



HAL
open science

Average Consensus: A Little Learning Goes A Long Way

Bernadette Charron-Bost, Patrick Lambein-Monette

► **To cite this version:**

Bernadette Charron-Bost, Patrick Lambein-Monette. Average Consensus: A Little Learning Goes A Long Way. 2020. hal-03044169

HAL Id: hal-03044169

<https://hal.science/hal-03044169>


Preprint submitted on 7 Dec 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Average Consensus: A Little Learning Goes A Long Way

Bernadette Charron-Bost^{*†}
charron@lix.polytechnique.fr

Patrick Lambein-Monette[†]
patrick@lix.polytechnique.fr
 <https://orcid.org/0000-0002-9401-8564>

Abstract When networked systems of autonomous agents carry out complex tasks, the control and coordination sought after generally depend on a few fundamental control primitives. Chief among these primitives is *consensus*, where agents are to converge to a common estimate within the range of initial values, which becomes *average consensus* when the joint limit should be the average of the initial values. To provide reliable services that are easy to deploy, these primitives should operate even when the network is subject to frequent and unpredictable changes. Moreover, they should mobilize few computational resources so that low powered, deterministic, and anonymous agents can partake in the network.

In this stringent adversarial context, we investigate the distributed implementation of these primitives over networks with bidirectional, but potentially short-lived, communication links. Inspired by the classic EqualNeighbor and Metropolis agreement rules for multi-agent systems, we design distributed algorithms for consensus and average consensus, which we show to operate in polynomial time in a synchronous temporal model. These algorithms are fully distributed, requiring neither symmetry-breaking devices such as unique identifiers, nor global control or knowledge of the network. Our strategy consists in making agents learn simple structural parameters of the network – namely, their largest degrees – which constitutes enough information to build simple update rules, implementable locally with little computational and memory overhead.

1 Introduction

1.1 Controlling networked systems

FROM an initial state of discord, a group is said to reach agreement when everyone eventually adopts the same opinion. This problem is central in multi-agent systems whether they be natural or artificial. Examples of the latter kind include flocks of autonomous vehicles or drones, distributed ledgers, smart power grids, decentralized contact-tracing apps, and sensors for the remote monitoring of environmental or industrial parameters, to name some vibrant areas in which applications are being developed.

We consider a networked system of n *agents* – a generic term we use to denote the autonomous nodes of the network regardless of their nature: robots, cellular phones, sensors... – denoted by the integer labels $1, \dots, n$. Each agent i starts with a value μ_i , which we call its *input* and is taken arbitrarily from the domain of the problem, assumed here to be the set of real numbers \mathbb{R} . The input represents the private observation made by the agent of some aspect of

^{*}CNRS

[†]LIX, École polytechnique, Institut Polytechnique de Paris

the environment that is relevant to the task at hand: for example a temperature reading, or the direction, speed, or location of the agent if it is mobile. We focus on two flavors of a control primitive where the entire network strives to reach a common value that is compatible with the input values μ_1, \dots, μ_n in a sense we now make precise.

The first one is *consensus*: each agent i maintains over time its own estimate $x_i(t)$ of the objective, and the estimates should asymptotically converge to some common limit ω within the range of the input values – we say that the estimates achieve *asymptotic consensus* over ω . Depending on the system being modeled, the estimate may be an internal variable storing an intermediate result of some long computation, but it may also directly measure some relevant parameter controlled by the agent, such as its speed or position.

Among many practical applications, a consensus primitive can serve to coordinate mobile agents: have them regroup, adopt a common heading, or control their relative positions while moving together – problems formally known as *rendez-vous* [1, 34], *flocking* [37, 17, 12], and *formation control* [2, 30]. Examples not grounded in mobile agents include implementing a distributed clock [19, 38, 29, 32], or indeed in natural systems to get cardiac pacemaker cells working in concert to produce heartbeats [20, 14] or fireflies to flash in unison [4, 20].

The second primitive, *average consensus*, adds the constraint that the common limit ω should be the arithmetic mean of the input values $\bar{\mu} = \frac{1}{n}(\mu_1 + \dots + \mu_n)$. This is sometimes required for specific applications such as sensor fusion [40], load balancing [13, 16], or distributed optimization and machine learning [26, 23].

We look at these primitives in the adversarial context of *dynamic networks*, where the communication links joining the agents evolve over time. Indeed, agreement primitives are often required in settings that are inherently dynamic and inadequately described by static networks: we cite for example mobile ad hoc networks, where the links change as the agents move in space due to external factors; autonomous vehicular networks, where again change is caused by mobility, but which results this time from our control; or peer-to-peer networks, which continuously reconfigure as agents join and leave.

A standard approach to consensus has agents regularly adjust their estimates as a convex combination of those of neighboring agents, defined by a *convex update rule*. We adopt a temporal model of *synchronized rounds*, and this can be rephrased, in each round t and for each agent i , as an update of the general form $x_i(t) = \sum_{j \in \mathcal{N}_i(t)} a_{ij}(t)x_j(t-1)$, where the weights $a_{ij}(t)$ are taken to form a convex combination, and the sum is over the incoming neighbors of agent i in the communication graph in round t .

1.2 Contribution

We study such updates from a computational angle: what sort of weights $a_{ij}(t)$ can be computed locally by the agents and produce good behavior? For a simple example, given enough connectivity, average consensus comes easily if we make each agent i pick $a_{ij}(t) = \frac{1}{n}$ for its proper neighbors j , and picks $a_{ii}(t) = \frac{d_i(t)-1}{n}$ for itself, where the degree $d_i(t)$ is the count of its neighbors, itself included, in the communication graph in round t . However simple this scheme is to describe, picking these weights in a distributed manner requires agents to know n , but evaluating the network size in a distributed manner is itself an entirely non-trivial feat. We will argue that the Metropolis rule – where the weights for neighbors

neighboring agents i and j are given by $a_{ij}(t) = \frac{1}{\max(d_i(t), d_j(t))}$ – breaks down over dynamic networks because of similar issues. On the other hand, implementing the EqualNeighbor weights $a_{ij}(t) = \frac{1}{d_i(t)}$ poses no problem, but convergence over dynamic networks can be dramatically slower than in the static case.

Our main contribution consists in a pair of algorithms for consensus and average consensus, the MaxWeight and MaxMetropolis algorithms, which operate over dynamic networks as long as the communication links are bidirectional, but without relying on some central control (e.g., identifiers or leaders) or global knowledge (e.g., a bound on the network size). The corresponding update rules depend on structural parameters of the dynamic network, learned along the way by each agent – namely, its largest degree in the communication topology. Our strategy induces a moderate delay in the temporal complexity when compared to the static case for the EqualNeighbor and Metropolis rules: these rules have respective temporal bounds in $O(n^3 \log n/\varepsilon)$ and $O(n^2 \log n/\varepsilon)$, while both our algorithms admit a bound in $O(n^4 \log n/\varepsilon)$.

We establish these complexity bounds with geometric arguments relying on spectral graph theory, but each update rule is subject to its own specific challenges. The MaxMetropolis update is not convex throughout the entire execution, causing the system to away from consensus, and inducing a delay that we have control. The MaxWeight update faces a subtler issue: different geometries best quantify its progress at different times, but switching geometries wipes off much of said progress, and we have to show that the succession of those switches cannot stall convergence too much.

1.3 Related works

Convex update rules are closely related to random walks over finite graphs, whose origins go back to the development of the field of probability theory itself. For a formulation and purpose that resemble ours, we mention early works in opinion dynamics by DeGroot [15], Chatterjee [9], and Chatterjee and Seneta [10], followed later by works on distributed control by Tsitsiklis [35] and Tsitsiklis et al. [36]. Modern interest in the matter, and especially in the EqualNeighbor rule, can be traced back to the work on the simulation of flocking done by Reynolds [31] and then by Vicsek et al. [37], followed by analytical analyses by Jadbabaie et al. [18] and by Olfati-Saber and Murray [25], after which the sustained attention that the topic has received precludes anything resembling a complete bibliographical review; see [39, 21, 3, 17, 5, 27, 24, 11, 8, 22] for a few more milestones.

Update rules for consensus and average consensus generally require the weights $a_{ij}(t)$ to be non-negative so that they form a convex combination. A notable exception is found in [39], where Xiao et al. look for the *fixed* weights a_{ij} that optimize the speed of convergence over a given fixed graph, and find that the weights may be negative. The MaxMetropolis algorithm is itself able to solve average consensus over dynamic networks precisely because its update weights can sometimes be negative. When compared with our approach, the important difference is that we consider dynamic graphs and focus on distributed implementation of the update rules, while the weights obtained in [39] are given by a centralized optimization problem and are incompatible with a distributed approach.

A number of strategies aim at speeding up convex update rules over static

networks by having the agents learn what amounts to spectral elements of the graph laplacian [6], which can dramatically improve convergence [33]. Like our own algorithms, these represent distributed methods by which the agents learn structural properties of the communication graph. However, these methods need agents to be issued unique identifiers, and they are gluttonous in memory and heavy in computation, with agents computing and memorizing, in each rounds, kernels of Hankel matrices of dimension $\Theta(n) \times \Theta(n)$. In contrast, our method works with anonymous agents, uses $\lceil \log n \rceil$ bits of memory and bandwidth, and has a trivial computational overhead.

Average consensus update rules for dynamic networks were notably studied by Olshevsky and Tsitsiklis [27] – which defined the convergence time and convergence rate of an algorithm – a work later expanded together with Nedić and Ozdaglar in [24]. These analytical approaches focus on aspects such as the temporal complexity and tolerance to quantization, whereas we address issues of a distributed nature, in particular the implementation of rules by distributed algorithms.

We now proceed along the following structure. In Section 2, we recall some mathematical terminology and known results, and detail our computational model. Section 3 presents update rules, in particular the EqualNeighbor rule for consensus, and the Metropolis rule for average consensus, and discusses their limitations. We give the MaxWeight consensus algorithm in Section 4, detailing the underlying learning strategy, which we then apply in Section 5 to the average consensus problem with the MaxMetropolis algorithm. We conclude in with a discussion in Section 6.

2 Preliminaries

2.1 Mathematical toolbox

Let us fix some notation. If k is a positive integer, we denote by $[k]$ the set $\{1, \dots, k\}$. If S is any non-empty finite set of real numbers, we denote its *diameter* by $\text{diam } S := \max S - \min S$.

A *directed graph* (or simply *graph*) $G = (V, E)$ with vertices in V and edges in $E \subseteq V \times V$ is called *reflexive* when $(i, i) \in E$ for all $i \in V$; G is *bidirectional* when $(i, j) \in E \iff (j, i) \in E$ for all $i, j \in V$; and G is *strongly connected* when directed paths join any pair of vertices.

All graphs that we consider here will be reflexive, bidirectional, and strongly connected graphs of the form $G = ([n], E)$. In such a graph, the vertices linked to a vertex i form its *neighborhood* $\mathcal{N}_i(G) := \{j \in [n] \mid (j, i) \in E\}$, and the count of its neighbors is its *degree* $d_i(G) := |\mathcal{N}_i(G)|$. By definition, the degree is at most n , and in a reflexive graph it is at least 1.

Matrices and vectors will be consistently denoted in bold italic style: upper case for matrices (\mathbf{A}, \mathbf{B}) and lower case for vectors (\mathbf{u}, \mathbf{v}), with their individual entries in regular italic style: A_{ij}, B_{jk}, u_i, v_j . We use the shorthand $\mathbf{v}^{\mathbb{N}}$ to denote the infinite vector sequence $\mathbf{v}(0), \mathbf{v}(1), \dots$.

The graph $G_{\mathbf{A}} = ([n], E)$ *associated* to a matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is defined by $(j, i) \in E \iff A_{ij} \neq 0$ for all $i, j \in [n]$. The matrix \mathbf{A} is said to be *irreducible* when $G_{\mathbf{A}}$ is strongly connected.

Given a vector $\mathbf{v} \in \mathbb{R}^n$, we write $\text{diam } \mathbf{v}$ to mean the diameter of the set $\{v_1, \dots, v_n\}$ of its entries. The diameter constitutes a seminorm over \mathbb{R}^n ; in particular, we have $\text{diam } \mathbf{v} \geq 0$. We call *consensus vectors* those of null diameter;

they form exactly the linear span of the constant vector $\mathbf{1} := (1, 1, \dots, 1)^\top$. The *identity matrix* \mathbf{I} is the diagonal matrix $\text{diag}(\mathbf{1})$.

A matrix or a vector with non-negative (resp. positive) entries is itself called *non-negative* (resp. positive). A non-negative vector is called *stochastic* if its entries sum to 1. A non-negative matrix \mathbf{A} is *stochastic* if *each of its rows* sums to 1 – equivalently, if $\mathbf{A}\mathbf{1} = \mathbf{1}$. If moreover its transpose \mathbf{A}^\top is stochastic, \mathbf{A} is *doubly stochastic*, and it is the case in particular when \mathbf{A} is *symmetric*.

We denote the mean value of a vector $\mathbf{v} \in \mathbb{R}^n$ by $\bar{v} := \frac{1}{n} \sum_i v_i$. Doubly stochastic matrices play a central role in the study of average consensus, as multiplying any vector \mathbf{v} by a doubly stochastic matrix \mathbf{A} preserves its average – that is, $\overline{\mathbf{A}\mathbf{v}} = \bar{v}$.

For any matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, we can arrange its n eigenvalues $\lambda_1, \dots, \lambda_n$, counted with their algebraic multiplicities, in decreasing order of magnitude: $|\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n|$. Under this convention, the *spectral radius* of the matrix \mathbf{A} is the quantity $\rho_{\mathbf{A}} := |\lambda_1|$, and its *spectral gap* is the quantity $\gamma_{\mathbf{A}} := |\lambda_1| - |\lambda_2|$. In particular, a stochastic matrix has a spectral radius of 1, which is itself an eigenvalue that has $\mathbf{1}$ for eigenvector.

2.2 Computing model

We consider a networked system of n agents, denoted $1, 2, \dots, n$. Computation proceeds in *synchronized rounds* that are communication closed, in the sense that no agent receives messages in round t that are sent in a different round. In round t ($t = 1, 2, \dots$), each agent i successively **a)** broadcasts a *single* message $m_i(t)$ determined by its state at the beginning of round t ; **b)** receives *some* of the messages $m_1(t), \dots, m_n(t)$; **c)** undergoes an internal transition to a new state; and **d)** produces a *round output* $x_i(t) \in \mathbb{R}$ and proceeds to round $t + 1$. Communications that occur in round t are modeled by a directed graph $\mathbb{G}(t) := ([n], E(t))$, called the round t *communication graph*, which may change from one round to the next. We assume each communication graph $\mathbb{G}(t)$ to be reflexive, as agents always know their own messages.

Messages to be sent in step **a)** and state transitions in step **c)** are determined by a *sending* and a *transition* functions, which together define the *local algorithm* for agent i . Collected together, the local algorithms of all agents in the system constitute a *distributed algorithm*.

An *execution* of a distributed algorithm is a sequence of rounds, as defined above, with each agent running the corresponding local algorithm. We assume that all agents start simultaneously in round 1, since the algorithms under our consideration are robust to asynchronous starts, retaining the same time complexity as when the agents start simultaneously. Indeed, asynchronous starts only induce an initial transient period during which the network is disconnected, which cannot affect the convergence and complexity results of algorithms driven by convex update rules.

The entire sequence $\mathbf{x}^{\mathbb{N}}$ in any execution of an algorithm is entirely determined by the input vector $\boldsymbol{\mu}$ and the patterns of communications in each round t , i.e., the sequence of communication graphs $\mathbb{G} := (\mathbb{G}(t))_{t \geq 1}$, called the *dynamic communication graph* of the execution, and so we write $\mathbf{x}^{\mathbb{N}} = \mathbf{x}^{\mathbb{N}}(\mathbb{G}, \boldsymbol{\mu})$. When the dynamic graph \mathbb{G} is understood, we let $\mathcal{N}_i(t)$ and $d_i(t)$ respectively stand for $\mathcal{N}_i(\mathbb{G}(t))$ and $d_i(\mathbb{G}(t))$. As no confusion can arise, we will sometimes identify an agent with its corresponding vertex in the communication graph, and speak

of the degree or neighborhood of an *agent* in a round of an execution.

We call a *network class* a set of dynamic graphs; given a class \mathfrak{C} , we denote by $\mathfrak{C}|_n$ the sub-class $\{\mathbb{G} \in \mathfrak{C} \mid |\mathbb{G}| = n\}$. In the rest of the paper, we will center our investigation on the class \mathfrak{G} of dynamic graphs of the following sort.

Assumption 1 *In each round $t \in \mathbb{N}_{>0}$, the communication graph $\mathbb{G}(t)$ is reflexive, bidirectional, and strongly connected.*

3 Update rules for asymptotic consensus

Our approach distinguishes local algorithms, as defined above, from the *update rules* that they implement: the latter are recurring formulas describing how the estimates $x_i(t)$ change over time, while the former specify the distributed implementation of such rules with local algorithms – that is, with each agent only using the information that is locally available. When we consider the EqualNeighbor rule, which involves purely local parameters and directly corresponds to a local algorithm, this distinction may appear exacting. However, it becomes crucial when we consider the Metropolis rule, which is easily defined, but challenging to implement because of its dependence on information in the neighborhood at distance 2 of each agent. Collecting this information in a distributed manner can only be done over specific graph classes, in a manner that does not generalize to the entire class \mathfrak{G} .

3.1 Update rules and convergence

We will focus on distributed algorithms which realize, in their executions, affine update rules of the general form

$$x_i(t) = \sum_{j \in \mathcal{N}_i(t)} a_{ij}(t) x_j(t-1) \quad , \quad \text{with} \quad \sum_{j \in \mathcal{N}_i(t)} a_{ij}(t) = 1 \quad , \quad \triangleright 1$$

where the time-varying affine weights $a_{ij}(t)$ may depend on the dynamic graph \mathbb{G} and the input vector $\boldsymbol{\mu}$. We then say that the algorithm *implements* the rule, but we insist again that *an algorithm is not the same thing as a rule*.

Under our assumptions, the convergence of update rules, including all those we consider here, is ensured by the following classic result, found in the literature under various forms [10, 35, 18, 21].

Proposition 1 *Let $\mathbf{x}^{\mathbb{N}}$ be a vector sequence satisfying the recurrence relation eq. (1) for weights $a_{ij}(t) \geq \alpha > 0$ – that is, the weights are positive and uniformly bounded away from 0. Under Assumption 1, the vectors $\mathbf{x}(t)$ converge to a consensus vector.*

We speak of *uniform convexity* when such a parameter $\alpha > 0$ exists, and we note that in that case Assumption 1 is much stronger than is necessary to achieve asymptotic consensus: as shown in [21], it suffices that the network never become permanently split.

We measure progress towards consensus with the *convergence time*: for a single sequence $\mathbf{z}^{\mathbb{N}}$, it is given for any $\varepsilon > 0$ by $T^{\mathbf{z}^{\mathbb{N}}}(\varepsilon) := \inf\{t \in \mathbb{N} \mid \forall \tau \geq t : \text{diam } \mathbf{z}(\tau) \leq \varepsilon\}$. For an update rule or an algorithm, we measure in fact the *worst-case* convergence time over a class \mathfrak{C} , defined for a system of size n by:

$$T_n^{\mathfrak{C}}(\varepsilon) := \sup_{\mathbb{G} \in \mathfrak{C}|_n, \boldsymbol{\mu} \in \mathbb{R}^n} T^{\mathbf{x}^{\mathbb{N}(\mathbb{G}, \boldsymbol{\mu})}}(\varepsilon \cdot \text{diam } \boldsymbol{\mu}) \quad . \quad \triangleright 2$$

By default, we will consider the class \mathfrak{G} , dropping the superscript and writing $T_n(\varepsilon)$ for $T_n^{\mathfrak{G}}(\varepsilon)$.

3.2 Example rules

3.2.1 EqualNeighbor

The prototypical example of an update rule is the *EqualNeighbor* rule, where the next estimate of an agent is the unweighted average of those of its incoming neighbors:

$$x_i(t) = \frac{1}{d_i(t)} \sum_{j \in \mathcal{N}_i(t)} x_j(t-1) . \quad \triangleright 3$$

This rule directly translates into an implementing local algorithm: an agent broadcasts its latest estimate in every round, and picks for new estimate the average of its incoming values. Since an agent's degree is at most n , the EqualNeighbor rule is uniformly convex with parameter $\alpha = 1/n$, and under Assumption 1 this algorithm solves consensus over the class \mathfrak{G} .

However, the convergence *time* is poor over the class \mathfrak{G} , where [28, Proposition 12] gives a *lower* bound of $T_n(\varepsilon) \geq 2^{\Omega(n)} \log 1/\varepsilon$. The convergence time is improved when the network displays additional structure: over the sub-class of \mathfrak{G} of *fixed* graphs, Olshevsky and Tsitsiklis [27, Corollary 5.2] use a result from Landau and Odlyzko to show a tight bound in $O(n^3 \log n/\varepsilon)$. This bound extends to the sub-class \mathfrak{G} of dynamic graphs for which each vertex has a fixed *degree* [11, Theorem 1.6].

Local update rules such as eq. (1) admit an equivalent matricial form. For the EqualNeighbor rule, as an example, eq. (3) is equivalent to the global rule $\mathbf{x}(t) = \mathbf{W}(\mathbb{G}(t))\mathbf{x}(t-1)$, with the EqualNeighbor matrix $\mathbf{W}(G)$ given for any graph G by

$$[\mathbf{W}(G)]_{ij} = \begin{cases} 1/d_i(G) & j \in \mathcal{N}_i(G) \\ 0 & j \notin \mathcal{N}_i(G) \end{cases} . \quad \triangleright 4$$

We note that this matrix is stochastic for any graph G , and has G for associated graph.

3.2.2 Metropolis

Let us call the *Metropolis-Hastings symmetrization* the transform $\mathbf{A} \mapsto \mathbf{M}(\mathbf{A})$ which, to any square matrix \mathbf{A} , associates the matrix given by

$$[\mathbf{M}(\mathbf{A})]_{ij} = \begin{cases} \min(A_{ij}, A_{ji}) & j \neq i \\ 1 - \sum_{k \neq i} \min(A_{ik}, A_{ki}) & j = i \end{cases} . \quad \triangleright 5$$

By construction, the matrix $\mathbf{M}(\mathbf{A})$ is symmetric and leaves consensus vectors invariant. Outside the main diagonal, we have $[\mathbf{M}(\mathbf{A})]_{ij} \leq A_{ij}$, and thus on the main diagonal we have $[\mathbf{M}(\mathbf{A})]_{ii} \geq A_{ii}$; the matrix $\mathbf{M}(\mathbf{A})$ is therefore doubly stochastic whenever the matrix \mathbf{A} is stochastic.

Xiao et al. [40] propose to approach the average consensus problem with the *Metropolis* update rule:

$$x_i(t) = x_i(t-1) + \sum_{j \in \mathcal{N}_i(t)} \frac{x_j(t-1) - x_i(t-1)}{\max(d_i(t-1), d_j(t-1))} ; \quad \triangleright 6$$

when viewed at the scale of the system, this rule corresponds to symmetrizing the EqualNeighbor rule with eq. (5) round-wise, hence its name. Proposition 1 ensures asymptotic consensus of the Metropolis rule over the entire class \mathfrak{G} as

it did for the EqualNeighbor rule, and since the EqualNeighbor matrices are stochastic, the Metropolis matrices are doubly stochastic, and the Metropolis rule results in fact in *average* consensus, with a convergence time in $O(n^2 \log n/\varepsilon)$ over the class \mathfrak{G} .

Unfortunately, no local algorithm is able to implement the Metropolis rule over the class \mathfrak{G} : the rule is local only in the weak sense that an agent's next estimate $x_i(t)$ depends on information present *within distance* 2 of agent i in round t , which is not local enough when the network is subject to change.

Indeed, since agent j only knows its round t degree $d_j(t)$ at the end of round t , it has to wait until round $t + 1$ to share this information with its neighbors. Any distributed implementation of this rule would require the communication links to evolve at a slow and regular pace. As an example, we may assume that the links only change at rounds t for which $t \equiv r \pmod k$ – e.g., at even rounds. Such conditions are all the more limitative in that they additionally require all agents to be loosely synchronized, as they have to agree on k and on the current round number – at least modulo k .

The situation is even worse when the network is subject to unpredictable changes, as we need to warn all agents, ahead of time, about any upcoming topology change. In effect, this amounts to having a global synchronizing signal precede every change in the communication topology. For a topology changing in round t_0 , this differs little from starting an *entirely new execution* with $(x_1(t_0 - 1), \dots, x_n(t_0 - 1))$ for new input.

To paraphrase, provided the dynamic communication graph is sufficiently stable, one “can” implement the Metropolis rule over dynamic networks, but the execution is fully distributed only as long as no change occurs.

4 Stabilizing weights for consensus

Consider the update rule given by

$$x_i(t) = x_i(t - 1) + \frac{1}{q_i} \sum_{j \in \mathcal{N}_i(t)} (x_j(t - 1) - x_i(t - 1)) , \quad \triangleright 7$$

which we call the FixedWeight rule for the parameters $q_1, \dots, q_n > 0$. When $d_i(t) \leq q_i$, it acts as a sort of lazy version of the EqualNeighbor rule, where agent i gives more importance to its own estimate $x_i(t - 1)$ than to those of its neighbors. Over the sub-class $\mathfrak{C} \subset \mathfrak{G}|_n$ of dynamic graphs for which $d_i(t) \leq q_i$ holds for all agents at all times, the FixedWeight rule with parameters q_1, \dots, q_n is shown in [11, Theorem 1.6] to solve consensus with a convergence time in $T_n^{\mathfrak{C}}(\varepsilon) = O(\sum_i q_i \cdot n \log n/\varepsilon)$. In particular, using $q_1 = q_2 = \dots = q_n = n$ yields a bound in $O(n^3 \log n/\varepsilon)$, comparable to the EqualNeighbor rule over *static* networks.

Unfortunately, the ability for the agents to pick good values for the parameters q_1, \dots, q_n is limited by what they know, ahead of time, about the structure of the communication graph: parameters that are too small risk breaking the degree condition $d_i(t) \leq q_i$ and cause the system to diverge, while parameters that are too large make convergence unacceptably slow if the network has a small degree.

Instead of relying on exogenous parameters, incompatible with a distributed approach, we propose making each agent i learn by itself what value of q_i work

for the current execution. We obtain the *MaxWeight* rule by replacing the fixed parameter q_i with $d'_i(t) := \max \{d_i(1), \dots, d_i(t)\}$, at each step of eq. (7).

As each sequence $d'_i(t)$ stabilizes over its limit $d'_i := \max_t d_i(t)$, the MaxWeight rule eventually behaves like the FixedWeight rule with parameters d'_1, \dots, d'_n . However, the MaxWeight rule can be implemented over the class \mathfrak{G} with no additional assumption, resulting in the *MaxWeight* algorithm, given in Algorithm 1.

Algorithm 1 The MaxWeight algorithm, code for agent i

Input: $\mu_i \in \mathbb{R}$

1 Initially:

2 $\mathbf{x}_i \leftarrow \mu_i$

3 $\mathbf{q}_i \leftarrow 2$

4 In each round do:

5 send $m_i = \langle \mathbf{x}_i \rangle$

6 receive m_{j_1}, \dots, m_{j_d} $\triangleright d$ neighbors

7 $\mathbf{q}_i \leftarrow \max(\mathbf{q}_i, \mathbf{d})$

8 $\mathbf{x}_i \leftarrow \mathbf{x}_i + \frac{1}{\mathbf{q}_i} \sum_{k=1}^d (\mathbf{x}_{j_k} - \mathbf{x}_i)$

9 output \mathbf{x}_i

Theorem 1 *The MaxWeight algorithm solves consensus over the class \mathfrak{G} of dynamic graphs that are reflexive, bidirectional, and strongly connected in each round, with a convergence time of $T_n(\varepsilon) = O(n^4 \log n / \varepsilon)$ for a system of n agents.*

4.1 Technical preliminaries

The proof of Theorem 1 will leverage some of the apparatus developed in [7], which we restate here briefly.

4.1.1 Weighted geometries

We fix a positive stochastic vector $\boldsymbol{\pi} \in \mathbb{R}^n$, and we define the $\boldsymbol{\pi}$ -weighted Euclidian geometry with the inner product $\langle -, - \rangle_{\boldsymbol{\pi}}$ and the norm $\|-\|_{\boldsymbol{\pi}}$:

$$\langle \mathbf{u}, \mathbf{v} \rangle_{\boldsymbol{\pi}} := \sum_i \pi_i u_i v_i \quad , \quad \|\mathbf{v}\|_{\boldsymbol{\pi}} := \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle_{\boldsymbol{\pi}}} \quad . \quad \triangleright 8$$

Controlling the dispersion of the estimates in an execution of the MaxWeight algorithm will specifically rely on the *variance* induced by this geometry:

$$\text{var}_{\boldsymbol{\pi}} \mathbf{v} := \|\mathbf{v} - \langle \mathbf{v}, \mathbf{1} \rangle_{\boldsymbol{\pi}} \mathbf{1}\|_{\boldsymbol{\pi}}^2 = \|\mathbf{v}\|_{\boldsymbol{\pi}}^2 - \langle \mathbf{v}, \mathbf{1} \rangle_{\boldsymbol{\pi}}^2 \quad . \quad \triangleright 9$$

We will need to switch between different geometries throughout an execution. As a consequence, we will use the following lemma to relate variances with one another.

Lemma 1 *Let $\boldsymbol{\pi}$ and $\boldsymbol{\pi}'$ be two positive stochastic vectors of \mathbb{R}^n . For any vector $\mathbf{v} \in \mathbb{R}^n$,*

$$\text{var}_{\boldsymbol{\pi}'} \mathbf{v} \leq \max_i \frac{\pi'_i}{\pi_i} \text{var}_{\boldsymbol{\pi}} \mathbf{v} \quad . \quad \triangleright 10$$

Proof. For any vector \mathbf{u} we have by definition

$$\|\mathbf{u}\|_{\pi'}^2 \leq \max_i \frac{\pi'_i}{\pi_i} \|\mathbf{u}\|_{\pi}^2, \quad \triangleright 11$$

and in particular for the vector $\mathbf{u} := \mathbf{v} - \langle \mathbf{v}, \mathbf{1} \rangle_{\pi} \mathbf{1}$. By the definition of the variance, we then have $\text{var}_{\pi} \mathbf{v} = \|\mathbf{u}\|_{\pi}^2$ and $\text{var}_{\pi'} \mathbf{v} = \text{var}_{\pi'} \mathbf{u} \leq \|\mathbf{u}\|_{\pi'}^2$, since adding a consensus vector does not change the variance, from which our claim follows. \blacktriangleleft

Moreover, we will have to relate the variance of a vector to its diameter, used to define the convergence time. To this effect, we assume the next lemma, which generalizes Lemma 5 to the π -geometries.

Lemma 2 *Let $\pi \in \mathbb{R}^n$ be a positive stochastic vector. For any vector $\mathbf{v} \in \mathbb{R}^n$,*

$$2 \text{var}_{\pi} \mathbf{v} < (\text{diam } \mathbf{v})^2 < \frac{4}{\min_i \pi_i} \text{var}_{\pi} \mathbf{v}. \quad \triangleright 12$$

4.1.2 Reversibility and Perron-Frobenius theory

As a corollary of the celebrated Perron-Frobenius theorem, an irreducible matrix \mathbf{A} admits a unique positive stochastic vector $\pi(\mathbf{A})$ satisfying $\pi(\mathbf{A}) = \mathbf{A}^{\top} \pi(\mathbf{A})$, usually called the *Perron vector* of the matrix \mathbf{A} . We will say that a matrix is *reversible* if it is self-adjoint with respect to any π -weighted inner product, and in the case of an irreducible matrix the vector in question is necessarily its Perron vector.

By definition, a reversible matrix \mathbf{A} is subject to the spectral theorem, and so is diagonalizable with real eigenvalues. In the case of a stochastic matrix, $\mathbf{1}$ is an eigenvector belonging to the eigenvalue 1, and the Rayleigh-Ritz variational characterization of the eigenvalues of the matrix \mathbf{A} gives us the following lemma.

Lemma 3 *For an irreducible stochastic matrix \mathbf{A} that is reversible and has positive diagonal entries,*

$$\forall \mathbf{v}: \text{var}_{\pi(\mathbf{A})} \mathbf{A} \mathbf{v} \leq (1 - \gamma_{\mathbf{A}})^2 \text{var}_{\pi(\mathbf{A})} \mathbf{v}. \quad \triangleright 13$$

We will control the contraction of the estimates using Lemma 3 with the following spectral bound, originally given in [7, Corollary 7], which generalizes a previous bound from [24, Lemma 9].

Lemma 4 *For an irreducible stochastic matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ that is reversible and has positive diagonal entries,*

$$\gamma_{\mathbf{A}} \geq \frac{\alpha(\mathbf{A})}{n-1}, \quad \triangleright 14$$

where $\alpha(\mathbf{A}) := \min \{ \pi_i(\mathbf{A}) A_{ij} \mid i, j \in [n] \setminus \{0\} \}$.

4.1.3 Proof of the theorem

Proof. We fix a dynamic graph $\mathbb{G} \in \mathfrak{G}_{|n}$, and we let

$$\begin{aligned} d'_i(t) &:= \max \{ d_i(\tau) \mid 0 \leq \tau \leq t \}, & d_i &:= \max \{ d_i(t) \mid t \in \mathbb{N} \}, \\ D'(t) &:= \sum_{i=1}^n d'_i(t), & D' &:= \sum_{i=1}^n d'_i, \\ \mathcal{T}_{\ell} &:= \{ t \in \mathbb{N}_{>0} \mid \exists i \in [n] : d'_i(t) \neq d'_i(t-1) \}, \end{aligned} \quad \triangleright 15$$

using the convention $d_i(0) = 2$. The set \mathcal{T}_ℓ has cardinal at most $|\mathcal{T}_\ell| \leq \sum_{i \in [n]} d'_i$, and so we let $t_s := \max \mathcal{T}_\ell$; for all $t > t_s$, we have $d'_i(t) = d'_i(t-1)$.

Let us then pick arbitrary values $\mu_1, \dots, \mu_n \in \mathbb{R}$, and consider the sequence of estimates $\mathbf{x}^\mathbb{N} = \mathbf{x}^\mathbb{N}(\mathbb{G}, \boldsymbol{\mu})$ produced by the MaxWeight update rule for our chosen parameters:

$$\begin{aligned} x_i(0) &= \mu_i \\ x_i(t) &= x_i(t-1) + \frac{1}{d'_i(t)} \sum_{j \in \mathcal{N}_i(t)} (x_j(t-1) - x_i(t-1)) . \end{aligned} \quad \triangleright 16$$

Since we have $2 \leq d'_i(t) \leq n$ for all rounds $t \in \mathbb{N}_{>0}$, the sequence $\mathbf{x}^\mathbb{N}$ satisfies Proposition 1 with uniform convexity parameter $\alpha = 1/n$, and so it achieves asymptotic consensus within the convex hull of the set $\{\mu_1, \dots, \mu_n\}$. This shows that the MaxWeight rule, and thereby its implementing algorithm, solve the consensus problem over the class \mathfrak{G} .

It remains to control the convergence time. If we could bound the round t_s – say, by $t_s \leq f(n)$ for some function f – then we could simply reuse the result from [11] about the FixedWeight update and deduce $T_n(\varepsilon) = O(f(n) + n^3 \log n / \varepsilon)$. However, there clearly are some dynamic graphs in the class $\mathfrak{G}_{|n}$ for which t_s is arbitrary large, and we need to control the contraction of the estimates over the time window $t \in [1, \dots, t_s]$.

Let us then fix a disagreement threshold $\varepsilon > 0$, and consider the set of ε -converged rounds: $\mathcal{T}_\varepsilon := \{t \in \mathbb{N} \mid \text{diam } \mathbf{x}(t) \leq \varepsilon \cdot \text{diam } \boldsymbol{\mu}\}$. The convexity of the sequence $\mathbf{x}^\mathbb{N}$, along with the fact that it achieves asymptotic consensus, imply that this set is an unbounded interval of the integers, and we let $t_\varepsilon := \inf \mathcal{T}_\varepsilon$ denote the earliest round at which the estimates agree with a relative error of ε .

Let us denote by $\mathbf{A}(t)$ the round t MaxWeight update matrix:

$$[\mathbf{A}(t)]_{ij} = \begin{cases} \frac{1}{d'_i(t)} & i \neq j \in \mathcal{N}_i(t) \\ 1 - \frac{d_i(t)-1}{d'_i(t)} & j = i \\ 0 & j \notin \mathcal{N}_i(t) . \end{cases} \quad \triangleright 17$$

Its associated graph is $G_{\mathbf{A}(t)} = \mathbb{G}(t)$, and by Assumption 1, this matrix is irreducible, with its Perron vector $\boldsymbol{\pi}(t) := \boldsymbol{\pi}(\mathbf{A}(t))$ given by

$$\boldsymbol{\pi}(t) = (d'_1(t)/D'(t), \dots, d'_n(t)/D'(t))^\top .$$

We can verify that the matrix $\mathbf{A}(t)$ is self-adjoint for the inner product $\langle -, - \rangle_{\boldsymbol{\pi}(t)}$. As $A_{ij}(t) \geq 1/d'_i(t)$ for all positive entries of the matrix $\mathbf{A}(t)$, Lemma 3 gives us $\gamma := \inf_{t \in \mathbb{N}_{>0}} \gamma_{\mathbf{A}(t)} \geq 1/(n-1)D'$.

Let us then define the potential $\mathcal{L}(t) := \text{var}_{\boldsymbol{\pi}(t)} \mathbf{x}(t)$ for positive t . For a round $t \neq 1$ outside of \mathcal{T}_ℓ , the matrices $\mathbf{A}(t)$ and $\mathbf{A}(t-1)$ share their Perron vector, and with Lemma 6 we have

$$\mathcal{L}(t) \leq (1 - \gamma)^2 \mathcal{L}(t-1) . \quad \triangleright 18$$

Equation (18) provides some control over the dispersion of the estimates: for any temporal interval $[t, t']$ which does not intersect \mathcal{T}_ℓ , applying eq. (18) round-wise yields $\mathcal{L}(t') \leq (1 - \gamma)^{2(t'-t)} \mathcal{L}(t)$ – we find again that the rule achieves asymptotic consensus, as this $\mathcal{L}(t) \rightarrow_t 0$. However, eq. (18) alone cannot control

the potential $\mathcal{L}(t)$ across the entire execution. We use Lemma 1 to piece the variations of the potential over \mathcal{T}_ℓ :

$$\forall t \geq 2 : \mathcal{L}(t) \leq \max_i \frac{\pi_i(t)}{\pi_i(t-1)} (1-\gamma)^2 \mathcal{L}(t-1) . \quad \triangleright 19$$

The rounds for which $\pi(t) \neq \pi(t-1)$ are exactly those of \mathcal{T}_ℓ , so eq. (19) is equivalent to eq. (18) when $t \notin \mathcal{T}_\ell$. Applying eq. (19) over \mathcal{T}_ℓ will induce a delay factor $\beta_\ell := \prod_{t \in \mathcal{T}_\ell} \max_i \frac{\pi_i(t)}{\pi_i(t-1)}$. Given $t \in \mathcal{T}_\ell$ we have

$$\begin{aligned} \max_i \frac{\pi_i(t)}{\pi_i(t-1)} &= \frac{d'(t-1)}{d'(t)} \max_i \frac{d'_i(t)}{d'_i(t-1)} \\ &\leq \frac{d'(t-1)}{d'(t)} \prod_i \frac{d'_i(t)}{d'_i(t-1)} , \end{aligned}$$

and with $d'_i(0) = 2$:

$$\beta_\ell \leq \frac{2n}{d'} \prod_i \frac{d'_i}{2} . \quad \triangleright 20$$

Finally, Lemmas 2 and 3 give us $\mathcal{L}(1) \leq \frac{1}{2} ((1-\gamma) \text{diam } \boldsymbol{\mu})^2$. Together with eq. (19), we have for any $t \geq 2$:

$$\begin{aligned} \mathcal{L}(t) &\leq \frac{1}{2} \prod_{\tau \leq t} \max_i \frac{\pi_i(\tau)}{\pi_i(\tau-1)} \cdot ((1-\gamma)^t \text{diam } \boldsymbol{\mu})^2 \\ &\leq \frac{\beta_\ell}{2} ((1-\gamma)^t \text{diam } \boldsymbol{\mu})^2 \end{aligned}$$

and c using Lemma 5,

$$\text{diam } \mathbf{x}(t) \leq \sqrt{\frac{2\beta_\ell}{\min_i \pi_i(t)}} (1-\gamma)^t \cdot \text{diam } \boldsymbol{\mu}$$

with eq. (20) and the fact that $\pi_i(t) \geq 2/d'$,

$$\leq \sqrt{n \prod_i \frac{d'_i}{2}} (1-\gamma)^t \text{diam } \boldsymbol{\mu} .$$

We now let $\beta := n \prod_i \frac{d'_i}{2}$. By definition of t_ε , we have

$$\sqrt{\beta} (1-\gamma)^{t_\varepsilon} \leq \varepsilon ;$$

equivalently,

$$t_\varepsilon \geq \frac{\log(1-\gamma)}{\log(\varepsilon/\sqrt{\beta})} ,$$

and since $\log(1-x) \leq -x$ when $x \in (0,1)$,

$$t_\varepsilon \leq \gamma^{-1} \log(\sqrt{\beta}/\varepsilon) . \quad \triangleright 21$$

Inserting in this expression our bound over γ ,

$$t_\varepsilon \leq \frac{(n-1)D'}{2} \left(\sum_i \log d'_i + \log n - 2 \log \varepsilon - n \log 2 \right), \quad \triangleright 22$$

and with $d'_i \leq n$ we have both $D' \leq n^2$ and $\sum_i \log d'_i \leq n \log n$, which yields indeed $T_n(\varepsilon) = O(n^4 \log n / \varepsilon)$. \blacktriangleleft

5 The MaxMetropolis algorithm

Girded with our stabilizing strategy, we now return to the problem of average consensus. Recall that we obtained the Metropolis rule by applying the Metropolis-Hastings symmetrization to the EqualNeighbor update matrices. Any consensus update rule can be given the same treatment; in particular, the symmetrized version of the FixedWeight rule – where round t neighbors i and j apply the weight $a_{ij}(t) = \frac{1}{\max(q_i, q_j)}$ – achieves asymptotic average consensus whenever the FixedWeight rule achieves asymptotic consensus.

This symmetrized FixedWeight rule is subject to the same limitations as the FixedWeight rule, which we now set out to circumvent as we did in the previous section. However, in doing so we must also avoid the trappings of the Metropolis rule, where issues of information locality prevent the distributed implementation. In particular, we observe that the symmetrized MaxWeight rule is no easier to implement than the Metropolis rule itself: for i and j neighbors in round t , the weight $a_{ij}(t)$ depends on $d'_j(t) = \max\{d_j(1), \dots, d_j(t)\}$; in particular, it depends on $d_j(t)$ like the Metropolis weight, which was the source of the problem with the Metropolis rule.

Our solution is to define the update rule in terms of an agent's largest degree *up to the previous round*, resulting in the *MaxMetropolis* update

$$x_i(t) = x_i(t-1) + \sum_{j \in \mathcal{N}_i(t)} \frac{x_j(t-1) - x_i(t-1)}{\max(d'_i(t-1), d'_j(t-1))}, \quad \triangleright 23$$

whose implementation by a local algorithm is given in Algorithm 2.

We can make a few immediate observations. First, the MaxMetropolis rule defines symmetric update weights, and so the initial average is the only admissible consensus value. Moreover, the weights are clearly stabilizing, and once they stop changing the MaxMetropolis rule behaves like the symmetrized FixedWeight rule with parameters d'_1, \dots, d'_n . From there, Proposition 1 shows that Algorithm 2 is an average consensus algorithm for the class \mathfrak{G} .

However, the right-hand side of eq. (23) no longer necessarily defines a *convex* combination, as the self-weight $a_{ii}(t)$ may be negative when $d'_i(t) < d_i(t)$. In the worst case, the next estimate $x_i(t)$ may leave the convex hull of the set $\{x_1(t-1), \dots, x_n(t-1)\}$, which delays the eventual convergence. We show in Theorem 2, that this is only by at most a linear factor, when compared to a bound of $O(n^3 \log n / \varepsilon)$ available for the symmetrized FixedWeight rule with parameters d'_1, \dots, d'_n .

We note that the broken convexity does not fully explain the discrepancy of the convergence times between the Metropolis and MaxMetropolis rules, the former admitting a bound in $O(n^2 \log n / \varepsilon)$. To explain the rest of the gap, observe that the Metropolis and MaxMetropolis rules take opposite approaches

towards selecting weights. The Metropolis update uses the exact degree of each agent, and is thus perfectly tailored to the graph in each round; on the other hand, the MaxMetropolis update only uses an *upper bound* over the degree, a pessimistic approach that induces a slower convergence.

Algorithm 2 The MaxMetropolis algorithm, code for agent i

Input: $\mu_i \in \mathbb{R}$

1 Initially:

2 $\mathbf{x}_i \leftarrow \mu_i$

3 $\mathbf{q}_i \leftarrow 2$

4 In each round do:

5 send $m_i = \langle \mathbf{x}_i, \mathbf{q}_i \rangle$

6 receive m_{j_1}, \dots, m_{j_d} $\triangleright d$ neighbors

7 $\mathbf{x}_i \leftarrow \mathbf{x}_i + \sum_{k=1}^d \frac{\mathbf{x}_{j_k} - \mathbf{x}_i}{\max(\mathbf{q}_i, \mathbf{q}_{j_k})}$

8 $\mathbf{q}_i \leftarrow \max(\mathbf{q}_i, d)$

9 output \mathbf{x}_i

Theorem 2 *The MaxMetropolis algorithm solves average consensus over the class \mathfrak{G} of dynamic graphs that are reflexive, bidirectional, and strongly connected in each round, with a convergence time of $T_n(\varepsilon) = O(n^4 \log n/\varepsilon)$ for a system of n agents.*

In contrast with Theorem 1, the proof of Theorem 2 will only involve the usual geometry over the space \mathbb{R}^n . As a consequence, we briefly restate Lemmas 5 to 7, specialized for the usual Euclidian norm $\|\cdot\|$.

Lemma 5 *For a non-null vector \mathbf{v} for which $\bar{\mathbf{v}} = 0$, we have*

$$\sqrt{2/n} \|\mathbf{v}\| < \text{diam } \mathbf{v} < 2 \|\mathbf{v}\| . \quad \triangleright 24$$

Lemma 6 *For an irreducible stochastic matrix \mathbf{A} that is symmetric and has positive diagonal entries, and any vector \mathbf{v} for which $\bar{\mathbf{v}} = 0$,*

$$\|\mathbf{A}\mathbf{v}\| \leq (1 - \gamma_{\mathbf{A}}) \|\mathbf{v}\| . \quad \triangleright 25$$

Lemma 7 *For an irreducible stochastic matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ that is symmetric and has positive diagonal entries*

$$\gamma_{\mathbf{A}} \geq \frac{A^-}{n(n-1)} , \quad \triangleright 26$$

where $A^- := \min\{A_{ij} \mid i, j \in [n]\} \setminus \{0\}$.

Proof of Theorem 2. We fix a dynamic graph $\mathbb{G} \in \mathfrak{G}$ of order n , and define $d'_i(t)$, d'_i , \mathcal{T}_ℓ , and t_s as in eq. (15), and recall from the proof of Theorem 1 that $|\mathcal{T}_\ell| \leq \sum_i d'_i$.

Let us then fix an execution of Algorithm 2 over the dynamic communication graph \mathbb{G} , using input values $\mu_1, \dots, \mu_n \in \mathbb{R}$. Without losing generality, we can assume $\bar{\boldsymbol{\mu}} = 0$, since a uniform translation of the input does not alter the relative positions of the estimates.

An immediate induction reveals that the estimate vector satisfies the recurrence equation $\mathbf{x}(t) = \mathbf{A}(t)\mathbf{x}(t-1)$, where the matrix $\mathbf{A}(t)$ is the round t MaxMetropolis matrix for the dynamic graph \mathbb{G} :

$$[\mathbf{A}(t)]_{ij} = \begin{cases} \frac{1}{\max(d'_i(t-1), d'_j(t-1))} & i \neq j \in \mathcal{N}_i(t) \\ 1 - \sum_{k=1}^n \frac{1}{\max(d'_i(t-1), d'_k(t-1))} & j = i \\ 0 & j \notin \mathcal{N}_i(t) \end{cases} \quad \triangleright 27$$

As such, it is a symmetric matrix for which $\mathbf{A}(t)\mathbf{1} = \mathbf{1}$, and in particular the average $\overline{\mathbf{x}(t)}$ is an invariant of the execution. Asymptotic consensus, if it happens, is necessarily over the average $\overline{\boldsymbol{\mu}} = 0$.

The matrix $\mathbf{A}(t)$ results from a Metropolis-Hastings symmetrization, which implies $A_{ii}(t) \geq 1 - \frac{d_i(t)-1}{d'_i(t-1)}$, and in particular for $t \notin \mathcal{T}_\ell$ we have $A_{ii}(t) \geq 1/n$. As the set \mathcal{T}_ℓ is finite, we let $t_s := \max \mathcal{T}_\ell$, and the above holds in particular for all subsequent rounds $t > t_s$.

Let us then define another sequence $\mathbf{z}^{\mathbb{N}}$, together with a dynamic graph \mathbb{G}' , by $\mathbf{z}(k) := \mathbf{x}(k+t_s+1)$ and $\mathbb{G}'(k) := \mathbb{G}(k+t_s+1)$, for each $k \in \mathbb{N}$. The sequence $\mathbf{z}^{\mathbb{N}}$ satisfies the assumptions of Proposition 1 for the dynamic graph $\mathbb{G}' \in \mathfrak{G}$ and uniform convexity parameter $\alpha = 1/n$, and so it achieves asymptotic consensus and the sequence of estimates $\mathbf{x}^{\mathbb{N}}$ does as well. Since the consensus value is necessarily the average $\overline{\boldsymbol{\mu}}$, we see that Algorithm 2 is an average consensus algorithm for the class \mathfrak{G} .

To bound the convergence time, let us first remark that the diagonal entry $A_{ii}(t)$ may be negative when $d'_i(t) \neq d'_i(t-1)$, which can cause the estimate $x_i(t)$ can then leave the convex hull of the set $\{x_j(t-1) \mid j \in \mathcal{N}_i(t)\}$. In fact, they can leave the convex hull of the set $\{x_j(t-1) \mid j \in [n]\}$, which moves the system away from consensus and delays the eventual convergence.

To bound the total delay accrued in this manner, we fix a disagreement threshold $\varepsilon > 0$, and define the set $\mathcal{T}_\varepsilon := \{t \in \mathbb{N} \mid \text{diam } \mathbf{x}(t) \leq \varepsilon \cdot \text{diam } \boldsymbol{\mu}\}$. Since the system achieve asymptotic consensus, this set contains an unbounded interval, and we let $t_\varepsilon := \inf \{t \in \mathcal{T}_\varepsilon \mid \forall \tau \geq t: \tau \in \mathcal{T}_\varepsilon\}$. Remark that since the estimates can leave their convex hull, it is possible for t_ε to be greater than $\inf \mathcal{T}_\varepsilon$.

We then follow the variations of the quantity $N(t) := \|\mathbf{x}(t)\|$ from one round to the next, distinguishing on whether $t \in \mathcal{T}_\ell$ or not. When $t \notin \mathcal{T}_\ell$, the update matrix $\mathbf{A}(t)$ has positive diagonal entries, and by Lemma 7 we have $\gamma := \inf_{t \notin \mathcal{T}_\ell} \gamma_{\mathbf{A}(t)} \geq 1/n^3$. Using Lemma 6, we have

$$\forall t \notin \mathcal{T}_\ell: N(t) \leq (1 - \gamma)N(t-1) \quad \triangleright 28$$

For rounds $t \in \mathcal{T}_\ell$, on the other hand, the update matrix $\mathbf{A}(t)$ may have negative entries, and we cannot use Lemma 6 to control $N(t)$. However, since the matrix $\mathbf{A}(t)$ is symmetric, it is diagonalizable, and for any vector \mathbf{v} , we have $\|\mathbf{A}(t)\mathbf{v}\| \leq \rho_{\mathbf{A}(t)}\|\mathbf{v}\|$, which gives us

$$\forall t \in \mathcal{T}_\ell: N(t) \leq \rho_{\mathbf{A}(t)}N(t-1) \quad \triangleright 29$$

We note that eq. (29) holds in fact for all $t \in \mathbb{N}$, but is strictly worse than eq. (28) outside of \mathcal{T}_ℓ .

To bound the spectral radius $\rho_{\mathbf{A}(t)}$, we define $\nu_{t,i} := 1 - \min(0, A_{ii}(t))$ and $\nu_t := \max_i \nu_{t,i}$. For any eigenvalue λ of the matrix $\mathbf{A}(t)$, the quantity $(1 + \frac{\lambda-1}{\nu_t})$ is an eigenvalue of the stochastic matrix $\frac{1}{\nu_t}(\mathbf{A}(t) + (\nu_t - 1)\mathbf{I})$, and so is less than 1 in absolute value. We have $1 - 2\nu_t \leq \lambda \leq 1$, and so $|\lambda| \leq 2\nu_t - 1 \leq \nu_t^2$, the latter since $x^2 - 2x + 1 \geq 0$ always holds. This holds for all eigenvalues of the matrix $\mathbf{A}(t)$, and so we have

$$\forall t \in \mathcal{T}_\ell: N(t) \leq \nu_t^2 N(t-1) . \quad \triangleright 30$$

The delay accrued during \mathcal{T}_ℓ will then depend on some factor $\beta_\ell := \prod_{t \in \mathcal{T}_\ell} \nu_t^2$. To bound ν_t for $t \in \mathcal{T}_\ell$, we observe that $\sum_{j \neq i} A_{ij}(t) \leq \frac{d_i(t)-1}{d_i(t-1)} \leq \frac{d_i(t)}{d_i(t-1)}$ for any $i \in [n]$, which yields $\nu_{i,t} \leq \frac{d_i(t)}{d_i(t-1)}$ since $d_i'(t)$ is weakly increasing. Given that $\nu_{i,t} \geq 1$, we have $\beta_\ell \leq \prod_{t \in \mathcal{T}_\ell} \prod_i \nu_{i,t}^2$, and since $d_i'(t) = d_i'(t-1)$ when $t \notin \mathcal{T}_\ell$, we finally have $\beta_\ell \leq \left(\prod_i \frac{d_i}{2}\right)^2$

Taking eqs. (28) and (30) together, we have

$$\begin{aligned} N(t) &\leq \prod_{\tau \leq t: \tau \in \mathcal{T}_\ell} \nu_\tau^2 \prod_{\tau \leq t: \tau \notin \mathcal{T}_\ell} (1 - \gamma_{\mathbf{A}(\tau)}) \cdot N(0) \\ &\leq \beta_\ell (1 - \gamma)^{t - |\mathcal{T}_\ell|} \cdot N(0) . \end{aligned}$$

Using Lemma 5, this gives us $\text{diam } \mathbf{x}(t) \leq 2\sqrt{n}\beta_\ell (1 - \gamma)^{t - |\mathcal{T}_\ell|} \text{diam } \boldsymbol{\mu}$. By definition of t_ε , we have $2\sqrt{n}\beta_\ell (1 - \gamma)^{t_\varepsilon - |\mathcal{T}_\ell|} \leq \varepsilon$, from which we deduce that $t_\varepsilon \leq \gamma^{-1} \log(2\sqrt{n}\beta_\ell/\varepsilon) + |\mathcal{T}_\ell|$. Using our upper bounds for $|\mathcal{T}_\ell|$, γ , and β_ℓ ,

$$t_\varepsilon \leq n^3 \left(2 \sum_i \log d_i' - \log \varepsilon - (2n - 1) \log 2 \right) + \sum_i d_i' - 2n , \quad \triangleright 31$$

and with $d_i' \leq n$ the convergence time of the MaxMetropolis algorithm over the class \mathfrak{G} is in $T_n(\varepsilon) = O(n^4 \log n/\varepsilon)$. \blacktriangleleft

6 Discussion

In the preceding sections, we discussed a couple of well-studied update rules traditionally directed at the problems of consensus and average consensus in multi-agent networks with bidirectional interactions. We argued that deploying these rules algorithmically is difficult over dynamic networks, where the communication links change often and in an unstructured manner. The EqualNeighbor rule can take an exponentially long time before reaching consensus with a dynamic topology, whereas the Metropolis rule cannot even be implemented unless the fluctuations in the network are tightly choreographed, and not too frequent.

We introduced a parcimonious approach to cope with dynamic topologies, which requires no global control, no symmetry breaking devices, and no global information about the network, only making each agent keep track of its largest degree over time. Defining the update rule in terms of the history of the network beyond its current configuration acts as a sort of low-pass filter, which dampens the effects of the fluctuations from pathological down to manageable.

Remarkably, adding *a little* memory helps *a lot*. Out of all possible history-based strategies, we pick a rather crude one: each agent keeps track of a

single parameter – its historically largest degree, taking up to $\lceil \log n \rceil$ bits of memory – which it adjusts only upwards in the pursuit of correctness and never downwards in that of optimization. Even so, it suffices to correct most of the deficiencies of the EqualNeighbor and Metropolis rules, at a moderate cost in the convergence time: compare the $O(n^3 \log n/\varepsilon)$ bound for EqualNeighbor over static networks and $O(n^2 \log n/\varepsilon)$ theoretical bound for Metropolis over dynamic networks with the $O(n^4 \log n/\varepsilon)$ bounds for both the MaxWeight and MaxMetropolis algorithms.

We find bounds of the same order of magnitude for both algorithms, but, interestingly, for opposite reasons. In an execution of the MaxWeight algorithm, the estimates keep moving towards a target value, but their convergence is set back when the target itself moves. In contrast, the MaxMetropolis algorithm rigidly targets the average $\bar{\mu}$, and keeping the target constant sometimes requires individual estimates to move away from consensus, undoing earlier progress.

References

- [1] Steve Alpern. The Rendezvous Search Problem. *SIAM Journal on Control and Optimization*, 33(3):673–683, 1995. doi:10.1137/S0363012993249195.
- [2] Tucker Balch and Ronald C. Arkin. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939, 1998. doi:10.1109/70.736776.
- [3] Vincent D. Blondel, Julien M. Hendrickx, Alex Olshevsky, and John N. Tsitsiklis. Convergence in Multiagent Coordination, Consensus, and Flocking. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 2996–3000, Seville, Spain, 2005. IEEE. doi:10.1109/CDC.2005.1582620.
- [4] John Bonner Buck. Synchronous Rhythmic Flashing of Fireflies. *The Quarterly Review of Biology*, 13(3):301–314, 1938. doi:10.1086/394562.
- [5] Ming Cao, A. Stephen Morse, and Brian D. O. Anderson. Reaching a Consensus in a Dynamically Changing Environment: A Graphical Approach. *SIAM Journal on Control and Optimization*, 47(2):575–600, 2008. doi:10.1137/060657005.
- [6] Themistoklis Charalambous, Michael G. Rabbat, Mikael Johansson, and Christoforos N. Hadjicostis. Distributed Finite-Time Computation of Digraph Parameters: Left-Eigenvector, Out-Degree and Spectrum. *IEEE Transactions on Control of Network Systems*, 3(2):137–148, 2016. doi:10.1109/TCNS.2015.2428411.
- [7] Bernadette Charron-Bost. Geometric Bounds for Convergence Rates of Averaging Algorithms. *CoRR*, 2020. arXiv:2007.04837.
- [8] Bernadette Charron-Bost, Matthias Függer, and Thomas Nowak. Approximate Consensus in Highly Dynamic Networks: The Role of Averaging Algorithms. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming*, volume 9135 of *Lecture Notes in Computer Science*, pages 528–539, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. doi:10.1007/978-3-662-47666-6_42.
- [9] Samprit Chatterjee. Reaching a Consensus: Some Limit Theorems. In *Proceedings of the 40th Session of the International Statistical Institute, Warsaw, Poland, 1975*, volume 3, pages 156–160, Warsaw, Poland, 1975.
- [10] Samprit Chatterjee and Eugene Seneta. Towards Consensus: Some Convergence Theorems on Repeated Averaging. *Journal of Applied Probability*, 14(1):89–97, 1977. doi:10.2307/3213262.
- [11] Bernard Chazelle. The Total s-Energy of a Multiagent System. *SIAM Journal on Control and Optimization*, 49(4):1680–1706, 2011. doi:10.1137/100791671.

- [12] Felipe Cucker and Steve Smale. Emergent Behavior in Flocks. *IEEE Transactions on Automatic Control*, 52(5):852–862, 2007. doi:10.1109/TAC.2007.895842.
- [13] George Cybenko. Dynamic load balancing for distributed memory multiprocessors. *Journal of Parallel and Distributed Computing*, 7(2):279–301, 1989. doi:10.1016/0743-7315(89)90021-X.
- [14] Ariel Daliot, Danny Dolev, and Hanna Parnas. Self-Stabilizing Pulse Synchronization Inspired by Biological Pacemaker Networks. In Shing-Tsaan Huang and Ted Herman, editors, *Self-Stabilizing Systems*, volume 2704 of *Lecture Notes in Computer Science*, pages 32–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. doi:10.1007/3-540-45032-7_3.
- [15] Morris H. DeGroot. Reaching a Consensus. *Journal of the American Statistical Association*, 69(345):118–121, 1974. doi:10.2307/2285509.
- [16] Michael Dinitz, Jeremy Fineman, Seth Gilbert, and Calvin Newport. Load balancing with bounded convergence in dynamic networks. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pages 1–9, Atlanta, GA, USA, 2017. IEEE. doi:10.1109/INFOCOM.2017.8057000.
- [17] Julien M. Hendrickx and Vincent D. Blondel. Convergence of Linear and Non-Linear Versions of Vicsek’s Model. In *Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems*, pages 1229–1240, Kyoto (Japan), 2006.
- [18] Ali Jadbabaie, Jie Lin, and A. Stephen Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 48(6):988–1001, 2003. doi:10.1109/TAC.2003.812781.
- [19] Leslie Lamport and P. Michael Melliar-Smith. Synchronizing clocks in the presence of faults. *Journal of the ACM (JACM)*, 32(1):52–78, 1985. doi:10.1145/2455.2457.
- [20] Renato E. Mirollo and Steven H. Strogatz. Synchronization of Pulse-Coupled Biological Oscillators. *SIAM Journal on Applied Mathematics*, 50(6):1645–1662, 1990. doi:10.1137/0150098.
- [21] Luc Moreau. Stability of multiagent systems with time-dependent communication links. *IEEE Transactions on Automatic Control*, 50(2):169–182, 2005. doi:10.1109/TAC.2004.841888.
- [22] Angelia Nedic and Ji Liu. On Convergence Rate of Weighted-Averaging Dynamics for Consensus Problems. *IEEE Transactions on Automatic Control*, 62(2):766–781, February 2017. doi:10.1109/TAC.2016.2572004.
- [23] Angelia Nedić, Alex Olshevsky, and Michael G. Rabbat. Network Topology and Communication-Computation Tradeoffs in Decentralized Optimization. *Proceedings of the IEEE*, 106(5):953–976, 2018. doi:10.1109/JPROC.2018.2817461.
- [24] Angelina Nedić, Alex Olshevsky, Asuman Ozdaglar, and John N. Tsitsiklis. On Distributed Averaging Algorithms and Quantization Effects. *IEEE Transactions on Automatic Control*, 54(11):2506–2517, 2009. doi:10.1109/TAC.2009.2031203.
- [25] Reza Olfati-Saber and Richard M. Murray. Consensus Problems in Networks of Agents With Switching Topology and Time-Delays. *IEEE Transactions on Automatic Control*, 49(9):1520–1533, 2004. doi:10.1109/TAC.2004.834113.
- [26] Alex Olshevsky. Linear Time Average Consensus and Distributed Optimization on Fixed Graphs. *SIAM Journal on Control and Optimization*, 55(6):3990–4014, 2017. doi:10.1137/16M1076629.
- [27] Alex Olshevsky and John N. Tsitsiklis. Convergence Speed in Distributed Consensus and Averaging. *SIAM Review*, 53(4):747–772, 2011. doi:10.1137/110837462.

- [28] Alex Olshevsky and John N. Tsitsiklis. Degree Fluctuations and the Convergence Time of Consensus Algorithms. *IEEE Transactions on Automatic Control*, 58(10):2626–2631, 2013. doi:10.1109/TAC.2013.2257969.
- [29] Qun Li and Daniela Rus. Global clock synchronization in sensor networks. *IEEE Transactions on Computers*, 55(2):214–226, 2006. doi:10.1109/TC.2006.25.
- [30] W. Ren. Consensus strategies for cooperative control of vehicle formations. *IET Control Theory & Applications*, 1(2):505–512, 2007. doi:10.1049/iet-cta:20050401.
- [31] Craig W. Reynolds. Flocks, Herds, and Schools: A Distributed Behavioral Model. *Computer Graphics*, 21(4):10, 1987.
- [32] Osvaldo Simeone and Umberto Spagnolini. Distributed Time Synchronization in Wireless Sensor Networks with Coupled Discrete-Time Oscillators. *EURASIP Journal on Wireless Communications and Networking*, 2007(1):057054, 2007. doi:10.1155/2007/57054.
- [33] Shreyas Sundaram and Christoforos N. Hadjicostis. Finite-Time Distributed Consensus in Graphs with Time-Invariant Topologies. In *2007 American Control Conference*, pages 711–716, 2007. doi:10.1109/ACC.2007.4282726.
- [34] Ichiro Suzuki and Masafumi Yamashita. Distributed Anonymous Mobile Robots: Formation of Geometric Patterns. *SIAM Journal on Computing*, 28(4):1347–1363, 1999. doi:10.1137/S009753979628292X.
- [35] John N. Tsitsiklis. *Problems in Decentralized Decision Making and Computation*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 1984. URL: <https://www.mit.edu/~jnt/Papers/PhD-84-jnt.pdf>.
- [36] John N. Tsitsiklis, Dimitri P. Bertsekas, and Michael Athans. Distributed Asynchronous Deterministic and Stochastic Gradient Optimization Algorithms. *IEEE Transactions on Automatic Control*, 31(9):803–812, 1986. doi:10.1109/TAC.1986.1104412.
- [37] Tamás Vicsek, András Czirók, Eshel Ben-Jacob, Inon Cohen, and Ofer Shochet. Novel Type of Phase Transition in a System of Self-Driven Particles. *Physical Review Letters*, 75(6):1226–1229, 1995. doi:10.1103/PhysRevLett.75.1226.
- [38] Jennifer Lundelius Welch and Nancy Lynch. A new fault-tolerant algorithm for clock synchronization. *Information and Computation*, 77(1):1–36, 1988. doi:10.1016/0890-5401(88)90043-0.
- [39] Lin Xiao and Stephen Boyd. Fast linear iterations for distributed averaging. *Systems & Control Letters*, 53(1):65–78, 2004. doi:10.1016/j.sysconle.2004.02.022.
- [40] Lin Xiao, Stephen Boyd, and Sanjay Lall. A Scheme for Robust Distributed Sensor Fusion Based on Average Consensus. In *Fourth International Symposium on Information Processing in Sensor Networks, 2005.*, pages 63–70, Los Angeles, CA, USA, 2005. IEEE. doi:10.1109/IPSN.2005.1440896.