2011 American Control Conference
on O'Farrell Street, San Francisco, CA, USA
June 29 - July 01, 2011

Distributed Averaging with Flow Constraints

Miroslav Barić and Francesco Borrelli*

Abstract— **A network of storage elements is considered. Each storage element is an integrator which can exchange stored resource with neighboring elements. The flow between the elements as well as the amount of the resource that each element can store are subject to constraints. The problem of averaging the state of each element is addressed. The particularity of this problem compared to standard consensus–based averaging is that the elements exchange not only the information but also the stored resource. Thus the elements are controlled–coupled through the flow constraints. A distributed algorithm for flow control is proposed. The algorithm is non–iterative and does not require centralized design procedure. The proposed scheme guarantees asymptotic convergence of states of all nodes to the same value equal to the average of initial values.**

I. Introduction

Networks of integrators are commonly used as a model of large–scale systems comprising smaller sub–systems whose primary purpose is to store some resource (water, goods, energy, data, etc.). Additionally these elements are often required to exchange the resource with other storage elements in the network or to satisfy external demand. In this note we address a problem which is often associated with networked storage devices - the *balancing problem*. In many applications it is preferable to store the resource in a manner which utilizes the available storage evenly across the whole network. One such example is a network of battery cells used in modern electric vehicles. In order to avoid potential damage caused by over– or under–charging of the individual cells, it is desirable to have the amount of charge (normalized to the cell's capacity) stored in the particular cell as close as possible to the normalized amount of charge stored in any other cell. Similar problem occurs in data storage networks in which balancing of the stored data helps avoiding data throughput bottlenecks caused by uneven distribution of the stored data over the nodes in the network. If the number of storage elements and their connections is large the centralized balancing may become intractable. In such cases one may consider consider decentralized or distributed algorithms for balancing. In its most distilled form the problem of balancing reduces to the problem of reaching an agreement on the stored amount of resource in each storage element, i.e. the problem of *consensus*.

In general, consensus protocols are distributed control laws used by agents (dynamical systems) to reach an agreement on a common value of some variable. The attribute "distributed" in this context means that the applied control policy uses only *local* information, i.e. each agent exchanges information only with its neighbors, where a "neighborhood" is typically formalized through a notion of communication graph. Consensus protocols have been considered as a tool for parallel (distributed) computation, including asynchronous averaging protocols ([1], [2]) and protocols that lead to an agreement on general functions ([3], [4]). Most recently consensus algorithms have been applied in areas like mobile robotics and distributed sensing [5], [6]. For a more thorough overview of various applications the reader is referred to [7]. Convergence properties of consensus algorithms have been studied under different assumptions, utilizing existing theory and developing new tools. For instance in [8] the authors apply methods of algebraic graph theory to give conditions for convergence of consensus protocols for directed and undirected communication graphs, possibly time–varying and in presence of comunication time–delays. A general framework for stability analysis of various discrete–time consensus protocols with time varying graph topologies is proposed in [9], introducing a notion of *set–valued Lyapunov functions*, and in [10], using the concept of *averaging maps*.

When considering application of consensus algorithms for resource balancing in storage networks, one must confront the facts that in every real instance of the problem the capacity of the links used for the resource transfers between the storage devices in the network is bounded and that each device can store limited amount of the resource. Nevertheless, the problem of consensus under constraints on dynamic behavior of agents, which is of particular importance for our purposes, has attracted relatively little attention. In [11] authors introduce consensus on several variables that are separated by hard constraints. Consensus in multi–agent systems with various constraints (on the agents' dynamics, connectivity and mission objectives) is discussed in [12]. In [13] the authors propose algorithms used by multiple agents for distributed computations of an estimate of some variable, with a restriction that the estimate of each agent lies in the corresponding constraint set. In particular, the proposed algorithm performs standard consensus–type averaging in which an agent updates its estimate using a convex combination of its neighbors' estimates and projects it on its set of constraints. The approach was subsequently applied to consensus problem of constrained linear systems in [14]. In [15] the authors deploy distributed receding–horizon control approach for consensus in a time–varying network of agents exhibiting single– and double–integrator dynamics and in the presence of control constraints.

In the above cited work dealing with constrained con-

Miroslav Barić is with the United Technologies Research Center (e–mail: baricm@utrc.utc.com) and Francesco Borrelli is with the Department of Mechanical Engineering, UC Berkeley (e–mail: fborrelli@me.berkeley.edu). This work was supported by U.S. Air Force Office of Scientific Research (AFOSR) under Grant FA9550-09-1-0106.

sensus the constraints are imposed on the control decision of each agent or on the consensus value. In all mentioned references the control decisions remain *decoupled*, i.e. each agent chooses a control within its own constraint set and does not consider current control decisions of other agents. In our problem, however, the agents (storage elements) exchange not only information locally, but are expected to exchange stored resources over links with limited capacity at *each discrete–time instance*. Therefore, at each time instance the agents must reach an agreement on the value of the constrained flow through the links. Furthermore, the decision on the exact value of the flow must be reached in *finitely many iteration* and the generated sequence of generated flows should ensure convergence of the amount of the resource stored by each agent to the same common value. In this paper we propose one such protocol which allows agents to reach a quick agreement on local flow values. We show that the proposed algorithm, which is optimization–based and in general generates non–linear control policies, guarantees convergence to the consensus value in which each agent's state equals to the state of its neighbors in the graph. We compare the performance of our algorithm to linear protocols considered in [2], which we modify to satisfy flow constraints in our problem. Computationally, our algorithm requires a solution to a quadratic program in each node of the network, while linear control policies demand very little implementational resources. On the other hand our algorithm is truly distributed in nature and does not require a centralized design procedure like the algorithm in [2]. Also, as indicated in Section III-B, the proposed distributed averaging algorithm is applicable also in case of time–varying network topology.

Notation and basic definitions: The set of real number is denoted by $\mathbb{R}$, the set of non–negative (greater or equal to 0) real numbers as $\mathbb{R}^{+}$ and the set of strictly positive (greater than 0) real numbers as $\mathbb{R}^{++}$. $\mathbb{N}$ is the set of natural numbers, i.e. $\mathbb{N} := \{1, 2, \dots\}$. $\mathbb{N}_{[q1,q2]}$ stands for $\{q1, \dots, q2\}$, where $q1, q2 \in \mathbb{N}$ and $q1 \leq q2$. Empty set is denoted as $\emptyset$. Given any set $\mathcal{S}$, $2^{\mathcal{S}}$ denotes power set of $\mathcal{S}$, i.e. the set of all subsets of $\mathcal{S}$. Cardinality of a set $\mathcal{S}$ is denoted as $|\mathcal{S}|$.

II. Problem Statement

Consider a graph $G = (\mathcal{N}, \mathcal{E})$ where $\mathcal{N} = \mathbb{N}_{[1,n]}$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ is the set of unordered pairs, *edges*. The network stores a resource in its nodes. The most basic notion in our study is *the flow* of such resource between the nodes. The flow is constrained by the *capacity* associated to the edges.

Definition 2.1: A flow is a function $f\colon \mathcal{N} \times \mathcal{N} \to \mathbb{R}$ such that for $(i, j) \in \mathcal{N}$:

(a) $f(i, j) = -f(j, i)$, and
(b) $f(i, j) \leq c(i, j)$,

where $c\colon \mathcal{N} \times \mathcal{N} \to \mathbb{R}^{++}$ is the *capacity function*.
We use a more convenient notation $f_{ij} = f(i, j)$. Each node $i \in \mathcal{N}$ is a discrete–time dynamical system with the state update equation:

$$x_i^{+} = x_i + \phi_i, \tag{1}$$

where x_i^{+} denotes the state at the next discrete time instance given the current state x_i. The state $x_i \in \mathbb{R}$ represents the current amount of the resource stored in the node i, whose level is controlled by the *total net inflow* ϕ_i into the node i, defined as:

$$\phi_i := \sum_{j \in \mathcal{A}_i} f_{ji}. \tag{2}$$

The set of nodes adjacent to node $i \in \mathcal{N}$ is denoted as:

$$\mathcal{A}_i = \{j\colon (i, j) \in \mathcal{E}\}.$$

For later convenience we introduce the sets $\mathcal{A}_i^{+}$ and $\mathcal{A}_i^{-}$ denoting, respectively, adjacent nodes that supply the node i or demand the stored resource from it. Formally, given a flow $f(\cdot, \cdot)$:

$$\mathcal{A}_i^{+} := \{j \in \mathcal{A}_i\colon f_{ji} > 0\}, \tag{3a}$$
$$\mathcal{A}_i^{-} := \{j \in \mathcal{A}_i\colon f_{ji} < 0\}. \tag{3b}$$

We can separate the total net inflow ϕ_i into the amount of the resource that is supplied to the node i, ϕ_i^{+}, and the amount that is taken from the node i, ϕ_i^{-}:

$$\phi_i = \phi_i^{+} - \phi_i^{-},$$

where:

$$\phi_i^{+} := \sum_{j \in \mathcal{A}_i^{+}} f_{ji}, \quad \text{and} \quad \phi_i^{-} := \sum_{j \in \mathcal{A}_i^{-}} f_{ij}.$$

Dynamics of the whole network can be compactly written as:

$$x^{+} = x + \phi,$$

where $x = [x_1\ x_2\ \dots\ x_n]^T$ and $\phi = [\phi_1\ \phi_2\ \dots\ \phi_n]^T$. In addition to flow constraints, we also assume that the amount of resource that can be stored in each node is bounded, i.e. that $x_i \in \mathcal{X}$. Any control problem for a given network with the capacity $c(\cdot, \cdot)$ can be formalized as selection of the flow $f(\cdot, \cdot)$ (in this case the selection of $|\mathcal{E}|$ values f_{ij}) such that the constraints $x_i \in \mathcal{X}$, $i \in \mathcal{N}$, hold for all time. We state our distributed consensus problem as follows.

Problem 1 (Constrained Distributed Consensus): At each time instance k select the flow $f(\cdot, \cdot)$ such that: (a) for a given $x_i(k) \in \mathcal{X}$ the successor state $x_i(k+1) \in \mathcal{X}$, and (b) for all $i \in \mathcal{N}$ the values of $x_i(k)$ converge to the same $\bar{x} \in \mathcal{X}$ as $k \to \infty$. The following restrictions apply at each time instance:

(a) each node $i \in \mathcal{N}$ knows the value of x_i,
(b) each node $i \in \mathcal{N}$ knows the capacity values $c(i, j)$ and $c(j, i)$ for $j \in \mathcal{A}_i$,
(c) each node i can communicate and exchange information only with adjacent nodes, i.e. with nodes $j \in \mathcal{A}_i$,
(d) the decision on the value of f_{ij} is made in one of the nodes i or j.

In order to have a solution for the above problem, an additional condition on the graph connectedness will be imposed later. Clearly, in that case the common limit value $\bar{x}$ corresponds to the average of initial values. We now propose an algorithm which solves the Problem 1 and discuss its properties.

III. Flow–Constrained Consensus Protocol

Assume that each node is associated to an agent which computes the flow values over the incident edges at each discrete–time instance k. To make the presentation more clear, we divide the algorithm executed by each agent into "computational" part, where the flow values for each edge are computed, and "implementational" part in which the actual values of the flows are implemented.

A. The Algorithm

First we present the algorithm used by each agent for computing of the flow values that are to be implemented in the second part of the process. Each agent computes these flow values using only the limited information from its neighborhood, as specified in Problem 1. The aim of each agent is to bring the state of the associated node close to the average of its own state and the states of its neighbors. In order to prevent agents to act selfishly in pursuing their goal of bringing or keeping their node's state close to the average, restrictions are imposed on the minimum flow that agents must allow in order to facilitate the resource transfer between the nodes in their neighborhood. As it will become apparent further in the text, such restrictions are essential for establishing convergence of the states $x_i(k)$ to the common value. At time instance $k \geq 0$, agent at node i computes the candidate flows using the Algorithm 1. Here is the summary of the algorithm clarifying the key points:

- In steps 1–2 the agent i measures the state $x_i(k)$, receives the measurement of the states $x_j(k)$ from its neighbors, determines minimal and maximal values of the states in nodes $\{i\} \cup \mathcal{A}_i$ and computes the average $\bar{x}_i$ of the states x_i and x_j, $j \in \mathcal{A}_i$.
- In step 3 the agent i decides to be supplied only by neighboring nodes which have more resource stored and to supply only the neighboring nodes storing strictly less resource. Agent i expects 0 flow from the neighbors containing the amount of resource equal to x_i. This is formally done, with some notational overloading, by assigning the nodes to the sets $\mathcal{A}_i^+$ and $\mathcal{A}_i^-$ defined in (3).
- In step 4 the agent decides on upper bounds of the incoming and outgoing flow represented by variables c_i^+ and c_i^-, respectively. These bounds are determined by the corresponding edge capacities and the differences of the state x_i and the states x_j, $j \in \mathcal{A}_i^+ \cup \mathcal{A}_i^-$.
- The crucial part of the algorithm is setting up the constraints on incoming and outgoing transfers ϕ_i^+ and ϕ_i^- in step 5. The total net inflow to node i, given by $\phi_i^+ - \phi_i^-$ must be smaller than the amount of resource the node can receive, i.e. smaller than $(x_{\max}^i - x_i)$, and greater than the amount the node can give away, i.e. $(x_i - x_{\min}^i)$. Lower bound for ϕ_i^+ and ϕ_i^- is set to the same value ϕ_{low}^i which is strictly positive for $x_i \in \left(x_{\min}^i, x_{\max}^i\right)$ and is equal to 0 only for $x_i \in \left\{x_{\min}^i, x_{\max}^i\right\}$. This way a node with states between $x_{\min}^i$ and $x_{\max}^i$ acts as a "middle–man", always transferring the resource from nodes in $\mathcal{A}_i^+$ to $\mathcal{A}_i^-$. As it will become clear later, this fact is important for establishing convergence of values x_i to the common consensus value.
- In step 6 the values of incoming and outgoing flows $\tilde{\phi}_i^+$ and $\tilde{\phi}_i^-$ are computed, optimal from the node i's perspective. Chosen cost is the distance from the local average $\bar{x}_i$. The second term in the cost $\rho\phi_i^+\phi_i^-$ is added to make the problem strictly convex and any choice of $\rho \in (0, 2)$ would do. This term can be used to tune the behavior of the algorithm, typically resulting in larger

Algorithm 1 Computations done by an agent i at time k

Require: $x_i(k) \in \mathcal{X}$ and $x_j(k) \in \mathcal{X}$, $j \in \mathcal{A}_i$

1: $x_i = x_i(k)$, $x_j = x_j(k)$, $j \in \mathcal{A}_i$

2: $x_{\min}^i = \min_{j \in \{i\} \cup \mathcal{A}_i} x_j, \quad x_{\max}^i = \max_{j \in \{i\} \cup \mathcal{A}_i} x_j$
$\bar{x}_i := \left(x_i + \sum_{j \in \mathcal{A}_i} x_j\right) / (1 + |\mathcal{A}_i|)$

3: $\mathcal{A}_i^+ = \{j \in \mathcal{A}_i \colon x_j > x_i\}$,
$\mathcal{A}_i^- = \{j \in \mathcal{A}_i \colon x_j < x_i\}$

4:

$$c_i^+ = \begin{cases} 0, & \text{if } \mathcal{A}_i^+ = \emptyset, \\ \sum_{j \in \mathcal{A}_i^+} \min\left\{x_j - x_i, c(j,i)\right\}, & \text{otherwise} \end{cases}$$

$$c_i^- = \begin{cases} 0, & \text{if } \mathcal{A}_i^- = \emptyset, \\ \sum_{j \in \mathcal{A}_i^-} \min\left\{x_i - x_j, c(i,j)\right\}, & \text{otherwise} \end{cases}$$

5: define the constraints:

$$\phi_{low}^i = \min\{x_{\max}^i - x_i, x_i - x_{\min}^i, c_i^+, c_i^-\},$$
$$\Delta_{\max}^i = x_{\max}^i - x_i,$$
$$\Delta_{\min}^i = x_i - x_{\min}^i,$$
$$\Phi_i = \left\{(\phi_i^+, \phi_i^-) \colon \begin{array}{l} \phi_{low}^i \leq \phi_i^+ \leq \min\{c_i^+, \Delta_{\max}^i\} \\ \phi_{low}^i \leq \phi_i^- \leq \min\{c_i^-, \Delta_{\min}^i\} \end{array}\right\},$$

6: compute:

$$(\tilde{\phi}_i^+, \tilde{\phi}_i^-) = \arg \min_{(\phi_i^+, \phi_i^-) \in \Phi_i} \left(x_i + \phi_i^+ - \phi_i^- - \bar{x}_i\right)^2 + \\ + \rho\phi_i^+\phi_i^-,$$

for some $\rho > 0$.

7: compute f_{ji}^i, $j \in \mathcal{A}_i^+$, which minimize:

$$\sum_{j \in \mathcal{A}_i^+} \left(f_{ji}^i\right)^2, \text{subj. to } \begin{cases} \tilde{\phi}_i^+ = \sum_{j \in \mathcal{A}_i^+} f_{ji}^i, \\ f_{ji}^i \leq \min\{x_j - x_i, c(j,i)\}, \\ j \in \mathcal{A}_i^+ \end{cases}$$

8: compute f_{ij}^i, $j \in \mathcal{A}_i^-$, which minimize:

$$\sum_{j \in \mathcal{A}_i^-} \left(f_{ij}^i\right)^2, \text{subj. to } \begin{cases} \tilde{\phi}_i^- = \sum_{j \in \mathcal{A}_i^-} f_{ij}^i, \\ f_{ij}^i \leq \min\{x_i - x_j, c(i,j)\}, \\ j \in \mathcal{A}_i^- \end{cases}$$

values $\tilde{\phi}_i^-$ and $\tilde{\phi}_i^+$ for smaller ρ.

- In steps 7 and 8, given the values $\tilde{\phi}_i^-$ and $\tilde{\phi}_i^+$, the values of the flow for the edges incident to node i are selected. The upper bounds for f_{ij}^i and f_{ji}^i are determined by the capacity values for the corresponding edges and the differences $|x_i - x_j|$. The constraints on ϕ_i^- and ϕ_i^+ in steps 3–5 ensure that the optimization problems in steps 7 and 8 are feasible.

The values f_{ij}^i for $j \in \mathcal{A}_i^-$ and f_{ji}^i for $j \in \mathcal{A}_i^+$ computed at node i in general differ from the values f_{ij}^j and f_{ji}^j computed at an adjacent node $j \in \mathcal{A}_i$. Hence, these values must be "negotiated" before the implementation and the simplest way to achieve that is to implement the flow for which $f_{ij} = \min\left\{f_{ij}^i, f_{ij}^j\right\}$. We will show later that with such choice the feasibility and required convergence as stated in Problem 1 are ensured for connected graph G. The implementation part of the consensus protocol is therefore given by the following procedure.

Algorithm 2 Implementation of the flow by an agent i at time k

Require: the sets $\mathcal{A}_i^+$ and $\mathcal{A}_i^-$ and the values f_{ij}^i, $j \in \mathcal{A}_i^-$, and f_{ji}^i, $j \in \mathcal{A}_i^+$, computed using Algorithm 1.

1: send: f_{ij}^i to nodes $j \in \mathcal{A}_i^-$,
f_{ji}^i to nodes $j \in \mathcal{A}_i^+$, and
$f_{ij}^i = 0$ to nodes $j \notin \mathcal{A}_i^- \cup \mathcal{A}_i^+$
2: receive: f_{ij}^j from nodes $j \in \mathcal{A}_i^-$,
f_{ji}^j from nodes $j \in \mathcal{A}_i^+$
3: **for all** $j \in \mathcal{A}_i$ such that $j < i$ **do**
4: **if** $j \in \mathcal{A}_i^-$ **then**
5: implement $f_{ij} = \min\left\{f_{ij}^i, f_{ij}^j\right\}$
6: **else if** $j \in \mathcal{A}_i^+$ **then**
7: implement $f_{ji} = \min\left\{f_{ji}^i, f_{ji}^j\right\}$
8: **else**
9: implement $f_{ij} = 0$
10: **end if**
11: **end for**

The procedure given by Algorithms 1–2 clearly terminates in finite time. The requirement is, however, that the decisions made by agents are synchronized: the actual flow cannot be implemented before all agents complete their computations and exchange the results. It should be clear that the selected rule prescribing that maximum of the nodes i and j implements f_{ij} is arbitrary and does not affect the resulting dynamic behavior of the network.

Next step is to demonstrate that the procedure in Algorithms 1–2 can actually be used to solve the Problem 1.

B. Convergence

We start with some auxiliary observations highlighting the relevant features of the Algorithms 1–2. In particular, we show that the flow value f_{ij} will be non–zero if and only if the difference $x_i - x_j$ is non–zero and use this to prove local "contractive" property of the generated flows, namely that the successor state x_i^+ is in the relative interior (if one exists) of the interval $[x_{\min}^i, x_{\max}^i]$. Finally, we show that the generated flow values change continuously with the state vector x. All these properties are then used to prove the main result stated by Theorem 3.1.

Unfortunately, due to the lack of space we are able only to list above properties. Detailed proof and the discussion is provided in the technical report [16].

Lemma 3.1:

(a) Let $\Delta_{ij} := x_i - x_j$, for $i \in \mathcal{N}$ and $j \in \mathcal{A}_i$. The value f_{ij}^i computed by the agent i using the Algortihm 1 equals 0 if and only if $\Delta_{ij} = 0$. In particular, $f_{ij}^i > 0$ if and only if $\Delta_{ij} > 0$ and $f_{ji}^i > 0$ if an only if $\Delta_{ij} < 0$.

(b) Let ϕ_i be the actual (implemented) net inflow for the node i, i.e. given the actual flow $f(\cdot,\cdot)$ computed in Algorithm 2, $\phi_i = \sum_{j\in\mathcal{A}_i} f_{ji}$. Then $x_i + \phi_i \in \left[x_{\min}^i, x_{\max}^i\right]$, where $x_{\min}^i$ and $x_{\max}^i$ are specified in Algorithm 1, step 1. Moreover, $x_i + \phi_i \in \left\{x_{\min}^i, x_{\max}^i\right\}$ if and only if $x_i = x_{\min}^i = x_{\max}^i$.

(c) Let $\phi_i \colon \mathcal{X}^n \to \mathbb{R}$ be the mapping from $x \in \mathcal{X}^n$ to the total net inflow of the node i computed by the Algorithms 1–2. The mapping $\phi_i(\cdot)$ is continuous.

We can now make a claim about the distributed algorithm that solves the Problem 1.

Theorem 3.1: Let the graph $G = (\mathcal{N}, \mathcal{E})$ be connected. Then the distributed algorithm given by Algorithms 1–2 solves the Problem 1. In particular, given an initial $x_i(0)$, sequences $\{x_i(k)\}_{k=0}^{\infty}$ for $i \in \mathcal{N}$ converge to the mean value $\frac{1}{|\mathcal{N}|}\sum_{i\in\mathcal{N}} x_i(0)$.

We remark that the the properties of our algorithm stated in Lemma 3.1 satisfy Assumption 1 in [9], which considers much more general framework including time–varying network topologies. In that respect, we can simply refer to the result in [9] and claim the convergence for the general case of time–varying graphs. For the particular case of time–invariant graphs the proof of Theorem 3.1 is provided in [16].

IV. Numerical Example

In this section we compare the proposed distributed averaging algorithm to the *fastest distributed linear averaging* (FDLA) algorithm in [2]. In [2] the authors consider flows, for a given x, generated by linear feedback laws:

$$f_{ij}(x) = k_{ij}(x_i - x_j), \quad (i,j) \in \mathcal{E}.$$

The gains k_{ij} which ensure optimal convergence rate (with respect to the considered *asymptotic convergence factor*, cf. [2]) can be obtained by solving the following optimization problem:

$$\min_{k_{ij}} \rho(W - \mathbb{1}^T\mathbb{1}/n), \tag{4a}$$

$$\text{subj. to } W = I - B\,\mathrm{diag}(k_{ij})\,B^T, \tag{4b}$$

where $\rho(\cdot)$ denotes the spectral radius of the argument matrix, B is the incidence matrix for the considered network and arbitrary graph orientation, $\mathrm{diag}(k_{ij})$ represents diagonal matrix with gains k_{ij} on the diagonal, $\mathbb{1}$ is the vector whose entries are all equal to 1 and I is the identity matrix of

appropriate dimension. Optimization problem (4) must be solved for the given network as a whole. As shown in [2], for symmetric weights, i.e. for $k_{ij} = k_{ji}$, the problem (4) can be solved by convex optimization. In order to satisfy capacity constraints imposed by Problem 1 we augment the constraints in (4) with the set of linear inequalities:

$$|k_{ij}(x_{\max} - x_{\min})| \leq \min\{c(i,j), c(j,i)\},\ (i,j) \in \mathcal{E}, \quad (5)$$

where $x_{\min}$ and $x_{\max}$ correspond to the minimal and maximal possible value of the integrator state.

We consider the network used as an example in [2], consisting of 8 nodes and 17 edges whose graph is depicted in Figure 1. The performance of our distributed algorithm is

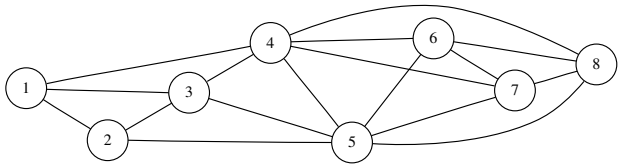


Fig. 1: Graph of the network used in the numerical example.

compared to the FDLA algorithm in [2] with and without constraints (5) on linear control laws. We assume that all capacities in the network are equal to 1, i.e. $c(i,j) = c(j,i) = 1$ for all $(i,j) \in \mathcal{E}$. The initial amount of resource in each node (the state) is selected randomly in the interval $[0, 10]$. Time evolution of the states and the flow values generated by the proposed Algorithms 1–2 is depicted in Figure 2.

We computed[1] the gains k_{ij} for the FDLA algorithm *without imposing the constraints* (5). The resulting spectral radius $\rho(W - \mathbb{1}^T\mathbb{1}/n)$ is 0.6, which is the value reported for the same example in [2]. Performance of the unconstrained linear averaging (FDLA) algorithm is shown in Figure 3.

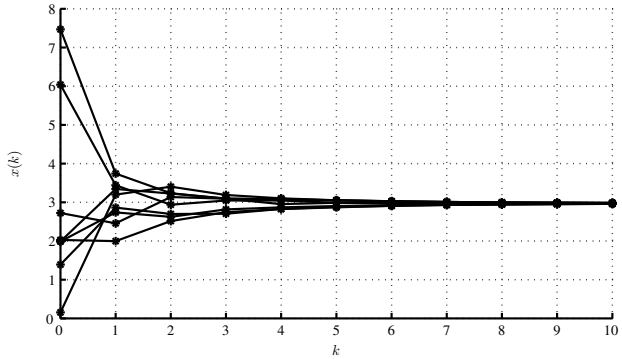
Finally, we computed the gains for the FDLA algorithm with capacity constraints (5), resulting with the spectral radius $\rho(W - \mathbb{1}^T\mathbb{1}/n)$ equal to 0.84774. The performance of the constrained FDLA algorithm is shown in Figure 4.

As can be seen clearly from the Figures 2 and 3, our distributed algorithm, which in general induces non–linear averaging control laws, provides practically the same performance as the fastest linear distributed averaging algorithm, while generating flows which satisfy capacity constraints. On the other hand, the original FDLA algorithm as presented in [2] generates flows which do not conform to the existing capacity constraints $c(i,j) \leq 1$, as can be seen in Figure 2(b). If we introduce capacity constraints (5) in the computation of FDLA gains, the performance of the linear algorithm expectedely deteriorates, as shown in Figure 4.

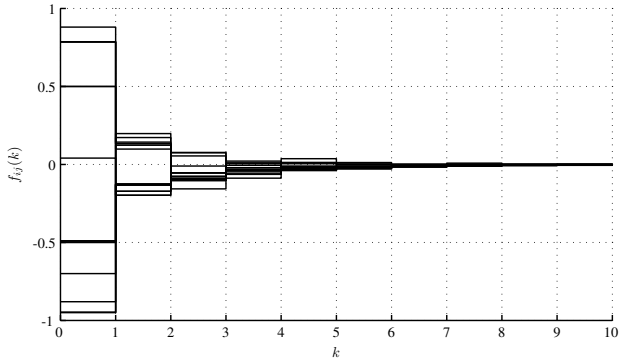
V. Conclusion

We propose an algorithm for distributed averaging (consensus) in networks of storage elements subject to constraints on the states and the control inputs. Unlike standard consensus problems in the literature where each agent is actuated

[1]With the help of YALMIP [17].



(a) State evolution.



(b) Generated flows.

Fig. 2: Time evolution of states and flows for the Algorithms 1–2. Generated flows respect capacity bounds $[-1, 1]$.

independently, in our problem we include constraints on the flows between the individual storage elements. In a distributed setup this type of constraints requires an agreement between the agents on the flows along the common edges and this agreement must be reached in finitely many iterations in order to generate the appropriate flow between the storage elements. We compare our algorithm to the distributed averaging scheme reported in [2], based on linear policies. As shown on the simple example, our algorithm provides fast convergence, while satisfying all constraints, which is not an inherent feature of the linear distributed algorithm.

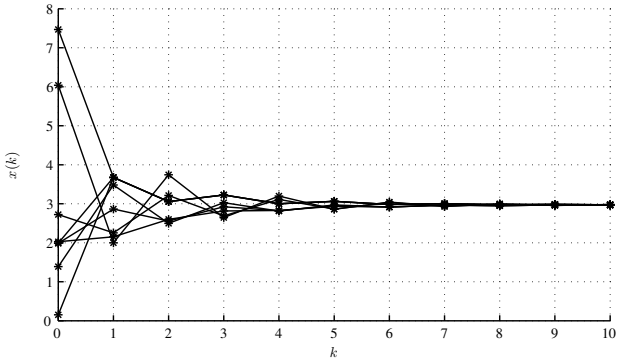
As a future research direction, one may consider the extension in which certain nodes can select the set of their neighbors with the aim of improving the convergence rate. Another research topic of immediate interest is the influence of time–delays, measurement and communication noise and dynamic disturbances.

References

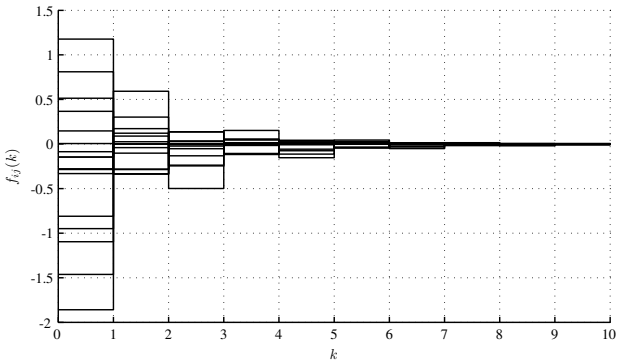
[1] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed Asynchronous Deterministic and Stochastic Gradient Optimization Algorithms," *IEEE Trans. Automat. Contr.*, vol. 31, no. 9, pp. 803–812, 1986.

[2] L. Xiao and S. Boyd, "Fast Linear Iterations for Distributed Averaging," *Syst. Control Lett.*, vol. 53, pp. 65–78, 2004.

[3] J. Cortés, "Distributed algorithms for reaching consensus on general functions," *Automatica*, vol. 44, pp. 726–737, 2008.

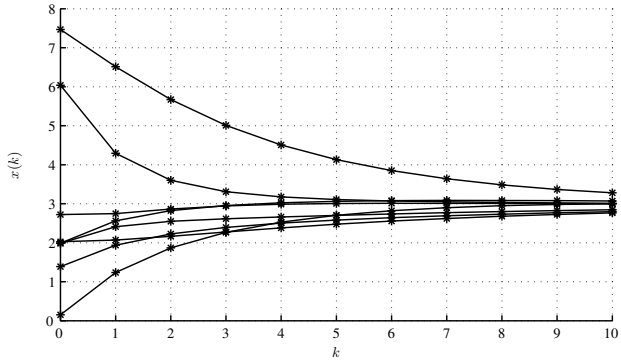


(a) State evolution.

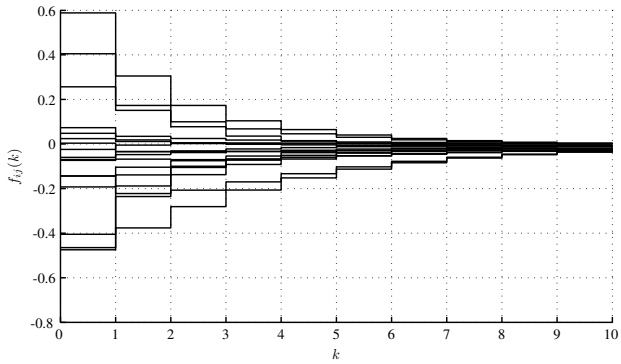


(b) Generated flows.

Fig. 3: Time evolution of states and flows for the FDLA algorithm as originally proposed in [2] *without capacity constraints*. The capacity bounds $[-1, 1]$ are violated.



(a) State evolution.



(b) Generated flows.

Fig. 4: Time evolution of states and flows for the FDLA algorithm in [2] *with imposed capacity constraints*.

[4] D. Bauso, L. Giearré, and R. Pesenti, "Non–linear protocols for optimal distributed consensus in networks of dynamic agents," *Syst. Control Lett.*, vol. 55, pp. 918–928, 2006.

[5] A. Jadbabaie, J. Lin, and A. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Trans. Automat. Contr.*, vol. 48, no. 6, pp. 988–1001, 2003.

[6] R. Olfati-Saber and J. Shamma, "Consensus Filters for Sensor Networks and Distributed Sensor Fusion," in *Proc. 44th IEEE Conf. on Decision and Control*, Sevilla, Spain, 2005, pp. 6698–6703.

[7] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and Cooperation in Networked Multi-Agent Systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[8] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Automat. Contr.*, vol. 49, no. 9, pp. 1520–1533, 2004.

[9] L. Moreau, "Stability of multiagent systems with time-dependent communication links," *IEEE Trans. Automat. Contr.*, vol. 50, no. 2, pp. 169–182, 2005.

[10] J. Lorenz and D. Lorenz, "On conditions for convergence to consensus," *IEEE Trans. Automat. Contr.*, vol. 55, no. 7, pp. 1651–1656, 2010.

[11] K. Moore and D. Lucarelli, "Forced and constrained consensus among cooperating agents," in *Proc. of IEEE Intl. Conf. on Networking, Sensing and Control*, 2005, pp. 449–454.

[12] M. Franceschelli, M. Egerstedt, A. Giua, and C. Mahulea, "Constrained invariant motions for networked multi-agent systems," in *Proc. American Control Conf.*, St. Louis, MO, USA, jun 2009, pp. 5749–5754.

[13] A. Nedić and A. Ozdaglar, "Distributed Subgradient Methods for Multi-Agent Optimization," *IEEE Trans. Automat. Contr.*, vol. 54, no. 1, pp. 48–61, 2009.

[14] B. Johansson, A. Speranzon, M. Johansson, and K. Johansson, "On decentralized negotiation of optimal consensus," *Automatica*, vol. 44, pp. 1175–1179, 2008.

[15] G. Ferrari-Trecate, L. Galbusera, M. Marciandi, and R. Scattolini, "Model Predictive Control Schemes for Consensus in Multi–Agent Systems with Single– and Double–Integrator Dynamics," *IEEE Trans. Automat. Contr.*, vol. 54, no. 11, pp. 2560–2572, 2009.

[16] M. Barić and F. Borrelli, "Distributed averaging with flow constraints," UC Berkeley, Tech. Rep., 2011, available from http://www.mpc.berkeley.edu.

[17] J. Löfberg, "YALMIP : A toolbox for modeling and optimization in MATLAB," in *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004, available from `http://control.ee.ethz.ch/~joloef/yalmip.php`.