \mathcal{N}_i$. As a result, each node will be updating its value as

$$\mu_j[k+1] = \frac{1}{1+\mathcal{D}_j^+}\mu_j[k] + \sum_{i\in\mathcal{N}_j} \frac{1}{1+\mathcal{D}_i^+}\mu_i[k], \quad (4)$$

where $\mathcal{D}_i^+$ is the number of nodes that i can transmit information to (the out-degree of node i). Then, provided the directed graph describing the communication modality between nodes is strongly connected, a simple algorithm for solving the distributed resource allocation problem is given in the following Lemma.

Lemma 2: Let $\mu_j[k] = [\hat{\mu}_j[k], \check{\mu}_j[k]]',\ \forall j$, be the result of iteration (4) with initial conditions $\hat{\mu}_j[0] = x_j - \pi_j^{min}$ and $\check{\mu}_j[0] = \pi_j^{max} - \pi_j^{min}$. Then, a solution to the distributed resource allocation problem can be asymptotically achieved as $\lim_{k\to\infty} \pi_j[k]$, where

$$\pi_j[k] = \pi_j^{min} + \frac{\hat{\mu}_j[k]}{\check{\mu}_j[k]}(\pi_j^{max} - \pi_j^{min}). \quad (5)$$

Proof: Define $\hat{\mu}[k] = [\hat{\mu}_1[k], \hat{\mu}_2[k], \ldots, \hat{\mu}_n[k]]',\ \forall k \geq 0$, and $\check{\mu}[k] = [\check{\mu}_1[k], \check{\mu}_2[k], \ldots, \check{\mu}_n[k]]',\ \forall k \geq 0$. Then, we can rewrite (4) in matrix form as

$$\hat{\mu}[k+1] = P_c\hat{\mu}[k],\ \hat{\mu}[0] = x - \pi^{min}, \quad (6)$$
$$\check{\mu}[k+1] = P_c\check{\mu}[k],\ \check{\mu}[0] = \pi^{max} - \pi^{min}, \quad (7)$$

where $x = [x_1, x_2, \ldots, x_n]'$. By construction, it is easy to see that P_c is a primitive column stochastic matrix. Then, the Perron-Frobenius theorem (see, e.g., [6]) states that P_c has a unique eigenvalue with largest modulus at $\lambda_1 = 1$.

Let v be the right eigenvector of P_c associated with λ_1 and let w be the left eigenvector of P_c associated with λ_1, normalized so that $v'w = 1$. From the fact that P_c is column stochastic, the entries of the vector w must be all equal. Without loss of generality, let $w = [1, 1, \ldots, 1]'$, and since $v'w = 1$, the entries of v must add up to one. Let $\hat{\mu}^{ss}$ and $\check{\mu}^{ss}$ be the steady-state solutions of (6) and (7) respectively. By

Perron-Frobenius, we have that $\lim_{k\to\infty} P_c^k = vw'$, therefore

$$\hat{\mu}^{ss} = vw'\hat{\mu}[0] = \big(\sum_{j=1}^{n}(x_j - \pi_j^{min})\big)v = \big(\rho_d - \sum_{j=1}^{n}\pi_j^{min}\big)v,$$
$$\check{\mu}^{ss} = vw'\check{\mu}[0] = \big(\sum_{j=1}^{n}(\pi_j^{max} - \pi_j^{min})\big)v, \quad (8)$$

from which it follows that

$$\lim_{k\to\infty}\pi_j[k] = \lim_{k\to\infty}\big(\pi_j^{min} + \frac{\hat{\mu}_j[k]}{\check{\mu}_j[k]}(\pi_j^{max} - \pi_j^{min})\big)$$
$$= \pi_j^{min} + \frac{\hat{\mu}_j^{ss}}{\check{\mu}_j^{ss}}(\pi_j^{max} - \pi_j^{min}),\ \forall j. \quad (9)$$

where $0 \le \frac{\hat{\mu}_j^{ss}}{\check{\mu}_j^{ss}} = \frac{\rho_d - \sum_{j=1}^{n}\pi_j^{min}}{\sum_{j=1}^{n}(\pi_j^{max} - \pi_j^{min})} \le 1$. From (9), it is obvious that (1) holds. It is important to note that $\frac{\hat{\mu}_j[k]}{\check{\mu}_j[k]}$ is a finite quantity for all $k \ge 0$ as $\check{\mu}_j[k] > 0,\ \forall k \ge 0$. ■

B. Algorithm 2

In this algorithm, and following the notation of (2), each node j will periodically update its request after a fixed number of iterations k_0 have elapsed, for k_0 sufficiently large. Let $k = rk_0,\ r = 1, 2, \ldots$, be the instants at which updates occur; then, we define

$$\pi_j^r = \pi_j^{min} + [1\ 0]\mu_j^r, \quad (10)$$

where $\mu_j^r \equiv \big[\hat{\mu}_j[rk_0], \check{\mu}_j[rk_0]\big]'$.

Also every k_0 steps, a node j will adjust the weights over which it has control (i.e, $p_{jl}[rk_o] = p_{jl}[rk_o + 1] = \cdots = p_{jl}[(r+1)k_o - 1]$ for all l such that $j \in \mathcal{N}_l$) based on its own μ_j^r and the $\{\mu_i^r \mid i \in \mathcal{N}_j\}$ of its neighbors. We use the term *super-iteration* to distinguish between the updates every k_0 steps and the k_0 iterative updates each node conducts with a fixed weight matrix. We focus on the iterative procedure by which the weights are updated, which is solely governed by evolution of $\check{\mu}_j^r = \check{\mu}_j[rk_0],\ r = 1, 2, \ldots$.

Initially, we start with a weight matrix that has each node j equally distribute its initial value $\check{\mu}_j[0] = \pi_j^{max} - \pi_j^{min}$ among itself and the nodes that have j as neighbor. Then, the resulting iteration is of the form in (4). Define $\check{\mu}[k] = [\check{\mu}_1[k], \check{\mu}_2[k], \ldots, \check{\mu}_n[k]]',\ \forall k \ge 0$; we can then write

$$\check{\mu}[k+1] = P_0\check{\mu}[k],\ \check{\mu}[0] = \pi^{max} - \pi^{min}, \quad (11)$$

where $P_0 = P_c$ (as defined in (7)). Note that matrix P_0 can also be rewritten as

$$P_0 = \overline{P}\Delta_0 + (I - \Delta_0), \quad (12)$$

where $\Delta_0 = \text{diag}\big(\delta_1^0, \delta_2^0, \ldots, \delta_j^0, \ldots, \delta_n^0\big)$ is a diagonal matrix with nonzero entries $\delta_j^0 = \frac{\mathcal{D}_j^+}{1+\mathcal{D}_j^+}$ satisfying[2] $\frac{1}{2} \le \delta_j^0 < 1,\ \forall j = 1, 2, \ldots, n$, and $\overline{P}$ is the matrix used for updating the values when each node distributes its own value equally among its neighbors and does not keep anything for itself. Note that the diagonal entries of matrix $\overline{P}$ are zero and, for a particular column, the nonzero entries are equal and add up to one (this implies that P_0 is the same matrix as in (7)). When (11) reaches steady-state, denoted by $\check{\mu}^0$ (we assume that k_0 is large enough so that this occurs after k_0 iterations), the weight matrix that governs node updates is changed to

$$P_1 = \overline{P}\Delta_1 + (I - \Delta_1), \quad (13)$$

where the j^{th} diagonal entry of $\Delta_1 = \text{diag}\big(\delta_1^1, \delta_2^1, \ldots, \delta_j^1, \ldots, \delta_n^1\big)$ is updated based on the corresponding entry of Δ_0 as follows:

$$\delta_j^1 = \begin{cases} \frac{\check{\mu}_j^0}{\Delta\pi_j}\delta_j^0, & \text{if } \check{\mu}_j^0 \le \Delta\pi_j, \\ 1 - \frac{\Delta\pi_j}{\check{\mu}_j^0}\big(1 - \delta_j^0\big), & \text{if } \check{\mu}_j^0 > \Delta\pi_j, \end{cases} \quad (14)$$

where $\Delta\pi_j := \pi_j^{max} - \pi_j^{min}$. By construction, it is easy to see that P_1 is a primitive column stochastic matrix (at least as long as $0 < \delta_j^1 \le 1,\ \forall j = 1, 2, \ldots, n$, and there exists at least one i for which $\delta_i^1 < 1$). Once P_1 is obtained, the nodes run $\check{\mu}[k+1] = P_1\check{\mu}[k]$, where the initial conditions are set to $\check{\mu}[0] = \pi^{max} - \pi^{min}$ until a new steady-state, denoted by $\check{\mu}^1$, is reached. The matrix $\Delta_2 = \text{diag}\big(\delta_1^2, \delta_2^2, \ldots, \delta_j^2, \ldots, \delta_n^2\big)$ is then used to obtain matrix $P_2 = \overline{P}\Delta_2 + (I - \Delta_2)$ with δ_j^2 updated based on δ_j^1 and $\check{\mu}_j^1$ as in (14), with superscript 1 replaced by 2 and superscript 0 replaced by 1.

This process continues in this fashion, and we can define an iterative relation for the diagonal matrix $\Delta_r = \text{diag}\big(\delta_1^r, \delta_2^r, \ldots, \delta_j^r, \ldots, \delta_n^r\big)$ that defines the weight matrix $P_r = \overline{P}\Delta_r + (I - \Delta_r)$ at the r^{th} super-iteration. If we denote the j^{th} diagonal entry of Δ_r by δ_j^r, then we have the following relationship:

$$\delta_j^r = \begin{cases} \frac{\check{\mu}_j^{r-1}}{\Delta\pi_j}\delta_j^{r-1}, & \text{if } \check{\mu}_j^{r-1} \le \Delta\pi_j, \\ 1 - \frac{\Delta\pi_j}{\check{\mu}_j^{r-1}}\big(1 - \delta_j^{r-1}\big), & \text{if } \check{\mu}_j^{r-1} > \Delta\pi_j. \end{cases} \quad (15)$$

Theorem 2 establishes that the algorithm described in (11)—(15) converges to a steady-state weight matrix P_{ss} that solves the allocation problem. To prove this theorem, we first need Theorem 1 and Corollary 1, which are stated next.

Theorem 1: Let $P_0 = P_c$ be the initial update matrix of iteration (11) (at the 0^{th} super-iteration), where P_0 can be written in terms of a diagonal matrix Δ_0 and a matrix $\overline{P}$ as described in (12). Let $\check{\mu}^0$ be the solution to $v = P_0 v$ normalized so that $\sum_{j=1}^{n}\check{\mu}_j^0 = \sum_{j=1}^{n}\Delta\pi_j$. Let $P_r = \overline{P}\Delta_r + (I - \Delta_r)$, where Δ_r is a diagonal matrix recursively obtained as described in (15), and let $v = \check{\mu}^r$ be the solution to $v = P_r v$ normalized so that $\sum_{j=1}^{n}\check{\mu}_j^r = \sum_{j=1}^{n}\Delta\pi_j$. Then, the following hold: i) If $\check{\mu}_j^r > \Delta\pi_j$ for some $r \ge 0$, then $\check{\mu}_j^{r+1} < \check{\mu}_j^r$ and $\delta_j^{r+1} > \delta_j^r \ge 1/2$; and ii) If $\check{\mu}_j^r \le \Delta\pi_j$ for some $r \ge 0$, then $\check{\mu}_j^{r+l} \le \Delta\pi_j, \forall l > 0$.

Proof: By induction. First, we consider the base case $r = 0$ by analyzing what happens at the end of the 0^{th} super-iteration. We partition the set $\mathcal{I} = \{1, 2, \ldots, n\}$ that indexes the nodes into two disjoint sets as follows: $\mathcal{I} = \mathcal{A}_0 \bigcup \mathcal{B}_0$, where $\check{\mu}_j^0 > \Delta\pi_j, \forall j \in \mathcal{A}_0$, and $\check{\mu}_j^0 \le \Delta\pi_j, \forall j \in \mathcal{B}_0$. We will establish that after the weight matrix is updated according to (13), it results in $\check{\mu}_j^1 < \check{\mu}_j^0, \forall j \in \mathcal{A}_0$, and $\check{\mu}_j^1 \le \Delta\pi_j, \forall j \in \mathcal{B}_0$.

[2]Initially, each node splits its value equally among itself and its neighbors. The strong connectivity of the graph implies that any given node can send its value to at least one other node, thus $\mathcal{D}_j^+ \ge 1$, and therefore $\delta_j^0 \ge \frac{1}{2}$.

Let $P_0 = P_c = \overline{P}\Delta_0 + (I - \Delta_0)$ and $P_1 = \overline{P}\Delta_1 + (I - \Delta_1)$, where Δ_0 and Δ_1 are the diagonal matrices in (12) and (13) respectively. We can write $P_1 = P_0\Delta_0^{-1}\Delta_1 + (I - \Delta_0^{-1}\Delta_1)$, and it follows from Lemma 1 that $\breve{\mu}^1 = \alpha^1\Delta_1^{-1}\Delta_0\breve{\mu}^0$ for some $\alpha^1 > 0$ (note that the requirements for Lemma 1 are satisfied: $0 < \frac{\delta_j^1}{\delta_j^0} \leq 1 < \frac{1}{1-P_0(j,j)}$ for $j \in \mathcal{B}_0$, and $0 < \frac{\delta_j^1}{\delta_j^0} < \frac{1}{\delta_j^0} = \frac{1}{1-(1-\delta_j^0)} = \frac{1}{1-P_0(j,j)}$ for $j \in \mathcal{A}_0$). Since P_0 is column stochastic, P_1 is also column stochastic and thus $\sum_{j=1}^n \breve{\mu}_j^1 = \sum_{j=1}^n \Delta\pi_j$. Since $\sum_{j=1}^n \breve{\mu}_j^1 = \alpha^1 \sum_{j=1}^n \frac{\delta_j^0}{\delta_j^1}\breve{\mu}_j^0$, it follows that

$$\alpha^1 = \frac{\sum_{j=1}^n \Delta\pi_j}{\sum_{j=1}^n \frac{\delta_j^0}{\delta_j^1}\breve{\mu}_j^0}. \tag{16}$$

Now, we show that α^1 is smaller than or equal to one. Recall from (14) that $\delta_j^1/\delta_j^0 = \breve{\mu}_j^0/\Delta\pi_j \leq 1,\ \forall j \in \mathcal{B}_0$, and $(1 - \delta_j^1)/(1-\delta_j^0) = \Delta\pi_j/\breve{\mu}_j^0 < 1,\ \forall j \in \mathcal{A}_0$. Then, we can rewrite the denominator in (16) as follows

$$\sum_{j=1}^n \frac{\delta_j^0}{\delta_j^1}\breve{\mu}_j^0 = \sum_{j\in\mathcal{B}_0} \Delta\pi_j + \sum_{j\in\mathcal{A}_0} \frac{\delta_j^0(1-\delta_j^0)}{\delta_j^1(1-\delta_j^1)}\Delta\pi_j. \tag{17}$$

Since $\delta_j^0 \geq 1/2$ and $\delta_j^1 > \delta_j^0,\ \forall j \in \mathcal{A}_0$, it follows that $\frac{\delta_j^0(1-\delta_j^0)}{\delta_j^1(1-\delta_j^1)} > 1,\ \forall j \in \mathcal{A}_0$, so that

$$\alpha^1 \leq \frac{\sum_{j=1}^n \Delta\pi_j}{\sum_{j\in\mathcal{A}_0} \Delta\pi_j + \sum_{j\in\mathcal{B}_0} \Delta\pi_j} = 1, \tag{18}$$

where the equality is possible only if $\mathcal{A}_0 = \emptyset$. Now, we have:

1) For $j \in \mathcal{A}_0$, $\breve{\mu}_j^1 = \alpha^1 \frac{\delta_j^0}{\delta_j^1}\breve{\mu}_j^0$. Since $\alpha^1 \leq 1$ and $\frac{\delta_j^0}{\delta_j^1} < 1$, it follows that $\breve{\mu}_j^1 < \breve{\mu}_j^0$.
2) For $j \in \mathcal{B}_0$, $\breve{\mu}_j^1 = \alpha^1 \frac{\delta_j^0}{\delta_j^1}\breve{\mu}_j^0 = \alpha^1\Delta\pi_j$. Since $\alpha^1 \leq 1$, it follows that $\breve{\mu}_j^1 \leq \Delta\pi_j$.

The inductive step, omitted for brevity, proceeds in a very similar fashion to the developments above and it results in

$$\alpha^{r+1} = \frac{\sum_{j=1}^n \Delta\pi_j}{\sum_{j=1}^n \frac{\delta_j^r}{\delta_j^{r+1}}\breve{\mu}_j^r} \leq 1. \tag{19}$$

■

Corollary 1: Let P_r be the update matrix at the r^{th} super-iteration of the recursive algorithm described in (11)—(15), and let $v = \breve{\mu}^r$ be the steady-state solution of $v = P_r v$ normalized so that $\sum_{j=1}^n \breve{\mu}_j^r = \sum_{j=1}^n \Delta\pi_j$. If $\breve{\mu}_j^0 > \Delta\pi_j$, then $\breve{\mu}_j^r$ either becomes smaller than $\Delta\pi_j$ for some finite r (and subsequently remains below $\Delta\pi_j$) or converges to $\Delta\pi_j$ with worst-case convergence given by

$$\breve{\mu}_j^{r+1} \leq \frac{1}{\theta^{r+1} + \frac{1}{\sum_{j=1}^n \Delta\pi_j}\frac{1-\theta^{r+1}}{1-\theta}}\breve{\mu}_j^0, \tag{20}$$

where $\theta_j = 1 - \frac{\Delta\pi_j}{\sum_{i=1}^n \Delta\pi_i}$.

The key idea in the proof (which we omit for brevity) is to see that, for every $j \in \mathcal{A}_r$, the denominator of (19) can be written as $\sum_{i=1}^n \frac{\delta_i^r}{\delta_i^{r+1}}\breve{\mu}_i^r = \sum_{i\in\mathcal{B}_r} \Delta\pi_i + \sum_{i\in\mathcal{A}_r, i\neq j} \frac{\delta_i^r(1-\delta_i^r)}{\delta_i^{r+1}(1-\delta_i^{r+1})}\Delta\pi_i + \frac{\delta_j^r}{\delta_j^{r+1}}\breve{\mu}_j^r$. Then, using an argument similar to the one used in the proof of Theorem 1, it can be established that

$$\alpha^{r+1} \leq \frac{\sum_{i=1}^n \Delta\pi_i}{\sum_{i=1}^n \Delta\pi_i + \frac{\delta_j^r}{\delta_j^{r+1}}\breve{\mu}_j^r - \Delta\pi_j},\ \forall j \in \mathcal{A}_r, \tag{21}$$

which is then used to establish the result in (20).

The next theorem establishes that the algorithm described in (11)—(15) converges to a solution where all the nodes reach a set of weights that solves the allocation problem.

Theorem 2: Let $P_r = \overline{P}\Delta_r + (I - \Delta_r)$ be the update matrix at the r^{th} super-iteration of the recursive algorithm described in (11)—(15), and let $v = \pi^r$ be the steady-state solution of $v = P_r v$ normalized so that $\sum_{j=1}^n \breve{\mu}_j^r = \sum_{j=1}^n \Delta\pi_j$. Then, the following hold:

1) $\lim_{r\to\infty} \breve{\mu}_j^r = \Delta\pi_j,\ \forall j = 1, 2, \ldots, n$;
2) $0 < \delta_j^r < 1, \forall r \geq 0,\ \forall j = 1, 2, \ldots, n$;
3) $\Delta_{ss} = \text{diag}(\delta_1^{ss}, \delta_2^{ss}, \ldots, \delta_n^{ss})$, where $\delta_j^{ss} = \lim_{r\to\infty} \delta_j^r,\ \forall j = 1, 2, \ldots, n$, exist;
4) $\delta_i^{ss} < 1$ for some i, and $0 < \delta_j^{ss} \leq 1,\ \forall j = 1, 2, \ldots, n$.

Proof: Due to space constraints, we will only prove the first three conclusions. Conclusion (1) can be proved as follows. For every $j \in \mathcal{A}_0$, we take the limit of (20) as $r \to \infty$, from where it follows that

$$\lim_{r\to\infty} \breve{\mu}_j^{r+1} \leq \lim_{r\to\infty} \frac{1}{\theta^{r+1} + \frac{1}{\sum_{i=1}^n \Delta\pi_i}\frac{1-\theta^{r+1}}{1-\theta}}\breve{\mu}_j^0 = \Delta\pi_j. \tag{22}$$

Now, we know that either (i) (22) holds in the limit for all $j \in \mathcal{A}_0$ or (ii) at some finite r the value of $\breve{\mu}_j^r$ becomes smaller or equal to one (and remains such). Either situation ensures that $\lim_{r\to\infty} \sum_{i\in\mathcal{A}_0} \breve{\mu}_i^r \leq \sum_{i\in\mathcal{A}_0} \Delta\pi_i$. This observation together with the fact that P_r is column stochastic for every $r = 1, 2, \ldots$, which ensures that $\sum_{j=1}^n \breve{\mu}_j^r = \sum_{j=1}^n \Delta\pi_j$ for every $r = 1, 2, \ldots$, implies that $\lim_{r\to\infty} \sum_{j\in\mathcal{B}_0} \breve{\mu}_j^r \geq \sum_{j\in\mathcal{B}_0} \Delta\pi_j$. However, Theorem 1 established that if $\breve{\mu}_j^0 \leq \Delta\pi_j$, then $\breve{\mu}_j^r \leq \Delta\pi_j, \forall r > 0$; thus, $\lim_{r\to\infty} \sum_{j\in\mathcal{B}_0} \breve{\mu}_j^r = \sum_{j\in\mathcal{B}_0} \Delta\pi_j$ and also $\lim_{r\to\infty} \sum_{j\in\mathcal{A}_0} \breve{\mu}_j^r = \sum_{j\in\mathcal{A}_0} \Delta\pi_j$. The above leads to $\lim_{r\to\infty} \breve{\mu}_j^r = \Delta\pi_j,\ \forall j$.

Conclusion (2) is proved by induction. Since $0 < \delta_j^0 < 1, \forall j = 1, 2, \ldots, n$, and $\breve{\mu}_j^0 > 0, \forall j = 1, 2, \ldots, n$, it follows from (14) that $0 < \delta_j^1 < 1,\ \forall j = 1, 2, \ldots, n$. Assuming that $0 < \delta_j^{r-1} < 1, \forall j$, and since $\breve{\mu}_j^{r-1} > 0, \forall j = 1, 2, \ldots, n$, it follows from (15) that $0 < \delta_j^r < 1,\ \forall j$.

Conclusion (3) is proved as follows. The existence of $\delta_j^{ss},\ \forall j$, follows from Conclusions (1) and (2), and the update rule in (15). Since $\lim_{r\to\infty} \breve{\mu}_j^r = \Delta\pi_j,\ \forall j$; $\delta_j^r \neq 0, \forall j, \forall r \geq 0$; and $1 - \delta_j^r \neq 0, \forall j, \forall r \geq 0$, then, for some $r = r_0 > 0$

$$\lim_{r\to\infty} \frac{\delta_j^{r_0+r+1}}{\delta_j^{r_0+r}} = \lim_{r\to\infty} \breve{\mu}_j^{r_0+r} = \Delta\pi_j,\ \forall j \in \mathcal{B}_{r_0},$$

$$\lim_{r\to\infty} \frac{1-\delta_j^{r_0+r}}{1-\delta_j^{r_0+r+1}} = \lim_{r\to\infty} \breve{\mu}_j^{r_0+r} = \Delta\pi_j,\ \forall j \in \mathcal{A}_{r_0}, \tag{23}$$

from where it follows that $\lim_{r\to\infty} \delta_j^r = \delta_j^{ss}$ for some δ_j^{ss} for all $j = 1, 2, \ldots, n$. ■

It is left to show that (10) solves the allocation problem asymptotically. This follows from the fact that (10) can be written as $\pi_j[k] = \pi_j^{min} + \hat{\mu}[k]$. The evolution of $\hat{\mu}[k]$ is governed by the same transition matrix $P_r,\ r = 0, 1, \ldots,$ as the evolution of $\check{\mu}[k]$. Theorem (2) established that P_r converges to a column-stochastic primitive matrix such that its eigenvector associated with the largest modulus eigenvalue is aligned with $[\Delta\pi_1, \Delta\pi_2, \ldots, \Delta\pi_n]'$. Since $\hat{\mu}_j[0] = x_j - \pi_j^{min}$, then $\sum_{j=1}^n \hat{\mu}_j[0] = \rho_d - \sum_{j=1}^n \pi_j^{min}$, and since $\sum_{j=1}^n \hat{\mu}_j[0] = \sum_{j=1}^n \hat{\mu}_j[k],\ \forall k > 0$, it follows that $\lim_{k\to\infty} \hat{\mu}[k] = \frac{\rho_d - \sum_{j=1}^n \pi_j^{min}}{\sum_{j=1}^n \Delta\pi_j}[\Delta\pi_1, \Delta\pi_2, \ldots, \Delta\pi_n]'$. Then, it is easy to check that (1) holds.

C. Algorithm 3

This algorithm is similar to the one described above but each "super-iteration" only involves one update, i.e., it is now a regular iteration. In this algorithm, and following the notation of (2), each node j will update its request as

$$\pi_j[k] = \pi_j^{min} + [1\ 0]\mu_j[k], \tag{24}$$

and will adjust the weights over which it has control (i.e, $p_{lj}[k]$ for all l such that $j \in \mathcal{N}_l$) based on its own $\check{\mu}_j[k]$ and the $\check{\mu}_i[k]$ of its neighbors (i.e., $i \in \mathcal{N}_j$). We focus on the iterative procedure by which the weights are updated, which is solely governed by the evolution of $\check{\mu}[k]$.

Initially, for $k = 0$, each node j chooses the weights on its out-going links so that it distributes its value $\check{\mu}_j[0] = \pi_j^{max} - \pi_j^{min}$ equally among itself and its neighbors. This results in each node updating its value to $\check{\mu}_j[k]$ as in (7), which can be rewritten in matrix form as

$$\check{\mu}[1] = P[0]\check{\mu}[0],\ \check{\mu}[0] = \pi^{max} - \pi^{min}. \tag{25}$$

The matrix $P[0] = P_c$ can be rewritten as

$$P[0] = \overline{P}\Delta[0] + (I - \Delta[0]), \tag{26}$$

where $\Delta[0]$ is a diagonal matrix with nonzero entries $\delta_j[0] = \frac{\mathcal{D}_j^+}{1+\mathcal{D}_j^+},\ j = 1, 2, \ldots, n$, and $\overline{P}$ is the weight matrix where each node distributes its value equally among its neighbors without keeping anything for itself.

After the first round of exchanges, $k = 1$, each node j will update the weights that it uses to distribute its value among itself and the nodes that j sends information to. The matrix associated to the new weights will be of the form $P[1] = \overline{P}\Delta[1] + (I - \Delta[1])$, where $\Delta[1] = \text{diag}\big(\delta_1[1], \delta_2[1], \ldots, \delta_j[1], \ldots, \delta_n[1]\big)$ is a diagonal matrix. Define $\rho_j[0] = \frac{\sum_{i\in\mathcal{N}_j\cup\{j\}} p_{ji}[0]\Delta\pi_i}{\Delta\pi_j}$, with $\Delta\pi_i = \pi_i^{max} - \pi_i^{min},\ \forall i \in \mathcal{N}_j \cup \{j\}$. Then, the $\delta_j[1]$'s are chosen as follows:

$$\delta_j[1] = \begin{cases} \delta_j[0]\rho_j[0], & \text{if } \rho_j[0] \le 1, \\ 1 - \frac{1}{\rho_j[0]}\big(1 - \delta_j[0]\big), & \text{if } \rho_j[0] > 1. \end{cases} \tag{27}$$

By construction, $P[1]$ is a primitive column stochastic matrix (refer to the reasoning in the proof of Lemma 1). This process continues and, for any $k \ge 0$, the nodes will update their value according to

$$\check{\mu}[k+1] = P[k]\check{\mu}[k], \tag{28}$$

where $P[k] = \overline{P}\Delta[k] + (I - \Delta[k])$, and $\Delta[k] = \text{diag}\big(\delta_1[k], \delta_2[k], \ldots, \delta_j[k], \ldots, \delta_n[k]\big)$ with

$$\delta_j[k] = \begin{cases} \delta_j[k-1]\rho_j[k-1], & \text{if } \rho_j[k-1] \le 1, \\ 1 - \frac{1}{\rho_j[k-1]}\big(1 - \delta_j[k-1]\big), & \text{if } \rho_j[k-1] > 1, \end{cases} \tag{29}$$

and $\rho_j[k-1] = \frac{\sum_{i\in\mathcal{N}_j\cup\{j\}} p_{ji}[k-1]\Delta\pi_i}{\Delta\pi_j}$.

The idea behind the weight update described in (29) is to drive the entries of the weight matrix to a steady-state matrix P_{ss} so that, in the limit as k goes to infinity, the right eigenvector u associated with the (unique) eigenvalue with largest modulus at $\lambda_1 = 1$ is such that $u = \gamma(\pi^{max} - \pi^{min}) \equiv \gamma\Delta\pi$, for some $\gamma > 0$. Then, taking into account the fact that the sequence $P[0]$, $P[1]$, ..., consists of column stochastic (and primitive) matrices that converge to a steady state P_{ss}, the steady-state solution of

$$\hat{\mu}[k+1] = P[k]\hat{\mu}[k],\ \hat{\mu}[0] = x - \pi^{min}, \tag{30}$$

is that is such that

$$\hat{\mu}_j^{ss} = \frac{\rho_d - \sum_{j=1}^n \pi_j^{min}}{\sum_{j=1}^n \Delta\pi_i}\Delta\pi_j. \tag{31}$$

Then, from (24), it is clear that (1) holds.

Note that $\rho_j[k-1] = \sum_{i\in\mathcal{N}_j\cup\{j\}} p_{ji}[k-1]\Delta\pi_i$ can be replaced by $\sum_{i=1}^n p_{ji}[k-1]\Delta\pi_i$ (since $p_{ji}[0] = 0$ for $i \notin \mathcal{N}_j \cup \{j\}$). It was written in this fashion in (29) to emphasize the fact that the update of node j is based purely on locally available information.

The result in (31) follows from Theorem 3 stated below, where we prove that indeed the weight matrix $P[k]$ reaches a limiting matrix P_{ss} that is column stochastic and primitive, and whose eigenvector associated with the largest modulus eigenvalue is aligned with $[\Delta\pi_1, \Delta\pi_2, \ldots, \Delta\pi_n]'$. To prove this theorem, we first need the following two lemmas.

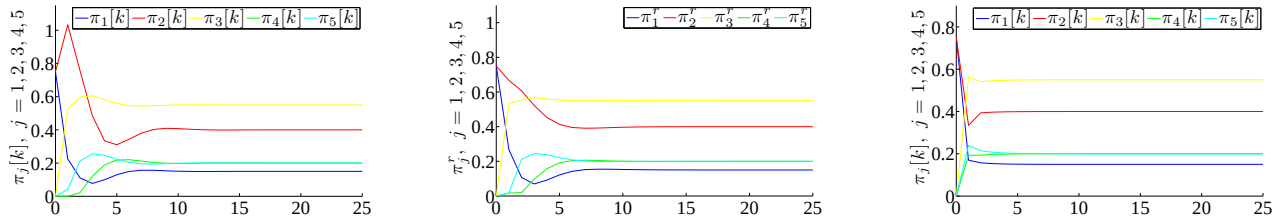
Lemma 3: Let $P[k-1]$ and $P[k]$ be the update matrices in (28) at steps $k-1$ and k respectively, and assume that the underlying connectivity graph associated with $P[k-1]$ (and thus $P[k]$) is strongly connected. Then, the following hold:

1) If $\sum_i p_{ji}[k-1]\Delta\pi_i \le \Delta\pi_j$, then $\sum_i p_{ji}[k-1]\Delta\pi_i \le \sum_i p_{ji}[k]\ \Delta\pi_i \le \Delta\pi_j$;
2) If $\sum_i p_{ji}[k-1]\Delta\pi_i > \Delta\pi_j$, then $\sum_i p_{ji}[k-1]\Delta\pi_i > \sum_i p_{ji}[k]\Delta\pi_i > \Delta\pi_j$.

Proof: If $\sum_i p_{ji}[k-1]\Delta\pi_i \le \Delta\pi_j$, then $\delta_j[k] = \delta_j[k-1]\frac{\sum_i p_{ji}[k-1]\Delta\pi_i}{\Delta\pi_j}$, from where it follows that $p_{jj}[k] = 1 - (1 - p_{jj}[k-1])\frac{\sum_i p_{ji}[k-1]\Delta\pi_i}{\Delta\pi_j}$. Since $\delta_i[k-1] = \delta_i[k], \forall i \ne j$, it follows that $p_{ji}[k-1] = p_{ji}[k],\ \forall i \ne j$. Then, since $\sum_i p_{ji}[k-1]\Delta\pi_i - \Delta\pi_j \le 0$, it can be shown that $\sum_i p_{ji}[k]\Delta\pi_i = p_{jj}[k]\Delta\pi_j + \sum_{i\in\mathcal{N}_j} p_{ji}[k]\Delta\pi_i$ satisfies $\sum_i p_{ji}[k]\Delta\pi_i = \Delta\pi_j + p_{jj}[k-1]\left(\sum_i p_{ji}[k-1]\Delta\pi_i - \Delta\pi_j\right) \le \Delta\pi_j$. Also, since $p_{ji}[k-1] = p_{ji}[k],\ \forall i \ne j$ and $p_{jj}[k] \ge p_{jj}[k-1]$, we have $\sum_i p_{ji}[k-1]\Delta\pi_i \le \sum_i p_{ji}[k]\Delta\pi_i$.

A similar argument, which we omit, can be used to show that $\sum_i p_{ji}[k-1]\Delta\pi_i > \sum_i p_{ji}[k]\Delta\pi_i$. ■

Define $\varepsilon[k] = \sum_{l=1}^n \varepsilon_l[k]$ where $\varepsilon_l[k] = |\sum_i p_{li}[k]\Delta\pi_i - \Delta\pi_l|$. Note that both $\varepsilon_l[\cdot]$ and $\varepsilon[\cdot]$ are nonnegative.



(a) (b) (c)

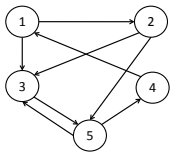
Fig. 1. Algorithms 1, 2, and 3 progress.

Lemma 4: Let $P[k-1]$ and $P[k]$ be the update matrices in (28) at steps $k-1$ and k respectively, and assume that the underlying connectivity graph associated with $P[k-1]$ (and thus $P[k]$) is strongly connected. Then, it follows that $\varepsilon[k] \leq \varepsilon[k-1]$.

Proof: Assume that $\sum_i p_{ji}[k-1]\Delta\pi_i \leq \Delta\pi_j$; it follows from Lemma 3 that $\sum_i p_{ji}[k]\Delta\pi_i \leq \Delta\pi_j$; and from (29) that $p_{jj}[k] \geq p_{jj}[k-1]$. Then, $\varepsilon_j[k] = \Delta\pi_j - \sum_i p_{ji}[k]\Delta\pi_i = \Delta\pi_j - p_{jj}[k]\Delta\pi_j + \sum_{i\in\mathcal{N}_j} p_{ji}[k-1]\Delta\pi_i \leq \Delta\pi_j - p_{jj}[k-1]\Delta\pi_j + \sum_{i\in\mathcal{N}_j} p_{ji}[k-1]\Delta\pi_i = \varepsilon_j[k-1]$. If we let $p_{jj}[k] = p_{jj}[k-1] + \Delta p_{jj}[k]$ for some positive $\Delta p_{jj}[k]$, it follows that $\varepsilon_j[k] - \varepsilon_j[k-1] = -\Delta p_{jj}[k]\Delta\pi_j$ and also that $p_{lj}[k] = p_{lj}[k-1] - \frac{1}{\mathcal{D}_j^+}\Delta p_{jj}[k]$ for all l such that $p_{lj}[k-1] \neq 0$; which can be used to establish that $|\varepsilon_l[k] - \varepsilon_l[k-1]| \leq \frac{1}{\mathcal{D}_j^+}\Delta p_{jj}[k]\Delta\pi_j$, $\forall l$ such that $l \in \mathcal{N}_j$, $l \neq j$. Thus, the worst case occurs when node j can only send information to nodes $l \neq j$ that also satisfy $\sum_{i\in\mathcal{N}_l\cup\{l\}} p_{li}[k-1]\Delta\pi_i \leq \Delta\pi_l$, which means that $\varepsilon_l[k] - \varepsilon_l[k-1] = \frac{1}{\mathcal{D}_j^+}\Delta p_{jj}[k]\Delta\pi_j$ and thus, in the worst case, $\varepsilon[k] - \varepsilon[k-1] = \sum_{l=1}^n \varepsilon_l[k] - \sum_{l=1}^n \varepsilon_l[k-1] = 0$. Otherwise $\varepsilon[k] - \varepsilon[k-1] < 0$. A similar argument can be made when $\sum_i p_{ji}[k-1]\Delta\pi_i[k] > \Delta\pi_j$. ■

Theorem 3: Let $P[k]$ be the update matrix at the k^{th} step of the recursive algorithm described in (25)—(29). Then, $\lim_{k\to\infty} P[k]$ exists and it is a column-stochastic and primitive matrix $P_{ss} = \overline{P}\Delta_{ss} + (I - \Delta_{ss})$, where $\Delta_{ss} = \text{diag}(\delta_1^{ss}, \delta_2^{ss}, \ldots, \delta_n^{ss})$ with $\delta_j^{ss} = \lim_{k\to\infty} \delta_j^k$ satisfying $0 < \delta_j^{ss} \leq 1$, $\forall j = 1,2,\ldots,n$, and $\delta_i^{ss} < 1$ for at least one $i \in \{1,2,\ldots,n\}$.

The key for the proof (which is omitted for brevity) is to show that $\varepsilon[k] = \sum_{j=1}^n |\sum_{i\in\mathcal{N}_j\cup\{j\}} p_{ji}[k]\Delta\pi_i - \Delta\pi_j|$ decreases monotonically with k. To see this, it is necessary to show that $\varepsilon[k] - \varepsilon[k-1] < 0$ unless $\varepsilon[k-1] = 0$ in which case $\varepsilon[k] = \varepsilon[k-1] = 0$. As in the proof of Theorem 1, we define the sets $\mathcal{A}_k$ and $\mathcal{B}_k$ as a partition of the index set $\mathcal{I} = \{1,2,\ldots,n\}$ (i.e., $\mathcal{I} = \mathcal{A}_k \bigcup \mathcal{B}_k$ and $\mathcal{I} = \mathcal{A}_k \bigcap \mathcal{B}_k = \emptyset$), where $\sum_{i\in\mathcal{N}_j\cup\{j\}} p_{ji}[k]\Delta\pi_i > \Delta\pi_j, \forall j \in \mathcal{A}_k$, and $\sum_{i\in\mathcal{N}_j\cup\{j\}} p_{ji}[k]\Delta\pi_i \leq \Delta\pi_j, \forall j \in \mathcal{B}_k$. Unlike that proof, however, it can be verified that it is not necessarily true that $\mathcal{A}_k \subseteq \mathcal{A}_{k-1}$. It is easy to show that Algorithm 3 solves the distributed allocation problem asymptotically using an argument similar to the one used in Algorithm 2.



(a)

$$\overline{P} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ .5 & 0 & 0 & 0 & 0 \\ .5 & .5 & 0 & 0 & .5 \\ 0 & 0 & 0 & 0 & .5 \\ 0 & .5 & 1 & 0 & 0 \end{bmatrix}$$

(b)

Fig. 2. Small directed graph used for illustration of the three algorithms and its corresponding $\overline{P}$.

IV. Example

Consider the directed graph shown in Fig. 2(a) with its corresponding $\overline{P}$ in Fig. 2(b). Assume that the minimum and maximum capacity values for the nodes are given by $\pi_{min} = [.1\ .3\ .4\ .1\ .1]'$ and $\pi_{max} = [.2\ .5\ .7\ .3\ .3]'$. Also assume that the total resource requested by the aggregator is $\rho_d = 1.5$ and that the aggregator can only talk to nodes 1 and 2 so that $x = [.75\ .75\ 0\ 0\ 0]'$.

For Algorithm 1, the evolution of the values $\pi_j[k]$, $j = 1,2,3,4,5$, in (5) are given in Fig. 1(a); as expected, the limiting values are $\pi^{ss} = [.15\ .4\ .55\ .2\ .2]'$ according to (9). The convergence in the case of Algorithms 2 and 3 is shown in Figs. 1(b) and 1(c), respectively. For Algorithm 2, one super-iteration is set to 100 iterations but we only plot the values at the end of each super-iteration.

V. Concluding Remarks and Future Work

In this paper, we have proposed several distributed algorithms that enable decentralized coordination and control of demand response resources and distributed energy resources. A direction for future work is to extend the distributed algorithms beyond finding a feasible solution (i.e., include some optimization criteria in the problem) and to make them robust to faulty or malicious nodes/links.

References

[1] J. Tsitsiklis, "Problems in decentralized decision making and computation," Ph.D. dissertation, MIT, Cambridge, MA, 1984.

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

[3] R. Olfati-Saber and R. M. Murray, "Agreement problems in networks with directed graphs and switching topology," in *Proc. of American Control Conference*, vol. 4, 2003, pp. 4123–4132.

[4] B. Gharesifard and J. Cortés, "Distributed strategies for generating weight-balanced and doubly stochastic digraphs," 2010. [Online]. Available: http://arxiv.org/abs/0911.0232

[5] A. D. Domínguez-García and C. N. Hadjicostis, "Coordination and control of distributed energy resources for provision of ancillary services," in *Proc. IEEE SmartGridComm*, 2010, pp. 537–542.

[6] R. Horn and C. Johnson, *Matrix Analysis*. New York, NY: Cambridge University Press, 1985.