Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Convergence and Stability Properties of a Dynamic Maximum Consensus Estimator

João C. Monteiro * **Alessandro Jacoud Peixoto** *

* *Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil*

Abstract: In this paper, we present a novel dynamic consensus algorithm capable of tracking the maximum value of a given measurement in a distributed network. Each node in the network implements a sliding-mode based algorithm and only uses the information provided by its neighbors to track this maximum value on the network. Thus, at any given time, a network node is not allowed to disclose information from one of its neighbors to any other neighbor. We demonstrate the convergence and stability properties of this technique and provide a guide to select the control parameters, initial conditions, and sampling period for discrete-time implementations. From a practical perspective, the proposed technique is very promising since it relies on selecting only three parameters, which are the same for the whole network, and solving one numerical integration on each node. Numerical simulations illustrate the results and help visualize the algorithm transient behavior.

Keywords: sliding-mode control; consensus

1. INTRODUCTION

This work is inspired by distributed optimization problems, such as swarm motion towards a target position and cost minimization in cooperative games. In fact, the idea of developing the maximum consensus technique described in this text occured while searching for a solution to estimating the Chebyshev distance of multiple objective functions in a distributed optimization problem. *Distributed* is used in the sense that the many nodes that constitute the network have an associated objective function, which varies with time, and the ability to exchange information with its neighbors. A node cannot, however, inquire information from a non-neighbor node, even if one of its neighbors can communicate with this other node. *Consensus* is used in the sense that each node aims at determining a common global performance of the system.

Problems such as this one are extensively studied in the area of networked systems. They usually appear in two forms: static consensus and dynamic consensus. In *static consensus*, a snapshot of the nodes' inputs at a given time is used to initialize the algorithm, but changes to these inputs are ignored. In *dynamic consensus*, algorithms are designed to track the desired network performance as the nodes' inputs change through time.

Some of the pioneering works on consensus estimators are due to Spanos et al. (2005); Ren and Beard (2005); Olfati-Saber et al. (2007). There are many works on consensus estimators dealing with average consensus. In the static case, authors have proposed many solutions, encompassing, for example, privacy-preserving algorithms (Manitara and Hadjicostis, 2013; Mo and Murray, 2016), robustness to switching topologies and time-delays (Olfati-Saber and Murray, 2004), and disturbance rejection (Bauso et al., 2009). Since the literature on static consensus is quite mature, one might be tempted to apply a static algorithm over fixed periods of time repeatedly. As discussed and exemplified by Kia et al. (2019), this is usually not the best approach.

In contrast, dynamic consensus algorithms are explicitly developed to deal with time-varying inputs. Although the literature is not as extensive as the one on static consensus, for specific consensus algorithms (mainly average consensus), authors have already tackled problems such as robustness to additive disturbances (Shi and Johansson, 2013), privacy-preserving schemes (Kia et al., 2015), and robustness to communication delays (Moradian and Kia, 2018). Very recently, Kia et al. (2019) wrote a survey paper on various applications and theoretical foundations of dynamic average consensus algorithms. For a thorough review of the state-of-the-art of consensus estimators, we strongly recommend (Kia et al., 2019) and the references therein. Also, a field of study that uses many of the techniques that come from the dynamic consensus literature is leader-follower networks of mobile agents.

Regarding maximum-value consensus, some authors have studied this problem to solve time synchronization in wireless sensor networks. In such a network, each sensor performs measurements in a given time and publishes this information to its neighbors. Thus, all logical times must be synchronized across the network. For this problem, authors have proposed appealing static consensus algorithms tackling the most common challenges of wireless sensor networks — privacy-preservation (Wang et al., 2019), security against malicious attacks (He et al., 2014a), and robustness to network delays (He et al., 2014b).

Although the literature on consensus algorithms is quite extensive, to the best of our knowledge, no author has considered the problem of dynamic maximum consensus. Thus, our contribution is the development of a dynamic maximum consensus algorithm capable of tracking the

time-varying maximum input of a directed and strongly connected network. Naturally, the algorithm also works for undirected and strongly connected networks. Our algorithm also differs from most works in the consensus literature because it is based on sliding-mode control, which ensures finite-time convergence to the network maximum. Theoretically, the convergence time and final tracking error can be made arbitrarily small. Nonetheless, we provide a practical guide to select the control parameters in order to meet two predefined design constraints, namely: maximum tracking error and convergence rate.

This paper is structured as follows. In section 2, we introduce the necessary notations and mathematical foundations needed to present the results and state the maximum consensus problem. In section 3, we develop the sliding-mode based dynamic maximum consensus algorithm capable of achieving consensus in finite-time and demonstrate its stability and convergence properties. Also, in this section, we provide a practical guide to help the control designer select the appropriate parameters to meet the desired network performance. In section 4, we perform simulations to illustrate the results, and in section 5 we conclude the paper and hint the direction on future works.

2. PRELIMINARIES AND PROBLEM STATEMENT

In this section, we provide the mathematical foundations needed to present the results and state the dynamic maximum consensus problem.

Consider a group of n labeled nodes, with labels belonging to the set $\mathcal{V} = \{1,\ 2,\ \dots\ ,\ n\}$, each one holding an input $u_j(t) \in \mathbb{R}$ and an estimate $x(t) \in \mathbb{R}$, both function of time. These nodes interact over a communication network, with topology represented by a directed graph $\mathcal{G} = (\mathcal{V},\ \mathcal{E})$, where $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is the set of all edges which connect two nodes. It is said that a node i receives information from node j if and only if $(i, j) \in \mathcal{E}$. When this is the case, node i has access to the inputs and the estimates of node j at any given time t.

Many properties can be associated with graphs, but we are particularly interested in the following. An *undirected path* is a sequence of nodes $i_1, i_2, \dots, i_p$ such that either $(i_j, i_{j+1}) \in \mathcal{E}$ or $(i_{j+1}, i_j) \in \mathcal{E}$. Using this concept, a graph is *weakly connected* if every pair of nodes lie on some undirected path. A *directed cycle* is a sequence of nodes $i_1, i_2, \dots, i_p$, with $i_1 = i_p$, such that $(i_j, i_{j+1}) \in \mathcal{E}$. A graph is *strongly connected* if every pair of nodes lie on some directed cycle. Considering undirected graphs, weakly connected $\iff$ strongly connected. An illustration of these concepts is shown in fig. 1. Similar to an undirected path, a *directed path* is a sequence of nodes $i_1, i_2, \dots, i_p$ such that $(i_j, i_{j+1}) \in \mathcal{E}$.

For a given node j,

$$\mathcal{U}_j(t) = \{u_i(t) \in \mathbb{R} : (j, i) \in \mathcal{E}\} \cup \{u_j(t)\} \tag{1a}$$

$$\mathcal{X}_j(t) = \{x_i(t) \in \mathbb{R} : (j, i) \in \mathcal{E}\} \cup \{x_j(t)\} \tag{1b}$$

are the sets containing all inputs and all estimates that node j is aware of, including its own input and estimate, which are contained in the singletons $\{u_j(t)\}$ and $\{x_j(t)\}$.

The *shortest path* between two nodes in a directed graph is the directed path with the least amount of edges. The

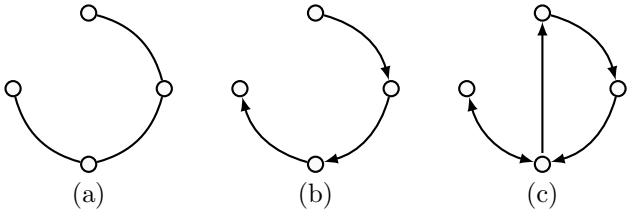

Fig. 1. Examples of (a) an undirected strongly connected graph, (b) a weakly connected but not strongly connected directed graph, and (c) a strongly connected directed graph.

distance between two nodes $\delta(i, j)$ is the number of edges in a shortest path, and the *eccentricity* of a node is its greatest distance to any other node. The *diameter*, denoted $d(\mathcal{G})$, is the greatest eccentricity of all nodes in a graph.

Given a vector $v \in \mathbb{R}^p$, $v_j \in \mathbb{R}$ denotes its j-th component. Therefore, the vectors containing all inputs and all estimates in the network are $u(t), x(t) \in \mathbb{R}^n$, respectively.

Depending on the context, the function max(.) can denote either the maximum component of a vector or the maximum element in a set. The sign function is defined as

$$\text{sign}(\zeta) = \begin{cases} -1\ ,\ \zeta < 0 \\ 1\ ,\ \zeta > 0 \end{cases} \tag{2}$$

If $\zeta = 0$ is not a sliding-surface, $\text{sign}(\zeta) = 0$ at $\zeta = 0$. Otherwise, $\text{sign}(\zeta)$ is undefined at $\zeta = 0$. We consider the following linear approximation of the sign function,

$$\text{sign}_\epsilon(\zeta) = \begin{cases} \text{sign}(\zeta)\ ,\ |\zeta| > \epsilon \\ \zeta/\epsilon\ \ \ \ \ ,\ |\zeta| \leq \epsilon \end{cases} \tag{3}$$

for any positive scalar $\epsilon > 0$.

Furthermore, all solutions to differential equations are defined for $t \geq t_0$, for any initial time $t_0 \in \mathbb{R}$. When dealing with systems described by differential equations with discontinuous right-hand sides, we adopt Filipov's definition of solutions (Filippov, 1964). To improve the paper clarity, the explicit dependence of time is omitted, kept only in contexts where this could cause ambiguity.

Finally, the objective of this paper is to develop an algorithm such that all estimates x_j are driven toward the maximum input $\max(u)$ in the network. In other words, for $t > t^*$, $|x_j(t) - \max(u(t))| < \epsilon$, for all $j \in \mathcal{V}$ and for an arbitrarily small scalar $\epsilon > 0$, where $t^* > t_0$ can be made arbitrarily close to t_0.

3. DYNAMIC MAXIMUM CONSENSUS ALGORITHM

In this section, we formulate the problem and present the sliding-mode based maximum consensus algorithm.

For the remaining of this paper, let $\mathcal{G} = (\mathcal{V},\ \mathcal{E})$ denote a directed and strongly connected graph, with nodes belonging to $\mathcal{V} = \{1,\ 2,\ \dots\ ,\ n\}$ and edges belonging to $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$.

For this class of networks, consider the update rule

$$\tau \dot{x}_j = (1 + \alpha)\,\text{sign}_\epsilon(\hat{e}_j) + \text{sign}(e_j) \tag{4a}$$

$$\hat{e}_j = \max(\mathcal{X}_j) - x_j \tag{4b}$$

$$e_j = \max(\mathcal{U}_j) - x_j \tag{4c}$$

where $\hat{e}_j(t) \in \mathbb{R}$ is the error between the node estimate and the maximum estimate it knows, $e_j(t) \in \mathbb{R}$ the error

between the node estimate and the maximum input it knows, $\tau > 0$ is a scalar that controls the convergence rate, and $\alpha \in (0, 1)$ and $\epsilon > 0$ are other design parameters.

The first component of (4) is responsible for driving all estimates toward a common value, which is the highest in the entire network. The second component is responsible for driving the estimates toward the maximum input.

We assume that $u(t)$ is differentiable almost everywhere, and that its rate of change $\dot{u}(t)$ is bounded, such that the following are true.

Assumption 1 (Differentiable Input Vector)
The input vector $u(t)$ is differentiable almost everywhere.

Assumption 2 (Bounded Input Derivative)
The input vector $u(t)$ is absolutely continuous and there is a known upper-bound $L > 0$ to its time derivative

$$\max_{j \in \mathcal{V}} |\dot{u}_j(t)| \leq L \ , \quad \forall t \geq t_0 \tag{5}$$

which is defined almost everywhere.

Assumption 2 is important to enable the control designer to select an appropriate value for the parameter τ.

Lemma 1 (Convergence to max(x))
Let $\mathcal{G}$ be a directed and strongly connected graph and let all nodes update their estimates through the update rule (4). Then, with $\tau < \alpha/L$, the network achieves a consensus and all estimates converge to $\max(x) - x_j \leq \mathcal{O}(\epsilon)$ after a finite-time $t_1 \leq t_0 + \mathcal{O}(\tau)$, remaining therein for $t \geq t_1$.

Proof. At any given time, a node's estimate is lower or equal to the highest estimate it knows, i.e. $x_j(t) \leq \max(\mathcal{X}_j(t))$, $\forall j \in \mathcal{V}$. Thus, considering also the ϵ-vicinity of $\max(\mathcal{X}_j)$, there are three possibilities for any node j: $x_j \leq \max(\mathcal{X}_j) - \epsilon$, $\max(\mathcal{X}_j) - \epsilon < x_j < \max(\mathcal{X}_j)$, and $x_j = \max(\mathcal{X}_j)$.

$\boldsymbol{x_j \leq \max(\mathcal{X}_j) - \epsilon}$**.** In this case, it follows that $\hat{e}_j \geq \epsilon \iff \text{sign}_\epsilon(\hat{e}_j) = 1$, and (4a) becomes

$$\tau \dot{x}_j = 1 + \alpha + \text{sign}(e_j) \geq \alpha \tag{6}$$

with a solution

$$x_j(t) \geq x_j(t_0') + (\alpha/\tau)(t - t_0') \tag{7}$$

where $t_0' > t_0$ is a time instant at which x_j enters the region $x_j \leq \max(\mathcal{X}_j) - \epsilon$. Therefore, while x_j is not close to $\max(\mathcal{X}_j)$ it increases at a rate α/τ.

$\boldsymbol{x_j = \max(\mathcal{X}_j)}$**.** In this case, $\hat{e}_j = 0 \iff \text{sign}_\epsilon(\hat{e}_j) = 0$, and (4a) becomes

$$\tau \dot{x}_j = \text{sign}(e_j) \tag{8}$$

such that the dynamics of e_j becomes

$$\dot{e}_j = \tau \frac{d}{dt} \max(\mathcal{U}_j) - \text{sign}(e_j) \tag{9}$$

Therefore, as long as $\tau < 1/L$, $e_j = 0$ is a sliding-surface while $x_j = \max(\mathcal{X}_j)$. Furthermore, to guarantee that the neighbors i of node j for which $\max(\mathcal{X}_i) = x_j$, have estimates converging to x_j, one must have $\alpha/\tau > L \iff \tau < \alpha/L$. This conclusion follows from (7).

These two cases are enough to show that $x_j \to \max(x)$, $\forall j \in \mathcal{V}$, entering an ϵ-vicinity of $\max(x)$ in finite-time, since all estimates are either increasing, according to (7) or bounded by their highest known input until another estimate exceeds it. When this happens, the exceeded estimate, for instance x_i, must track its current maximum known estimate $\max(\mathcal{X}_i)$. This process repeats itself and propagates through the network until all estimates reach $x_j \geq \max(x) - d(\mathcal{G})\,\epsilon$, converging in a finite-time

$$t_1 \leq t_0 + \Delta \left(\frac{\alpha}{\tau} - L\right)^{-1} \leq t_0 + \mathcal{O}(\tau) \tag{10a}$$

$$\Delta = \max\left(\begin{bmatrix} u(t_0) \\ x(t_0) \end{bmatrix}\right) - \min(x(t_0)) \tag{10b}$$

The propagation is guaranteed because $\mathcal{G}$ is strongly connected. The factor $d(\mathcal{G})$ acts as a worst case bound because the estimate error might propagate through the network from $\max(x)$ to the node furthest from it. This distance is, at most, equal to the graph diameter $d(\mathcal{G})$. □

From the perspective of consensus, lemma 1 is enough to show that the network reaches a consensus with the proposed technique. It does not, however, establish any relationship between the nodes' inputs and the consensus value. The next theorem shows that the update rule (4) is able to enforce tracking of the maximum network input.

Theorem 1 (Convergence to max(u))
Let $\mathcal{G}$ be a directed and strongly connected graph and let all nodes update their estimates through the update rule (4). Then, with $\tau < \alpha/L$, the network achieves a consensus and all estimates converge to $|\max(u) - x_j| < \mathcal{O}(\epsilon)$ after a finite-time $t^* \leq t_0 + \mathcal{O}(\tau)$, remaining therein for $t \geq t^*$.

Proof. From lemma 1 we already know that $\max(\mathcal{X}_j) - x_j \leq \epsilon$ and $\max(x) - x_j \leq d(\mathcal{G})\,\epsilon$ hold $\forall j \in \mathcal{V}$ for $t \geq t_1$. Thus, it suffices to show that $\max(x) \to \max(u)$. The proof is then split into two parts. First we show that, if $\max(x) < \max(u)$, all estimates increase. Otherwise, if $\max(x) > \max(u)$, all estimates decrease.

$\boldsymbol{\max(x) < \max(u)}$**.** Let $j : u_j = \max(u)$, which implies $u_j = \max(\mathcal{U}_j)$, and let

$$k \in \{i \in \mathcal{V} : (j, i) \in \mathcal{E}\} \cup \{j\} \tag{11}$$

that is, k correspond to all nodes that node j has access to, including itself. Considering these nodes and $t > t_1$, such that $\max(\mathcal{X}_j) - x_j \leq \epsilon \iff \text{sign}_\epsilon(\hat{e}_j) = \hat{e}_j/\epsilon$,

$$\tau \dot{x}_k = \frac{1 + \alpha}{\epsilon}\, \hat{e}_k + \text{sign}(e_k) \tag{12}$$

Since $\max(x) < \max(u)$, then $\text{sign}(e_k) = \text{sign}(\max(u) - x_k) = 1$. Furthermore, from its definition, it follows that $\hat{e}_k \geq 0$, and equation (12) can be converted to the inequality

$$\tau \dot{x}_k \geq 1 \tag{13}$$

which yields

$$x_k(t) \geq x_k(t_2) + (t - t_2)/\tau \tag{14}$$

where $t_2 \geq t_1$ is any time instant for which $\max(x) < \max(u)$. Hence, for $t > t_1$, $\max(\mathcal{X}_j)$ must reach $\max(u)$ in a finite-time

$$t_1^* \leq t_1 + \Delta^* \left(\frac{1}{\tau} - L\right)^{-1} \tag{15a}$$

$$\Delta^* = |\max(u(t_1)) - \max(\mathcal{X}_j(t_1))| \tag{15b}$$

with L from assumption 2. From lemma 1, $|x_j - \max(u)| \leq d(\mathcal{G})\,\epsilon$, $\forall j \in \mathcal{V}$, is reached in a finite-time bounded by (15).

$\max(x) > \max(u)$. Consider the error function

$$e = \max(x) - \max(u) = 0 \tag{16}$$

with time derivative

$$\tau \dot{e} = \text{sign}(e_i) - \tau \frac{d}{dt} \max u(t) \tag{17}$$

where $i : x_i = \max(x)$ and $\text{sign}_\epsilon(x_i) = 0$ was omitted because x_i is not in sliding-mode. Since $\max(x) > \max(u)$, then $\text{sign}(e_i) = -1$. Therefore, the following is valid whenever $\max(x) > \max(u)$:

$$\tau \dot{e} \leq \tau L - 1 \tag{18a}$$

$$e(t) \leq e(t_3) - \left(\frac{1}{\tau} - L\right)(t - t_3) \tag{18b}$$

where $t_3 \geq t_1$ is any time instant for which $\max(x) > \max(u)$. Hence, for $t > t_1$ and $\max(x) > \max(u)$, the maximum estimate $\max(x)$ reaches $\max(u)$ in a finite-time

$$t_2^* \leq t_1 + \Delta^* \left(\frac{1}{\tau} - L\right)^{-1} \tag{19}$$

which equals the bound (15) for the previous case. Once again, invoking lemma 1, $|x_j - \max(u)| \leq d(\mathcal{G})\,\epsilon$, $\forall j \in \mathcal{V}$, is reached in a finite-time bounded by (19).

Finally, since it was shown that $\max(x)$ is driven toward $\max(u)$, we conclude from (10), (15), and (19) that $|\max(u) - x_j| \leq d(\mathcal{G})\,\epsilon \leq \mathcal{O}(\epsilon)$ is reached in a finite-time

$$t^* \leq t_0 + t_1 + \Delta^* \left(\frac{1}{\tau} - L\right)^{-1} \leq t_0 + \mathcal{O}(\tau) \tag{20}$$

□

Remark 1 (Finite-Time Upper-Bounds)
It is worth pointing out that both convergence time upper-bounds (10) and (20) are very conservative, since, to compute them, it is assumed that the inputs are always growing at their maximum possible rate $L \geq \max_{j\in\mathcal{V}} |\ddot{u}_j|$.

Theorem 1 guarantees that all estimates track the maximum network input, with an arbitrarily small error of order $\mathcal{O}(\epsilon)$. Naturally, in practical discrete-time implementations, even though the theory ensures arbitrarily fast convergence rates and small tracking errors, there is a tradeoff between improving these values and selecting an appropriate sampling period. The faster the system dynamics, the smaller the sampling period. Likewise, the smaller the desired tracking error, the smaller the sampling period. To help implementing the proposed consensus algorithm (4), we highlight some guidelines in the following remark.

Remark 2 (Discrete-Time Implementation)
To avoid undesired chattering, we have experienced better results using the trapezoidal integration rule and using $\text{sign}_\epsilon(e_j)$ instead of $\text{sign}(e_j)$ in (4a). For the initial states, we suggest using $x_j(t_0) = u_j(t_0)$. Let $p_{\text{error}} > 0$ and $p_{\text{rate}} > L$ denote the desired maximum error and minimum convergence rate, with L from assumption 2. Using these specifications, the control parameters are defined as

$$\epsilon = p_{\text{error}}/d(\mathcal{G}) \tag{21a}$$

$$\tau = \alpha/p_{\text{rate}} \tag{21b}$$

Although we let $\alpha \in (0, 1)$, we usually select $\alpha = 0.5$.

There are two basic rules to select the sampling period. It can be either a function of the convergence rate or a function of the desired tracking error. To ensure convergence and that both specification are met, the sampling period t_s should be

$$t_s \leq \min(\epsilon\ ,\ \tau/100) \tag{22}$$

Naturally, if the sampling period t_s is pre-defined, good choices of ϵ and τ are

$$\epsilon \geq t_s \tag{23a}$$

$$\tau \geq 100\, t_s \tag{23b}$$

There is a margin on τ, such that the 10^2 factor can be relaxed to 10 without much impact on performance.

Finally, we stress that it is essential to select an appropriate sampling period. Otherwise, there might be convergence issues that may hinder the algorithm performance. If reducing the sampling period or increasing τ are not viable options, one might actually lower the parameter τ. Although this seems counterintuitive, it mitigates the problem, at the expense of increasing chattering.

4. SIMULATIONS

In this section, we illustrate the proposed consensus algorithm properties through two numerical simulations. The first one serves to illustrate the algorithm tracking performance and its convergence properties, while the second displays its robustness to the network size. The graphs topologies for these simulations are illustrated in fig. 2. Note that, regarding the network connectivity, the second topology is the worst possible, since every node has the same eccentricity, which equals the network diameter $d(\mathcal{G}_2) = 99$.

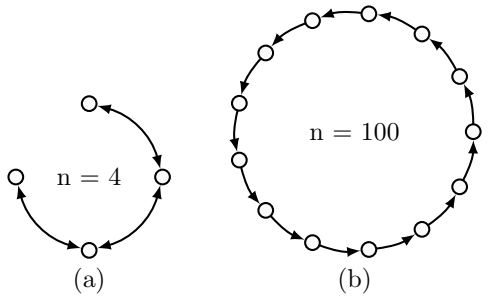

Fig. 2. Topology of the two graphs, $\mathcal{G}_a$ on the left and $\mathcal{G}_b$ on the right, considered in the simulations. For simplicity, we draw only 13 nodes of $\mathcal{G}_b$, but the actual simulation runs with 100 nodes.

As suggested in remark 2, all simulations are solved using the trapezoidal integration rule.

4.1 Small Network

The first simulation consists of a graph $\mathcal{G}_a$ with four nodes connected as in fig. 2(a), each with a sinusoidal input

$$u_j(t) = \sin(2\pi t/j) \tag{24}$$

Nodes are numbered starting from the topmost node in fig. 2(a) and increase clockwise until the last node is reached. The target consensus value $\max(u)$ is shown in fig. 3. All parameters, together with those of the other

simulation, are listed in table 1. The control parameters are chosen according to (21) and (22) to achieve a desired maximum consensus error $p_{\text{error}} = 0.01$ and a desired minimum convergence rate $p_{\text{rate}} = 2\pi$.

Table 1. Parameters used to run the simulations on graphs $\mathcal{G}_a$ and $\mathcal{G}_b$ of fig. 2.

	τ	α	ϵ	t_s
Simulation of $\mathcal{G}_a$	$8 \cdot 10^{-2}$	$5 \cdot 10^{-1}$	$2.5 \cdot 10^{-3}$	10^{-4}
Simulation of $\mathcal{G}_b$	$5 \cdot 10^{-2}$	$5 \cdot 10^{-1}$	10^{-3}	10^{-4}

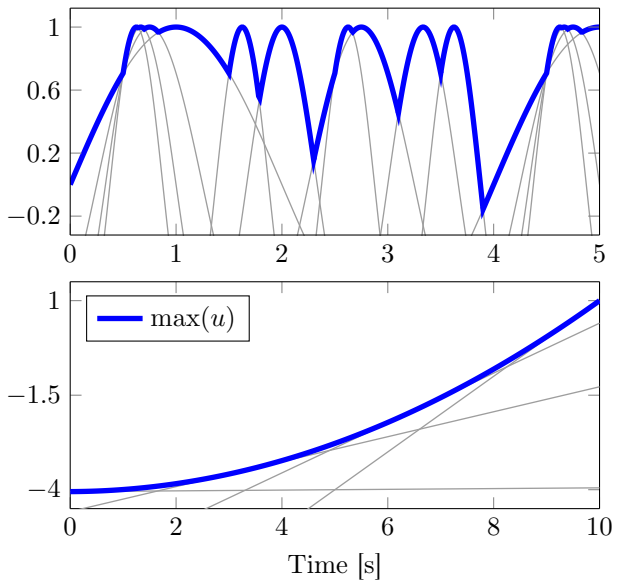

Fig. 3. Maximum consensus of networks $\mathcal{G}_a$ and $\mathcal{G}_b$ shown in fig. 2, with inputs (24) and (25). In blue the value of $\max(u)$ and in light gray the nodes inputs u_j. Only the inputs u_1, u_{33}, u_{66}, and u_{100} are shown for $\mathcal{G}_b$.

4.2 Large Network with Sparse Connectivity

This simulation consists of a graph $\mathcal{G}_b$ with one hundred nodes connected as in fig. 2(b), labeled clockwise, with ramp inputs

$$u_j(t) = \left(\frac{j}{n}\right) t + b_j \tag{25a}$$

$$b_{j-1} = \left(\frac{j-1}{jn}\right) t_f + b_j \tag{25b}$$

$$b_n = 1 - t_f \quad , \quad t_f = 10 \tag{25c}$$

Out of curiosity, note that $\lim_{n\to\infty} \max(u(t)) = [(t/t_f)^2 - 1]\, t_f/2 + 1$, for $t \in [0, t_f]$, with $u(t)$ from (25), is a parabola. The target consensus value $\max(u)$ is shown in fig. 3. All control and simulation parameters are listed in table 1. The control parameters are chosen according to (21) and (22) to achieve a maximum consensus error $p_{\text{error}} = 0.1$ and a minimum convergence rate $p_{\text{rate}} = 10$.

4.3 Simulation Results

The simulation results are shown in fig. 4 for both scenarios. Note that, as expected, during both simulations all nodes estimates converge to the maximum network input and proceed to track this value afterwards. The bottom graphs display the tracking errors, which remain smaller than the prescribed values of 0.01 and 0.1.

Analyzing the results from the first simulation, we observe some of the convergence properties of the proposed consensus algorithm. A lot can be said about the interaction between nodes 2 (in solid black), 3 (in dashed black), and 4 (in solid red). Focusing on a short frame at the begining of the simulation, one can study several convergence properties discussed in the proof of lemma 1, and also get a good feeling of the algorithm transient behavior.

When the simulation starts, node 4 knows no higher estimate than its own, and, hence, x_4 tracks its own input, since $\max(\mathcal{U}_4) = u_4$. Meanwhile, not only does node 3 knows an estimate higher than its own, $\max(\mathcal{X}_3) = x_2$, but also $\max(\mathcal{U}_3) = u_2 > x_3$. Thus, node's 3 estimate increases at a rate $(2 + \alpha)/\tau$. At approximately $t = 8.4$ ms x_3 surpasses x_4, and node 4 will then start increasing its estimate at a rate α/τ, since $\max(\mathcal{X}_4) = u_4 < x_4$, and follow x_3. Shortly after, at approximately $t = 11$ ms, x_3 surpasses $\max(\mathcal{U}_3) = x_2$, node's 3 highest known maximum input, and x_3 will then start increasing at a rate α/τ, the same as x_4. Both estimates continue to grow until x_3 reaches the ϵ-vicinity below x_2 at approximately $t = 157$ ms. From this moment on, the network reaches a consensus and all estimates track $\max(u)$.

The results of the second simulation illustrate a phenomenon that, although very unlikely, can occur on some occasions. It is the error propagation from one node to another across a long path in the network, in this case, across the entire network. In the current example, it happens because $\max(u)$ is always changing from u_j to u_{j+1} as time goes by. These consecutive changes imply that x_{j-1} is always chasing x_j. Since all estimates are increasing at the same rate, without ever reaching one another, the final consensus error for each node becomes

$$|\max(u) - x_j|_{t\geq 10} \approx \delta(j, 100)\, r\, \epsilon = (100 - j)\, r\, \epsilon$$

where $\delta(j, 100)$ is the distance from node j to node 100 and $r \in (0, 1)$ a ratio which determines the separation between each estimate. On this simulation scenario, we have $r \approx 0.7$. On average, the overall final consensus errors is reduced if the node inputs of this second example are randomly reordered or, even better, if more edges are added to the network, such that the network diameter $d(\mathcal{G}_2)$ decreases.

5. CONCLUSION

We have presented a dynamic maximum consensus algorithm that works with directed and strongly connected networks. To the best of our knowledge, this problem had remained hitherto unsolved for dynamic consensus. Both theoretically and through simulations, we have demonstrated the algorithm finite-time convergence and stability properties. From a practical perspective, the algorithm is promising since it requires tuning only three parameters, which we describe as functions of two usual design constraints: convergence rate and maximum tracking error.

There are many directions for future works for the proposed technique. We are currently developing a dual approach that finds the minimum input in a network, and we are using both of these techniques to perform scalarization

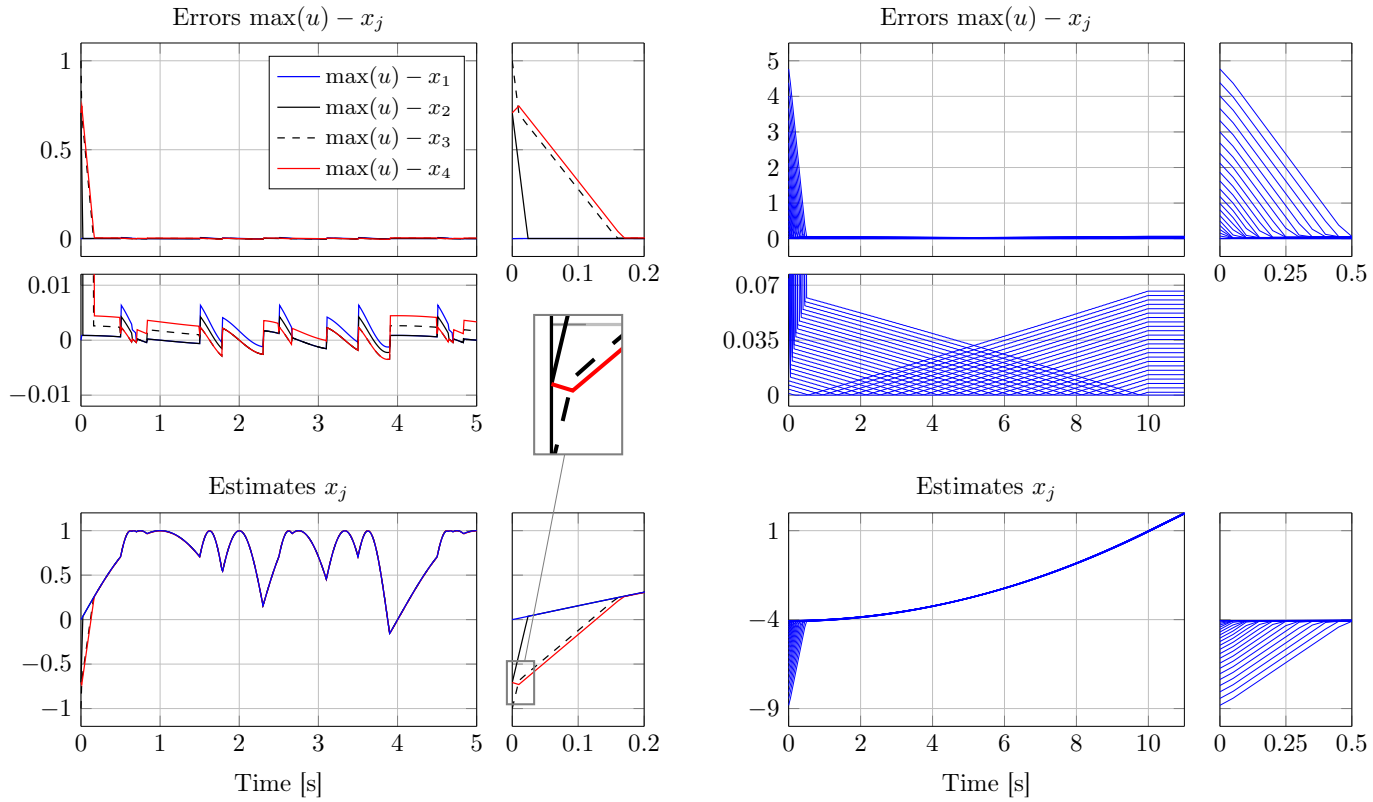

Fig. 4. Simulation results for networks $\mathcal{G}_a$ and $\mathcal{G}_b$, with inputs (24) and (25), using the proposed consensus algorithm (4). To the left, the solution of $\mathcal{G}_a$. To the right, the solution of $\mathcal{G}_b$, where only 25 out of the 100 nodes are shown.

of multiple objectives and solve distributed optimization problems in real-time. Other exciting possibilities for future development are adapting the technique to preserve privacy, hiding a node's input from its neighbors, testing the algorithm robustness to topology changes, network delays, and robustness to malicious attacks.

ACKNOWLEDGEMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

REFERENCES

Bauso, D., Giarré, L., Pesenti, R., 2009. Consensus for networks with unknown but bounded disturbances. SIAM J. on Control and Optim. 48 (3), 1756–1770.

Filippov, A. F., 1964. Differential equations with discontinuous right-hand side. American Math. Soc. Translations 42 (2), 199–231.

He, J., Chen, J., Cheng, P., Cao, X., 2014a. Secure time synchronization in wirelesssensor networks: A maximumconsensus-based approach. IEEE Trans. on Parallel and Distributed Syst. 25 (4), 1055–1065.

He, J., Cheng, P., Shi, L., Chen, J., Sun, Y., 2014b. Time synchronization in wsns: A maximum-value-based consensus approach. IEEE Trans. Auto. Control 59 (3), 660–675.

Kia, S. S., Cortés, J., Martinez, S., 2015. Dynamic average consensus under limited control authority and privacy requirements. Int. J. of Robust and Nonlinear Control 25 (13), 1941–1966.

Kia, S. S., Van Scoy, B., Cortes, J., Freeman, R. A., Lynch, K. M., Martinez, S., 2019. Tutorial on dynamic average consensus: The problem, its applications, and the algorithms. IEEE Control Syst. Mag. 39 (3), 40–72.

Manitara, N. E., Hadjicostis, C. N., 2013. Privacy-preserving asymptotic average consensus. In: European Control Conf. IEEE, pp. 760–765.

Mo, Y., Murray, R. M., 2016. Privacy preserving average consensus. IEEE Trans. Auto. Control 62 (2), 753–765.

Moradian, H., Kia, S. S., 2018. On robustness analysis of a dynamic average consensus algorithm to communication delay. IEEE Trans. on Control of Network Sys. 6 (2), 633–641.

Olfati-Saber, R., Fax, J. A., Murray, R. M., 2007. Consensus and cooperation in networked multi-agent systems. Proceedings of the IEEE 95 (1), 215–233.

Olfati-Saber, R., Murray, R. M., 2004. Consensus problems in networks of agents with switching topology and time-delays. IEEE Trans. Auto. Control 49 (9), 1520–1533.

Ren, W., Beard, R. W., 2005. Consensus seeking in multiagent systems under dynamically changing interaction topologies. IEEE Trans. Auto. Control 50 (5), 655–661.

Shi, G., Johansson, K. H., 2013. Robust consensus for continuous-time multiagent dynamics. SIAM Journal on Control and Optimization 51 (5), 3673–3691.

Spanos, D. P., Olfati-Saber, R., Murray, R. M., 2005. Dynamic consensus on mobile networks. In: IFAC World Congress. Citeseer, pp. 1–6.

Wang, X., He, J., Cheng, P., Chen, J., 2019. Differentially private maximum consensus: Design, analysis and impossibility result. IEEE Trans. on Network Sci. and Eng. 6 (4), 928–939.