Average consensus in anonymous dynamic networks : An algorithmic approach

Patrick Lambein

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2020IPPAX103

Thèse de doctorat

Consensus de moyenne dans les réseaux dynamiques anonymes

Une approche algorithmique

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à l'École polytechnique

École doctorale n°626
École Doctorale de l'Institut Polytechnique de Paris (ED IP Paris)
Spécialité de doctorat : Informatique, Données et Intelligence Artificielle

Thèse présentée et soutenue au Bois-Plage-en-Ré, le 16 décembre 2020, par

PATRICK LAMBEIN

Composition du Jury :

Éric Goubault Professeur, École polytechnique (LIX)	Président
Pierre Fraigniaud Directeur de Recherche, CNRS (IRIF)	Rapporteur
Julien Hendrickx Professeur, Université Catholique de Louvain (ICTEAM)	Rapporteur
François Baccelli Professeur, University of Texas in Austin (Department of Mathematics, Department of Electrical and Computer Engineering)	Examinateur
Angelia Nedić Professeur, Arizona State University (School of Electrical, Computer and Energy Engineering)	Examinatrice
Laurent Viennot Directeur de Recherche, INRIA (IRIF)	Examinateur
Bernadette Charron-Bost Directrice de Recherche, CNRS (LIX)	Directrice de thèse

Average Consensus in Anonymous Dynamic Networks

An Algorithmic Approach

κολοφών

This manuscript was prepared with the LaTeX document preparation system created by Leslie Lamport, using the `memoir` class created by Peter Wilson and maintained by Lars Madsen. Bibliographic facilities were provided by the `biblatex` package created by Philipp Lehman, and maintained by Philip Kime. Knowledge management facilities were provided by the `knowledge` package created by Thomas Colcombet. Much of the knowledge itself was facilitated by Alexandra Elbakyan. Donald Knuth should probably also be mentioned here.

The body text is set in 11/14pt on a 27pc measure with Libertinus Serif, which is also used to typeset mathematical formulae. Other fonts used include Zilla Slab and `Inconsolata`.

Processed for (approximatly) the 1,308th time on February 25, 2021 using the LuaTeX typesetting engine.

To my number one fan.

Il nous faut regarder
Ce qu'il y a de beau
Le ciel gris ou bleuté
Les filles au bord de l'eau
L'ami qu'on sait fidèle
Le soleil de demain
Le vol d'une hirondelle
Le bateau qui revient.

Il nous faut regarder
Jacques Brel

CONTENTS

ACKNOWLEDGEMENTS

> A hero ventures forth from the world of common day into a region of supernatural wonder: fabulous forces are there encountered and a decisive victory is won: the hero comes back from this mysterious adventure with the power to bestow boons on his fellow man.
>
> *The Hero with a Thousand Faces*
> Joseph Campbell

Boy are these some strange, *strange*, *s t r a n g e* times. It feels like *a lot* of history has been going on recently – up to, and very much including, the last few days. In the short years since I completed my Master's degree in late 2014, the human collective sure had to bear the whips and scorns of time – and the rest. And, frankly, who knows what other ills we'll fly to next?

At the scale of my own microcosm, I, too, served as practice target for fortune's outrage much more than I could have imagined. Being dealt a dismal hand makes you viscerally appreciate those few good cards that you are holding. I have *a lot* to be grateful for, and many people to thank, as this point would not have been reached without them. Let us proceed.

I owe my life, and most of my sanity, to the courage, dedication, generosity, and kindness of countless health-care workers, public servants, and good samaritans, some of which whose name I'll never know, and certainly too many to name here. For want of ever repaying my debts, let them all be hereby acknowledged. I also thank all the friends who supported and comforted me when it was severely needed; again, there are just too many to name them all.

Next in line, obviously, comes Bernadette, whom I thank first for giving me the opportunity of this mysterious adventure, and then for relentlessly pursuing good and clear science, and for dispensing enough rear-end-bludgeoning that I might have a shot at that same pursuit. I learned a lot.

School and I never really agreed with one another, and it's therefore kind of hilarious that I would end up spending so much time there. I am thankful for some of my high school teachers, who helped nurture my curiosity in the midst of some dull, dull years: Didier Belin and Michèle

Otz, who taught history and geography, and Bernadette Meyer, who taught mathematics and who tried to teach me that you don't get far by being a smart-ass.

I owe a great deal to my *prépa* teachers; my life as a young adult would have been very different had we not crossed paths. Thanks go to Christophe More, who is the best educator whose classes I have had the pleasure to attend, and to Hervé Pépin, who, more than anyone else, opened my eyes to the supernatural wonder of mathematical thought. They also go to Christian Lair, a fantastic educator with a wit to arm a yak barber, with whom I first heard the call to adventure a long, long time ago that seems far, far away.

I probably wouldn't have started down this path without the insistant pushing of dedicated friends, in particular Geoffroy Couteau, – who is entirely to blame for my enrolling in the MPRI in the first place – Simone De Liberato, and Xavier Koegler, who really should appear in many different places in these acknowledgements. Although less of a friend than a friendly stranger, Paul-André Melliès played an outsized role through his kind encouragements over a chance encounter.

I certainly could not have *stayed* on this path were it not for Solène's unwavering support, kindness, love, and hardly-ever-wavering patience. My love, thank you for those years; here's to many more.

I am deeply grateful to Solène's family as well for being so welcoming and helpful. In particular, I thank Marie-Solange for suffering a brat like me near her daughter, and for giving us so much, especially during the wacky ride that this last year has been, and I thank Max for guarding the house while we were there – sorry for locking you up at night by mistake.[1] I knew Jean too briefly before his untimely exit; we all miss him often, and I thank him for the beautiful marks he left behind.

I would like to thank Shlomo Moran and Jennifer Welch for enlightening scientific discussions – of which I hope there will be more – and Pierre Fraigniaud and Stéphane Gaubert for their encouraging feedback during my *entretien de mi-parcours*.

The scientist's journey is a solitary one; thanks to my colleagues at the LIX, it was not overly lonesome. Let me acknowledge and thank Frédéric Ayrault, François Bidet, Olivier Bournez, Maria Costa, Pierre-Yves Coursolle, Uli Fahrenberg, Quentin Guilmant, Emmanuel Haucourt, Nan Li, Samuel Mimram, Sergio Mover, Jean-Marc Notin, and Pierre-Yves Strub, to name a few lunch companions. I had the privilege of sharing my office with Thibaut Benjamin and Simon Forest, whom I even managed to

[1] No animals were hurt during the writing of this manuscript.

chance upon from time to time. It was always a pleasure to be distracted from work with either of them, and it was humbling to be thinking about my mundane graphs while they were hacking away at their ~~abstract nonsense~~ hard problems. I wish them the best for what's coming next – frankly, I'm not too worried.

It was also my pleasure to be teaching programming for Cécile Braunstein, Olivier Marchetti, and Maria Potop-Butucaru, and in particular to be working alongside Guillaume Matheron. I never expected teaching in pair would be so rewarding.

Few fabulous forces will pulverize your heart, smother your spirit, desecrate your soul, and obliterate your hope as much as the banal evil of bureaucracy. The unsung heroes of the LIX are its wonderful administrative assistants, who battle daily to carve pockets of purgatory within that ninth circle of Hell. I want to thank Jackie Gardin, Helena Kutniak, Vanessa Molina, Evelyne Rayssac, and especially Sylvie Jabinet for all their help, with trifles as with tragedies. For their patient councel during my mysterious adventures in the administrative jungle, my thanks go to Alain Couvreur and Benjamin Doerr as well.

Lastly, but by no means least, I want to thank my family; any decisive victory of mine is really theirs. In particular, for all they ever did for me, for their unflinching support, and for their boundless love that's even older than I am, I thank my parents: my mom, who's got the biggest of all hearts, and my dad, inexhaustibly inspiring and the wisest man I know – wise enough that he won't mind not sharing the dedication of this manuscript.

On with the boons, fellow *sapiens*.

Le Bois-Plage-en-Ré, 2020-11-09

Post-scriptum. As I painfully came to learn while writing these words, Prof. Christian Lair passed away weeks before I could send him this manuscript to read. There is surely a lesson on procrastination in here.

Rest in power, fellow traveler.

Post-post-scriptum. Well, that was one of the most stressful things I was ever given the opportunity to do. After taking a couple of months to breathe, I would like to add a few mentions to the above ones. Thanks go to Pierre Fraigniaud and Julien Hendrickx for their careful and enthusiastic reviews, which touched me deeply. To the rest of the jury as well, who

honored me with astute questions. To Thomas Clausen for wielding magic so that the viva could happen on-line, and to Esther for helping during my live tests. To Whitney for proof-reading these acknowledgements, and Wazou for helping me fix the front page. Finally, because I initially forgot, to my little brother Xavier, for many things, starting with being in my life.

INTRODUCTION

The work that we present in this monograph is situated at the meeting point of the *algorithmic* study of distributed systems and their *control theoretic* study. We look at problems originating from distributed control, but with an algorithmic perspective, with great attention directed to aspects such as the model assumptions and the localization of events in space and time.

Specifically, we focus on the AVERAGE CONSENSUS problem: given a multi-agent system with each agent $u \in V$ starting with some value μ_u, make all agents compute the arithmetic mean of the input, $\overline{\mu} = \frac{1}{|V|} \sum_u \mu_u$. The agents must do so through local interactions alone, in a communication topology that may be incomplete – that is, we do *not* assume that the network permits all possible pairwise interactions – and without the help of a centralized coordinator. Because they are intimately related, we also consider to a lesser extent the unconstrained CONSENSUS problem, where agents must reach a common opinion on *some* value $\omega \in \text{range } \boldsymbol{\mu}$ which need not be the exact average $\overline{\mu}$.

Looking at *control* problems means, in a sense, that we are interested in the *trajectory* of the system as much as in its destination. In computational terms, we will be looking at systems that *weakly compute* an average $\overline{\mu}$ or a consensus ω – in the sense that agreement need only be reached asymptotically, with agents never definitively committing to a result – but at the same time we will be interested in quantifying regular progress towards that goal. This approach is motivated by numerous practical applications in the control of autonomous networked systems, such as coordinating mobile agents to have them regroup, adopt a common heading, or control their relative positions while moving together – problems formally known as *rendez-vous* [3, 94], *flocking* [98, 55, 41], and *formation control* [12, 90], – implement a distributed load balancing [42, 74, 11], aggregating the measures of many sensors, formally known as *sensor fusion* [100], – or distributed optimization and machine learning [80, 77]. It is likewise of interest in the modeling of natural systems such as flocking [98, 55, 41], or the synchronization of cardiac pacemaker cells to produce heartbeats [71, 43], or that of fireflies to flash in unison [21, 71]; of particular interest to the topics of this thesis, the distributed implementation of sensor fusion scheme – which rely on asymptotic average consensus as a core primitive – serves as a model of brain function in

computational neurology [86]. Finally, it is of relevance in the social sciences, notably in the modelization of opinion formation in groups [46, 54, 4]. Adopting a weak computational model also sheds light over the different sources of difficulties in distributed computing: here, we will contrast obstacles due to the networked nature of the system from those that arise from the need to take coordinated irreversible actions while only having a partial view of the system.

In many cases, consensus is usefully fulfilled by having the network agents reach an answer reasonably fast without ever committing to it, rather than wait for a long time for perfect certainty. To develop an example, let us consider pacemaker cells, which are the heart neurons responsible for producing coherent heartbeats. Pacemaker cells can be modeled as frequently *firing*, inducing a change in the electric potential in neighboring heart muscle cells, causing a chain reacting that produces a global contraction – the *heartbeat* – provided that the pacemaker cells fired simultaneously.

The pacemaker cells are themselves electrically networked, and the system of pacemaker cells can then be modeled as networked coupled oscillators, which depend on agreeing on the *phase* of the oscillation to fulfill their function. In other words, distributed consensus processes are at play behind all animal – and, as far as we know, sentient – life. If the pacemaker cells happen to become slightly desynchronized, the cardiovascular system will be better served[2] by having them keep firing while they move back to consensus, producing weaker heartbeats in the meantime, than if they cease all activity while they patiently wait for an agreement subroutine to complete.

[2] This ceases to be true in some extreme cases such as *ventricular fibrillation*, where pacemaker cells become so desynchronized that they fail to produce any sort of coordinated behavior and the heart ceases to operate properly. In this case, suspending cardiac pacemaking all at once is *exactly* the right thing to do, a medical intervention known as defibrillation. Note that even then, pacemaker cell activity re-starts right away; defibrillation acts less as a call to a blocking subroutine than as a global signal which helps the agents re-synchronize.

Much of the algorithmic theory of distributed systems, and especially that of consensus, is concerned with terminating primitives, because they underpin the fundamental tool *replication*, powering the resiliency and consistency of distributed systems. Replication, and irrevocable consensus, are hard problems in and of themselves, and the study of agreement problems in distributed algorithms and in distributed control tend to rely on different sets of assumptions. Here, we adopt the latter kind, where agents are typically *anonymous* and their communication is mediated by a *partial graph*.

Several of the applications outlined above – vehicular and mobile ad-hoc networks, animal flocking – are only adequately described as *dynamic* networks, by which we mean that the communication topology

linking the agents is susceptible to change over time. In fact, considering mobile agents motivates us to adopt a communication model based on *local broadcast*, hereafter simply referred to as *broadcast*, by which we mean that agents cast messages destined to potentially reach all other agents, but without *a priori* knowledge of its actual recipients or their number. In particular, an agent cannot individually address its neighbors – there is no low-level primitive providing port numbers – and the content of a message is independent from the set of its recipients.

We take an agnostic view as to how this set comes to be: it may be that agents are in fact communicating *via* radio, but what we consider to be "the communication topology" may as well be an overlay network built upon some unspecified substrate. The framework of dynamic networs will lead us, moreover, to avoid relying on *structural* assumptions about the network: while we will generally suppose the network sufficiently connected – global behaviors are generally impossible over split networks – we will refrain from imposing, or excluding, any particular *shape* on the network.

Our model shares familiar traits with the LOCAL model [88] studied in distributed computing, but with important differences: we assume agents to be initially indistinguishable – they are not uniquely identified – and the communication graph is susceptible to change over time. This makes it impossible to solve the traditional agreement problem,[3] in which all agents should eventually halt and output the same value $\mu^* \in \{ \mu_u \mid u \in V \}$, but for different reasons than, for example, the traditional FLP result [48] about asynchronous networks where agents may suddenly stop working. Anonymous agents have no reliable way of learning the size of the system if it is not provided to them by an external source, and as such they cannot ever be certain that they can make a decision, or that new information that could alter the decision are yet to arrive.[4] Adopting a weak model of global computation means that such concerns are relegated to the background.

Approaching asymptotic problems with an algorithmic outlook means that we pay a detailed attention to the model assumptions, as well as to what "knowledge" agents can rely on at given times. In particular, in dynamic settings, information pertaining to the local topology of the network – say, the shape of the communication graph within some bounded distance of an agent – may travel too slowly to remain relevant once it reaches its target. As an example, the MAXMETROPOLIS algorithm, the main contribution of Chapter 2, is shown to possess a convergence time

[3] Throughout this thesis, we will use the term "consensus" to denote the problem of *asymptotically converging within the convex hull of the input values*, which is how it is called by the relevant litterature.

[4] Such situations – different networks, or different parts of networks, looking alike to the agents – are generally referred to by the umbrella term of SYMMETRIES of the networks.

in $\widetilde{O}(n^4)$, compared to the $\widetilde{O}(n^2)$ convergence time of the Metropolis method to which it is compared. However, our algorithm is shown to operate on all dynamic networks with bidirectional communication that are connected in each round, whereas the Metropolis method cannot in fact be implemented by local algorithms outside of very specific kinds of networks, and even then are fragile to any deviation from these specific conditions.

Thus we provide in the MaxMetropolis algorithm the first, to our knowledge, local and deterministic algorithm for average consensus in anonymous bidirectional dynamic networks, in the complete absence of centralized coordination. The MaxWeight algorithm, also introduced in Chapter 2, is a *consensus* algorithm – it computes *some* consensus value ω that need not be the average $\overline{\mu}$ – with a proven convergence time of $\widetilde{O}(n^4)$ in the same conditions as the MaxMetropolis algorithm.

The MaxWeight and MaxMetropolis algorithms are considered over networks with bidirectional communication links, and in Chapter 3 we drop the assumption of bidirectionality. Directed networks generally display too many symmetries for the average to be computable, even weakly, and so we resort to the classic tool of *randomized* algorithms, which here means that we give agents access to *private* random oracles: our randomized model remains local. Because randomization is such a powerful breaker of symmetries, we take a closer look at the space efficiency of algorithms, as well as stronger forms of computation while relaxing the common, and highly non-trivial, assumption that agents start executing their code simultaneously.

Table 1 summarizes the characteristics of the principal algorithms contributed in this thesis.

Summary of the argument We place ourselves in a model of *closed communication rounds*, which corresponds to the model usually described as "synchronous". Here, we will refrain from using this term, as a way to emphasize that the only "synchrony" assumed is logical: at no point will we ever assume anything about the system displaying *real-time* guarantees. Our communication model embeds no absolute bound over the *physical* timing of events, delays and message losses are treated the same way as the inherent dynamicity of the network – by the absence of the corresponding edges in the communication graph.

We consider two cases: deterministic local algorithms operating over bidirectional networks, and randomized local algorithms operating over arbitrary networks.

Table 1: Characteristics of contributed algorithms

Algorithm	Time	Message size	Limit	Comments
MaxWeight	$\widetilde{\mathrm{O}}(n^4)$	−*	$\omega \in \text{range } \boldsymbol{\mu}$	unconstrained consensus bidirectional networks
MaxMetropolis	$\widetilde{\mathrm{O}}(n^4)$	−*	$\overline{\mu}$	bidirectional networks
$\overline{\mathcal{R}}$	$\mathrm{O}(n)$	$\mathrm{O}(\log\log n)$	$\widehat{\mu} \in [\overline{\mu} - r, \overline{\mu} + r]$	Monte Carlo†
$\overline{\mathcal{D}}$	$\mathrm{O}(n)$	$\mathrm{O}(\log N + \log\log n)$	$\widehat{\mu} \in [\overline{\mu} - r, \overline{\mu} + r]$	terminating $N \geqslant n$ externally provided Monte Carlo†

* The MaxWeight and MaxMetropolis algorithms are studied in the idealized model where agents manipulate real numbers.
† By Monte Carlo, we mean that the algorithm can fail to produce a correct result with probability p, where p is a tuneable parameter of the algorithm.

In the bidirectional case, we build on the theory of convex update rules, following the prototype

$$x_u(t) = \sum_v a_{uv}(t) x_v(t-1)$$

with a positive weight $a_{uv}(t)$ assigned to each neighbor of u's in round t and the weights $(a_{uv})_v$ forming a convex combination. Such systems have received a sustained interest in recent decades, and are routinely used to model opinion dynamics, flocking, and more generally distributed agreement in multi-agent systems. If the weights are all symmetric – that is, if $a_{uv}(t) = a_{vu}(t)$ in all rounds and for all pairs of neighboring agents – then given a sufficiently connected network the estimates $x_u(t)$ are known to jointly converge to the average $\overline{\mu}$, and to jointly converge towards *some* consensus value ω in every case. The study of convex update rules is intertwined with that of asymptotic consensus itself, to the point that they are sometimes referred to as "the consensus algorithm".

We take a close look at the kind of rule that produces symmetric weights, particularly the Metropolis rule:

$$x_u(t) = x_u(t-1) + \sum_v \frac{x_v(t-1) - x_u(t-1)}{\max(d_u(t), d_v(t))}\ ,^{5}$$

and argue that the implementation by local algorithms over dynamic networks is difficult and fragile, as it necessarily relies on information

[5] By $d_u(t)$ we mean agent u's degree in the communication graph at round t.

beyond the immediate vicinity of each agent and has to be collected *via* ad-hoc methods that depend on the specific communication graph and do not generalize to large classes of networks.

We leverage the good convergence properties of convex update rules by making the agents progressively learn weights that can be used in place of the METROPOLIS weights given above; namely, each agent uses its *largest previous degree* in place of $d_u(t)$. Because the self-assigned weight $a_{uu}(t)$ may then sometimes be negative, the resulting update rule is not *convex*, but *affine*; however, it remains convergent because convexity is only broken finitely many times, with a penalty of $O(n^2)$ when compared to the theoretical convergence time of the METROPOLIS update.

When trying to design algorithms for directed networks more generally, we are confronted with the fact that the network symmetries make it essentially impossible to keep track of the multiplicities of the input values, and local algorithms cannot reliably compute the average under assumptions of connectivity alone. An additional symmetry-breaking device is thus needed in order to solve average consensus over general directed networks, and here we resort to randomization by giving each agent access to a private random oracle. As randomization can be extremely powerful in breaking symmetries, we take a finer look at the memory and bandwidth requirements of our algorithms, and show that, with high probability, our algorithm $\overline{\mathcal{R}}$ computes a good approximation of the average with a memory and bandwidth footprint in $O(\log\log n)$. Moreover, if we allow *some* global knowledge and provide each agent with a bound $N \geqslant n$, the algorithm $\overline{\mathcal{R}}$ can be turned into the algorithm $\overline{\mathcal{D}}$, which computes an approximation of the average $\overline{\mu}$ in a *terminating* manner with messages in $O(\log N + \log\log n)$. As both algorithms rely on a information propagation primitive that consists in finding a global minimum in the network, these algorithms have a temporal complexity that is linear in the number of agents.

RELATED WORKS

General books covering distributed computing as a whole notably include Lynch [69], Tel [95], Attiya and Welch [8], and for a clear, short introduction,[6] we cite the recent textbook by Hirvonen and Suomela [58]. For an approach grounded in locality, let us cite Peleg [88]. Bertsekas and Tsitsiklis [13] constitutes a general reference with an approach closer to that of the distributed control community.

[6] Which has the good taste of being distributed under an open licence.

Agreement problems have long been central to the study of distributed systems. For the classic problem of terminating consensus, we note the classic references [66, 87, 48].

The topic of distributed computation over dynamic networks has received much interest in recent years, notably following the seminal works of O'Dell and Wattenhofer [79], and later of Kuhn, Lynch, and Oshman [63], which adopt models broadly similar to ours. Among many works on the topic, we note in this vein [65, 64, 47]. This topic has much deeper roots, let us cite for example [10, 1]; in fact, dynamicity in the form of lost messages, of which Santoro and Widmayer [92] is of interest in connexion with the present work, has been studied for much longer [2, 51]. Closer to this thesis, an algorithmic approach to *asymptotic* consensus was undertaken by Chazelle and by Charron-Bost [28], and Charron-Bost, Függer, and Nowak [30, 31], building on approaches developed in the distributed control community, notably the graph composition-centric approach of Cao, Morse, and Anderson [24, 25]. *Stabilizing* consensus, situated in-between terminating and asymptotic consensus in terms of computational strength, has been notably studied by Angluin, Fischer, and Jiang [7], and more recently by Charron-Bost and Moran [35].

More distant to us, the algorithmic study of dynamic networks includes Kempe, Kleinberg, and Kumar [62], as well as Casteigts et al. [27, 26]. More distant still, we note the different model with broader aims studied by Latapy et al. [67], and the model of *population protocols* studied by Angluin et al. [6].

For the specific aspect of *anonymous* computation, we build our arguments on the seminal work of Angluin [5], and extensions by Yamashita and Kameda [101, 102, 103], Boldi et al. [15], and Boldi and Vigna [16, 17, 19, 18]; considerations centered on *connectivity* of the communication graphs, but not on its shape, include Hendrickx, Olshevsky, and Tsitsiklis [56], Hendrickx and Tsitsiklis [57].

For the distributed control approach to consensus, we note the seminal work of DeGroot [46] on static convex update rules for complete graphs, extended to dynamic rules by Chatterjee [38] and Chatterjee and Seneta [39]. Of importance, we note Tsitsiklis [97], Tsitsiklis et al. [96], and much work on flocking and consensus that followed the publication of [91, 98], of which important milestones include [99, 72, 14, 55, 24, 83, 78, 40, 76].

PLAN

This monograph is divided in three chapters: Chapter 1 introduces our communication model, which is built on the fundamental tool of *dynamic graphs*, which are sequences of directed graphs, studied with the help of *graph composition*. Chapter 2 then presents the theory around the MaxWeight and MaxMetropolis algorithms for asymptotic consensus and average consensus. Chapter 3 considers the *randomized* algorithms $\mathcal{R}$, $\overline{\mathcal{R}}$, and $\overline{\mathcal{D}}$. Finally, we offer concluding thoughts and directions for further works in Section 3.6.

COMPUTATION IN DYNAMIC NETWORKS

1.1 INTRODUCTION

In this chapter, we are mostly concerned with bringing up the tools needed to discuss computation and consensus in dynamic networks. We recall some of the classic concepts pertaining to binary relations and directed graphs, and then extend these notions to our main tool for modeling time-varying networks, called here *directed graphs*, which are infinite sequences of directed graphs with a self-loop at each vertex. Using the tool of graph composition, we obtain a compact description of how information propagates in dynamic networks.

We then describe the communication and computational models that we use in this thesis: we consider infinite executions in which agents communicate by local broadcast over discrete rounds that are communication closed, with the pattern of communications in each round captured by a directed graph, and the communications for the entire execution then captured by a dynamic graph. As we are concerned with problems of an asymptotic nature, we take executions to be infinite and do not embed an explicit notion of termination. However, we define conditions over the behavior of the agents that amout to varying degrees of resoluteness in the computation, with the strongest versions being equivalent to having the execution actually terminate, since all agents then know that nothing else remains to be done.

Describing this hierarchy will allow us to relate the asymptotic problems that we study with agreement problems more traditional in the theory of distributed algorithms. Moreover, as we will be mainly concerned with the bottom layers of this hierarchy – that is, with weaker notions of computability – the difficulties met will arise from symmetries in the geometry of the network rather than on considerations of a global nature which arise when agents are to make coordinated and irrevocabe decisions while only ever having a partial and out-dated view of the state of the system. We will recall results pertaining to the theory of anonymous computability in distributed networks, and how they contraint the problems we seek to study. In particular, the problem of CONSENSUS can be interpreted as one of computing a relation and is therefore much easier to solve than that of AVERAGE CONSENSUS, which correponds to the computation of a *function* which enough symmetries in the network can make impossible even to approximate.

In terms of communication and computational models, the closest to the one we consider is the Heard-Of model introduced by Charron-Bost and Schiper [37], whose precise semantics constitutes a useful basis to study the propagation of messages in dynamic networks. Not too dissimilar to our model are classic models of synchronous distributed computation by message-passing like the LOCAL – where the bandwidth is typically unrestrained – or CONGEST – where messages are usually taken to occupy $O(\log n)$ bits, which precludes solutions based on full information protocols.

We note the important difference that the LOCAL and CONGEST models usually suppose centralized naming authority assigning unique identifiers to the agents, whereas we take the opposite stance and suppose that the agents start indistinguishable from one another. This distinction is central to matters of computability: identifiers can be used to break all symmetries in the network – in theory, if not easily in practice. If we assume an initial synchronization of the system, as is usually done, in a sense the only remaining challenge is one of efficiency: getting agents to solve a given task as fast as possible, with the smallest possible message or memory footprint, or any other measure that the designer seeks to optimize.

1.2 RELATIONS, GRAPHS, DYNAMIC GRAPHS

RELATIONS

Binary relations are at the basis of our model of communications in dynamic networks.

Definition 1 (Relations)**.** A BINARY RELATION (or simply RELATION) over sets A and B is a subset $R \subseteq A \times B$. If R is a relation over $A \times A$, we say that it is a relation over A. ✒

As is usual, we will generally write $x \ R \ y$ instead of $(x, y) \in R$. Although the main use that we will make of relations will be in capturing snapshots of communication patterns in networks, they constitute a basic building block of mathematics. Let us recall some classic terminology.

Definition 2 (Common properties of relations)**.** We say that a relation R is

FUNCTIONAL if $x \ R \ y$ and $x \ R \ z$ implies $y = z$;

REFLEXIVE if $x \; R \; x$ holds for all x;

SYMMETRIC if $x \; R \; y$ if and only if $y \; R \; x$;

ANTISYMMETRIC if $x \neq y$ implies that at most one of $x \; R \; y$ or $y \; R \; x$ is true;

TRANSITIVE if $x \; R \; y$ and $y \; R \; z$ implies $x \; R \; z$;

a PARTIAL ORDER is a relation that is reflexive, transitive, and antisymmetric; an EQUIVALENCE RELATION is one that is reflexive, transitive, and symmetric. ❧

Given a relation R over a set A, its REFLEXIVE, SYMMETRIC, or TRANSITIVE CLOSURES, are, respectively, the smallest reflexive, symmetric, or transitive relations over A containing R. Since $A \times A$ is an equivalence relation, these closures always exist.

Definition 3 (Relation composition)**.** Given sets A, B, and C, the COMPOSITION of relations $R \subseteq A \times B$ and $S \subseteq B \times C$, denoted $R \,;\, S$, is the relation over $A \times C$ given by

$$x \; R \,;\, S \; z \Leftrightarrow \exists y \in B \colon x \; R \; y \wedge y \; S \; z \quad . \tag{1.1}$$

❧

The relation composition is associative, and its identity is the IDENTITY RELATION $\mathcal{I}_A \coloneqq \{ \, (x, x) \mid x \in A \, \}$. For $k \in \mathbb{N}$, the k-th POWER of a relation $R \subseteq A \times A$ is then given recursively by

$$R^k = \begin{cases} \mathcal{I}_A & k = 0 \\ R \,;\, R^{k-1} & k > 0 \quad . \end{cases} \tag{1.2}$$

Reflexive behave monotonously for the composition.

Lemma 1.2.1. Let R and S be relations. If S is reflexive, we have

$$R \,;\, S \subseteq R \quad \text{and} \quad S \,;\, R \subseteq R \quad . \tag{1.3}$$

❀

It is well known that the transitive closure R^+ of a relation R, and its transitive and reflexive closure R^*, are given by

$$R^+ = \bigcup_{k=1}^{\infty} R^k \ , \qquad R^* = \bigcup_{k=0}^{\infty} R^k \ . \tag{1.4}$$

In the case of a reflexive relation, R^+ and R^* are equal, and since by Lemma 1.2.1 we have $R^0 \subseteq R^1 \subseteq R^2 \subseteq \cdots$, they are both given by the limit $\lim_{k\to\infty} R^k$. When the support of the relation R is finite, the geometric sequence R^k stabilizes over R^* after finitely many iterations.

DIRECTED GRAPHS

Throughout this monograph, we will mean by graph a *simple directed labeled graph*, which is given by a relation, tagged by the set over which this relation is considered.

Definition 4 (Directed graphs)**.** A DIRECTED GRAPH (or simply GRAPH) over a set V is a pair (V, E), where E is a relation over V. Given a graph $G = (V, E)$, the elements of V are the VERTICES of G, and their cardinal $|V|$ is the ORDER of G. The elements of E are the EDGES of G, and their cardinal $|E|$ is the SIZE of G. ✒

That is, graphs in our model are directed and can have SELF-LOOPS (u, u), but may only have one edge in each direction between two vertices – they are not *multi-graphs*. They are distinct from the *geometric* object to which they correspond, in the sense that we do not regard isomorphic graphs to be the same object. In other words, a graph $G = (V, E)$ is *labeled* by the set V, which is not to be confounded with the fact that agents were issued unique identifiers; we strive to design algorithms which are *specifically* designed to cope with the absence of unique identifiers. Here, we will be mainly concerned with graphs defined over finitely many vertices, which will be silently assumed whenever it is required for the discussion to make sense.

For any graph G, we denote its vertex set and its edge set respectively by $\mathrm{V}(G)$ and $\mathrm{E}(G)$. If (u, v) is an edge of the graph G, we will generally use the notation $u \xrightarrow{G} v$ rather than $(u, v) \in \mathrm{E}(G)$ or $u \,\mathrm{E}(G)\, v$. When the graph G is clear from the context, we simply write $u \longrightarrow v$.

Definition 5 (Neighborhoods, degrees)**.** Given a graph G the IN- and OUT-NEIGHBORHOODS of a vertex $v \in \mathrm{V}(G)$ are the sets

$$\begin{aligned} \mathrm{In}_v(G) &\coloneqq \{\, u \in \mathrm{V}(G) \mid u \xrightarrow{G} v \,\} \\ \text{and} \quad \mathrm{Out}_v(G) &\coloneqq \{\, w \in \mathrm{V}(G) \mid v \xrightarrow{G} w \,\} \ . \end{aligned} \tag{1.5}$$

The IN and OUT-DEGREES of the vertex v are the count of its in- and out-neighbors:

$$d^-_u(G) \coloneqq |\mathrm{In}_v(G)| \quad \text{and} \quad d^+_u(G) \coloneqq |\mathrm{Out}_v(G)| \ . \tag{1.6}$$

The COMPLETE GRAPH over a set V is the graph $K_V \coloneqq (V, V \times V)$; the IDENTITY GRAPH is $I_V \coloneqq \{\, (u,u) \mid u \in V \,\}$. The NULL GRAPH [52] is the graph $(\varnothing, \varnothing)$.

We inherit for graphs the terminology of relations: we will say that a graph $G = (V, E)$ is REFLEXIVE or TRANSITIVE when the relation E is. If the relation E is symmetric, the graph G is traditionally said to be BIDIRECTIONAL rather than "symmetric".

A graph G is a SUBGRAPH of a graph H when $\mathrm{V}(G) \subseteq \mathrm{V}(H)$ and $\mathrm{E}(G) \subseteq \mathrm{E}(H)$. Of particular importance are induced subgraphs: given a graph $G = (V, E)$ and a set S, the (sub)graph of G INDUCED by S is the graph $G_{|S} \coloneqq (V \cap S, E \cap S \times S)$.

Given two graphs G and H, a function $f\colon \mathrm{V}(G) \to \mathrm{V}(H)$ is a GRAPH MORPHISM for G and H if $f(u) \xrightarrow{H} f(v)$ holds whenever $u \xrightarrow{G} v$ does. If f is a bijective map whose inverse f^{-1} is also a graph morphism, then f is a GRAPH ISOMORPHISM.

We extend the notion of relation composition to that of graphs: given graphs $G = (U, E)$ and $H = (V, F)$,[1] their COMPOSITION is the graph

$$G \circ H \coloneqq (U \cup V, E \,;\, F) \ . \tag{1.7}$$

The POWERS of a graph (V, E) are then defined by $G^k = (V, E^k)$.

[1] Most of the time, we consider the case $U = V$, in which case we simply have $G \circ F = (U, E\,;\,F)$; composing graphs over different vertex sets will matter mainly when the agents of the network engage in the execution at different time.

ECCENTRICITY AND CONNECTIVITY

For any graph G, we will denote by $- \xRightarrow{G} -$ the transitive and reflexive closure of the relation $- \xrightarrow{G} -$. We say that v is REACHABLE by u in G when $u \xRightarrow{G} v$.

Reachability is intimately connected with the GEODESIC FORWARD DISTANCE (or simply DISTANCE) induced by the graph G over the set $\mathrm{V}(G)$:

$$\mathrm{d}_G(u,v) \coloneqq \inf\{\, k \in \mathbb{N} \mid u \xrightarrow{G^k} v \,\} \ , \tag{1.8}$$

with as usual $\inf \varnothing = \infty$. Indeed, we have $\mathrm{d}_G(u,v) < \infty$ if and only if $u \overset{G}{\Longrightarrow} v$.

We say that a vertex u is a ROOT of a graph G is $u \Rightarrow v$ for all $v \in \mathrm{V}(G)$. The ECCENTRICITY of the root u is then given by

$$\mathrm{ecc}_G(u) \coloneqq \max_{v \in \mathrm{V}(G)} \mathrm{d}_G(u,v) \ ; \tag{1.9}$$

if u is not a root, then we let $\mathrm{ecc}_G(u) \coloneqq \infty$. If any vertex u of a graph G is a root, then we say that G is ROOTED in u; G is ROOTED if it is rooted in any of its vertices.[2] The RADIUS of a graph G is given by

$$\operatorname{rad} G \coloneqq \min_{u \in \mathrm{V}(G)} \mathrm{ecc}_G(u) \ , \tag{1.10}$$

which is finite if and only if G is rooted, in which case we have $\operatorname{rad} G < |\mathrm{V}(G)|$.

[2] In general, it is not true that $u \Rightarrow v$ implies $v \Rightarrow u$, as is evident from considering the two-vertex graph $\circ \to \circ$. As a consequence, rootedness is a strictly weaker notion than strong connectivity, defined below.

WHEN every vertex is reachable from every other, we say that the graph is STRONGLY CONNECTED. We define the DIAMETER of a graph G by

$$\operatorname{diam} G \coloneqq \max_{u \in \mathrm{V}(G)} \mathrm{ecc}_G(u) \ , \tag{1.11}$$

which is finite if and only if G is strongly connected, and in that case we have $\operatorname{diam} G < |\mathrm{V}(G)|$, and $\operatorname{rad} G \leqslant \operatorname{diam} G \leqslant 2 \operatorname{rad} G$.

This notion can be strengthened into c-strong connectivity, with $c \in \mathbb{N}_{>0}$: we say that a strongly connected graph G is c-STRONGLY CONNECTED if for any set $T \subseteq \mathrm{V}(G)$ with cardinal $|T| < c$ the induced graph $G_{|\mathrm{V}(G)\setminus T}$ remains strongly connected. Classic combinatorial arguments related to Menger's theorem show that in that case $\operatorname{diam} G < \frac{|\mathrm{V}(G)|}{c}$; note that $c = 1$ corresponds to the strongly connected case.

Since we will be mostly interested in reflexive graphs, we will use the following property of connectivity to control the temporal complexity of our algorithms.

Lemma 1.2.2. Let $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ be reflexive graphs with same vertex set V. If G_1 is c_1-strongly connected and G_2 is c_2-strongly connected with $c_1, c_2 \in \mathbb{N}_{>0}$, then their composition $G_1 \circ G_2$ is either the complete graph K_V, or is $c_1 + c_2$-strongly connected. ❀

DYNAMIC GRAPHS

In our discrete time model, the communications occurring between agents in V in a given instant will be captured by a graph over V. The pattern of communication over time are then naturally captured by the following notion.

Definition 6 (Dynamic graphs)**.** A DYNAMIC GRAPH $\mathbb{G}$ with vertices in a finite set $\mathrm{V}(\mathbb{G}) = V$ is an infinite sequence $\mathbb{G}(1), \mathbb{G}(2), \dots$ of reflexive directed graphs over V.[3] ⬲

[3] In some rare instances, unambigously announced as such, we will allow the sequence to be finite or to have the vertex set vary over time.

Notice that each graph $\mathbb{G}(t)$ is assumed to be reflexive and that the set $\mathrm{V}(\mathbb{G})$ is assumed to be finite. We speak again of the ORDER of a dynamic graph $\mathbb{G}$ to denote the cardinal $|\mathrm{V}(\mathbb{G})|$.

Instead of $\mathrm{In}_u(\mathbb{G}(t))$, we write $\mathrm{In}_u(t; \mathbb{G})$, dropping the parameter $\mathbb{G}$ when it is clear from the context, and similarly for other notions: $\mathrm{Out}_u(t), d_u^-(t), d_u^-(t), \dots$

The CUMULATIVE GRAPH of a dynamic graph $\mathbb{G}$ over the interger interval $\{ t, \dots, t' \}$ is the graph

$$\mathbb{G}(t\colon t') \coloneqq \mathbb{G}(t) \circ \cdots \circ \mathbb{G}(t') \ , \tag{1.12}$$

where by convention $\mathbb{G}(t\colon t') = I_V$ if $t' < t$. Here as well, we will simplify notation for the neighborhoods and degrees and let for example $\mathrm{In}_u(t\colon t'; \mathbb{G})$ stand for $\mathrm{In}_u(\mathbb{G}(t\colon t'))$; as above, we will drop the parameter $\mathbb{G}$ when it is clear from the context.

DYNAMIC REACHABILITY

The notions pertaining to reachability for static graphs admit several different forms of generalizations, connected with one another. Here, we attempt to briefly sketch these notions and the relations between them.

The simplest way to generalize, say, rooted graphs, is to consider dynamic graphs that are term-wise rooted – that is, each individual graph $\mathbb{G}(1), \mathbb{G}(2), \dots$ is rooted. A common relaxation is to introduce a delay: given $\Delta \in \mathbb{N}_{>0}$, we say that a dynamic graph $\mathbb{G}$ is Δ-DELAYED rooted when each cumulative graph $\mathbb{G}(t\colon t + \Delta - 1)$ is rooted. We find yet another, much weaker condition when no such *uniform* bound Δ exists: we say that $\mathbb{G}$ is EVENTUALLY ROOTED if for each $t \in \mathbb{N}_{>0}$ there exists *some* $t' \geqslant t$ for which the cumulative graph $\mathbb{G}(t\colon t')$ is rooted.

Clearly, nothing singles out rootedness in this manner, and we similarly define, for example, term-wise, delayed, or eventual c-strong connectivity. In particular, we will simply say that a dynamic graph is EVENTUALLY CONNECTED rather than "eventually strongly connected".

The above notions have in common that they relegate to the background the dynamic aspect of dynamic graphs: they approach the problem of dynamic reachability by constructing fixed graphs over which we can apply traditional notions. Another approach, which makes a first-class citizen of the dynamic graph, starts with a spatio-*temporal* notion of reachability: a vertex v is REACHABLE from u in a dynamic graph $\mathbb{G}$ at time t if $u \xrightarrow{\mathbb{G}(t\colon t')} v$ holds for some $t' \geqslant t$, which we alternatively note $u \overset{t\colon t'}{\Longrightarrow}_{\mathbb{G}} v$ or simply $u \overset{t\colon t'}{\Longrightarrow} v$ if $\mathbb{G}$ is clear.

Similarly, we can define the GEODESIC FORWARD DISTANCE (or, again, simply DISTANCE) at time t by

$$\mathrm{d}_t(u, v; \mathbb{G}) \coloneqq \inf\{\, k \in \mathbb{N} \mid u \overset{t\colon t+k-1}{\Longrightarrow} v \,\} \;, \tag{1.13}$$

where as usual the annotation $\mathbb{G}$ is dropped when the context is clear. The notion of dynamic ECCENTRICITY naturally follows:

$$\mathrm{ecc}_t(u; \mathbb{G}) \coloneqq \sup_{v \in \mathrm{V}(\mathbb{G})} \mathrm{d}_t(u, v; \mathbb{G}) \;. \tag{1.14}$$

As opposed to the static case, we may well have $\mathrm{d}_t(u, v) < \infty$, but $\mathrm{d}_{t'}(u, v) = \infty$ for the same vertices u and v but a later time t'. Generalizing the radius and diameter thus requires some care. To start with, for any $t \in \mathbb{N}_{>0}$ we define the MINIMAL and MAXIMAL ECCENTRICITIES at time t of a dynamic graph $\mathbb{G}$ by

$$\begin{aligned} \min \mathrm{ecc}_t\, \mathbb{G} &\coloneqq \min_{u \in V} \mathrm{ecc}_t(u; \mathbb{G}) \;, \\ \max \mathrm{ecc}_t\, \mathbb{G} &\coloneqq \max_{u \in V} \mathrm{ecc}_t(u; \mathbb{G}) \;. \end{aligned} \tag{1.15}$$

It is possible for a dynamic graph $\mathbb{G}$ to have $\max \mathrm{ecc}_t\, \mathbb{G} < \infty$ for some $t \in \mathbb{N}_{>0}$, but $\max \mathrm{ecc}_{t'}\, \mathbb{G} = \infty$ at a later time $t' > t$, and the same is true of the minimal eccentricity. Although they are well defined, such graphs inadequately model network dynamics over which to reason about computability: clearly, whether the system is connected or not, or rooted or not, has to impact what it can achieve. A dynamic graph whose minimal eccentricity is finite for some time, then becomes infinite, may permit

some global behaviors that cannot be achieved if the eccentricity is *never* finite, but only because it starts with the right timing; running the same algorithm on the same network at a later time could lead to different results, and so we deem such a dynamic graph ineffective at capturing the conditions enabling certain kinds of computation.

We shall therefore limit our considerations to the case where the extremal eccentricities do not display such behavior. We can convince ourselves that the dynamic graphs for which $\min \operatorname{ecc}_t \mathbb{G} < \infty$ holds for all t are exactly those that are eventually rooted; similarly, $\max \operatorname{ecc}_t \mathbb{G} < \infty$ holds for all t if and only if $\mathbb{G}$ is eventually connected. When uniform bounds are available, we obtain the classic notions of the radius and diameter of a dynamic graph.

Definition 7 (Dynamic radius and diameter)**.** The DYNAMIC RADIUS and DYNAMIC DIAMETER of a dynamic graph $\mathbb{G}$ are given by

$$\operatorname{rad} \mathbb{G} \coloneqq \sup_{t \geqslant 1} \min \operatorname{ecc}_t \mathbb{G} \ , \qquad \operatorname{diam} \mathbb{G} \coloneqq \sup_{t \geqslant 1} \max \operatorname{ecc}_t \mathbb{G} \ . \tag{1.16}$$

Since no confusion can arise, we simply speak of the radius and diameter of a dynamic graph. To connect these notions with those of static graphs, observe that for any graph G we have $\operatorname{rad} G = \operatorname{rad}(G, G, \cdots)$, and similarly $\operatorname{diam} G = \operatorname{diam}(G, G, \cdots)$.

We find an important example of dynamic graphs whose radius can be bounded in delayed connected graphs.

Proposition 1.2.3. Let $\mathbb{G}$ be a dynamic graph of order $n \in \mathbb{N}_{>0}$. Suppose that $\mathbb{G}$ is Δ-delayed c-strongly connected, for some constants $\Delta, c \in \mathbb{N}_{>0}$ – that is, each cumulative graph $\mathbb{G}(t\colon t + \Delta - 1)$ is c-strongly connected. In this case, we have

$$\operatorname{diam} \mathbb{G} \leqslant \left\lceil \frac{\Delta(n-1)}{c} \right\rceil \ . \tag{1.17}$$

In particular, $\operatorname{diam} \mathbb{G} \leqslant n - 1$ for a term-wise strongly connected dynamic graph $\mathbb{G}$.

As we have just seen, the notion of delayed strong connectivity is in fact equivalent to having a finite diameter – a dynamic graph $\mathbb{G}$ with finite diameter is by definition $\operatorname{diam} \mathbb{G}$-delayed complete, and

thus diam $\mathbb{G}$-delayed strongly connected. Dynamic graphs of finite diameter thus do not constitute a new class. Why then introduce the notion? The bounds over the diameter derived in Proposition 1.2.3 are necessarily linear in the order of the network, which translates in linear factors in the complexity of some algorithms, see for example our temporal bounds in [32]. This is in fact an artefact of assuming a termwise strongly connected graph: the "real" factor at play is the diameter, which turns out to be at most $n - 1$ given the assumptions. By considering the diameter as a first-class parameter, we can better distinguish a factor n that inherently depends on the scale of the system from one that only reflects a connectivity assumption. A similar argument could be made about delayed rooted dynamic graphs and dynamic graphs with a finite radius.

Given dynamic graphs $\mathbb{G}$ and $\mathbb{H}$, a function $f\colon \mathrm{V}(\mathbb{G}) \to \mathrm{V}(\mathbb{H})$ is a DYNAMIC GRAPH MORPHISM for $\mathbb{G}$ and $\mathbb{H}$ if it is a graph morphism for each pair $\mathbb{G}(t), \mathbb{H}(t)$. It is a DYNAMIC GRAPH ISOMORPHISM if it is a graph isomorphism for each pair $\mathbb{G}(t), \mathbb{H}(t)$. Notice that for two dynamic graphs $\mathbb{G}$ and $\mathbb{H}$ to be isomorphic, it does not suffice that each pair $\mathbb{G}(t), \mathbb{H}(t)$ be isomorphic; the *same* mapping of vertices $f\colon \mathrm{V}(\mathbb{G}) \to \mathrm{V}(\mathbb{H})$ must induce a graph isomorphism for *each* t.

NETWORK CLASSES

Defining what is computable by anonymous agents over a given network is made difficult by the fact that the characteristics of a *specific* network might make certain things possible, but in a non-generalizable manner. As an example, the classic example of symmetry breaking that is *leader election* is trivial if the communication graph is a star graph, – with a vertex connected to every other, and no other edge –all agents figure out whether they are at the center, and the only agent which satisfies the test picks up the crown. Although simple and effective, it is clear that this approach cannot be transposed to, for example, a ring graph. Indeed, as shown by Angluin in her seminal article [5], leader election is in fact impossible when the communication graph is a cycle.

The aforementioned algorithm, in a sense, *embeds information* about the network over which it operates. We follow the approach used by, for example, Boldi and Vigna in [17] and other works, and capture this notion of available information or knowledge by the means of network classes. We will call a NETWORK CLASS a set $\mathfrak{C}$ of dynamic graphs that is stable for dynamic graph isomorphisms – that is, given $\mathbb{G} \in \mathfrak{C}$, the network

class $\mathfrak{C}$ contains every dynamic graph $\mathbb{H}$ with vertices $\mathrm{V}(\mathbb{H}) = \mathrm{V}(\mathbb{G})$ that is isomorphic to $\mathbb{G}$. A network class (or simply CLASS) thus captures the kind of dynamic communication topology that a network might display – or, rather, what is *known*[4] about this topology – without giving any *a priori* information about individual roles in the network, as any role that an agent may play in a given class could as well be played by all other agents. For any class $\mathfrak{C}$, we will denote by $\mathfrak{C}_{|n}$ its restriction to dynamic graphs of order n.

[4] To cite [17]: "[...] classes specify knowledge: the larger the class, the smaller the knowledge (the class of all networks corresponds to no knowledge at all; a singleton to maximum knowledge)".

Sometimes, particularly to express impossibility results, it will be convenient to speak of "all dynamic graphs" of a certain kind. We only use dynamic graphs as abstractions capturing the patterns of message exchange in the network; typically, for a system of n agents in a set V, we will identify the agents with the vertices of the dynamic communication graphs of the execution – that is, we suppose $\mathrm{V}(\mathbb{G}) = V$. We could as well arbitrarily order the agents – so that $V = \{u_1, \ldots, u_n\}$ – and look at the isomorphic dynamic graph with vertex set $[n]$ given by the isomorphism $u_i \mapsto i$. As a consequence, when needed, we will default to representing dynamic graphs with n vertices as over the vertex set $[n]$. When considering network classes, the specific bijection $u_i \mapsto i$ does not matter, as a dynamic graph over $[n]$ belongs to a class $\mathfrak{C}$ if and only if all isomorphic dynamic graphs do as well.

1.3 DISTRIBUTED MODEL

Here, we seek to sketch our communication and computational models with the aim of capturing the abstract behavior of distributed algorithms running over dynamic networks. It corresponds to common models studied in the literature on algorithms for dynamic network as well as on distributed control. In contrast to some classic models used when studying time-invarint networks, individual edges of the communication graph follow *delivery semantics* in the spirit of the Heard-Of model [37]: the presence of an edge (u, v) in a graph means that a message coming from u was effectively received by v.

Because we model connexions that are fundamentally uncertain and fleeting, we assume in particular that individual agents do not control, or indeed know of, who receives their messages. Lacking such control and knowledge, agents are unable to individually address their neighbors, which will turn out to have deep implications for computability. This characteristic, referred to as a *local broadcast* communication primitive,

is commonplace when studying dynamic networks, as can be seen for example in [63].

Adopting this view means that our model has no notion of "link failure": a link failing simply corresponds to a missing edge in the communication graph. As a more subtle consequence, although we consider a discrete time model in which agents act in *virtual synchrony*, the model embeds no assumption that the network behaves anything like a *real-time* system: agents might well be badly desynchronized, which would translate as their being disconnected in the communication graph.

TIME AND COMMUNICATION

We thus consider a set V of $|V| = n$ networked agents engaged in a distributed computation. The computation is sliced into successive discrete layers called *rounds*, each denoted by its round number $t = 1, 2, \ldots$; we mark from the onset that an agent of the system is *not* assumed to have access to the current round number t.

In a round, each agent $u \in V$ successively **a)** casts a *single* message $m_u(t)$; **b)** receive some of the messages $m_v(t)$ sent by other agents in the same round; **c)** undergoes an internal transition to a new state; and **d)** produces an OUTPUT VALUE $x_u(t)$ and proceeds to round $t + 1$. The rounds are COMMUNICATION CLOSED, which means that agents only receive in round t messages that were sent in round t.

Notice in particular that all agents hearing from agent u in a given round t receive the same message $m_u(t)$ – we speak of *local broadcast*, and here, since no confusion can arise, of *broadcast*. An agent u need not hear from the entire set V in any given round, and we allow the set of agents it does hear from to change arbitrarily from one round to the next; as no message loss can occur between an agent and itself, our only *a priori* assumption is that an agent always receives its own message. Apart from its own, the messages received by an agent in a given round arrive unsorted, which we refer to as *mailbox reception*; this corresponds to the fact that the messages received by agent u during round t form a multi-set $[\, m_{v_1}(t), \ldots, m_{v_k}(t) \,]$, for some agents $v_1, \ldots, v_k \in V$.

In each round, this model naturally defines a binary relation $E_t \subseteq V \times V$ by $(u \; E_t \; v) \Leftrightarrow (v$ receives u's round t message$)$, and the communications over the entire sequence of rounds are then captured by a dynamic graph $\mathbb{G} = (\mathbb{G}(t) = (V, E_t))_{t \in \mathbb{N}}$, which we call the DYNAMIC COMMUNICATION GRAPH.

NETWORKED AUTOMATA AND ALGORITHMS

From the communication model, we discern three moments when an agent has the opportunity to make choices: when crafting its round message, when transitioning in each round, and when producing its round output. In addition, the agent has latitude in determining its initial state, to be discussed shortly. Everything else is beyond the agent's control: the content of other messages, which messages gets delivered to whom, the choices made by other agents, and so forth.

Each of these choices is driven by the programming of the agent, which, taken together, we call an ALGORITHM. Supposing that agents take states $\sigma \in \Sigma$, transmit messages $m \in M$, and output values $x \in \Omega$, then an agent's possible transitions are given by a TRANSITION RELATION over $(M^{\oplus} \times \Sigma) \times \Sigma$, its messages by a SENDING RELATION over $\Sigma \times M$, and its round outputs by an OUTPUT FUNCTION $\Sigma \to \Omega$. Messages and transitions to be relations rather than functions so as to be able to define non-deterministic algorithms; outputs are always deterministically obtained from an agent's state at the end of the round. Round outputs are less of an intrinsic part of the algorithm than a manner of singling out the observable part of an agent's behavior.

In addition to the above, we assume that each agent u starts by picking up an INPUT VALUE $\mu_u \in \Xi$ from its environment, used to set up its initial state *via* an INITIALIZATION RELATION over $\Xi \times \Sigma$. We will be mainly interested in *local* and *deterministic* algorithms, where all aforementioned relations are in fact functional.

Definition 8 (Local deterministic algorithm)**.** A LOCAL AND DETERMINISTIC ALGORITHM (hereafter, simply a LOCAL ALGORITHM) with INPUT SPACE Ξ, OUTPUT SPACE Ω, STATE SPACE Σ, and MESSAGE SPACE M is a collection $(\iota, \xi, \tau, \zeta)$ comprised of

- an INITIALIZATION FUNCTION $\iota \colon \Xi \to \Sigma$;
- a SENDING FUNCTION $\xi \colon \Sigma \to M$;
- a TRANSITION FUNCTION $\tau \colon M^{\oplus} \times \Sigma \to \Sigma$; and
- an OUTPUT FUNCTION $\zeta \colon \Sigma \to \Omega$.

A local algorithm is relative to an agent; the local algorithms of all agents in the system, collected together, form a DISTRIBUTED ALGORITHMS. Here, we will only consider UNIFORM distributed algorithms, taken to mean that all agents run the same local algorithm.

We call such algorithms local because the only mean for agents to acquire information is through interacting with their immediate incoming neighbors in the communication graph. Information such as their out-degree in any round, for example, is not *a priori* available in this model, as it cannot be deduced from an agent's prior state and the collection of its incoming messages. Under our terminology, an algorithm such as PUSHSUM [61, 77][5] is not "local", because it depends on agents being provided information – their out-degree – that cannot be gathered locally. Formally, we can say that the PUSHSUM algorithm does not have a sending *function*, but a sending *relation* $\xi \subseteq \Sigma \times M$. Alternatively, this can be viewed as the PUSHSUM sending function $\xi \colon \Sigma \times \mathbb{N}_{>0} \to M$, expecting at each step a certificate in $\mathbb{N}_{>0}$. Claims over the correctness of the PUSHSUM algorithm can then be interpreted in the same manner as claims made over non-deterministic Turing machines: there exists a collection of certificates which lead the system to behaved in the desired manner. Similarly, a uniform distributed algorithm relying on unique identifiers will be viewed as non-local,[6] as two agents with the same input value are not initialized in the same state. Thus, whenever mentioning identifiers, we shall speak of a CENTRALIZED AUTHORITY issuing such identifiers, to highlight the embedded assumption of centralized control.

On the other hand, the *randomized* algorithms discussed in Chapter 3 *will* be local, in the sense that the agents use only local information and the outcomes of private random coins.

[5] In the PUSHSUM algorithm, agents make two parallel passes of a linear iterative update scheme of the sort considered in Chapter 2, but with the weight assigned to each value $x_i(t)$ depending on the *out-degree* $d_i^+(t)$; the ratio of the two values thus computed converges towards the average $\overline{\mu}$.

[6] Unfortunately, this interpretation conflicts with the standard terminology in the contemporary of distributed algorithms, where the LOCAL model roughly corresponds to the model used here, plus unique identifiers.

EXECUTIONS AND I/O RELATIONS

We call an EXECUTION a complete run of a distributed algorithm – it is the collection, for all rounds, of the states, messages, and outputs of all agents. In a given execution, we denote by $x_u(t)$ agent u's round t output, and the infinite sequence $x_u(1), x_u(2), \ldots$ is agent u's OUTPUT STREAM in the execution. If all agents run local algorithms and start engaging in the execution simultaneously in round t, then an agent's output stream is a function of the input values, $\boldsymbol{\mu}$, and of the dynamic communication graph of the execution, and so is the OUTPUT STREAM of the *system* – the sequence $\boldsymbol{x}(1), \boldsymbol{x}(2), \ldots$ of the output vector in each round.

However, achieving simultaneous starts is *in itself* a complex distributed task, particularly for anonymous agents – see [36] which studies this question in our model of dynamic networks. Most often, we will make the simplifying assumption that agents *do* start simultaneously, but only when we are justified in doing so by the fact that we discuss algorithms that are, in fact, resilient to asynchronous starts. Otherwise, we will explicitly consider the possibility for agents to start asynchronously, by adopting the model used in [36].

In this model, every agent begin the execution in an *inactive* state; to each agent u corresponds an additional parameter $s_u \in \mathbb{N}$, called the START SIGNAL of agent u, which describes the last round of the execution during which u is inactive, and which we assume here to be driven by an external process beyond our control. At the end of round s_u, the agent sets up its initial state as described above, and, starting in round $s_u + 1$, it normally engages with the system according to its local algorithm. *Until* round s_u, however, agent u only emits NULL MESSAGES – messages devoid of content with respect to the current execution. Likewise, an inactive agent has an empty output.

A null message may be, for example, a regular message pertaining to another execution, or it may be a *heartbeat*, an empty message periodically emitted by a sleeping agent to remind the network of its existence. Although null messages have no content to be interpreted by the rest of the system, they are picked up by neighboring agents according to the communication graph in each round. If the graph is sufficiently connected the system can detect the presence of inactive agents in its midst, and refrain from moving too far along in the computation until all agents are active. The *firing squad* problem, as studied in [36], consists in ensuring that agents re-synchronize after starting in arbitrarily desynchronized states.

To summarize, an execution of a distributed algorithm produces an infinite stream of output vectors $\boldsymbol{x}(1), \boldsymbol{x}(2), \ldots$, which is a function of the dynamic communication graph, the input values, and of the collection of start signals unless simultaneous starts are assumed. The communication graph, the input values, and when applicable the start signals, are assumed to be arbitrarily taken from some domain and are referred to as the *parameters* of the execution. In other words, a distributed algorithm induces a correspondance between a choice of parameters and the output stream of the system, which we can interpret as a complex input/output relationship between the input values and the output stream, mediated

by the other parameters. In the following section, we isolate situations in which we define simpler input/output relationships, leading to the notion of algorithmic function computation, of consensus, and of average consensus.

1.4 BIG-STEP OPERATIONAL SEMANTICS

COMPUTING AGENTS

A given execution of a distributed algorithm turns an input vector $\boldsymbol{\mu}$ into an infinite sequence of output vectors $\boldsymbol{x}(0), \boldsymbol{x}(1), \cdots$. We next introduce some definitions in order to equip our model with simpler evaluation semantics, allowing simple expressions of our problems. At the same time, we will sketch a hierarchy of computational difficulty that is orthogonal to the task being solved. Throughout this section, V is a finite set of cardinal $n \in \mathbb{N}_{>0}$, and (Ω, d) is a metric space.

Asymptotic computation We describe three increasingly strict ways an agent can be said to be "computing" something in an execution. Let $u \in V$ be an agent of a distributed system, and let $x_u : \mathbb{N} \to \Omega$ denote the sequence of its round outputs in a given execution of an algorithm.

Definition 9 (Asymptotic computation)**.** Let $u \in V$ be an agent in a distributed system, and let $x_u : \mathbb{N} \to \Omega$ denote the sequence of its round outputs in an execution. If this sequence satisfies

CONVERGENCE $\exists\, \omega_u \in \Omega : x_u(t) \to_t \omega_u$,

we say that agent u ASYMPTOTICALLY COMPUTES the value ω_u in the execution. ◂

Stabilization and eventual computation This definition is relative to the metric considered over the set Ω. If the agent asymptotically computes a value for a given metric, it then computes the same value for coarser metrics, but not necessarily for finer ones. We distinguish the important case of the DISCRETE METRIC $d(x, y) = 0$ if $x = y$, $d(x, y) = 1$ otherwise. Indeed, as it is the finest topology over any set, asymptotic computation for the discrete metric implies asymptotic computation for every other. A sequence is STABILIZING when it converges with respect to the discrete metric.

Definition 10 (Stabilizing computation)**.** We say that agent u engages in a STABILIZING, or EVENTUAL computation when its output satisfies

STABILIZATION $\exists t_u^*, \forall t, t' \geqslant t_u^* \colon x_u(t) = x_u(t')$.

Irrevocable decisions In order to relate our discussion to traditional models, we next describe how the act of returning a value can be expressed in our model of infinite executions. To this effect, we fix an element $\perp \notin \Omega$, and we suppose that agent u runs an algorithm with output set $\Omega' := \Omega \cup \{ \perp \}$ – that is, we either have $x_u(t) \in \Omega$ or $x_u(t) = \perp$. We interpret the value $\perp$ as an *absence* of output in Ω.

Definition 11 (Returning computation)**.** We say that agent u engages in a RETURNING COMPUTATION *over the set* Ω when its output satisfies

DECISION $\exists t_u^* \in \mathbb{N} \colon x_u(t_u^*) \neq \perp$, and

IRREVOCABILITY $\forall t, t' \in \mathbb{N}, t \leqslant t' \colon x_u(t) = \perp \vee x_u(t') = x_u(t)$.

Remark that irrevocability implies that the agent's output changes finitely many times – at most once. Since decision implies that the output will eventually take values in Ω, a returning computation is stabilizing in Ω, and thus also asymptotic. Since these three notions are of increasing strengths, and the topology over Ω is generally unambiguous, we will simply say that an agent COMPUTES a value in an execution when it does asymptotically. We will reserve the term asymptotic computation to highlight the fact that the computation happens in this weaker form, but not necessarily for stronger ones.

ADOPTING an outside vantage provides a simple interpretation for the above computational notions. Suppose that a third party observer can query agent u for its current output value, and seeks to learn the result ω_u of the computation. If the computation is asymptotic, then querying the agent at intervals should give increasingly better estimates of the result of the computation – after a finite period where the output may display an arbitrary behavior. For a stabilizing computation, the output will eventually stop changing, and thus if the output remains stable for

some time the observer may become increasingly convinced that it has learned the true result ω_u. Finally, for a returning computation, the agent itself may signal to the observer that it has reached the result ω_u.

COORDINATION

Termination We are now able to interpret, in some executions, the behavior of *an individual agent* as a computation. We easily extend this interpretation to the entire system: we will say that the system performs a computation (asymptotic, stabilizing, returning) when each individual agent does, and in this case the result of the computation is the vector $\boldsymbol{\omega} = \lim_t \boldsymbol{x}(t)$. Taking the view of the entire system also allows us to express a fourth, stronger form of computation.

Definition 12 (Termination)**.** We say that an execution is TERMINATING if it performs a returning computation, and if the round output of the system satisfy

SIMULTANEOUS DECISIONS $\forall u, v \in V, \forall t \in \mathbb{N}, x_u(t) = \bot \Leftrightarrow x_v(t) = \bot$.

An agent engaged in a computation that is simply returning may have to keep participating actively to help other agents finish. Thus, even when all agents have privately irrevocably decided on a value, the system remains active forever, because no individual agent can detect that the computation has reached its end. A terminating execution is not subject to this issue: an agent only decides on a value when *all* agents are ready to do so. Once it decides on an output value, the agent may safely cease to participate in the network; it need no longer send any messages or mobilize any computational resources.

1.5 COMPUTATIONAL TASKS

COMPUTING OVER SEMILATTICES

Let us now look at a concrete example of a distributed algorithm, given in Algorithm 1, which eventually computes the infimum of the input values. This algorithm, which here we call FLOODINF, is natural to write and is indeed commonplace: see for example the reference book [95, Section 6.1.5].

Consider a partial order $\preccurlyeq$ over a set A. An element $x \in A$ is a LOWER BOUND of a subset $S \subseteq A$ when $x \preccurlyeq y$ whenever $y \in S$, which we denote by $x \preccurlyeq S$. An INFIMUM is a maximal lower bound: $z = \inf S \Leftrightarrow \forall x \preccurlyeq S : x \preccurlyeq z$; the infimum of the subset S does not necessarily exist, but it is unique when it does.

When $\inf\{x, y\}$ is defined for any two elements $x, y \in A$, we note $x \wedge y \coloneqq \inf\{x, y\}$[7], and we say that $(A, \wedge)$ is a MEET-SEMILATTICE (hereafter "semilattice"). In this case, the operation $\wedge$ is associative ($x \wedge (y \wedge z) = (x \wedge y) \wedge z$), commutative ($x \wedge y = y \wedge z$) and idempotent ($x \wedge x = x$).

[7] The operation $\wedge$ should be read "meet".

Semilattices provide us with an important class of computable functions. Assuming that an agent may compute $x \wedge y$ for any given x and y, we can make agents eventually compute the infimum of initial values in a distributed manner, using Algorithm 1

Algorithm 1: the algorithm FLOODINF, code for agent u.

Input: $\mu_u \in A$, a semilattice
Initially:
    $\mathtt{x}_u \longleftarrow \mu_u$
In each round:
    send $\mathtt{m}_u = \langle \mathtt{x}_u \rangle$
    receive $[\, m_{v_1}, \ldots, m_{v_d} \,]$ ▷ d neighbors
    $\mathtt{x}_u \longleftarrow x_{v_1} \wedge \cdots \wedge x_{v_k}$
    output $\mathtt{x}_u$

Proposition 1.5.1. All executions of the FLOODINF algorithm are stabilizing. For an input vector $\boldsymbol{\mu}$, and a dynamic communication graph $\mathbb{G}$ that is eventually connected, all agents eventually compute the infimum $\inf \boldsymbol{\mu}$. Once all agents are active, stabilization happens in at most $\operatorname{diam} \mathbb{G}$ rounds if the diameter is finite.

Proof. In any round t, we denote agent u's round output by $x_u(t)$, and we denote the set of its active neighbors by $\mathrm{In}^*_u(t) \coloneqq \{\, v \in \mathrm{In}_u(t) \mid s_v < t \,\}$, where s_v denotes the last rond during which agent v is inactive. For any round $t > s_u$, we then have $x_u(t) = \inf\{\, x_v(t-1) \mid v \in \mathrm{In}^*_u(t) \,\}$; $x_u(t)$ is thus weakly decreasing and takes values in a finite set, and therefore stabilizes in finite time.

Suppose then that the communication graph is eventually connected, and let $s_{\max}$ denote the last round with inactive agents. Clearly, by

round $t^* \coloneqq s_{\max} + \max \operatorname{ecc}_{s_{\max}} \mathbb{G}$ all agents have heard from the entire network, and so $x_u(t) = \inf \boldsymbol{\mu}$ for all $t \geqslant t^*$, and in particular for all $t \geqslant \operatorname{diam} \mathbb{G}$. □

We will consider a few important instances of semilattices.

Power sets Any set A induces a semilattice $(\mathcal{P}(A), \cup)$, where the partial order $\preccurlyeq_\cup$ is the *reverse* of the order given by the set inclusion: $\forall x, y \in \mathcal{P}(A) \colon x \preccurlyeq_\cup y \Leftrightarrow y \subseteq x$.

The union semilattice can be used to compute the set $X_{\boldsymbol{\mu}} = \{\, \mu_u \mid u \in V \,\}$ of the input values, and thus also any function or relation depending on $X_{\boldsymbol{\mu}}$ – that is, any function depending on the presence or absence of input values, but not on their multiplicity.

Moreover, if a centralized authority issues an unique identifier id_u to each agent u is given an unique identifier i_u, running Algorithm 1 using for input the pair (μ_u, id_u) computes the *multi*-set $[\, \mu_u \mid u \in V \,]$. As a consequence, given enough connectivity and unique identifiers, any function that is computable by a centralized algorithm is eventually computable.

Totally ordered sets A totally ordered set $(A, \leqslant)$ induces a semilattice $(A, \wedge)$ with $x \wedge y \coloneqq \min(x, y)$. In particular, the minimum of real input values is computable.

Cartesian products. Given k semilattices $(A_1, \wedge_1), \ldots, (A_k, \wedge_k)$, their cartesian product $A^\times \coloneqq A_1 \times \cdots \times A_k$ forms a semilattice $(A^\times, \wedge^\times)$ for the operation defined by

$$(x_1, \cdots, x_k) \wedge^\times (y_1, \cdots, y_k) \coloneqq (x_1 \wedge_1 y_1, \cdots, x_k \wedge_k y_k) \ . \quad (1.18)$$

When agents agree over k, it is possible to compute the latter example while prioritizing the size of the messages over the computation time, exchanging one vector entry at a time. We illustrate this approach in Algorithm 2.

Algorithms 1 and 2 offer a clear trade-off between the bandwidth needs and computation time for the computation of the same end result: the former is k times as fast as the latter, and transmits k times as much information per round – if we assume that each value $\mu_{vu}^{(i)}$ is represented over $\mathrm{O}(1)$ bits. Intermediate designs can be used as well: sharing ℓ entries

Algorithm 2: the algorithm FLOODINF, entry-wise version, code for agent u.

Input: $\boldsymbol{\mu}_u = (\mu_u^{(1)}, \cdots, \mu_u^{(k)}) \in (A^\times, \curlywedge^\times)$
Initially:
  $\mathbf{x}_u \leftarrow \boldsymbol{\mu}$
  $i \leftarrow 0$
In each round:
  send $m_u = \left\langle x_u^{(i)} \right\rangle$
  receive $[\, m_{v_1}, \ldots, m_{v_d} \,]$ ▷ d neighbors
  $x_u^{(i)} \leftarrow x_{v_1}^{(i)} \curlywedge_i \cdots \curlywedge_i x_{v_k}^{(i)}$
  $i \leftarrow (i+1) \bmod k$

in each round, multiplies the speed of computation and bandwidth needs by a factor of about k/ℓ.

The specific choice of how many entries to share in each round can thus be left to the application designer. However, while this choice does not impact the outcome of the computation under simultaneous start signals and over a dynamic graph that is strongly connected in each round, Algorithm 2 need not compute the correct value when we relax those assumptions. In particular, even with simultaneous starts, the convergence of the i-th entry $x^{(i)}$ depends on the behavior of the product $\mathbb{G}(i) \circ \mathbb{G}(i+k) \circ \cdots \circ \mathbb{G}(i+p\cdot k)$ as p grows. We remark that even for a dynamic graph with a finite radius, this product need not ever tend to the complete graph over V, and $x_u^{(i)}(t)$ need not ever tend to $\inf \boldsymbol{x}^{(i)}$. Substituting Algorithm 2 for Algorithm 1 is only permissible under sufficient guarantees of connectivity of the network.

COMPUTING RELATIONS

We will say that an execution of a distributed algorithm achieves ASYMPTOTIC CONSENSUS when all agents asymptotically compute the same value.

Let Ξ, Ω be sets, and R be a relation over $\Xi^\oplus \times \Omega$. We say that a distributed algorithm computes the relation R over a network class if **a)** all of its executions over the class achieve asymptotic consensus **b)** over an input vector $\boldsymbol{\mu}$, the consensus value ω satisfies[8] $\boldsymbol{\mu}\ R\ \omega$; and this for arbitrary input values taken in the set Ξ. The first example of relation of interest to us is given by $\boldsymbol{\mu}\ R\ \omega \Leftrightarrow \omega \in \operatorname{range} \boldsymbol{\mu}$.

[8] We slightly abuse notation here, as we denote by $\boldsymbol{\mu}$ the multi-set of the input values.

Definition 13 (Consensus)**.** An execution with input vector $\boldsymbol{\mu}$ solves CONSENSUS if it achieves asymptotic consensus over some value $\omega \in \operatorname{range} \boldsymbol{\mu}$; an *algorithm* solves consensus over a network class $\mathfrak{C}$ and a domain Ξ if it does in all of its executions over $\mathfrak{C}$ with input values taken in Ξ. ⬑

When the relation being considered is functional, we speak of computing the *function* $f : \Xi^{\oplus} \to \Omega$ when $\omega = f(\boldsymbol{\mu})$ in all executions. The main problem studied in this thesis is one of functional computation – namely, that of computing the average $\overline{\mu} = \frac{1}{|V|} \sum_{u \in V} \mu_u$.

Definition 14 (Average consensus)**.** An execution with input vector $\boldsymbol{\mu}$ solves AVERAGE CONSENSUS if it achieves asymptotic consensus over the average $\overline{\mu}$ of the input values. An *algorithm* solves average consensus over a network class $\mathfrak{C}$ and a domain Ξ if it does in all of its executions over $\mathfrak{C}$ with input values taken in Ξ. ⬑

WHAT the FLOODINF algorithm and Proposition 1.5.1 show is that functions defined by means of infima over the input values are computable whenever the communication graph is eventually connected. Unfortunately, these are about the *only* functions that are computable if we only make assumptions about the connectivity. Indeed, building upon the theory of computability in anonymous networks, Hendrickx and Tsitsiklis show in [57, Theorem 2] the following result.

Proposition 1.5.2 (Hendrickx et Tsitsiklis)**.** Let $f : \Xi^{\oplus} \to \Omega$ be a fonction, computed by a uniform distributed deterministic local algorithm over the class of strongly connected fixed communication graphs. Then, there exists a function $F : \mathcal{P}(\Xi) \to \Omega$ such that $f(\boldsymbol{x}) = F(\{\, x \mid x \in \boldsymbol{x} \,\})$ – that is, the function f is insensitive to the multiplicities of the input values. ⬑

As the main problem studied in this monograph, average consensus, is one of computing a function that *is* sensitive to multiplicities – the average – Proposition 1.5.2 clearly delineates boundaries to the kinds of networks where we can hope to do so in our local model. Each of the following chapter offers a type of answer to Hendrickx and Tsitsiklis's impossibility theorem: Chapter 2 shows that *asymptotic* average consensus can be obtained in any sufficiently connected dynamic networks where all communication links are bidirectional; Chapter 3 shows that stabilizing and terminating *approximate* average consensus – where the output is a close approximation of the average – can be done efficiently over arbitrary strongly connected networks.

SYMMETRIC COMMUNICATIONS: A LITTLE LEARNING GOES A LONG WAY

The results of this chapter originally appeared in pre-publication form in [Bernadette Charron-Bost and Patrick Lambein-Monette. Average Consensus: A Little Learning Goes A Long Way. *Oct. 12, 2020. arXiv:* `2010.05675 [cs]`*], with an extended abstract submitted at* STACS *2021.*

2.1 INTRODUCTION

Here, we consider systems with *bidirectional interactions*: I interact with you means that you interact with me, and vice versa. Many interactions in natural settings are of this kind: conversations and handshakes,[1], being neighbors or Facebook friends,[2] working towards a common goal or being opponents in a game. Many digital systems suppose reciprocal interactions as well, in some idealized form at least: such as radio communications modeled with geometric graphs or peer-to-peer networks – but notice that these interactions are *truly* bidirectional only as long as no one-sided faults ever appear.

[1] Although at the time of writing the practice is quickly losing popularity.

[2] But not Twitter followers: social networks are not necessarily bidirectional.

The family of *affine updates* that we study in this chapter is historically motivated by both kinds of interactions. Its genealogy can be traced back at least to DeGroot [46], who envisioned interactions over a complete graph to model *opinion dynamics* within a group trying, literally, to *reach a consensus* about a topic. On the other hand, Tsitsiklis [97] is considering similar kinds of updates for the distributed control of networked digital systems, subject to various failure modes such as transmission delays. Modern interest in the matter, however, was essentially kick-started by the problem of *flocking* – that of describing the coordinate behavior of flocks of birds after influential papers by Reynolds [91] and Vicsek et al. [98] that led to a sustained interest that has persisted since then.

We seek in particular to look at such update schemes with a computational angle: what schemes can be implemented by deterministic and local algorithms over dynamic networks? This is a subtle matter: the METROPOLIS update, for example, although introduced in [100] to be used over dynamic networks, precisely *because* it relies only on information present within bounded distance of each agent, involves in fact information that is *too far* for distributed agents to collect locally when the network is subject to frequent or impredictable change.

Our contribution in this chapter will be to provide a pair of algorithms, the MaxWeight and MaxMetropolis algorithms, implementing eponymous affine update rules over bidirectional dynamic networks. The MaxWeight rule is a *convex* update rule – the convex hull of the agents' outputs keeps shrinking – for *consensus*: the common limit is generally not the average $\overline{\mu}$; in contrast, the MaxMetropolis rule *does* compute the average, but it not convex, which may temporarily set back the convergence of the system.

CHAPTER NOTATION

If S is any non-empty finite set of real numbers, we denote its DIAMETER by $\operatorname{diam} S \coloneqq \max S - \min S$; no confusion can arise with the diameter of a graph or a dynamic graph.

In this chapter, we will be essentially looking at *linear* updates as a mean to achieve asymptotic agreement. As a consequence, the behavior of the system will be better discussed in terms of vectors, matrices, and other tools taken from the linear algebraist's belt. For convenience, instead of supposing the agents taken in an arbitrary finite set V, we identify them with the integers $1, \dots, n$,[3] so that we write $x_i(t)$ to denote agent i's round t output.

[3] We do *not* suppose the agents to be aware of those labels; they exist only for analysis and discussion.

We will consistently denote matrices and vectors in bold italic style, using capital letters for matrices $\boldsymbol{A}, \boldsymbol{B}$, and lowercase letters for vectors $\boldsymbol{u}, \boldsymbol{v}$, with their individual entries in regular weight: A_{ik}, B_{kj}, u_i, v_j. If $\boldsymbol{z}(0), \boldsymbol{z}(1), \boldsymbol{z}(2), \dots$ is an infinite sequence of vectors of $\mathbb{R}^n$, we will denote the entire sequence with the doubly struck letter $\mathbb{z}$. To any $n \times n$ complex matrix $\boldsymbol{A}$ is ASSOCIATED a unique graph $\mathrm{G}_{\boldsymbol{A}} = ([n], E_{\boldsymbol{A}})$, whose edges are given by $i \xrightarrow{\mathrm{G}_{\boldsymbol{A}}} j \Leftrightarrow A_{ji} \neq 0$.

Given a vector $\boldsymbol{v} \in \mathbb{R}^n$, we write $\operatorname{diam} \boldsymbol{v}$ to mean the diameter of the *set* $\{ v_1, \cdots, v_n \}$ of its entries. The diameter constitutes a seminorm over $\mathbb{R}^n$; in particular, we have $\operatorname{diam} \boldsymbol{v} \geqslant 0$. We call CONSENSUS VECTORS those of null diameter; they form exactly the linear span of the constant vector $\mathbf{1} \coloneqq (1, 1, \cdots, 1)^\intercal$. The IDENTITY MATRIX is the diagonal matrix $\mathbf{I} \coloneqq \operatorname{diag}(\mathbf{1})$.

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 $\boldsymbol{A}$ is STOCHASTIC if *each of its rows* sums to 1 – equivalently, if $\boldsymbol{A}\mathbf{1} = \mathbf{1}$. We denote by $\boldsymbol{A}^\intercal$ the TRANSPOSE of a matrix $\boldsymbol{A}$, and by $\boldsymbol{v}^\intercal$ that of a vector $\boldsymbol{v}$.

When both $\boldsymbol{A}$ and $\boldsymbol{A}^\intercal$ are stochastic, we say that the matrix $\boldsymbol{A}$ is DOUBLY STOCHASTIC. It is in particular the case when $\boldsymbol{A}$ is SYMMETRIC – that is, when $\boldsymbol{A}^\intercal = \boldsymbol{A}$.[4]

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

Recall that the CHARACTERISTIC POLYNOMIAL of a $n \times n$ complex matrix $\boldsymbol{A}$ is given by $\chi_{\boldsymbol{A}}(X) \coloneqq \det(\mathbf{I}X - \boldsymbol{A})$, and that its complex roots are the EIGENVALUES of the matrix $\boldsymbol{A}$. The set of all eigenvalues is denoted by $\operatorname{Sp} \boldsymbol{A}$. We can always arrange the n eigenvalues $\lambda_1, \dots, \lambda_n$ in weakly decreasing order of magnitude: $|\lambda_1| \geqslant \cdots \geqslant |\lambda_n|$. Under this convention, the quantity $\rho_{\boldsymbol{A}} \coloneqq |\lambda_1|$ is called the SPECTRAL RADIUS of the matrix $\boldsymbol{A}$, and the quantity $\gamma_{\boldsymbol{A}} \coloneqq |\lambda_1| - |\lambda_2|$ is called its SPECTRAL GAP. An eigenvalue is SIMPLE when it is a simple root of the characteristic polynomial; if λ_1 is a simple eigenvalue and that moreover the spectral gap is positive, we say that λ_1 is DOMINANT. By definition, a stochastic matrix $\boldsymbol{A}$ has 1 for eigenvalue; as is commonly known, it is also a largest eigenvalue – that is, $\rho_{\boldsymbol{A}} = 1$.

The MAXWEIGHT and MAXMETROPOLIS achieve asymptotic consensus in all bidirectional networks that are eventually connected; to discuss their convergence time – defined in the next section – we will restrict our considerations to the class $\mathfrak{G}$ of dynamic graphs that are bidirectional and strongly connected in every round; recall that its restriction to a system of n agents is denoted by $\mathfrak{G}_{|n}$.

[4] Note that symmetry of a matrix implies TYPE-SYMMETRY: the associated graph $\mathrm{G}_{\boldsymbol{A}}$ is bidirectional. However, type-symmetry is strictly weaker than symmetry.

2.2 AFFINE UPDATE RULES

An AFFINE UPDATE RULE is a condition over the output stream of an algorithm, where the individual round outputs $x_i(t)$ follow affine recurrence relations of the general form

$$x_i(t) = x_i(t-1) + \sum_{j \in \mathrm{In}_i(t), j \neq i} a_{ij}(t) x_j(t-1) \ , \tag{2.1}$$

where the time-varying weights $a_{ij}(t)$ may depend on the parameters of the execution: the dynamic graph $\mathbb{G}$ and the vector of input values $\boldsymbol{\mu}$. By convention, we let $a_{ii}(t) = 1 - \sum_{j \neq i} a_{ij}$, so that eq. (2.1) can be alternatively written $x_i(t) = \sum_{j \in \mathrm{In}_i(t)} a_{ij}(t) x_j(t-1)$; We say that the algorithm IMPLEMENTS the rule eq. (2.1) in its execution.

We call such a rule AFFINE, because by design an agent's local weights $a_{ij}(t)$ always form an affine combination – that is, we always have $\sum_{j\in\mathrm{In}_i(t)} a_{ij}(t) = 1$. The update rule is CONVEX if moreover all weights are non-negative. By and large, this subset of affine update rules has received the most attention. In particular, the convergence of convex update rules to asymptotic consensus is well understood, and is obtained whenever the dynamic communication graph is sufficiently connected and the weights are non-vanishing.

Proposition 2.2.1. Let $\mathbb{x}$ be a vector sequence satisfying the recurrence relation eq. (2.1) for a dynamic graph $\mathbb{G}$ and weights $a_{ij}(t) \geqslant \alpha > 0$ – that is, the weights are positive and uniformly bounded away from 0. If the dynamic graph $\mathbb{G}$ either **a)** has a finite radius, or **b)** is eventually rooted and bidirectional in each round, then the sequence $\mathbb{x}$ converges to a consensus vector.

We speak of UNIFORM CONVEXITY when such a parameter $\alpha > 0$ exists. The result expressed in Proposition 2.2.1 follows a long line of works that can be traced back at least to DeGroot [46], in the case of fixed weights a_{ij} and a complete graph, and to Chatterjee [38] and Chatterjee and Seneta [39] for time-varying weights over a partial graph. Later important works on the matter notably include Tsitsiklis [97] and Tsitsiklis et al. [96], then Jadbabaie et al. [60] and Cao et al. [24]. Condition **b)** is essentially due to Moreau [72], but we note a similar result with a simpler proof in Hendrickx and Blondel [55].

Traditionally, the temporal complexity of a system obeying eq. (2.1) is measured by means of the convergence time, first defined by Olshevsky and Tsitsiklis [81]. For a single sequence $\mathbb{z}$ and error $r > 0$, we let $\mathrm{T}(r;\mathbb{z}) \coloneqq \inf\{\, t \in \mathbb{N} \mid \forall \tau \geqslant t : \operatorname{diam} z(\tau) \leqslant r \,\}$. When considering an update rule, the temporal complexity will evidently depend on the order of the system being considered, and so we look at the WORST-CASE RELATIVE CONVERGENCE TIME, defined over a class $\mathfrak{C}$ for parameters r and n by

$$\mathrm{T}(r;n) \coloneqq \sup_{\mathbb{G}\in\mathfrak{C}_{|n},\, \boldsymbol{\mu}\in\mathbb{R}^n} \mathrm{T}(r\cdot\operatorname{diam}\boldsymbol{\mu};\mathbb{x}(\mathbb{G},\ \boldsymbol{\mu}))\ , \tag{2.2}$$

where $\mathbb{x}(\mathbb{G},\boldsymbol{\mu})$ denotes the sequence of estimates obtained by applying the update rule over the pair $\mathbb{G}$, $\boldsymbol{\mu}$, since for a deterministic algorithm it is uniquely defined.

What kind of convergence times can we expect when considering the general family of convex update rules? As a general matter, a lower

bound of $\mathrm{T}(r;n) = \Omega(n^2)$ is known to apply [82] over the class of fixed *line graphs*. Under the assumptions of Proposition 2.2.1, it is clear that no general bound can be given for the class of networks that are merely eventually connected, as it contains communication graphs that remain disconnected for arbitrarily long periods. For dynamic graphs with a finite radius, the best general bound involving no other assumption, to our knowledge, is that given by Charron-Bost et al. [31][5] using the tool of non-split graphs: a graph G is NON-SPLIT when any two of its vertices $u, v \in \mathrm{V}(G)$ share an in-neighbor – that is, $\mathrm{In}_i \cap \mathrm{In}_j \neq \varnothing$ for all pairs $i, j \in \mathrm{V}(G)$. Using the fact that the composition of $|V| - 1$ rooted graphs with same vertex set V is non-split, they provide the following result.

[5] Non-split graphs were previously considered by Cao, Morse, and Anderson [24], where they are called *neighbor-shared* rather than non-split

Proposition 2.2.2. Let a dynamic graph $\mathbb{G}$ be Δ-delayed non-split, for some delay $\Delta \in \mathbb{N}_{>0}$, and let $\mathbb{x}$ be a vector sequence satifsying eq. (2.1) over $\mathbb{G}$. If the update is uniformly convex for some parameter $\alpha > 0$, then for any error $r > 0$ we have:

$$\mathrm{T}(r;\mathbb{x}) \leqslant \Delta\alpha^{-\Delta} \log \frac{1}{r} + \Delta - 1 \quad . \tag{2.3}$$

Remark that the dynamic graphs that are delayed-non-split are in fact those with a finite radius, and that we have $\Delta \leqslant \operatorname{rad} \mathbb{G}$. For a dynamic graph with n vertices that is D-delayed rooted and a parameter $\alpha \geqslant \frac{1}{kn}$ for positive constants D and k, we then have the bound

$$\mathrm{T}(r;\mathbb{x}) = \mathrm{O}\big(n^n \log \tfrac{1}{r}\big) \quad , \tag{2.4}$$

showing that the best general bound available is super-exponential in the number of agents. Moreover, as seen next, there are known examples involving the EQUALNEIGHBOR update rule – over a fixed strongly connected graph, and over a dynamic graph that is bidirectional and strongly connected at each step – which display a convergence time in $2^{\Omega(n)}$. Polynomial bounds thus cannot be obtained from connectivity arguments alone.

The relationship between the radius and the delay to non-splitness is more involved than the immediate bound $\Delta \leqslant \operatorname{rad} \mathbb{G}$. It was shown by Charron-Bost and Schiper [37] that the radius of a non-split graph with n vertices is at most $\mathrm{O}(\log n)$, and more recently by Függer et al. [49] that it is bounded by $\mathrm{O}(\log \log n)$.

To summarize, convex update rules are advantageous when it comes to their *convergence*, but their convergence *time* is poor in general. As will be made clear by the case of the EqualNeighbor rule, the directivity of influence is hugely responsible, as the rule moves from polynomial convergence time over fixed bidirectional graphs down to an exponential one over directed ones. In this chapter, we thus focus on networks with bidirectional communications in the hope of retaining polynomial convergence times. This is not sufficient by itself: as will be seen in the second example given by the EqualNeighbor rule, dynamic communications re-introduce directionality in the patterns of influence, even if all links are bidirectional in every round. Our contribution is to provide distributed local algorithms producing update rules that retain a polynomial convergence time under arbitrary fluctuations in the communication graphs.

2.3 EUCLIDIAN GEOMETRIES FOR UPDATE RULES

Here, we introduce the elements that will be needed in the main proofs of our theorems Theorems 2.5.1 and 2.6.1; their articulation towards convex update rules mostly originates in [29], to which we refer the reader for their detailed treatment.

PERRON-FROBENIUS THEORY OF ERGODIC MATRICES

Ergodic matrices are at the center of the study of the convergence of convex update rules.

Definition 15 (Ergodic matrices)**.** A stochastic matrix $\boldsymbol{A}$ is said to be ergodic when the geometric sequence $\boldsymbol{A}^k$ is convergent and there exists a positive stochastic vector $\boldsymbol{\pi}$ such that

$$\lim_{k\to\infty} \boldsymbol{A}^k = \mathbf{1}\boldsymbol{\pi}^\intercal \ . \tag{2.5}$$

The vector $\boldsymbol{\pi}$ involved in eq. (2.5) is then unique, and called the Perron vector of the matrix $\boldsymbol{A}$, which we denote by $\boldsymbol{\pi}(\boldsymbol{A})$. ⬲

The spectral theory of non-negative matrices is usually referred to as Perron-Frobenius theory, after two important early investigators; see for example [93] for a detailed reference. Let us recall some of its central points that will be essential to our upcoming argument: a stochastic

matrix $\boldsymbol{A}$ is ergodic if and only if $\boldsymbol{A}^k > 0$ for some power k, which is in particular the case when its associated graph $\mathrm{G}_{\boldsymbol{A}}$ is reflexive and strongly connected. An ergodic matrix has 1 for dominant eigenvalue – that is, 1 is a simple root of its characteristic polynomial $\chi_{\boldsymbol{A}}(X)$, and the rest of its eigenvalues are contained in the open unit disk.

Here, we will principally consider ergodic matrices that arise from update rules: for a given round t, the matrix $\boldsymbol{A}$, defined from eq. (2.1) by letting $A_{ij} = a_{ij}(t)$ if i and j are neighbors and $A_{ij} = 0$ otherwise, is ergodic whenever the affine weights $a_{ij}(t)$ are all positive and the communication graph in round t is reflexive and strongly connected. The effect of the update $\boldsymbol{x}(t-1) \mapsto \boldsymbol{x}(t)$ induced by an ergodic matrix is best understood under a geometric lens, which will be foundational to the complexity proof of the MaxWeight algorithm in Theorem 2.5.1.

WEIGHTED GEOMETRIES

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

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle_{\boldsymbol{\pi}} \coloneqq \sum_i \pi_i u_i v_i \ , \qquad \|\boldsymbol{v}\|_{\boldsymbol{\pi}} \coloneqq \sqrt{\langle \boldsymbol{v}, \boldsymbol{v} \rangle_{\boldsymbol{\pi}}} \ . \tag{2.6}$$

Here, we will consider weighted geometries defined by the Perron vectors of ergodic matrices: the action of an ergodic update matrix $\boldsymbol{A}$ over a vector of estimates $\boldsymbol{x}(t)$ induces a reduction in the variance defined by the geometry weighted by $\boldsymbol{\pi} = \boldsymbol{\pi}(\boldsymbol{A})$:

$$\mathbf{V}_{\boldsymbol{\pi}}(\boldsymbol{v}) \coloneqq \|\boldsymbol{v} - \langle \boldsymbol{v}, \mathbf{1} \rangle_{\boldsymbol{\pi}} \mathbf{1}\|^2_{\boldsymbol{\pi}} = \|\boldsymbol{v}\|^2_{\boldsymbol{\pi}} - \langle \boldsymbol{v}, \mathbf{1} \rangle^2_{\boldsymbol{\pi}} \ . \tag{2.7}$$

The weighted variances have in common with the diameter diam that they are semi-norms over the space $\mathbb{R}^n$ whose zeroes are exactly the consensus vectors – that is, they both represent ways of measuring the "distance to consensus" of the system. As the convergence time is expressed in terms of the diameter, but the geometric approach we take here is based on weighted variances, we will rely on the following result given in [29] to connect the two.

Lemma 2.3.1. Let $\boldsymbol{\pi} \in \mathbb{R}^n$ be a positive stochastic vector. For any vector $\boldsymbol{v} \in \mathbb{R}^n$,

$$2\,\mathbf{V}_{\boldsymbol{\pi}}(\boldsymbol{v}) < (\operatorname{diam} \boldsymbol{v})^2 < \frac{4}{\min_i \pi_i} \mathbf{V}_{\boldsymbol{\pi}}(\boldsymbol{v}) \ . \tag{2.8}$$

❀

Reversibility Given any matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, we easily verify that its adjoint for the $\boldsymbol{\pi}$-weighted inner product $\langle -, - \rangle_{\boldsymbol{\pi}}$ is given by $\operatorname{diag}^{-1} \boldsymbol{\pi} \cdot \boldsymbol{A}^{\intercal} \operatorname{diag} \boldsymbol{\pi}$, where $\operatorname{diag}^{-1} \boldsymbol{\pi}$ and $\operatorname{diag} \boldsymbol{\pi}$ denote the diagonal matrices with respectively π_i^{-1} and π_i as their i-th diagonal entry. We will say that a matrix is REVERSIBLE if it is self-adjoint for *any* weighted inner product. By definition, a reversible matrix is subject to the spectral theorem: it is diagonalizable in an orthonormal (for the right weighted inner product) basis, and its eigenvalues are real numbers. For a stochastic reversible matrix $\boldsymbol{A}$, we have in particular $\operatorname{Sp} \boldsymbol{A} \subset [-1, 1]$.

Let us denote the $\boldsymbol{\pi}$-weighted adjoint of a matrix $\boldsymbol{A}$ by $\boldsymbol{A}^{\dagger_{\boldsymbol{\pi}}}$. Letting $\boldsymbol{D} \coloneqq \operatorname{diag} \boldsymbol{\pi}$, we then have, for any $\boldsymbol{\pi}$-self-adjoint matrix $\boldsymbol{A}$

$$\begin{aligned} \boldsymbol{A}^{\intercal} \boldsymbol{\pi} &= (\boldsymbol{D}\boldsymbol{\pi} \cdot \boldsymbol{A}\boldsymbol{D}^{-1}\boldsymbol{\pi})\boldsymbol{\pi} \\ &= \boldsymbol{D} \cdot \boldsymbol{A}\mathbf{1} \\ &= \boldsymbol{\pi} \ , \end{aligned}$$

from which we deduce that if the matrix $\boldsymbol{A}$ is ergodic, then necessarily we have $\boldsymbol{\pi} = \boldsymbol{\pi}(\boldsymbol{A})$. As a consequence, the natural geometry with which to study an ergodic matrix $\boldsymbol{A}$ is that given by the inner product $\langle -, - \rangle_{\boldsymbol{\pi}(\boldsymbol{A})}$: if we denote its $\boldsymbol{\pi}(\boldsymbol{A})$-adjoint by $\boldsymbol{A}^{\dagger}$, then the matrix $\boldsymbol{A}^{\dagger}\boldsymbol{A}$ is reversible and ergodic, with the same Perron vector as $\boldsymbol{A}$. The min-max variational characterization of the eigenvalues of the matrix $\boldsymbol{A}^{\dagger}\boldsymbol{A}$ known as the Rayleigh-Ritz theorem then gives us:

$$\sup_{\substack{\lambda \in \operatorname{Sp} \boldsymbol{A}^{\dagger}\boldsymbol{A} \\ \lambda \neq 1}} \lambda = \sup_{\substack{\boldsymbol{v} \in \mathbb{R}^n \setminus \{\mathbf{0}\} \\ \langle \boldsymbol{v}, \mathbf{1} \rangle_{\boldsymbol{\pi}(\boldsymbol{A})} = 0}} \frac{\langle \boldsymbol{A}^{\dagger}\boldsymbol{A}\boldsymbol{v}, \boldsymbol{v} \rangle_{\boldsymbol{\pi}(\boldsymbol{A})}}{\|\boldsymbol{v}\|^2_{\boldsymbol{\pi}(\boldsymbol{A})}} \ , \tag{2.9}$$

from which we deduce the following bound over the reduction in variance due to the multiplication by an ergodic matrix.

Lemma 2.3.2. For an ergodic matrix $\boldsymbol{A}$,

$$\forall \boldsymbol{v} \colon \mathbf{V}_{\boldsymbol{\pi}(\boldsymbol{A})}(\boldsymbol{A}\boldsymbol{v}) \leqslant (1 - \gamma_{\boldsymbol{A}^{\dagger}\boldsymbol{A}}) \, \mathbf{V}_{\boldsymbol{\pi}(\boldsymbol{A})}(\boldsymbol{v}) \ . \tag{2.10}$$

If the matrix $\boldsymbol{A}$ is furthermore reversible, then we have in fact

$$\forall \boldsymbol{v} \colon \mathbf{V}_{\boldsymbol{\pi}(\boldsymbol{A})}(\boldsymbol{A}\boldsymbol{v}) \leqslant (1 - \gamma_{\boldsymbol{A}})^2 \, \mathbf{V}_{\boldsymbol{\pi}(\boldsymbol{A})}(\boldsymbol{v}) \ . \tag{2.11}$$

❀

Proof. We let $\boldsymbol{\pi} \coloneqq \boldsymbol{\pi}(\boldsymbol{A})$. Fix a vector $\boldsymbol{v} \neq \mathbf{0}$, and let us show that $\mathbf{V}_{\boldsymbol{\pi}} \boldsymbol{A} \boldsymbol{v} \leqslant (1 - \gamma_{\boldsymbol{A}^\dagger \boldsymbol{A}}) \mathbf{V}_{\boldsymbol{\pi}} \boldsymbol{v}$. For any $\alpha \in \mathbb{R}$, we have $\mathbf{V}_{\boldsymbol{\pi}} \boldsymbol{v} + \alpha \mathbf{1} = \mathbf{V}_{\boldsymbol{\pi}} \boldsymbol{v}$ and $\mathbf{V}_{\boldsymbol{\pi}} \boldsymbol{A}(\boldsymbol{v} + \alpha \mathbf{1}) = \mathbf{V}_{\boldsymbol{\pi}} \boldsymbol{A} \boldsymbol{v} + \alpha \mathbf{1}$. Without loss of generality, we can then assume that $\langle \boldsymbol{v}, \mathbf{1} \rangle_{\boldsymbol{\pi}} = 0$, and it suffices to show that $\|\boldsymbol{A}\boldsymbol{v}\|_{\boldsymbol{\pi}}^2 \leqslant (1 - \gamma_{\boldsymbol{A}^\dagger \boldsymbol{A}}) \|\boldsymbol{v}\|_{\boldsymbol{\pi}}^2$.

We have

$$\begin{aligned} \|\boldsymbol{A}\boldsymbol{v}\|_{\boldsymbol{\pi}}^2 &= \langle \boldsymbol{A}\boldsymbol{v}, \boldsymbol{A}\boldsymbol{v} \rangle_{\boldsymbol{\pi}} \\ &= \langle \boldsymbol{A}^\dagger \boldsymbol{A}\boldsymbol{v}, \boldsymbol{v} \rangle_{\boldsymbol{\pi}} \\ &\leqslant \sup_{\substack{\lambda \in \operatorname{Sp} \boldsymbol{A}^\dagger \boldsymbol{A} \\ \lambda \neq 1}} \lambda \cdot \|\boldsymbol{v}\|_{\boldsymbol{\pi}}^2 \qquad \text{eq. (2.9) and } \langle \boldsymbol{v}, \mathbf{1} \rangle_{\boldsymbol{\pi}} = 0 \\ &= (1 - \gamma_{\boldsymbol{A}^\dagger \boldsymbol{A}}) \|\boldsymbol{v}\|_{\boldsymbol{\pi}}^2 \ , \end{aligned}$$

the latter by definition of the spectral gap, which proves the general case. If the matrix $\boldsymbol{A}$ is moreover reversible, then we have $\boldsymbol{A}^\dagger \boldsymbol{A} = \boldsymbol{A}^2$, and so $1 - \gamma_{\boldsymbol{A}^\dagger \boldsymbol{A}} = (1 - \gamma_{\boldsymbol{A}})^2$, which gives eq. (2.11). □

An additional feature of interest to us is that reversible ergodic matrices admit reasonable bounds over their spectral gap. We borrow the following spectral bound from [29, Corollary 7], which generalizes a previous bound given by Nedić et al. [78, Lemma 9].

Lemma 2.3.3. For a reversible ergodic matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$,

$$\gamma_{\boldsymbol{A}} \geqslant \frac{\alpha(\boldsymbol{A})}{n - 1} \ , \tag{2.12}$$

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

Metropolis and others In the theory of finite Markov chains, a stochastic matrix $\boldsymbol{A}$ has an interpretation as a transition matrix of a Markov chain $X(t)$ for which $\mathbb{P}[X(t+1) = j \mid X(t) = i] = A_{ij}$. If the matrix $\boldsymbol{A}$ is reversible and ergodic, then the Markov chain that it describes is reversible and ergodic as well for the traditional meaning given to those terms in the theory of Markov chain; the Perron vector $\boldsymbol{\pi} = \boldsymbol{\pi}(\boldsymbol{A})$ then gives the unique stationary distribution of the corresponding Markov chain.

In their seminal work [70], Metropolis et al. introduced the method of *Monte Carlo sampling* for the fast approximation of parameters that are hard to compute exactly. Their algorithm, later generalized by Hastings [53], consists at the core of turning a reversible ergodic matrix $\boldsymbol{A}$ with

arbitrary Perron vector $\boldsymbol{\pi}$ into another matrix $\boldsymbol{A}'$ with a *chosen* Perron vector $\boldsymbol{\pi}'$ by:

$$A'_{ij} = \begin{cases} \min(A_{ij}, \frac{\pi'_j}{\pi'_i} A_{ji}) & j \neq i \\ 1 - \sum_{k \neq i} A'_{ik} & j = i \end{cases} , \tag{2.13}$$

in a method now known as the Metropolis-Hastings algorithm.[6]

Here, we shall be interested mostly in the case where $\boldsymbol{\pi}'$ is the constant vector $(\frac{1}{n}, \cdots, \frac{1}{n})^\intercal$, in which case the resulting matrix $\boldsymbol{A}'$ is symmetric.

[6] Or, as perhaps unsurprisingly nobody calls it, the Metropolis-Rosenbluth-Rosenbluth-Teller-Teller-Hastings algorithm.

2.4 THE EQUALNEIGHBOR AND METROPOLIS UPDATE RULES

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:

$$x_i(t) = \frac{1}{d_i(t)} \sum_{j \in \mathrm{In}_i(t)} x_j(t-1) \ . \tag{2.14}$$

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 thus solves consensus over the class $\mathfrak{G}$.

However, the convergence *time* is poor over the class $\mathfrak{G}$, where [84, Proposition 12] gives a *lower* bound of $\mathrm{T}(r; n) \geqslant 2^{\Omega(n)} \log 1/r$. The convergence time is improved when the network displays additional structure: over the sub-class of $\mathfrak{G}$ of *fixed* graphs, Olshevsky and Tsitsiklis [83, Corollary 5.2] use a result from Landau and Odlyzko to show a tight bound in $\mathrm{O}(n^3 \log n/r)$. This bound extends to the sub-class $\mathfrak{G}$ of dynamic graphs for which each vertex has a fixed *degree* [40, Theorem 1.6].

Local update rules such as eq. (2.1) admit an equivalent matrix form. For the EqualNeighbor rule, as an example, eq. (2.14) is equivalent to the global rule $\boldsymbol{x}(t) = \boldsymbol{W}(\mathbb{G}(t))\boldsymbol{x}(t-1)$, with the EqualNeighbor matrix $\boldsymbol{W}(G)$ given for any graph G by

$$[\boldsymbol{W}(G)]_{ij} = \begin{cases} 1/d_i(G) & j \in \mathrm{In}_i(G) \\ 0 & j \notin \mathrm{In}_i(G) \end{cases} \ . \tag{2.15}$$

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

Metropolis

Since the Metropolis-* algorithm is not an "algorithm" in the sense we use in this thesis, and since Metropolis and Hastings are not wanting in fame, we shall call the METROPOLIS-HASTINGS SYMMETRIZATION the transformation $\boldsymbol{A} \mapsto \mathrm{M}(\boldsymbol{A})$ given by

$$[\mathrm{M}(\boldsymbol{A})]_{ij} = \begin{cases} \min(A_{ij}, A_{ji}) & j \neq i \\ 1 - \sum_{k \neq i} \min(A_{ik}, A_{ki}) & j = i \end{cases} . \tag{2.16}$$

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

Xiao et al. [100] propose to approach the average consensus problem with the METROPOLIS update rule:

$$x_i(t) = x_i(t-1) + \sum_{j \in \mathrm{In}\, i(t)} \frac{x_j(t-1) - x_i(t-1)}{\max(d_i(t-1), d_j(t-1))} \ ; \tag{2.17}$$

when viewed at the scale of the system, this rule corresponds to symmetrizing the EQUALNEIGHBOR rule with the Metropolis-Hastings symmetrization round-wise, hence its name. Proposition 2.2.1 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 $\mathrm{O}(n^2 \log n/r)$ 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 $t \equiv r \bmod k$ – 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 $(x_1(t_0-1),\cdots,x_n(t_0-1))$ 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.

2.5 DEGREE TRACKING FOR STABILIZING WEIGHTS

Consider the update rule given by

$$x_i(t) = x_i(t-1) + \frac{1}{q_i}\sum_{j\in\mathcal{N}_i(t)}\left(x_j(t-1)-x_i(t-1)\right)\ , \tag{2.18}$$

which we call the FIXEDWEIGHT rule for the parameters $q_1,\cdots,q_n > 0$. When $d_i(t) \leqslant q_i$, it acts as a sort of sluggish 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 $d_i(t) \leqslant q_i$ holds for all vertices at all times, the FIXEDWEIGHT rule with parameters $q_1,\cdots,q_n$ is shown in [40, Theorem 1.6] to solve consensus with a convergence time in $\mathrm{T}^{\mathfrak{C}}(r;n) = \mathrm{O}(\sum_i q_i \cdot n\log n/r)$. In particular, using $q_1 = q_2 = \cdots = q_n = n$ yields a bound in $\mathrm{O}(n^3\log n/r)$, comparable to the EQUALNEIGHBOR rule over static networks.

Unfortunately, the uniform programming of the agents in our model precludes the distributed implementation of this rule, which would require to individually communicate its parameter q_i to each agent i. Moreover, even if we have the capacity to do so, or if we batch program all agents with the same parameter q, our ability to pick good values for these parameters is limited by what is known, ahead of time, about the structure of the communication graph: parameters that are too small risk breaking the degree condition $d_i(t) \leqslant q_i$ 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 the agent learn by themselves what values of $q_1, \cdots, q_n$ work for the current execution. We call MaxWeight the rule obtained by replacing the fixed parameter q_i with $d_i'(t) := \max\{ d_i(1), \cdots, d_i(t) \}$ at each step of eq. (2.18).

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 $d_1', \cdots, d_n'$. However, the MaxWeight rule can be implemented over the class $\mathfrak{G}$ with no additional assumption, and we can show the following about the resulting MaxWeight algorithm, given in Algorithm 3.

Algorithm 3: The MaxWeight algorithm, code for agent i

Input: $\mu_i \in \mathbb{R}$
Initially:
  $\mathsf{x}_i \leftarrow \mu_i$
  $\mathsf{q}_i \leftarrow 2$
In each round:
  send $m_i = \langle \mathsf{x}_i \rangle$
  receive $m_{j_1}, \ldots, m_{j_d}$ ▷ d neighbors
  $\mathsf{q}_i \leftarrow \max(\mathsf{q}_i, d)$
  $\mathsf{x}_i \leftarrow \mathsf{x}_i + \frac{1}{\mathsf{q}_i} \sum_{k=1}^{d} (\mathsf{x}_{j_k} - \mathsf{x}_i)$
  output x_i

Theorem 2.5.1. 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 $\mathrm{T}(r; n) = \mathrm{O}(n^4 \log n/r)$ for a system of n agents. 🏵

As the update matrices given by the MaxWeight update rule will generally not all share the same Perron vector, we will use the following lemma to connect the various weighted geometries along the execution.

Lemma 2.5.2. Let $\boldsymbol{\pi}$ and $\boldsymbol{\pi}'$ be two positive stochastic vectors of $\mathbb{R}^n$. For any vector $\boldsymbol{v} \in \mathbb{R}^n$,

$$\mathbf{V}_{\boldsymbol{\pi}'}(\boldsymbol{v}) \leqslant \max_i \frac{\pi_i'}{\pi_i} \mathbf{V}_{\boldsymbol{\pi}}(\boldsymbol{v}) \ . \tag{2.19}$$

❁

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

$$\|\boldsymbol{u}\|^2_{\boldsymbol{\pi}'} \leqslant \max_i \frac{\pi'_i}{\pi_i} \|\boldsymbol{u}\|^2_{\boldsymbol{\pi}} \ , \tag{2.20}$$

and in particular for the vector $\boldsymbol{u} \coloneqq \boldsymbol{v} - \langle \boldsymbol{v}, \mathbf{1} \rangle_{\boldsymbol{\pi}} \mathbf{1}$. By the definition of the variance, we then have $\mathbf{V}_{\boldsymbol{\pi}}(\boldsymbol{v}) = \|\boldsymbol{u}\|^2_{\boldsymbol{\pi}}$ and $\mathbf{V}_{\boldsymbol{\pi}'}(\boldsymbol{v}) = \mathbf{V}_{\boldsymbol{\pi}'}(\boldsymbol{u}) \leqslant \|\boldsymbol{u}\|^2_{\boldsymbol{\pi}'}$, since adding a consensus vector does not change the variance, from which the claim follows. □

Proof of Theorem 2.5.1. We fix a dynamic graph $\mathbb{G} \in \mathfrak{G}$ of order n, and we let

$$\begin{gathered} d'_i(t) \coloneqq \max \{\, d_i(\tau) \mid 0 \leqslant \tau \leqslant t \,\} \ , \qquad d'_i \coloneqq \max\{\, d_i(t) \mid t \in \mathbb{N} \,\} \ , \\ D'(t) \coloneqq \sum_{i=1}^{n} d'_i(t) \ , \qquad D' \coloneqq \sum_{i=1}^{n} d'_i \ , \\ \mathcal{T}_\ell \coloneqq \{\, t \in \mathbb{N}_{>0} \mid \exists i \in [n] : d'_i(t) \neq d'_i(t-1) \,\} \ , \end{gathered} \tag{2.21}$$

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

We then pick arbitrary values $\mu_1, \cdots, \mu_n \in \mathbb{R}$ and consider the sequence of estimates $\mathbb{x}$ produced by the MaxWeight update rule over these input values and the dynamic graph $\mathbb{G}$:

$$\begin{aligned} x_i(0) &= \mu_i \\ x_i(t) &= x_i(t-1) + \frac{1}{d'_i(t)} \sum_{j \in \mathcal{N}_i(t)} (x_j(t-1) - x_i(t-1)) \ . \end{aligned} \tag{2.22}$$

Since for any round $t \in \mathbb{N}_{>0}$ we have $2 \leqslant d'_i(t) \leqslant n$, the sequence $\mathbb{x}$ satisfies the assumptions of Proposition 2.2.1 with uniform convexity parameter $\alpha = 1/n$, and so it achieves asymptotic consensus within the convex hull of the set $\{\, \mu_1, \cdots, \mu_n \,\}$. 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 $t_s \leqslant f(n)$ for some function f – then we could simply reuse the result from [40] about the FixedWeight update and deduce $\mathrm{T}(r; n) = \mathrm{O}(f(n) + n^3 \log n/r)$. 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 $t \in [1, \cdots, t_s]$.

We fix a disagreement threshold $r > 0$, and we define the set $\mathcal{T}_r \coloneqq \{\, t \in \mathbb{N} \mid \operatorname{diam} \boldsymbol{x}(t) \leqslant r \cdot \operatorname{diam} \boldsymbol{\mu} \,\}$. The convexity of the sequence $\mathbb{x}$ and the fact that it achieves asymptotic consensus imply that the set $\mathcal{T}_r$ is an unbounded interval of the integers, and we let $t_r \coloneqq \inf \mathcal{T}_r$ denote the earliest round at which the estimates approximately agree with a relative error of r.

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

$$[\boldsymbol{A}(t)]_{ij} = \begin{cases} \frac{1}{d_i'(t)} & i \neq j \in \mathcal{N}_i(t) \\ 1 - \frac{d_i(t)-1}{d_i'(t)} & j = i \\ 0 & j \notin \mathcal{N}_i(t) \ . \end{cases} \tag{2.23}$$

Its associated graph is $G_{\boldsymbol{A}(t)} = \mathbb{G}(t)$, and by our graph assumptions, this matrix is ergodic, with its Perron vector $\boldsymbol{\pi}(t) \coloneqq \boldsymbol{\pi}(\boldsymbol{A}(t))$ given by $\boldsymbol{\pi}(t) = (d_1'(t)/D'(t), \cdots, d_n'(t)/D'(t))^\intercal$. We can verify that the matrix $\boldsymbol{A}(t)$ is self-adjoint for the inner product $\langle -, - \rangle_{\boldsymbol{\pi}(t)}$. As $A_{ij}(t) \geqslant \frac{1}{d_i'(t)}$ holds for all positive entries of the matrix $\boldsymbol{A}(t)$, we have by Lemma 2.3.2 $\gamma \coloneqq \inf_{t \in \mathbb{N}_{>0}} \gamma_{\boldsymbol{A}(t)} \geqslant 1/(n-1)D'$.

Let us then define the potential $\mathcal{L}(t) \coloneqq \mathbf{V}_{\boldsymbol{\pi}(t)}(\boldsymbol{x}(t))$ for positive t. For a round $t \neq 1$ outside of $\mathcal{T}_\ell$, the matrices $\boldsymbol{A}(t)$ and $\boldsymbol{A}(t-1)$ share their Perron vector, and with Lemma 2.6.3 we have

$$\mathcal{L}(t) \leqslant (1-\gamma)^2 \mathcal{L}(t-1) \ . \tag{2.24}$$

Equation (2.24) provides some control over the dispersion of the estimates: for any temporal interval $[t, t']$ which does not intersect $\mathcal{T}_\ell$, applying eq. (2.24) round-wise yields $\mathcal{L}(t') \leqslant (1-\gamma)^{2(t'-t)} \mathcal{L}(t)$, which implies $\mathcal{L}(t) \to_t 0$ since $\mathcal{T}_\ell$ is finite. However, we cannot control the potential $\mathcal{L}(t)$ across the entire execution with the help of eq. (2.24) alone. To piece the variations of the potential over $\mathcal{T}_\ell$, we use Lemma 2.5.2:

$$\forall t \geqslant 2 : \mathcal{L}(t) \leqslant \max_i \frac{\pi_i(t)}{\pi_i(t-1)} (1-\gamma)^2 \mathcal{L}(t-1) \ . \tag{2.25}$$

The rounds for which $\boldsymbol{\pi}(t) \neq \boldsymbol{\pi}(t-1)$ are exactly those of $\mathcal{T}_\ell$, so eq. (2.25) is equivalent to eq. (2.24) when $t \notin \mathcal{T}_\ell$. Applying eq. (2.25) over $\mathcal{T}_\ell$ will induce a delay factor $\beta_\ell \coloneqq \prod_{t \in \mathcal{T}_\ell} \max_i \frac{\pi_i(t)}{\pi_i(t-1)}$, which we need to control. Given $t \in \mathcal{T}_\ell$ we have

$$\max_i \frac{\pi_i(t)}{\pi_i(t-1)} = \frac{d'(t-1)}{d'(t)} \max_i \frac{d_i'(t)}{d_i'(t-1)} \leqslant \frac{d'(t-1)}{d'(t)} \prod_i \frac{d_i'(t)}{d_i'(t-1)} \ ,$$

and with $d_i'(0) = 2$:

$$\beta_\ell \leqslant \frac{2n}{d'} \prod_i \frac{d_i'}{2} \ . \tag{2.26}$$

Finally, Lemmas 2.3.1 and 2.3.2 give us $\mathcal{L}(1) \leqslant \frac{1}{2}\left((1-\gamma)\ \operatorname{diam} \boldsymbol{\mu}\right)^2$. Together with eq. (2.25), we have for any $t \geqslant 2$:

$$\begin{aligned}
\mathcal{L}(t) &\leqslant \tfrac{1}{2} \textstyle\prod_{\tau \leqslant t} \max_i \frac{\pi_i(\tau)}{\pi_i(\tau-1)} \cdot \left((1-\gamma)^t\ \operatorname{diam} \boldsymbol{\mu}\right)^2 && \text{by definition of } \beta_\ell \\
&\leqslant \tfrac{\beta_\ell}{2} \left((1-\gamma)^t\ \operatorname{diam} \boldsymbol{\mu}\right)^2 && \text{Lemma 2.6.2} \\
\operatorname{diam} \boldsymbol{x}(t) &\leqslant \sqrt{\tfrac{2\beta_\ell}{\min_i \pi_i(t)}} (1-\gamma)^t \cdot \operatorname{diam} \boldsymbol{\mu} && \text{eq. (2.26), } \pi_i(t) \geqslant 2/n \\
&\leqslant \sqrt{n \textstyle\prod_i \frac{d_i'}{2}}\ (1-\gamma)^t\ \operatorname{diam} \boldsymbol{\mu} \ .
\end{aligned}$$

We now let $\beta \coloneqq n \prod_i \frac{d_i'}{2}$. By definition of t_r, we have

$$\sqrt{\beta}\,(1-\gamma)^{t_r} \leqslant r \ ;$$

equivalently,

$$t_r \geqslant \frac{\log(1-\gamma)}{\log(r/\sqrt{\beta})} \ ,$$

and since $\log(1-x) \leqslant -x$ for every $x \in (0,1)$,

$$t_r \leqslant \gamma^{-1} \log(\sqrt{\beta}/r) \ . \tag{2.27}$$

Inserting in this expression our bound over γ,

$$t_r \leqslant \frac{(n-1)D'}{2} \left(\sum_i \log d_i' + \log n - 2\log r - n \log 2 \right) \ , \tag{2.28}$$

and with $d_i' \leqslant n$ we have both $D' \leqslant n^2$ and $\sum_i \log d_i' \leqslant n \log n$, which yields indeed $\mathrm{T}(r;n) = \mathrm{O}(n^4 \log n / r)$. □

2.6 AN AFFINE ALGORITHM FOR AVERAGE CONSENSUS

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 in each round to the EQUALNEIGHBOR update matrix. Any consensus update rule can be given the same treatment; in particular, the symmetrized version of the FIXEDWEIGHT rule – which gives the weight $a_{ij}(t) = \frac{1}{\max(q_i,q_j)}$ to proper round t neighbors i and 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 rule obtained by symmetrizing the MAXWEIGHT rule is no easier to implement than the METROPOLIS rule itself, as both rules use the same information.

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

$$x_i(t) = x_i(t-1) + \sum_{j \in \mathcal{N}_i(t)} \frac{x_j(t-1) - x_i(t-1)}{\max(d'_i(t-1), d'_j(t-1))} \ , \tag{2.29}$$

whose implementation by a local algorithm is given in Algorithm 4.

We immediately observe the following about the MAXMETROPOLIS rule. First, it 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 $d'_1, \cdots, d'_n$. From there, Proposition 2.2.1 shows that Algorithm 4 is an average consensus algorithm for the class $\mathfrak{G}$.

However, we now face the additional difficulty that the right-hand side of eq. (2.29) does not necessarily define a *convex* combination: the self-weight $a_{ii}(t)$ may be negative at times 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 $\{ x_1(t-1), \cdots, x_n(t-1) \}$, which delays the eventual convergence. We show in Theorem 2.6.1, that this is only by at most a linear factor, when compared to a bound of $O(n^3 \log n/r)$ available for the symmetrized FIXEDWEIGHT rule with parameters $d'_1, \cdots, d'_n$.

We note that the broken convexity does not fully explain the discrepancy between the convergence times of the Metropolis and MaxMetropolis rules, the latter admitting a bound in $O(n^2 \log n/r)$. 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, whereas the MaxMetropolis update only uses an upper bound over the degree, and this pessimistic approach results in a slower convergence.

Algorithm 4: The MaxMetropolis algorithm, code for agent i

Input: $\mu_i \in \mathbb{R}$
Initially:
  $\mathsf{x}_i \leftarrow \mu_i$
  $\mathsf{q}_i \leftarrow 2$
In each round:
  send $m_i = \langle \mathsf{x}_i, \mathsf{q}_i \rangle$
  receive $m_{j_1}, \dots, m_{j_d}$ ▷ d neighbors
  $\mathsf{x}_i \leftarrow \mathsf{x}_i + \sum_{k=1}^{d} \frac{\mathsf{x}_{j_k} - \mathsf{x}_i}{\max(\mathsf{q}_i, \mathsf{q}_{j_k})}$
  $\mathsf{q}_i \leftarrow \max(\mathsf{q}_i, d)$
  output x_i

Theorem 2.6.1. 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 $T(r; n) = O(n^4 \log n/r)$ for a system of n agents. ֍

In contrast with Theorem 2.5.1, the proof of Theorem 2.6.1 will only involve the usual geometry over the space $\mathbb{R}^n$. As a consequence, we briefly restate Lemmas 2.6.2 to 2.6.4, specialized for the usual Euclidian norm $\| - \| = \sqrt{\langle -, - \rangle}$.

Lemma 2.6.2. For a non-null vector v for which $\langle v \rangle = 0$, we have

$$\sqrt{2/n}\,\|v\| < \operatorname{diam} v < 2\,\|v\| \quad . \tag{2.30}$$

❀

Lemma 2.6.3. For an ergodic matrix $\boldsymbol{A}$ that is symmetric, and any vector $\boldsymbol{v}$ for which $\langle \boldsymbol{v} \rangle = 0$,

$$\| \boldsymbol{A}\boldsymbol{v} \| \leqslant (1 - \gamma_{\boldsymbol{A}}) \, \| \boldsymbol{v} \| \quad . \tag{2.31}$$

❁

Lemma 2.6.4. For an ergodic matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ that is symmetric,

$$\gamma_{\boldsymbol{A}} \geqslant \frac{A^-}{n(n-1)} \quad , \tag{2.32}$$

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

Proof of Theorem 2.6.1. We fix a dynamic graph $\mathbb{G} \in \mathfrak{G}$ of order n, and define $d'_i(t)$, d'_i, $\mathcal{T}_\ell$, and t_s as in eq. (2.21), and recall from the proof of Theorem 2.5.1 that $|\mathcal{T}_\ell| \leqslant \sum_{i \in [n]} d'_i$.

Let us then fix an execution of Algorithm 4 over the dynamic communication graph $\mathbb{G}$, using input values $\mu_1, \cdots, \mu_n \in \mathbb{R}$. Without losing generality, we can assume $\overline{\mu} = 0$, 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 $\boldsymbol{x}(t) = \boldsymbol{A}(t)\boldsymbol{x}(t-1)$, where the matrix $\boldsymbol{A}(t)$ is the round t MaxMetropolis matrix for the dynamic graph $\mathbb{G}$:

$$[\boldsymbol{A}(t)]_{ij} = \begin{cases} \frac{1}{\max(d'_i(t-1), d'_j(t-1))} & i \neq j \in \mathcal{N}_i(t) \\ 1 - \sum_{k=1}^{n} \frac{1}{\max(d'_i(t-1), d'_k(t-1))} & j = i \\ 0 & j \notin \mathcal{N}_i(t) \quad . \end{cases} \tag{2.33}$$

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

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

Let us then define a sequence $\mathbb{z}$ and a dynamic graph $\mathbb{G}'$ by $\boldsymbol{z}(k) \coloneqq \boldsymbol{x}(k + t_s + 1)$ and $\mathbb{G}'(k) \coloneqq \mathbb{G}(k + t_s + 1)$, for each $k \in \mathbb{N}$. The sequence $\mathbb{z}$ satisfies the assumptions of Proposition 2.2.1 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 $\mathbb{x}$ does as well. Since

the consensus value is necessarily the average $\overline{\mu}$, we see that Algorithm 4 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 $d_i'(t) \neq d_i'(t-1)$. 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 $r > 0$, and define the set $\mathcal{T}_r \coloneqq \{ t \in \mathbb{N} \mid \operatorname{diam} \boldsymbol{x}(t) \leqslant r \cdot \operatorname{diam} \boldsymbol{\mu} \}$. Since the system achieve asymptotic consensus, this set contains an unbounded interval, and we let $t_r \coloneqq \inf \{ t \in \mathcal{T}_r \mid \forall \tau \geqslant t \colon \tau \in \mathcal{T}_r \}$. Remark that since the estimates can leave their convex hull, it is possible for t_r to be greater than $\inf \mathcal{T}_r$.

We then follow the variations of the quantity $N(t) \coloneqq \|\boldsymbol{x}(t)\|$ 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 $\boldsymbol{A}(t)$ has positive diagonal entries, and by Lemma 2.6.4 we have $\gamma \coloneqq \inf_{t \notin \mathcal{T}_\ell} \gamma_{\boldsymbol{A}(t)} \geqslant 1/n^3$. Using Lemma 2.6.3, we have

$$\forall t \notin \mathcal{T}_\ell \colon N(t) \leqslant (1-\gamma) N(t-1) \ . \tag{2.34}$$

For rounds $t \in \mathcal{T}_\ell$, on the other hand, the update matrix $\boldsymbol{A}(t)$ may have negative entries, and we cannot use Lemma 2.6.3 to control $N(t)$. However, since the matrix $\boldsymbol{A}(t)$ is symmetric, it is diagonalizable, and for any vector $\boldsymbol{v}$, we have $\|\boldsymbol{A}(t)\boldsymbol{v}\| \leqslant \rho_{\boldsymbol{A}(t)} \|\boldsymbol{v}\|$, which gives us

$$\forall t \in \mathcal{T}_\ell \colon N(t) \leqslant \rho_{\boldsymbol{A}(t)} N(t-1) \ . \tag{2.35}$$

We note that eq. (2.35) holds in fact for all $t \in \mathbb{N}$, but is strictly worse than eq. (2.34) outside of $\mathcal{T}_\ell$.

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

$$\forall t \in \mathcal{T}_\ell \colon N(t) \leqslant \nu_t^2 N(t-1) \ . \tag{2.36}$$

The delay accrued during $\mathcal{T}_\ell$ will then depend on some factor $\beta_\ell \coloneqq \prod_{t \in \mathcal{T}_\ell} \nu_t^2$. To bound ν_t for $t \in \mathcal{T}_\ell$, we observe that $\sum_{j \neq i} A_{ij}(t) \leqslant \frac{d_i(t)-1}{d_i'(t-1)} \leqslant$

$\frac{d'_i(t)}{d'_i(t-1)}$ for any $i \in [n]$, which yields $\nu_{i,t} \leqslant \frac{d'_i(t)}{d'_i(t-1)}$ since $d'_i(t)$ is weakly increasing. Given that $\nu_{i,t} \geqslant 1$, we have $\beta_\ell \leqslant \prod_{t\in\mathcal{T}_\ell} \prod_i \nu_{i,t}^2$, and since $d'_i(t) = d'_i(t-1)$ when $t \notin \mathcal{T}_\ell$, we finally have $\beta_\ell \leqslant \left(\prod_i \frac{d'_i}{2}\right)^2$

Taking eqs. (2.34) and (2.36) together, we have

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

Using Lemma 2.6.2, this gives us $\operatorname{diam} \boldsymbol{x}(t) \leqslant 2\sqrt{n}\beta_\ell\,(1-\gamma)^{t-|\mathcal{T}_\ell|} \operatorname{diam} \boldsymbol{\mu}$. By definition of t_r, we have $2\sqrt{n}\beta_\ell\,(1-\gamma)^{t_r-|\mathcal{T}_\ell|} \leqslant r$, from which we deduce that $t_r \leqslant \gamma^{-1}\log(2\sqrt{n}\beta_\ell/r) + |\mathcal{T}_\ell|$. Using our upper bounds for $|\mathcal{T}_\ell|$, γ, and β_ℓ,

$$t_r \leqslant n^3\left(2\sum_i \log d'_i - \log r - (2n-1)\log 2\right) + \sum_i d'_i - 2n \ , \qquad (2.37)$$

and with $d'_i \leqslant n$ the convergence time of the MaxMetropolis algorithm over the class $\mathfrak{G}$ is in $\mathrm{T}(r;n) = \mathrm{O}(n^4 \log n/r)$. □

RANDOMIZATION AND QUANTIZATION FOR DIRECTED DYNAMIC NETWORKS

The results of this chapter were published as an extended abstract in [Bernadette Charron-Bost and Patrick Lambein-Monette. "Randomization and Quantization for Average Consensus". In: 2018 IEEE Conference on Decision and Control (CDC). *IEEE, Dec. 2018, pp. 3716–3721. DOI:* 10.1109/CDC.2018.8619817*] and appeared in pre-publication form in [Bernadette Charron-Bost and Patrick Lambein-Monette.* Randomization and Quantization for Average Consensus. *Apr. 29, 2018. arXiv:* 1804.10919 [cs.MA]*].*

3.1 INTRODUCTION

In the previous chapter, we discussed an asymptotic average consensus algorithm for *bidirectional* dynamic networks. Here, we look at *directed* networks, where functions that depend on the multiplicities the input cannot be computed in general.

We turn to randomized algorithms to compute the average in this adversarial setting. That is, we allow the agents to make their moves depend on the outcomes of private *random oracles*, which capture an external source of randomness available only to the agent. In this sense, although we have made agents more powerful, the model remains local, in the sense that agents only learn new information through direct interactions.

The randomized algorithms that we discuss here are Monte Carlo: they produce a desirable behavior – here, appoximating the average $\overline{\mu}$, – but only in some executions, under the condition that the random oracles did not produce pathological outcomes. Specifically, in any given execution these algorithms behave incorrectly with probability at most $p \in]0, 1/2[$, where the parameter p can be adjusted in the code of the algorithm.

This pivot – introducing randomness and accepting a small probability of errors – vastly increases the set of computable functions: under the condition that agents initially know a bound over n, they can assign to themselves identifiers that are unique with high probability. Anything that can be computed by agents with unique identities can thus be computed by anonymous agents in a Monte Carlo manner.

We present three randomized algorithms, which are grounded in an idea due to Mosk-Aoyama and Shah and developed in a similar context by Kuhn, Lynch, and Oshman, and which compute an approximation of the average in the shortest possible time while satisfying increasingly strong constraints: the first algorithm $\mathcal{R}$ is stabilizing, the second algorithm $\overline{\mathcal{R}}$ additionally only manipulates values with an efficient space representation – with high probability. The last algorithm $\overline{\mathcal{D}}$ not only stabilizes, but in fact *terminates* – results in simultaneous decisions – even when agents start asynchronously.

In Table 3.1, we compare our algorithms with other methods aimed at computing the average in directed dynamic networks, of which we now give a brief explanation. When a centralized authority assigns unique identifiers to the agents, computing the exact average can be reduced to the problem of *all-to-all token dissemination* – that of getting all agents to learn the multi-set of input values. This problem is studied in [63], in the case of dynamic graphs that are strongly connected in each round; when no other assumption can be made over the network, we single out two methods. First, a full information protocol, based on Algorithm 1 and denoted *Flooding* in Table 3.1. Second, a protocol by which agents take turns to disseminate their values, given in [63, Section 5] and denoted *Sequential* in Table 3.1.

In the absence of identifiers, the PushSum algorithm achieves asymptotic consensus on directed dynamic networks that are strongly connected in each round. It is not a local algorithm, as each agent needs to be supplied with its exact out-degree at the onset of each round; under the same assumptions, the issues of the Metropolis rule at the core of Chapter 2 vanish entirely. The proof of convergence of the PushSum algorithm, given in [77, Theorem 12], gives an implicit bound over the convergence time of $n^{\mathrm{O}(n)}$, which we reproduce in Table 3.1; we know of no better bound.

In contrast with the previous chapter, we will measure the convergence of our algorithms either in the time it takes for their output to *stabilize*, or, for the $\overline{\mathcal{D}}$ algorithm, for the time before the agents *decide*.

Close to our algorithms, but not directly comparable in terms of complexity due to the use of a different communication model, we note the randomized average consensus algorithm proposed by Lucchese and Varagnolo [68].

If the agents start with a bound $N \geqslant n$ ahead of the executions, they can generate unique identifiers over $\mathrm{O}(\log N)$ bits in a Monte Carlo

Algorithm	Time	Message size	Restrictions
Flooding	$O(n)$	$O(n \log n)$	unique identifiers
Sequential	$O(n^2)$	$O(\log n)$	unique identifiers
PushSum	$n^{O(n)}$	∞^*	$d_u^+(t)$ provided to each agent ahead of each round asymptotic consensus
$\overline{\mathcal{R}}$	$O(n)$	$O(\log \log n)$	approximate consensus Monte Carlo
$\overline{\mathcal{D}}$	$O(n)$	$O(\log N + \log \log n)$	$N \geqslant n$ externally provided approximate consensus Monte Carlo

* The PushSum algorithm is generally considered over real values.

Table 3.1: Average consensus algorithms for directed networks

manner. With this approach, the Flooding and Sequential methods admit Monte Carlo variants, with identical temporal complexities, and using messages in $O(n \log N)$ and $O(\log N)$ respectively.

3.2 PRELIMINARIES

A PROBABILITY THEORY REFRESHER

We briefly recall some classic notions from probability theory. Let U be the universal set defining the outcomes of a random experiment. Given a Borel space (E, Σ)[1], any measurable function $f : U \to E$ is called a RANDOM VARIABLE valued in E, which is again interpreted as the space of possible outcomes of a random experiment; the distribution of the random variable f is given by $\mathbb{P}[f \in A] := \mathbb{P}[f^{-1}(A)]$. We say that random variables $f_1, \ldots, f_k : U \to E$ are INDEPENDENT when

$$\forall I \subseteq [k], I \neq \varnothing, \forall (A_i)_{i \in I} \in \Sigma^I : \\ \mathbb{P}[\bigcap_{i \in I} \{ f_i \in A_i \}] = \prod_{i \in I} \mathbb{P}[f_i \in A_i] \ . \tag{3.1}$$

[1] In other words, Σ is the Borel σ-algebra for a given topology over E. In practice, we essentially consider the usual topology over $E = \mathbb{R}$.

It is customary to use end-of-alphabet capital roman letters $Z, Y, X, \ldots$ to denote random variables; relegating U and Σ to the background, we simply say that Z is a random variable over E.

Here, the central case will be $E = \mathbb{R}$; the distribution of a real-valued random variable Z is fully determined by its CUMULATIVE DISTRIBUTION FUNCTION $F_Z : z \in [0,1] \mapsto F_Z(z) \coloneqq \mathbb{P}[Z \leqslant z]$, as then we have $\mathbb{P}[Z \in]a,b]] = F_Z(b) - F_Z(a)$.

As an example, given $\lambda > 0$, the EXPONENTIAL DISTRIBUTION with rate λ is characterized by the cumulative distribution function

$$F(z;\lambda) \coloneqq \begin{cases} 1 - \mathrm{e}^{-\lambda z} & z \geqslant 0 \\ 0 & z < 0 \end{cases}; \tag{3.2}$$

we use the shorthand $Z \sim \mathrm{Exp}(\lambda)$ to denote that the random variable Z is exponentially distributed with rate λ – that is, $F_Z = F(-;\lambda)$.

FAMOUS PLACES TO LOSE MONEY

In this chapter we will consider classes of communication topologies where, in general, the average cannot be computed. Recall for example that for the class of all fixed strongly connected graphs, functions that depend on the multiplicity of the input do not admit an algorithmic implementation. Sidestepping this issue requires bringing in additional tools, and as is often the case we resort to randomized algorithms. Here, we extend our computational model to integrate randomized aspects into our discussion.

Recall that we characterized a local algorithm by the quadruplet $(\iota, \tau, \chi, \zeta)$ defining how an agent is to initialize, transition, and generate messages and outputs. In the case of a *deterministic* algorithm, ξ, τ, and μ are all *functions*, but in general we allow them to merely be relations – generation of outputs is always assumed to be governed by a function ζ.

When considering instead *randomized* algorithms, we will use the notion of a random oracle and specify the type of an algorithm's constitutive elements. For example, instead of $\xi(\mu_u)$ being an element $\sigma \in \Sigma$, $\xi(\mu_u)$ is now a *random variable* with image Σ; alternatively, we can suppose that $\xi : \Xi \times S \to \Sigma$ is a fonction of two variables, assigning an initial state in Σ to each pair of an input $\mu \in \Xi$ and an auxiliary input $s \in S$.

Under this view, agent u obtains the auxiliary input s by querying a private RANDOM ORACLE at initialization; the random oracle is merely a black box abstraction of the true source of randomness in the execution. Here, the random oracle is merely viewed as a random variable whose

image is the set of auxiliary inputs S; the fact that it is *private* means that the random oracles of all agents are independent random variables. Note in particular that under this view, from the point of view of agent u the initialization is a *deterministic* process, some inputs of which are random. We may similarly define randomized variants of the functions τ and χ, taking random auxiliary inputs in addition to their regular inputs.

Recall that in the case of a deterministic algorithm, an execution is entirely defined by the pair $(\mathbb{G}, \boldsymbol{\mu})$ of the dynamic communication graph of the execution and the vector of input values. With the addition of random oracles, such a pair defines a *random variable* $\mathcal{E}(\mathbb{G}, \boldsymbol{\mu})$ over the set of executions.

Here, we will focus on the case where the communication graph results from a process that is independent of either the input values, the start signals, or the outcomes of the random oracles – a case commonly referred to as an OBLIVIOUS ADVERSARY, alternatively described as the dynamic communication being fixed, once and for all, ahead of the execution, before the input or the outcomes of the random oracles are known. For a predicate P over possible executions, we will say that an algorithm satisfies P with probability p for an oblivious adversary if over a class $\mathfrak{C}$ and inputs in Ξ

$$\forall \mathbb{G} \in \mathfrak{C}, \forall \boldsymbol{\mu} \in \Xi^{\mathrm{V}(\mathbb{G})} : \ \mathbb{P}[P(E \sim \mathcal{E}(\mathbb{G}, \boldsymbol{\mu}))] \geqslant p \ . \tag{3.3}$$

AMONG the usual relaxations of algorithmic problems for randomized settings, we will use the following two notions. If P is a *correctness* predicate – for example, $P(E)$ = "E achieves asymptotic consensus" – then an algortihm is MONTE CARLO if it satisfies P with probability $p > 1/2$. Typically, there are executions of the algorithm which fail to produce the expected result, but they are a minority. If Q is a *complexity* predicate – for example, $Q(E)$ = "messages exchanged over E use at most k bits of memory" – then an algorithm is LAS VEGAS[2] if it satisfies Q with probability $q > 1/2$. Algorithms with both Monte Carlo and Las Vegas aspects are sometimes called "Atlantic City".

[2] Traditionally, Las Vegas algorithms offer probabilistic guarantees over the *temporal* complexity of the computation; we make the natural extension of calling Las Vegas algorithms with probabilistic guarantees over the use of any resource.

RANDOM IDENTIFIERS

To illustrate the power of random oracles, consider the fact that independent random variables $Z_1, \ldots, Z_n$, each uniformly distributed over the set $\{ 0, \ldots, 2^k - 1 \}$, are pairwise distinct with probability at least p when $k \geqslant \lfloor 2 \log n - \log p \rfloor$. As a consequence, if all agents agree on a bound

$N \geqslant n^3$, they can simulate a centralized authority issuing unique identifiers, by each picking a value in the set $\{ 0, \ldots, 2^{\lfloor 2 \log N - \log p \rfloor} \}$, uniformly at random, to be used as an identifier. With probability at least $1 - p$, these identifiers are pairwise distinct, and any deterministic algorithm for identified networks can be turned into a Monte Carlo variant in this manner.

[3] In fact, nothing requires agents to use the *same* bound N; it suffices that each agent u use *a* bound.

We note that, although this method dispenses with the centralized authority, it still requires *some* external information in the form of a bound over the number of agents. Moreover, while $\log n$ bits suffice to represent each identifier when they are assigned within the set $\{ 0, \ldots, n - 1 \}$, the footprint of a random identifier generated as above depends on the target risk p of a Monte Carlo failure, and on the quality of the bound N. In particular, a poor bound leads to a much less efficient representation than is possible for a truly centralized authority.

The method of random identifiers is nonetheless useful as a baseline to compare other Monte Carlo algorithms: as the footprint of all identifiers is in $\mathrm{O}(n(\log N - \log p))$ bits. As we will see, our algorithms $\overline{\mathcal{R}}$ and $\overline{\mathcal{D}}$ fare substantially better.

3.3 RANDOMIZED ALGORITHM

We begin with a randomized Monte Carlo algorithm which approximately computes the average under the assumption that the agents in the system may store and transmit real numbers. Specifically, all of its executions eventually compute a value if the network is eventually connected, and this value is close to the average with high probability; the probability of failure p and the tolerated margin of error r are tuneable parameters of the algorithm.

The average is obtained by computing, in parallel, approximations of the order of the network, n, and the sum of the input values, $s \coloneqq \sum_u \mu_u$, and use the ratio of these approximations as an approximation of the average $\boldsymbol{\mu}$. By essence, these quantities depend on multiplicities, and as such they cannot be computed by deterministic distributed algorithms over static directed graphs in general, let alone over dynamic ones; in fact, we cannot even reliably approximate such values.

Randomized algorithms do not face this restriction, which is why our approach succeeds. In particular, we follow the approach proposed by Mosk-Aoyama and Shah in [73] for (asymptotically) computing *separable functions*, of which n and s are examples, based on the following concentration inequality.

Lemma 3.3.1. Let $Z_1, \ldots, Z_k$ be independent identically distributed exponential random variables with same rate $\lambda > 0$. For any error $\alpha \in]0, 1/2[$, we have

$$\mathbb{P}\left[\left|\frac{1}{k}(Z_1 + \cdots + Z_k) - \frac{1}{\lambda}\right| \geqslant \frac{\alpha}{\lambda}\right] \leqslant 2\exp\left(-\frac{k\alpha^2}{3}\right) \quad . \tag{3.4}$$

❁

Lemma 3.3.1 results from applying the Cramér-Chernoff method to exponential random variables, which is developed for example in [20, Sections 2.2 and 2.4]. To apply this result towards computing n and s, observe that sampling, for example, the exponential distribution $\mathrm{Exp}(s)$ several times over allows for computing, with high probability, a good approximation of the inverse $1/s$. The same holds for n.

This reduces the problem of approximating n and s, or rather their multiplicative inverses $1/n$ and $1/s$, to that of sampling the corresponding exponential distributions in a distributed manner. Following again Mosk-Aoyama and Shah, this is easily achieved due to the fact that exponential distributions transform minima into sums.

Lemma 3.3.2. Given independent exponential random variables $Z_1 \sim \mathrm{Exp}(\lambda_1), \ldots, Z_k \sim \mathrm{Exp}(\lambda_k)$, the random variable $Z \coloneqq \min(Z_1, \ldots, Z_k)$ is itself exponential, and distributed as $Z \sim \mathrm{Exp}(\lambda_1 + \cdots + \lambda_k)$. ❁

Proof. A distribution is characterized by its cumulative distribution function; for the exponential distribution $\mathrm{Exp}(\lambda)$, it is given by

$$\mathbb{P}[(X \sim \mathrm{Exp}(\lambda)) < z] = F(z;\lambda) \coloneqq \begin{cases} 1 - \mathrm{e}^{-\lambda z} & z \geqslant 0 \\ 0 & z < 0 \end{cases} . \tag{3.5}$$

Letting $\lambda \coloneqq \lambda_1 + \cdots + \lambda_k$, we need only show that $\mathbb{P}[Z < z] = F(z;\lambda)$ to establish our claim.

It is immediate for negative values of z, so we suppose that $z \geqslant 0$, and we have

$$\begin{aligned} \mathbb{P}[Z \geqslant z] &= \mathbb{P}[\min(Z_1, \ldots, Z_k) > z] \\ &= \mathbb{P}[\textstyle\bigcap_{i=1}^k Z_i > z] && \big) \; Z_1, \ldots, Z_k \text{ independent} \\ &= \textstyle\prod_{i=1}^k \mathbb{P}[Z_i > z] && \big) \; \text{by definition of } F(z;-) \\ &= \textstyle\prod_{i=1}^k \exp(-\lambda_i z) \\ &= \exp(-\lambda z) \quad , \end{aligned}$$

and as a consequence $\mathbb{P}[Z < z] = 1 - \mathrm{e}^{-\lambda z} = F(z;\lambda)$. □

The plan to compute the average with Mosk-Aoyama and Shah's sheme should now be clear: with Lemma 3.3.2, we can implement the sampling of exponential distributions of rates n and s in a distributed manner by agents equipped with random oracles. The resulting values, call them $\tilde{n}$ and $\tilde{s}$, do not directly give an approximation of the average in their ratios $\tilde{n}/\tilde{s}$, as $\mathbb{E}[1/Z] = +\infty$ for any exponential random variable Z, and in particular $\mathbb{E}[\tilde{n}] = +\infty$.

This technicality disappears once we duplicate this procedure and compute each approximation $\tilde{n}$ and $\tilde{s}$, and in fact Lemma 3.3.1 shows that duplication allows us to tune the quality of the approximations, and so of that of the average. The resulting algorithm takes a precision parameter $r \in]0, 1/2[$ and a tolerated probability of failure $p]0, 1/2[$ and computes an r-approximation of the average in a Monte Carlo manner, with probability at least $1 - p$.

The exponential distribution $\mathrm{Exp}(\lambda)$ is only defined for $\lambda > 0$. To use the above scheme over any input interval $\Xi = [a, b]$, we replace each value μ_u with the translated value $\mu_u - a + 1 \geqslant 1$, and obtain the estimation of the actual average $\overline{\mu}$ by translating back the result of the computation. The choice of 1 in this expression is arbitrary, in the sense that we could use any constant $\gamma > 0$ instead.

To guarantee a precision of r in the computation with probability $1-p$, sampling each distribution $\ell := \left\lceil 3(1 + 2/r)^2(b - a + 1)^2(\ln 4 - \ln p)\right\rceil$ times in parallel suffices. Thus this first randomized algorithm is parametrized by the four values r, p, a, b representing the target precision, the tolerated error, and bounds over the domain of the input values. We give the pseudocode in Algorithm 5, and establish that the algorithm $\mathcal{R}_{r,p,a,b}$ is a Monte Carlo in Theorem 3.3.3.

Theorem 3.3.3. Let p, r, a, and b be real numbers, with $p, r \in]0, 1/2[$ and $a < b$. Let $\mathbb{G}$ be a dynamic graph, assumed to be eventually connected, and let $\boldsymbol{\mu} \in [a, b]^{\mathrm{V}(\mathbb{G})}$.

When considering the executions of the randomized algorithm $\mathcal{R}_{r,p,a,b}$ over the dynamic communication graph $\mathbb{G}$ with input values $\boldsymbol{\mu}$, the system always achieves stabilizing consensus, and the distribution of the consensus value ω satisfies

$$\mathbb{P}[|\omega - \overline{\mu}| \leqslant r] \geqslant 1 - p \quad . \tag{3.6}$$

Once all agents are active, the stable consensus is reached in at most $\operatorname{diam}\mathbb{G}$ rounds – provided that the diameter is finite. ⊛

Algorithm 5: The algorithm $\mathcal{R}_{r,p,a,b}$ code for agent u.

1 **Input:** $\mu_u \in [a, b]$
2 **Initially:**
3 **for** $i \leftarrow 1$ **to** $\ell := \lceil 3(1+2/r)^2(b-a+1)^2(\ln 4 - \ln p)\rceil$ **do**
4 $\mathsf{X}_u^{(i)} \leftarrow \sigma_u^{(i)} \sim \mathrm{Exp}(\mu_u - a + 1)$
5 $\mathsf{Y}_u^{(i)} \leftarrow \nu_u^{(i)} \sim \mathrm{Exp}(1)$
6 **In each round:**
7 send $m_u = \langle \mathbf{X}_u, \mathbf{Y}_u \rangle$
8 receive $m_{v_1}, \ldots, m_{v_d}$ ▷ d neighbors
9 **for** $i \leftarrow 1$ **to** ℓ **do**
10 $\mathsf{X}_u^{(i)} \leftarrow \min(\mathsf{X}_{v_1}^{(i)}, \cdots, \mathsf{X}_{v_k}^{(i)})$
11 $\mathsf{Y}_u^{(i)} \leftarrow \min(\mathsf{Y}_{v_1}^{(i)}, \cdots, \mathsf{Y}_{v_k}^{(i)})$
12 output $(\sum_{i=1}^{\ell} Y_u^{(i)} / \sum_{i=1}^{\ell} X_u^{(i)} + a - 1)$

Proof. In terms of the propagation of the initial values, Algorithm 5 consists in 2ℓ parallel instances of the FloodInf algorithm, one for every entry of the sample vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$. As a consequence of Proposition 1.5.1, a stabilizing consensus is achieved in each individual entry $\boldsymbol{X}_u^{(i)}$ or $\boldsymbol{Y}_u^{(i)}$, in at most $\operatorname{rad}\mathbb{G}$ rounds after the last start signal if the radius is finite. The system itself then achieves stabilizing consensus under the same conditions, as the output value $x_u(t)$ only depends on the parameter a and on the values taken in round t by the arrays $\mathbf{X}_u$ and $\mathbf{Y}_u$.

It remains to show eq. (3.6). We fix an execution of the algorithm by agents in V, and use the notation:

$$\begin{gathered} \widetilde{\sigma}^{(i)} := \min_{u\in V} \sigma_u^{(i)} \ , \qquad \widetilde{\nu}^{(i)} := \min_{u\in V} \nu_u^{(i)} \ , \\ \widetilde{\sigma} := \frac{1}{\ell}\sum_{i=1}^{\ell} \widetilde{\sigma}^{(i)} \ , \qquad \widetilde{\nu} := \frac{1}{\ell}\sum_{i=1}^{\ell} \widetilde{\nu}^{(i)} \ , \qquad \widetilde{\mu} := \frac{\widetilde{\nu}}{\widetilde{\sigma}} \ . \end{gathered} \tag{3.7}$$

For all agents, the variable $\mathsf{X}_u^{(i)}$ eventually stabilizes over the value $\widetilde{\sigma}^{(i)}$ for all $i \in [\ell]$; similarly, each variable $\mathsf{Y}_u^{(i)}$ stabilizes over $\widetilde{\nu}^{(i)}$. By line 12, the consensus value of the execution is given by $\omega = a - 1 + \widetilde{\mu}$. We can assume that $a = 1$ without loss of generality, as we can always reduce to this case by considering the problem over the translate values $\mu'_u = \mu_u - a + 1$.

So we assume that $a = 1$, and in this case each execution of the algorithm over the pair $\mathbb{G}, \boldsymbol{\mu}$ produces a consensus value $\widetilde{\mu}$. This value depends on the input vector $\boldsymbol{\mu}$ and the outcomes of the random oracles, and it remains to show that

$$\mathbb{P}[\,|\widetilde{\mu} - \overline{\mu}| \leqslant r\,] \geqslant 1 - p \ . \tag{3.8}$$

We define $\alpha \coloneqq \frac{r}{(2+r)b}$, and we have

$$\begin{aligned} \frac{1+\alpha}{1-\alpha} &= 1 + \frac{2r}{2b + (b-1)r} \\ &\leqslant 1 + \frac{r}{b} \ , \end{aligned} \qquad \Big\downarrow\, b \geqslant a \geqslant 1 \tag{3.9}$$

$$\text{and similarly } \frac{1-\alpha}{1+\alpha} \geqslant 1 - \frac{r}{b} \ .$$

Suppose now that we have

$$|\widetilde{\sigma} - \frac{1}{s}| \leqslant \frac{\alpha}{s} \quad \text{and} \quad |\widetilde{\nu} - \frac{1}{n}| \leqslant \frac{\alpha}{n} \ , \tag{3.10}$$

and let us show that $\widetilde{\mu}$ is admissible as a consensus value. Indeed, we have equivalently

$$\begin{aligned} &\begin{cases} \frac{1}{s}(1-\alpha) \leqslant \widetilde{\sigma} \leqslant \frac{1}{s}(1+\alpha) \\ \frac{1}{n}(1-\alpha) \leqslant \widetilde{\nu} \leqslant \frac{1}{n}(1+\alpha) \end{cases} \qquad \Big\downarrow\, \textit{eq.}\ (3.9) \\ \Longrightarrow &\left(1 - \frac{r}{b}\right)\frac{s}{n} \leqslant \frac{\widetilde{\nu}}{\widetilde{\sigma}} \leqslant \left(1 + \frac{r}{b}\right)\frac{s}{n} \ , \end{aligned}$$

which is to say that $|\overline{\mu} - \widetilde{\mu}| \leqslant \frac{r}{b}\,\overline{\mu} \leqslant r$.

To conclude, it suffices to show that an execution satisfies eq. (3.10) with good probability. Each value $\sigma_u^{(i)}$ is sampled from the distribution $\mathrm{Exp}(\mu_u)$ and since the random oracles produce independent outcomes, Lemma 3.3.2 the value $\widetilde{\sigma}^{(i)} = \min_{u \in V} \sigma_u^{(i)}$ behaves as if sampled from $\mathrm{Exp}(s)$, and similarly $\widetilde{\nu}^{(i)} = \min_{u \in V} \nu_u^{(i)}$ behaves as if sampled from $\mathrm{Exp}(n)$. We then use Lemma 3.3.1: with the values

$$\begin{aligned} \alpha &= \frac{r}{(2+r)b} \\ \ell &= \left\lceil 3(2+r)^2(b-a+1)^2(\ln 4 - \ln p)/r^2 \right\rceil \ , \end{aligned}$$

instantiating eq. (3.4) gives

$$\mathbb{P}\left[\left|\frac{1}{\ell}(Z_1 + \cdots + Z_\ell) - \frac{1}{\lambda}\right| \geqslant \frac{\alpha}{\lambda}\right] \leqslant 2\left(\frac{p}{4}\right)^{\left((1+\frac{2}{r})b\alpha\right)^2} = \frac{p}{2} \ , \tag{3.11}$$

for any random variables $Z_1, \dots, Z_\ell$ independently following the same exponential distribution $\mathrm{Exp}(\lambda)$. In particular, $\mathbb{P}\left[|\widetilde{\sigma} - \frac{1}{s}| \geqslant \frac{\alpha}{s}\right] \leqslant \frac{p}{2}$ and $\mathbb{P}\left[|\widetilde{\nu} - \frac{1}{n}| \geqslant \frac{\alpha}{n}\right] \leqslant \frac{p}{2}$, and since these are independent event the probability of their union is less than p. That is to say, the probability of an execution satisfying eq. (3.10) is at least $1 - p$, and so

$$\mathbb{P}[|\widetilde{\mu} - \overline{\mu}| \leqslant r] \geqslant 1 - p \quad . \tag{3.12}$$

□

3.4 QUANTIZATION

We now factor in a key aspect of digital systems: they manipulate discrete values and dispose of limited bandwidth and storage. As a general matter, this means that we can no longer have agents store or transmit real values, and that we must pay attention to the memory footprint and the size of messages exchanged over the course of an execution. The process of mapping a large set onto a smaller, discrete set, whose elements admit an efficient representation is called *quantization*, and in this section we propose a quantized refinement of Algorithm 5.

To this effect, we adopt a quantization method used by Kuhn, Lynch, and Oshman in [63, Section 7], and detailed by Oshman in [85, Section 3.6]. These authors extend Mosk-Aoyama and Shah's approach in a manner very similar to ours, considering the *counting problem* – that of algorithmically evaluating the order n of the network. In fact, they look at the stronger problem of terminating computation under synchronous starts under the assumption that agents initially agree on an upper bound $N \geqslant n$. For now, we remain focused on stabilization and make no such assumption; the next section will be devoted to the problem of termination under *asynchronous* starts.

The quantization strategy used by Kuhn et al. consists in rounding real values down along a logarithmic scale, to the previous integer power of some pre-defined base. We will apply a similar method here: for some fixed parameter of the algorithm, $\beta > 0$, we approximate any value $z \in \mathbb{R}$ with its β-LOGARITHMIC ROUNDING $[z]_\beta$, defined as

$$[-]_\beta : z \in \mathbb{R}_{>0} \mapsto [z]_\beta \coloneqq (1+\beta)^{\left\lfloor \log_{1+\beta} z \right\rfloor} \quad . \tag{3.13}$$

Since β is a fixed parameter of the algorithm, it is only necessary for agents to store and transmit the integer exponent $\left\lfloor \log_{1+\beta} z \right\rfloor$, satisfying the requirements of only storing and transmitting discrete values.

This quantization method brings two other assets. First, the outcomes of the random oracles are all β-rounded, usually, into values with a very efficient representation, requiring only $O(\log\log n)$ bits each with high probability. This guarantee of low resource consumption is only probabilistic, in the sense that there is a positive probability, albeit small, that a random oracle outputs a value z for which $\left|\log_{1+\beta} z\right|$ is arbitrarily large. In this sense, the algorithm $\overline{\mathcal{R}}$ discussed in this section is Las Vegas in addition to being Monte Carlo.

In addition, logarithmic rounding is compatible with the exponential nature of the random variables, in the sense that we can turn the concentration inequalities given in Lemma 3.3.1, concerning an exponential random variable $Z \sim \mathrm{Exp}(\lambda)$ into ones over the β-rounded random variable $[Z]_\beta$, from which we can, in broad strokes, continue to use the approach of the previous section.

Lemma 3.4.1. Let $Z_1, \ldots, Z_k$ be independent random variables distributed as $\mathrm{Exp}(\lambda)$ for some $\lambda \in \mathbb{R}_{>0}$. Given any rounding parameter $\beta \in \mathbb{R}_{>0}$ and error $\alpha \in \left]0, 1/2\right[$, we have

$$\mathbb{P}\left[\left|\frac{1}{k}\left([Z_1]_\beta + \cdots + [Z_k]_\beta\right) - \frac{1}{\lambda}\right| \geqslant \frac{\alpha + \beta + \alpha\beta}{\lambda}\right] \leqslant 2\exp\left(-\frac{k\alpha^2}{3}\right) \quad . \tag{3.14}$$

❀

Proof. By eq. (3.13), we have $[z]_\beta \leqslant z < (1+\beta)[z]_\beta$ for any positive real number z, which implies, for all $i \in [k]$:

$$0 \leqslant Z_i - [Z_i]_\beta < \beta[Z_i]_\beta \leqslant \beta Z_i \quad . \tag{3.15}$$

Let us then define random variables Z and Z' by:

$$Z \coloneqq \frac{1}{k}(Z_1 + \cdots + Z_k) \quad , \qquad Z' \coloneqq \frac{1}{k}([Z_1]_\beta + \cdots + [Z_k]_\beta) \quad . \tag{3.16}$$

We have

$$\begin{aligned} \left|Z' - \frac{1}{\lambda}\right| &\leqslant |Z' - Z| + \left|Z - \frac{1}{\lambda}\right| \\ &\leqslant \beta Z + \left|Z - \frac{1}{\lambda}\right| \quad . \end{aligned} \qquad \text{eq. (3.15)}$$

By Lemma 3.3.1, with probability $p \geqslant 1 - 2\exp\left(-\frac{k\alpha^2}{3}\right)$ we have $\left|Z - \frac{1}{\lambda}\right| \leqslant \frac{\alpha}{\lambda}$, in which case the above gives us

$$\left|Z' - \frac{1}{\lambda}\right| \leqslant \frac{\alpha + \beta + \alpha\beta}{\lambda} \quad .$$

□

For a random variable $Y \sim \mathrm{Exp}(\lambda)$, we will write $Z \sim [\mathrm{Exp}(\lambda)]_\beta$ for the random variable $Z \coloneqq [Y]_\beta$.

We give the algorithm $\overline{\mathcal{R}}$ in Algorithm 6; it differs from the algorithm $\mathcal{R}$ in that the outcomes of the random oracles are logarithmically rounded with $[-]_{\frac{r}{6(b-a+1)}}$ and that the number ℓ of samples is larger by a constant multiplicative factor, in order to account for the probability of a Las Vegas failure.

Algorithm 6: The algorithm $\overline{\mathcal{R}}_{r,p,a,b}$ code for agent u.

Input: $\mu_u \in [a, b]$
Initially:
  let $\ell \coloneqq \left\lceil 3\left[(1 + \frac{4}{r})(b - a + 1)\right]^2 (\ln 8 - \ln p)\right\rceil$
  let $\beta \coloneqq \frac{r}{6(b-a+1)}$
  for $i \leftarrow 1$ **to** ℓ **do**
    $\mathsf{X}_u^{(i)} \leftarrow \left[\sigma_u^{(i)}\right]_\beta$, with $\sigma_u^{(i)} \sim \mathrm{Exp}(\mu_u - a + 1)$
    $\mathsf{Y}_u^{(i)} \leftarrow \left[\nu_u^{(i)}\right]_\beta$, with $\nu_u^{(i)} \sim \mathrm{Exp}(1)$
In each round:
  send $m_u = \langle X_u, Y_u \rangle$
  receive $m_{v_1}, \ldots, m_{v_d}$ ▷ d neighbors
  for $i \leftarrow 1$ **to** ℓ **do**
    $\mathsf{X}_u^{(i)} \leftarrow \min(\mathsf{X}_{v_1}^{(i)}, \cdots, \mathsf{X}_{v_d}^{(i)})$
    $\mathsf{Y}_u^{(i)} \leftarrow \min(\mathsf{Y}_{v_1}^{(i)}, \cdots, \mathsf{Y}_{v_d}^{(i)})$
  output $(\sum_{i=1}^{\ell} Y_u^{(i)} / \sum_{i=1}^{\ell} X_u^{(i)} + a - 1)$

We show that the algorithm $\overline{\mathcal{R}}$ is Monte Carlo in Proposition 3.4.2, and Las Vegas with respect to storage and bandwidth in Proposition 3.4.3, collecting both results in Theorem 3.4.4. For the rest of this section, we

fix real numbers p, r, a, and b, with $p, r \in]0, 1/2[$ and $a < b$, and we define β and ℓ as in Algorithm 6.

Proposition 3.4.2. Let $\mathbb{G}$ be a dynamic graph, assumed to be eventually connected, and let $\boldsymbol{\mu} \in [a, b]^{V(\mathbb{G})}$.

All executions of the randomized algorithm $\overline{\mathcal{R}}_{r,p,a,b}$ over the dynamic communication graph $\mathbb{G}$ with input values $\boldsymbol{\mu}$ achieve stabilizing consensus. The random variable given by the consensus value ω satisfies:

$$\mathbb{P}[|\omega - \overline{\mu}| \leqslant r] \geqslant 1 - \frac{p}{2} \ . \tag{3.17}$$

Proof. Convergence to consensus follows from the study of the FloodInf algorithm, as it did for Algorithm 5. To show eq. (3.17), we proceed as for Algorithm 5: we can assume that $a = 1$ without loss of generality, and for each execution we let

$$\begin{aligned} \widehat{\sigma}^{(i)} &:= \min_{u \in V} \sigma_u^{(i)} \ , \qquad \widehat{\nu}^{(i)} := \min_{u \in V} \nu_u^{(i)} \ , \\ \widehat{\sigma} := \frac{1}{\ell} \sum_{i=1}^{\ell} \widehat{\sigma}^{(i)} \ , \qquad \widehat{\nu} &:= \frac{1}{\ell} \sum_{i=1}^{\ell} \widehat{\nu}^{(i)} \ , \qquad \widehat{\mu} := \frac{\widehat{\nu}}{\widehat{\sigma}} \ . \end{aligned} \tag{3.18}$$

where these values depend on the dynamic graph, the input values, and the outcomes of the random oracles. In particular, $\widehat{\mu}$ is the consensus value, and so we need to show that $\mathbb{P}[|\widehat{\mu} - \overline{\mu}| \leqslant r] \geqslant 1 - \frac{p}{2}$.

The β-rounding operator $[-]_\beta$ is weakly increasing, and so we have $\min([x]_\beta, [y]_\beta) = [\min(x, y)]_\beta$; as a consequence, we have for each $i \in [\ell]$:

$$\widehat{\sigma}^{(i)} = [Z \sim \mathrm{Exp}(s)]_\beta \qquad \text{and} \qquad \widehat{\nu}^{(i)} = [Z \sim \mathrm{Exp}(n)]_\beta \ . \tag{3.19}$$

We now define $\alpha := \frac{1}{(1+\frac{4}{r})b}$; using Lemma 3.4.1 with those values and $\ell = \left\lceil 3\left[(1 + \frac{4}{r})(b - a + 1)\right](\ln 8 - \ln p)\right\rceil$, we have $\mathbb{P}[|\widehat{\sigma} - \frac{1}{s}| \geqslant \frac{\alpha+\beta+\alpha\beta}{s}] \leqslant \frac{p}{4}$ and $\mathbb{P}[|\widehat{\nu} - \frac{1}{n}| \geqslant \frac{\alpha+\beta+\alpha\beta}{n}] \leqslant \frac{p}{4}$, which by the union bound, gives

$$\mathbb{P}[|\widehat{\mu} - \overline{\mu}| \leqslant r] \geqslant 1 - \frac{p}{2} \ . \tag{3.20}$$

□

Proposition 3.4.3. When n agents run the algorithm $\overline{\mathcal{R}}_{r,p,a,b}$ over input values in $[a, b]$, with probability at least $1 - \frac{p}{2}$, all outcomes of the random oracles can be represented over $Q = \mathrm{O}(\frac{1}{r}(\log n - \log p - \log r))$ quantization levels after β-quantization.

Proof. For $\lambda \geqslant 1$, the cumulative function $\mathbb{P}[Z \leqslant z] = 1 - \mathrm{e}^{-\lambda z}$ of an exponential random variable $Z \sim \mathrm{Exp}(\lambda)$ gives

$$\forall z \in]0, 1[: \begin{cases} \mathbb{P}[Z \leqslant z] & \leqslant \lambda z \\ \mathbb{P}[Z > -\ln z] & \leqslant \quad z \ , \end{cases} \tag{3.21}$$

and as a consequence we have

$$\forall z \in]0, 1[: \ \mathbb{P}[Z \notin [z, -\ln z]] \leqslant z + \lambda z \ . \tag{3.22}$$

We let $z_Q \coloneqq \frac{p}{4\ell n(b-a+2)} \in]0, \frac{1}{16}[$, and by the above we have for all pairs u, i:

$$\begin{aligned} \mathbb{P}[\sigma_u^{(i)} \notin [z_Q, -\ln z_Q]] &\leqslant \frac{p}{4\ell n} \\ \mathbb{P}[\nu_u^{(i)} \notin [z_Q, -\ln z_Q]] &\leqslant \frac{p}{4\ell n} \ , \end{aligned} \tag{3.23}$$

which gives, since the random oracles produce independent outcomes,

$$\mathbb{P}[\forall u, \forall i : \sigma_u^{(i)} \in [z_Q, -\ln z_Q] \wedge \nu_u^{(i)} \in [z_Q, -\ln z_Q]] \geqslant 1 - \frac{p}{2} \ . \tag{3.24}$$

Let us now see that this results in a compact representation for the β-roundings of the outcomes.

Specifically, we let Q denote the cardinal of the set $\{ [z]_\beta \mid z \in [z_Q, -\ln z_Q] \}$; by eq. (3.24), all outcomes of the random oracles are mapped onto this set with probability $1 - \frac{p}{2}$, in which case Q quantization levels suffice to represent them all. We can bound the quantization levels required to represent values in an interval $[c, d]$ with

$$\left| [[c, d]]_\beta \right| \leqslant \lceil \log_{1+\beta} d \rceil - \lfloor \log_{1+\beta} c \rfloor \ ; \tag{3.25}$$

for the interval $[z_Q, -\ln z_Q]$, this results in

$$Q = \mathrm{O}(\log_{1+\beta} \frac{\ell n}{p}) \ . \tag{3.26}$$

Using the fact that $\log_{1+\beta} - < 2 \log \frac{-}{\beta}$ for $\beta \in]0, 1[$ and plugging in the values for β and ℓ given in Algorithm 6, we finally have

$$Q = \mathrm{O}(\frac{1}{r}(\log n - \log p - \log r)) \ . \tag{3.27}$$

□

Theorem 3.4.4. Let p, r, a, and b be real numbers, with $p, r \in]0, 1/2[$ and $a < b$. Let $\mathbb{G}$ be a dynamic graph of order n, assumed to be eventually connected, and let $\boldsymbol{\mu} \in [a, b]^{\mathrm{V}(\mathbb{G})}$.

Any execution of the randomized algorithm $\overline{\mathcal{R}}_{r,p,a,b}$ over the dynamic communication graph $\mathbb{G}$ with input values $\boldsymbol{\mu}$ achieves stabilizing consensus. With probability at least $1 - p$, the consensus value is in the interval $[\overline{\mu} - r, \overline{\mu} + r]$, and the execution has a memory and bandwidth footprint of $\mathrm{O}(\frac{-\log p}{r^2}(\log(\log n - \log p) - \log r))$ bits. 🙃

Proof. The convergence of the algorithm $\overline{\mathcal{R}}_{r,p,a,b}$ follows exactly the same patterns as that of the un-quantized algorithm $\mathcal{R}_{r,p,a,b}$, discussed in Theorem 3.3.3. For a given dynamic graph and input assignment, Proposition 3.4.2 shows that the consensus value is within distance r of the average $\overline{\mu}$ with probability at least $1 - \frac{p}{2}$. Likewise, Proposition 3.4.3 shows that the individual values used throughout the execution admit a representation over $Q = (\log n - \log p - \log r)$ quantization levels with probability at least $1 - \frac{p}{2}$; in this case, they each have a footprint of $\log Q = \mathrm{O}(\log(\log n - \log p - \log r))$ bits, and since there are $2\ell = \mathrm{O}(\frac{-\log p}{r^2})$ of them, their total footprint is as claimed.

With probability at least $1 - p$, both events hold at the same time, which proves the theorem. □

Remark 3.4.5. The exact primitive used to disseminate the minimal values $\widehat{\sigma}^{(i)}$ and $\widehat{\nu}^{(i)}$ is orthogonal to the result of the computation and its memory footprint. However, it does matter for the bandwidth. In Algorithm 6, we have used the FloodInf algorithm as the information dissemination primitive, but we could as well have used the entry-wise version of the FloodInf algorithm instead to drive down the bandwidth footprint even further. Sending one entry of each array $\mathbf{X}$ and $\mathbf{Y}$ at a time results in a bandwidth footprint of $\mathrm{O}((\log(\log n - \log p) - \log r))$ at the cost of delaying the computation time by a factor $\ell = \mathrm{O}(\frac{-\log p}{r^2})$. As mention when we discussed the entry-wise version of the FloodInf algorithm, the connectivity requirements to reach the correct value are stronger than when the agents simply broadcast the entire content of their memory.

3.5 DECISION

Finally, we turn to the fullest version of our average consensus randomized algorithm. While the algorithms $\mathcal{R}$ and $\overline{\mathcal{R}}$ both produce stabilizing

executions, the algorithm $\overline{\mathcal{D}}$ that we now present is terminating, in a Monte Carlo sense: all agents eventually decide on a value, and with high probability **a)** all agents simultaneously decide on the same value ω; **b)** ω is an admissible approximation of the average $\overline{\mu}$; and **c)** all local variables admit an efficient representation. That is, the algorithm $\overline{\mathcal{D}}$ is Las Vegas in the same way that the algorithm $\overline{\mathcal{R}}$ is.

The decision mechanism is derived from that of the randomized firing squad algorithm introduced by Charron-Bost and Moran in [36, Section 6], which is in turn grounded in the randomized counting algorithm developed in [63, 85] that we mentioned in the previous section. As such, the domain of application of the algorithm $\overline{\mathcal{D}}$ is the same as that of [36]: its Monte Carlo guarantees apply over any system with asynchronous starts, provided that all agents eventually become active and that the dynamic communication graph be strongly connected in every round.

For a system of n agents, the radius of such a dynamic graph is at most $n-1$. With high probability, all agents reach an admissible approximation of the average in at most $n-1$ rounds after all agents become active. It therefore suffices to provide a mechanism by which agents can determine that they have reached a round $t^* \geqslant s_{\max} + n$,[4] since they can then all simultaneously return their approximation in that round.

[4] Recall that $s_{\max} \in \mathbb{N}_{>0}$ denotes the latest round with an inactive agent.

As is the case for the algorithms given in [63, 85, 36], the algorithm $\overline{\mathcal{D}}$ will have to take as a parameter a bound $N \geqslant n$. In a model of simultaneous starts, this suffices to ensure simultaneous decisions that are late enough: agents can simply run the algorithm $\overline{\mathcal{R}}$ while maintaining a local round counter that is incremented in each round. Once this counter reaches N, they return their estimate, which by Theorem 3.4.4 is likely to be admissible.

This approach is insufficient in two respects. First, it strongly depends on the assumption of simultaneous starts: without that assumption, all agents no longer decide simultaneously, and in fact some agents may well decide while others are inactive, leading to an incorrect result. Moreover, although the estimates *stabilizes* quickly – in $n-1$ rounds, as with the algorithm $\overline{\mathcal{R}}_{r,p,a,b}$ over the same dynamic graph – it only *decide* after N rounds. In other words, the temporal complexity is no longer in the *order* of the network, but in the *bound* we provide over the order; for a very loose bound, the convergence time may be abysmal.[5]

[5] To take an extreme example, the IPV6 communication protocol [45] implicitly embeds a bound of the order of $N = 2^{128}$; for this value, and at the improbable rate of 10^{18} rounds per second, reaching termination would take of the order of 10^{13} years.

We address the latter issue by using the fact that the algorithm $\overline{\mathcal{R}}$ is already computing an approximation of the order n; using the value computed in this manner, rather than the bound N, allows us to make the

agents simultaneously terminate within $O(n)$ rounds of $s_{\max}$ with high probability. This is indeed the method used by Kuhn et al. in their approximate counting algorithm: the value being computed is simultaneously used by each agent to determine when to return.

For the former issue, we rely on results by Charron-Bost and Moran, and make the agents implement local round counters that remain bounded by n while some agents are inactive, and synchronize in at most n rounds once they all become active. Agents are able to detect this synchronization by round $s_{\max} + \frac{3}{2}n$ with high probability, which allows them to terminate. We give the pseudocode of the algorithm $\overline{\mathcal{D}}$ in Algorithm 7.

Theorem 3.5.1. Let N be a positive integer, and let p, r, a, and b be real numbers, with $p, r \in]0, 1/2[$ and $a < b$. Let $\mathbb{G}$ be a dynamic graph of order n, strongly connected in each round, and let $\boldsymbol{\mu} \in [a, b]^{V(\mathbb{G})}$.

When considering the executions of the algorithm $\overline{\mathcal{D}}$ over the dynamic communication graph $\mathbb{G}$, input values $\boldsymbol{\mu}$, and arbitrary start signals, with probability at least $1 - p$: **a)** all agents eventually return the same value ω; **b)** ω lies in the admissible interval $[\overline{\mu} - r, \overline{\mu} + r]$; **c)** once all agents are active, the execution terminates within $2n$ rounds – that is, all agents return simultaneously; and **d)** the execution has a memory and bandwidth footprint of $O\Big(\log N - \frac{\log p}{r^2}(\log(\log n - \log p)\log r)\Big)$. ⊛

Proof. We let s_u denote the latest round during which agent u is inactive, and $s_{\max}$ denote the latest round during which *any* agent is inactive. For any round $t \in \mathbb{N}_{>0}$ we let

$$V^*(t) \coloneqq \{\, u \in V \mid s_u < t \,\} \tag{3.28}$$

denote the set of agents that are active during round t.

Recall that we denote by $c_u(t)$ the value taken by the counter variable c_u at the *end* of round t. In any round, this value depends on how recently the agent last heard of an inactive agent:

$$\begin{aligned} &\forall t \in \mathbb{N}_{>0}, \forall u \in V^*(t), \forall \tau \leqslant t\colon \\ &\quad c_u(t) > t - \tau \Leftrightarrow \mathrm{In}_u(\tau\colon t) \subseteq V^*(\tau)\ , \end{aligned} \tag{3.29}$$

which is easily seen by a finite induction over τ.

From eq. (3.29), we deduce first that, in the initial period $t \in \{\, 1, \ldots, s_{\max} \,\}$, no agent's counter can be larger than $\operatorname{diam}\mathbb{G}$. Moreover, for any agent u for which $s_u = s_{\max}$ and $t > s_{\max}$, we have $c_v(t) = t - s_{\max}$

Algorithm 7: The algorithm $\overline{\mathcal{D}}_{N,r,p,a,b}$ code for agent u.

Input: $\mu_u \in [a, b]$
Initially:
  $\mathsf{c}_u \leftarrow 0$
  $\widehat{\mathsf{n}}_u \leftarrow 1$
  let $\ell_C := \lceil 3[(1 + \frac{4}{r})(b - a + 1)]^2(\ln 24 - \ln p)\rceil$
  let $\ell_D := \lceil 243(\ln 6N^2 - \ln p)\rceil$
  let $\ell := \max(\ell_C, \ell_D)$
  let $\beta := \frac{r}{6(b-a+1)}$
  for $i \leftarrow 1$ **to** ℓ **do**
    $\mathsf{X}_u^{(i)} \leftarrow \left\lfloor \sigma_u^{(i)} \right\rfloor_\beta$, with $\sigma_u^{(i)} \sim \mathrm{Exp}(\mu_u - a + 1)$
    $\mathsf{Y}_u^{(i)} \leftarrow \left\lfloor \nu_u^{(i)} \right\rfloor_\beta$, with $\nu_u^{(i)} \sim \mathrm{Exp}(1)$
In each round:
  send $m_u = \langle \mathsf{c}_u, \boldsymbol{X}_u, \boldsymbol{Y}_u \rangle$
  receive $m_{v_1}, \ldots, m_{v_k}$ from neighbors $v_1, \ldots, v_k$
  if *at least one of* $m_{v_1}, \ldots, m_{v_k}$ *is a null message* **then**
    $\mathsf{c}_u \leftarrow 0$
  else
    $\mathsf{c}_u \leftarrow 1 + \min(c_{v_1}, \ldots, c_{v_k})$
  for $i \leftarrow 1$ **to** ℓ **do**
    $\mathsf{X}_u^{(i)} \leftarrow \min(\mathsf{X}_{v_1}^{(i)}, \ldots, \mathsf{X}_{v_k}^{(i)})$
    $\mathsf{Y}_u^{(i)} \leftarrow \min(\mathsf{Y}_{v_1}^{(i)}, \ldots, \mathsf{Y}_{v_k}^{(i)})$
  $\widehat{\mathsf{n}}_u \leftarrow \frac{\ell}{\mathsf{Y}_u^{(1)} + \cdots + \mathsf{Y}_u^{(\ell)}}$
  if $\mathsf{c}_u > \frac{3}{2}\widehat{\mathsf{n}}_u$ **then**
    return $(\sum_{i=1}^{\ell} \mathsf{Y}_u^{(i)} / \sum_{i=1}^{\ell} \mathsf{X}_u^{(i)} + a - 1)$

for all agents $v \in \mathrm{Out}_u(s_{\max} + 1 \colon t)$, and so starting in some round $t_{\mathrm{sync}} \in \{ s_{\max} + 1 \ldots, s_{\max} + \operatorname{diam} \mathbb{G} \}$, all counters satisfy $c_u(t) = t - s_{\max}$ for all rounds $t \geqslant t_{\mathrm{sync}}$. Since we assume a dynamic graph that is strongly connected in each round, we can rephrase these observations by:

$$\begin{aligned} \forall t \in \{ 1, \ldots, s_{\max} \} \colon c_u(t) < n \\ \forall t \geqslant s_{\max} + n \colon c_u(t) = t - s_{\max} \quad . \end{aligned} \tag{3.30}$$

Since the counters keep increasing while $\widehat{n}(t)$ is obviously bounded, each agent eventually outputs a value. Because the local counters eventually become synchronized, the local test of Line 23 can result in a global, synchronous termination – provided that the condition of the test is never true *before* the round t_{sync}. This requires tracking the behavior of each agent's estimate $\widehat{n}_u(t)$ of the order n.

So we let, in any round t, $\mathbb{G}^*(t)$ denote the graph induced by $\mathbb{G}$ over the set $V^*(t)$ – that is, $\mathbb{G}^*(t)$ is the graph of message receptions between all *active* agents in round t. We let $\mathcal{I}_u^*(t) \coloneqq \mathrm{In}_u(\mathbb{G}^*(s_u + 1 \colon t))$[6] denote the set of all active agents of which agent u has heard of in round t, and we then easily verify that

$$\forall t > s_u, \forall i \in \{ 1, \ldots, \ell \} \colon \begin{cases} X_u^{(i)}(t) & = \min_{v \in \mathcal{I}_u^*(t)} [\sigma_v^{(i)}]_\beta \\ Y_u^{(i)}(t) & = \min_{v \in \mathcal{I}_u^*(t)} [\nu_v^{(i)}]_\beta \end{cases} , \tag{3.31}$$

from which we deduce

$$\forall t > s_u \colon \widehat{n}_u(t) = \frac{\ell}{\min\limits_{v \in \mathcal{I}_u^*(t)} [\nu_v^{(1)}]_\beta + \cdots + \min\limits_{v \in \mathcal{I}_u^*(t)} [\nu_v^{(\ell)}]_\beta} \quad . \tag{3.32}$$

By the assumption of round-wise strong connectivity, in any round $t > s_u$ there is an incoming edge from $V \setminus \mathcal{I}_u^*(t)$ into $\mathcal{I}_u^*(t)$, unless the latter is the entire set V. From eq. (3.29), we then have $c_u(t) < |\mathcal{I}_u^*(t)|$ whenever $\mathcal{I}_u^*(t) \neq V$. To ensure that the condition of line 23 is never triggered before $\mathcal{I}_u^*(t) = V$, it suffices to have

$$\widehat{n}_u(t) \geqslant \frac{2}{3} |\mathcal{I}_u^*(t)| \tag{3.33}$$

in each round t until $\mathcal{I}_u^*(t) = V$, which happens at the latest in round $t = s_{\max} + \operatorname{diam} \mathbb{G}$. As the set $\mathcal{I}_u^*(t)$ is weakly increasing over time with respect to inclusion, it suffices to verify eq. (3.33) for each of the $k_u < n$ rounds $t_u^1, \ldots, t_u^{k_u}$ in which this set strictly increases, since outside of

[6] In contrast with the way the notation $\mathbb{H}(t \colon t')$ used in most other places in this monograph, the individual graphs $\mathbb{G}^*(\tau)$ need not all be defined over the same vertex set. Let us simply recall here that the definition of the graph composition is perfectly compatible with this use.

those rounds eqs. (3.31) and (3.32) show that both $|\mathcal{I}_u^*(t)|$ and $\widehat{n}_u(t)$ remain unchanged.

We denote by B_u^j the event $\widehat{n}_u(t_u^j) \geqslant \frac{2}{3}|\mathcal{I}_u^*(t_u^j)|$ which depends on the dynamic graph $\mathbb{G}$, the start signals s_u, and the outcomes of the random oracles. If agent u only decides once $\mathcal{I}_u^*(t) = V$, then by eq. (3.31) it never revokes its decision; moreover, if it is the case for all agents, then their counters have synchronized by the time they decide, and we have the inclusion

$$\{\text{ the execution terminates }\} \subseteq B \coloneqq \bigcap_{u\in V} \bigcap_{j\in[k_u]} B_u^j \ , \tag{3.34}$$

and the return value of the execution is in fact $\widehat{\mu}$, defined as for Proposition 3.4.2 in eq. (3.18).

Let us then bound the probability of an execution over $\mathbb{G}$ and $(s_u)_u$ realizing event B. We can proceed as in Proposition 3.4.2: using Lemma 3.4.1 for $k \geqslant \ell_D$ samples and random variables $Z_i \sim [\mathrm{Exp}(\widehat{n}_u(t^j))]_\beta$, we find that

$$\mathbb{P}[\widehat{n}_u(t_u^j) < \frac{2}{3}|\mathcal{I}_u^*(t_u^j)|] \leqslant \frac{p}{3N^2} \ , \tag{3.35}$$

from which a union bound gives us $\mathbb{P}[B] \geqslant 1 - \frac{p}{3}$. Combined with eq. (3.34), we have

$$\mathbb{P}[\text{the execution terminates and returns } \widehat{\mu}] \geqslant 1 - \frac{p}{3} \ . \tag{3.36}$$

When does the execution terminate? The approximation of the order n computed by the agents in an execution is given by

$$\widehat{n} = \frac{\ell}{\min_u[\nu_u^{(1)}]_\beta + \cdots + \min_u[\nu_u^{(\ell)}]_\beta} \ . \tag{3.37}$$

With $\ell \geqslant \ell_D$, Lemma 3.4.1 gives $\mathbb{P}[\widehat{n} \leqslant \frac{3}{2}n] \geqslant 1 - \frac{p}{3}$; a union bound with eq. (3.36) gives us[7]

$$\mathbb{P}\begin{bmatrix}\text{the execution terminates, and}\\ \text{returns } \widehat{\mu} \text{ by round } s_{\max} + 2n\end{bmatrix} \geqslant 1 - \frac{2p}{3} \ . \tag{3.38}$$

Finally, the validity of the consensus value $\widehat{\mu}$, as well as the variables being well behaved in a Las Vegas sense, follow from exactly the same arguments as for Proposition 3.4.2; with $\ell \geqslant \ell_C$, these properties hold with probability at least $1 - \frac{p}{3}$, and we conclude with a union bound. □

[7] The arguments leading to eq. (3.38) are essentially those of Charron-Bost and Moran in [36, Section 6]. I would like to thank Shlomo Moran for insightful conversations that helped shape the work presented in this chapter.

3.6 SIMULATIONS

In this section, we seek to illustrate the behavior of our algorithms. For this, we fix a set $V = [n]$ of agents, which in our illustrations is each given a distinct color, shown in Figure 3.1 for two different graphs over V.

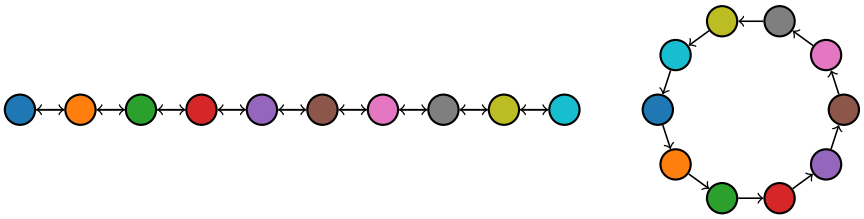

Figure 3.1: The line topology (left) and directed ring topology (right) used in our simulations.

We furthermore fix a set of inputs $\{ 1 + k/9, k = 0, \ldots, 9 \}$, which we assign once and for all to the agents in V: each agent $u \in V$ has a unique input $\mu_u = 1 + k_u/9$, and the average $\overline{\mu}$ is equal to $1/2$.

SIMULATING THE ALGORITHM $\mathcal{R}$

Given that Section 3.4 establishes the robustness of our algorithm to the rounding of the values in use, we run simulations for the simpler algorithm $\mathcal{R}$ rather than for $\overline{\mathcal{R}}$. We fix parameters $r = p = 1/3$. Using standard floating-point registers of a modern computer, the simulated computation is thus much more precise than our accepted error, ensuring that the effects of rounding are negligible.

On any dynamic graph $\mathbb{G}$, continuously strongly connected and with all self-loops at each round, our algorithm is shown to stabilize in at most diam $\mathbb{G}$ rounds. Hence, there are no specific difficulties arising from executing the algorithm $\mathcal{R}$ over a dynamic communication topology. For simplicity of exposition, we therefore illustrate our algorithm on fixed graphs, where the dynamic diameter coincides with the classical notion of diameter for directed graphs.

We plot in Figure 3.2 the individual estimates $x_u(t)$ for each agent $u \in V$ on the two different graphs of Figure 3.1, for the same input assignment $\boldsymbol{\mu}$ and parameters $r = p = 1/3$ and $(a, b) = (1, 2)$. The average $\overline{\mu}$ is marked by a dashed line. Note that the color used to draw each individual estimate $x_u(t)$ is reflected in the color of agent u in Figure 3.1.

We observe that the trajectory of individual estimates and the limit value $\widehat{\mu}$ vary with each execution. However, stabilization of the system always happens at round $t = n - 1$. Over the line graph, central agents stabilize at earlier times than extremal agents, whereas on the directed ring, all agents stabilize in the same round $t = n - 1$. This reflects the fact that an agent stabilizes when it has received the information originating in each other agent, which requires $\operatorname{diam} \mathbb{G} = 9$ rounds in the worst case.

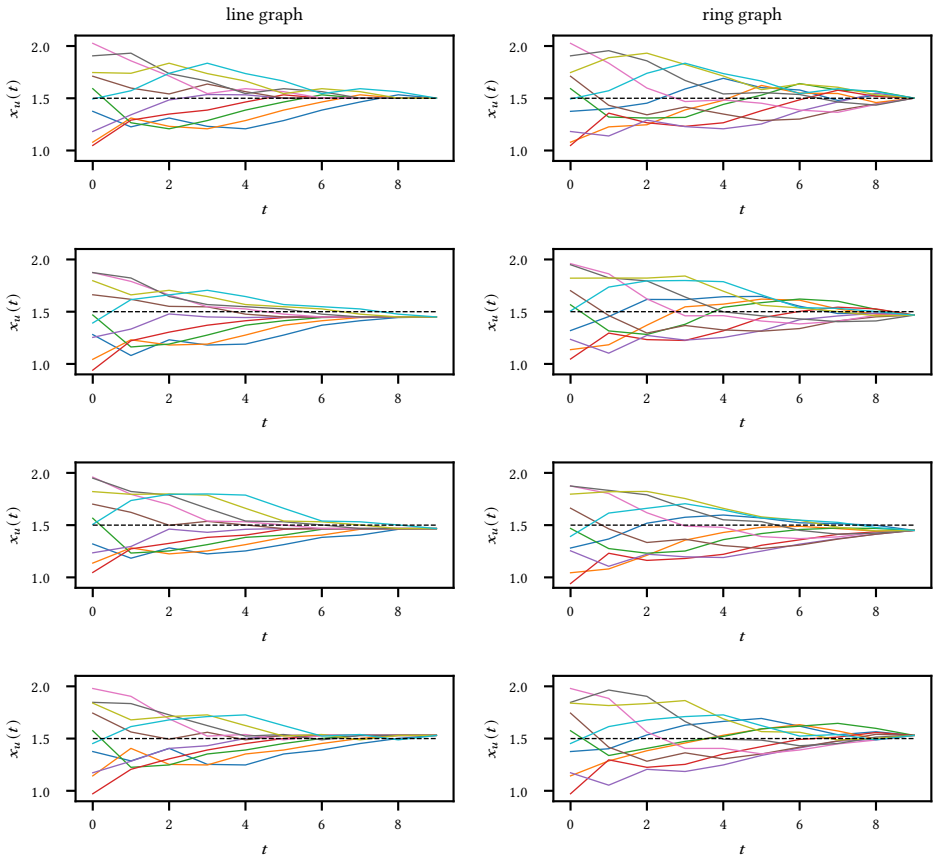

Figure 3.2: Agent estimates $x_u(t)$ for several executions of the algorithm $\mathcal{R}$ over the line graph and directed ring graph. Figure realized with Matplotlib [59].

SIMULATING THE ALGORITHM $\overline{\mathcal{D}}$

In Figure 3.3, we illustrate a typical execution of the algorithm $\overline{\mathcal{D}}$ on the line graph of Figure 3.1. As before, we use the input assignment $\boldsymbol{\mu}$ and parameters $r = p = 1/3$ and $(a, b) = (1, 2)$, with the additional parameter $N = 20$. We fix individual wake-up times chosen in $\{0, \ldots, 2n - 1\}$. As earlier, the color in use for each estimate $x_u(t)$ corresponds to the color of node u in the graph of Figure 3.1.

We observe five different phases, demarked in Figure 3.3 with dashed vertical rules. Initially, all agents are inactive. In the second phase, agents wake up and start computing local minima for each of their vector entries, sometimes with long periods with no progress for lack of new information. In the third phase, agents are all awake, information is disseminating, and the clocks c_u are synchronizing. In the fourth phase, all clocks c_u are equal and the estimates finish stabilizing at round t^c. Finally, in the fifth phase, the estimates have become stable and the system awaits decision. The final vertical rule marks the round t^d, where all agents simultaneously write in their variable d_u.

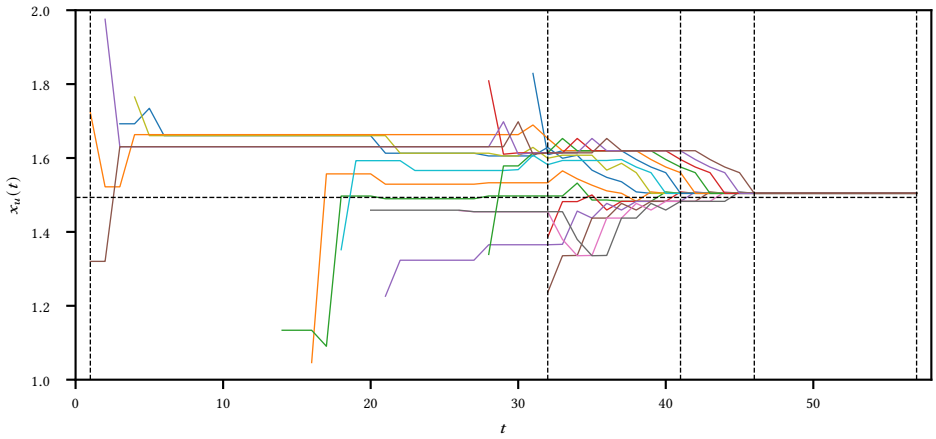

Figure 3.3: Agent estimates $x_u(t)$ for a typical execution of $\overline{\mathcal{D}}$ over the line graph of Figure 3.1. Figure realized with Matplotlib [59].

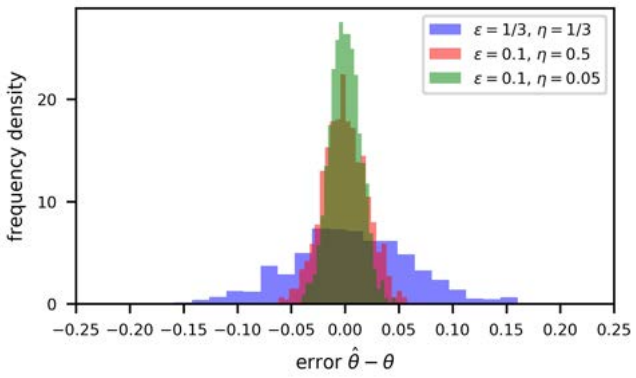

Figure 3.4: Distribution of the error $\widehat{\mu} - \overline{\mu}$ over 1000 simulations of $\mathcal{R}$, for various choices of r and p. Figure realized with Matplotlib [59].

PRECISION OF THE ALGORITHM $\mathcal{R}$

In the executions of our algorithms depicted in Figures 3.2 and 3.3, the computed estimate $\widehat{\mu}$ is much closer to the average $\overline{\mu}$ than what is allowed by fixing an accepted error $r = 1/3$. Here, we purport to show that this is not an accident of the specific executions we chose to depict.

Using $\boldsymbol{\mu}$ as the input assignment, we run many ($k = 1000$) executions of the algorithm $\mathcal{R}$ for various sets of parameters r and p, registering each time the *error* $\widehat{\mu} - \overline{\mu}$. Note that this is only a function of the input assignment $\boldsymbol{\mu}$ and the outcomes of the random oracles, not of the communication topology.

For each choice of parameters (r, p), we collect all errors into a histogram, which approximates experimentally the probability distribution of the error. These histograms are shown in Figure 3.4.

We observe that, while the distribution of the error is sensitive to the choice of r and p, the observed precision of the algorithm $\mathcal{R}$ is much stronger than what Theorem 3.3.3 entails. This suggests that our target precision can be met using fewer variables in the computations of s and n. Indeed, in designing our algorithms, we have focused on presenting the underlying ideas, rather than on optimizing the parameters. Whether the asymptotic complexity of the parameter ℓ can be improved is an open question.

> Thus, from the war of nature, from famine and death, the most exalted object which we are capable of conceiving, namely, the production of the higher animals, directly follows. There is grandeur in this view of life, with its several powers, having been originally breathed into a few forms or into one; and that, whilst this planet has gone cycling on according to the fixed law of gravity, from so simple a beginning endless forms most beautiful and most wonderful have been, and are being, evolved.
>
> *On the Origin of Species*
> Charles Darwin

Let us summarize the argument that we put forth in the preceding pages. The problems that arise in distributed computing are instrinsically hard: the field is littered with impossibility results that preclude the most basic tasks from admitting deterministic implementations that are truly distributed and local. Much progress in the theoretical study of distributed systems is thus built over a bedrock of simplifying assumptions – a complete communication topology, messages getting delivered under bounded delays, agents being initially synchronized, initial common knowledge in the form of the number of agents, the diameter of the communication graph, or the graph itself, infinite memory, computational power, and bandwidth... – as well as assumed features which may or may not correspond to the nature of specific networked systems – unique identifiers assigned to each agent, numbered ports locally assigned to the communication links, bidirectional communications, static communication topology...

We focused on two important classes of limitations. First, those that arise from the necessity to take irreversible and coordinated actions with each agent having only a partial, and partially outdated, view of the system. Second, those that arise from the symmetries of the network, which may cause some situations to look indistinguishable to the agents, event though they call for different behaviors. Much of the theory of distributed computing considers systems where symmetry is easily broken and focuses on problems of the former class. To some extent, this has to

do with the digital nature of the object of study: given unique identifiers (MAC or IP addresses, cryptographic public keys…), all symmetries can theoretically broken – not necessarily easily – and irreversibly committing together to some value underpins the general principle of *replication*, in turn foundational to the ability of distributed systems to reliably offer a coherent service at scale. In addition, such problems are *extremely* hard in the absence of simplifying assumptions like that of a system with real-time guarantees[8] making theoricians keen to not make their problems harder than is needed. In the synchronous model, on the other hand, in particular for a fixed communication graph and named agents that start and remain globally synchronized, – what amounts to the LOCAL model – there remains no fundamental limitation that prevents the system from acting as a single unit, and the "only" remaining problems are of efficiency in terms of temporal complexity, resource management, or other parameters to be optimized.

[8] We note for example the celebrated FLP result due to Fischer, Lynch, and Paterson [48], who established that the agreement problem of achieving termination, agreement, and validity is impossible to solve if even one agent may crash and there is no upper bound over the delay before messages get delivered.

We take the opposite tack and focus on cases where symmetries are generally unbreakable, assuming in particular that the agents do not come equipped with unique identifiers. This is motivated by examples in natural systems which do not necessarily offer a way to uniquely identify individuals, by low-powered agent systems which need not have the capacity to store or transmit identifiers, as well as from very simple multi-agent computational models such as cellular automata.

In these conditions, and lacking other ways of breaking symmetries, it becomes impossible for agents to definitively commit to a coordinated answer because information from far away in the network might yet change what the answer should be, and there are no ways of reliably evaluating the scale of the networks in a local manner. As a consequence, we focus on weak computational models – where it suffices to reach an answer without committing to it, or even to asymptotically tend towards an answer – motivated by applications coming from distributed *control*. Broadly speaking, some classes of applications are better served by starting with approximate answers that get refined over time rather than waiting for a long time before a question is definitively settled. One example can be found in distributed *load balancing*, where networked processing units assigned different amouts of work seek to distribute evenly the total work among themselves, which amounts to computing the average load. Clearly, the load will be processed faster if all units start processing some of the work right away than if they wait for an exact average consensus routine to terminate before they even begin.

Motivated by vehicular, mobile, and wireless networks, as well as by natural systems, we positioned our study in the general framework of *dynamic* networks, where communication links can frequently change. The problems of consensus and average consensus can be expressed as the system computing a relation or a function, and we recalled some of the fundamental constraints that apply to the latter kind of problem: in general, functions that are computable by an algorithm over a given connected network of unknown geometry can depend only on the set of the input values, but not on their multiplicities, even if the network is entirely static. In other words, all anonymous local schemes for computing specific multiplicity-sensitive functions have to exploit the geometry of the network in some manner.

We looked at two conditions under which the average is computable: by considering bidirectional networks – an example of geometric assumption – and by making agents more powerful through the use of random oracles. In both settings, we introduced novel algorithms which compare favorably in their respective leagues.

For bidirectional networks, we built upon the rich theory of convex updates for asymptotic consensus in multi-agent networks. Although they are known to converge under relatively connectivity of the network, and although they are often explicitly introduced in a context of dynamic communications, we argued that their good qualities over static networks do not readily extend to dynamic ones, for two reasons. First, their good analytical performance under static communications is generally only shown for *bidirectional* networks, but allowing the network to change over time reintroduces directionality in the flow of information, which can degrade the polynomial convergence time of the EqualNeighbor rule over static bidirectional networks down to an exponential one over static directed networks. Second, enforcing global invariants – such as maintaining the average of all agent estimates constant in order to enforce the average $\overline{\mu}$ as the consensus value – generally requires collecting information in the network beyond an agent's immediate horizon in a single round; but this information may never reach the agent if the network changes too fast, and schemes designed to cope with slow-paced change are fragile to a change of assumptions in the characteristics of the network.

We provided algorithms for consensus and average consensus that implement affine updates with a convergence time in $O(n^4)$ over dynamic networks that are strongly connected and bidirectional in each round.

These leverage a classic technique of discovering a parameter in a stabilizing manner, here, an agent's historically largest degree. We note that it was not guaranteed that the average *could* be obtained in this manner, given that dynamic networks reintroduce directionality even in bidirectional networks.

Over dynamic networks, we abandoned determinism and considered local algorithms where agents may consult random oracles. Randomization is a powerful method for breaking symmetries – given a bound over the order of the network, agents can assign themselves unique identities with high probability – so we focused in addition on *space efficiency* – designing algorithms with a small memory and bandwidth footprint – and on stronger computational semantics – getting executions to terminate rather than simply converge – in the particularly challenging settings of agents that are initially desynchronized. The stabilizing algorithm $\overline{\mathcal{R}}$, and its terminating version $\overline{\mathcal{D}}$, are Monte Carlo algorithms which, with a probability at least $1 - p$ for some parameter p tuneable in the code of the algorithm, produce a good approximation of the average $\overline{\mu}$ in a number of rounds that is linear in the number of agents.

Directions for future works We conclude by pointing at some research topics related to those we touched here and whose investigation looks potentially fruitful to us.

The line of research investigated by Boldi and Vigna in [17], which we could summarize in one sentence by "What do given classes of directed graphs allow for computing anonymously?", could potentially yield much finer results than the general considerations we make here. As classes of dynamic graphs are much more complex objects than classes of static graphs – the former may be of uncountable cardinality, whereas the latter are at most countably infinite – a central question is whether there are realistic non-trivial classes for which computable functions admit an effective description in the manner that Boldi and Vigna obtain for static classes. Moreover, the relevant generalizations of the graph theoretic tools of coverings and fibrations to dynamic graphs are yet to be fully fleshed out.

In a similar vein, it was shown by Avin, Koucky, and Lotker [9] that a random walk over a bidirectional graph that can change after each step – a dynamic graph in our terminology – can take an exponentially long time to mix, – to forget about the starting position of the walker – whereas it is well known that static bidirectional graphs produce a mixing

time that is at worst cubic. This idea was picked up by Olshevsky and Tsitsiklis [84] to produce a pathological case for the convergence time of the EqualNeighbor rule. What kinds of networks give rise to these pathologies? For the static case, the butterfly graph gives an example of how to produce a random walk that is poorly mixing: make a vertex so that there are many ways to arrive to it but few ways to visit the entire graph from it. Is there a threshold at which the polynomial mixing time becomes exponential? How much directivity is too much? These questions seem particularly intricate over dynamic graphs, where the principal source of directivity may be the arrow of time rather than the geometry of instantaneous communication patterns.

We note in particular recent results by Dinitz et al. [47] who, applying the technique of *smoothed analysis* to dynamic graph problems, show that small random perturbations in the network used by Avin, Koucky, and Lotker make the dynamic random walk polynomial again. Smoothed analysis thus seems a promising avenue for exploring the performance of affine update rules when the graph adversary is not all-powerful. Given the aforementioned result, it seems likely that the exponential lower bound for the EqualNeighbor update rule would be fragile to the smoothed analysis. Under these conditions, how do the MaxWeight and EqualNeighbor rules compare?

It appears to us that the study of convex update rules has much to gain[9] from finer geometric approaches in the spirit of Charron-Bost's [29]. One obvious trail would be to look at, for example, the bounded confidence model of opinion dynamics investigated by Hegselmann and Krause [54], – which is essentially an instance of the EqualNeighbor rule over specific dynamic graphs[10] – for which, among other open problems, the convergence times observed in simulations are much smaller than known bounds. Approaches based on the finer geometric characteristics of the influence graph might help understand better the behaviors of this class of dynamical systems.

Finally, we think that looking at algorithmic problems with an asymptotic perspective may give a new light, and a renewed interest, to classic problems of distributed computing. To take a clear example: in the nearly twelve years since the publication of the Bitcoin whitepaper [75] by an unknown author only known by the pseudonym "Satoshi Nakamoto", *block chain systems* have received a sustained interest from renowed scholars [50], to promising young hackers [22], to clueless amateurs [89]. A defining feature of Bitcoin is that the integrity of its central distributed

[9] The author might be biased here: it was being exposed to a previous iteration of this material in Prof. Charron-Bost's MPRI course on the computational theory of emergent phenomena that led him on the tortuous path resulting in the present monograph.

[10] Specifically, the graph for each round t is obtained by building a geometric graph with the estimates $x_u(t)$.

data structure, the *bitcoin ledger*, is only preserved if honest participants keep extending the ledger. That is, the system as a whole is progressing towards consensus over an *infinite* data structure by agreeing with more and more certainty over larger and larger finite parts of this infinite structure.

In other words, ledger consensus is best understood as an asymptotic consensus whose finite-time behavior satisfies certain desirable properties, much like convex update rules progress towards a consensus value, which need never be reached, while displaying some guarantees of progress in finite time as expressed by their convergence time. Given the sustained interest that block chain systems have received due to their promise of making it possible to solving a reasonable approximation of traditional consensus problems over large-scale permissionless networks, we take this as an indication that asymptotic forms of consensus open the way to fruitful new approaches to perennial problems.

REFERENCES

[1] Yehuda Afek, Baruch Awerbuch, and Eli Gafni. "Applying Static Network Protocols to Dynamic Networks". In: *28th Annual Symposium on Foundations of Computer Science (Sfcs 1987)*. Oct. 1987, pp. 358–370. DOI: 10.1109/SFCS.1987.7.

[2] E. A. Akkoyunlu, K. Ekanadham, and R. V. Huber. "Some Constraints and Tradeoffs in the Design of Network Communications". In: *Proceedings of the Fifth Symposium on Operating Systems Principles - SOSP '75*. ACM Press, 1975, pp. 67–74. DOI: 10.1145/800213.806523.

[3] Steve Alpern. "The Rendezvous Search Problem". In: *SIAM Journal on Control and Optimization* 33.3 (May 1995), pp. 673–683. DOI: 10.1137/S0363012993249195.

[4] Claudio Altafini. "Consensus Problems on Networks With Antagonistic Interactions". In: *IEEE Transactions on Automatic Control* 58.4 (Apr. 2013), pp. 935–946. DOI: 10.1109/TAC.2012.2224251.

[5] Dana Angluin. "Local and Global Properties in Networks of Processors". In: *Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing - STOC '80*. Ed. by R. E. Miller, S. Ginsburg, W. A. Burkhard, and R. J. Lipton. ACM Press, 1980, pp. 82–93. DOI: 10.1145/800141.804655.

[6] Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. "Computation in Networks of Passively Mobile Finite-State Sensors". In: *Distributed Computing* 18.4 (Mar. 2006), pp. 235–253. DOI: 10.1007/s00446-005-0138-3.

[7] Dana Angluin, Michael J. Fischer, and Hong Jiang. "Stabilizing Consensus in Mobile Networks". In: *Distributed Computing in Sensor Systems*. Ed. by Phillip B. Gibbons, Tarek Abdelzaher, James Aspnes, and Ramesh Rao. Lecture Notes in Computer Science. Springer, 2006, pp. 37–50. DOI: 10.1007/11776178_3.

[8] Hagit Attiya and Jennifer Welch. *Distributed Computing*. Red. by Albert Y. Zomaya. Wiley Series on Parallel and Distributed Computing. Hoboken, NJ, USA: John Wiley & Sons, Inc., Apr. 8, 2004. ISBN: 978-0-471-45324-6. DOI: 10.1002/0471478210.

[9] Chen Avin, Michal Koucky, and Zvi Lotker. "How to Explore a Fast-Changing World". In: (Feb. 10, 2008), p. 14.

[10] Baruch Awerbuch and Shimon Even. "Efficient and Reliable Broadcast Is Achievable in an Eventually Connected Network(Extended Abstract)". In: *Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing*. PODC '84. Association for Computing Machinery, Aug. 27, 1984, pp. 278–281. DOI: 10.1145/800222.806754.

[11] Jacques Bahi, Raphaël Couturier, and Flavien Vernier. "Synchronous Distributed Load Balancing on Dynamic Networks". In: *Journal of Parallel and Distributed Computing* 65.11 (Nov. 2005), pp. 1397–1405. DOI: 10.1016/j.jpdc.2005.05.007.

[12] Tucker Balch and Ron C. Arkin. "Behavior-Based Formation Control for Multirobot Teams". In: *IEEE Transactions on Robotics and Automation* 14.6 (Dec. 1998), pp. 926–939. DOI: 10.1109/70.736776.

[13] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Belmont, Mass.: Athena Scientific, 2014. ISBN: 978-1-886529-15-1.

[14] Vincent D. Blondel, Julien M. Hendrickx, Alexander Olshevsky, and John N. Tsitsiklis. "Convergence in Multiagent Coordination, Consensus, and Flocking". In: *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE, 2005, pp. 2996–3000. DOI: 10.1109/CDC.2005.1582620.

[15] Paolo Boldi, Shella Shammah, Sebastiano Vigna, Bruno Codenotti, Peter Gemmell, Albuquerque Nm, and Janos Simon. "Symmetry Breaking in Anonymous Networks: Characterizations". In: (1996), p. 11.

[16] Paolo Boldi and Sebastiano Vigna. "Computing Anonymously with Arbitrary Knowledge". In: *Proceedings of the Eighteenth Annual ACM Symposium on Principles of Distributed Computing - PODC '99*. ACM Press, 1999, pp. 181–188. DOI: 10.1145/301308.301355.

[17] Paolo Boldi and Sebastiano Vigna. "An Effective Characterization of Computability in Anonymous Networks". In: *DISC 2001: Distributed Computing*. Ed. by Jennifer Welch. Vol. 2180. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 33–47. DOI: 10.1007/3-540-45414-4_3.

[18] Paolo Boldi and Sebastiano Vigna. "Fibrations of Graphs". In: *Discrete Mathematics* 243.1-3 (Jan. 2002), pp. 21–66. DOI: 10.1016/S0012-365X(00)00455-6.

[19] Paolo Boldi and Sebastiano Vigna. "Universal Dynamic Synchronous Self-Stabilization". In: *Distributed Computing* 15.3 (July 1, 2002), pp. 137–153. DOI: 10.1007/s004460100062.

[20] Stéphane Boucheron, Gabor Lugosi, and Pascal Massart. *Concentration Inequalities : A Non Asymptotic Theory of Independence*. First. Oxford University Press, 2013. 481 pp. ISBN: 978-0-19-174710-6. DOI: 10.1093/acprof:oso/9780199535255.001.0001/acprof-9780199535255.

[21] John Bonner Buck. "Synchronous Rhythmic Flashing of Fireflies". In: *The Quarterly Review of Biology* 13.3 (Sept. 1938), pp. 301–314. DOI: 10.1086/394562.

[22] Vitalik Buterin. *Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform*. 2013. URL: https://web.archive.org/web/20150328054135/https://github.com/ethereum/wiki/wiki/White-Paper.

[23] Joseph Campbell. *The Hero with a Thousand Faces*. 1949. ISBN: 978-1-57731-593-3.

[24] Ming Cao, A. Stephen Morse, and Brian D. O. Anderson. "Reaching a Consensus in a Dynamically Changing Environment: A Graphical Approach". In: *SIAM Journal on Control and Optimization* 47.2 (Jan. 2008), pp. 575–600. DOI: 10.1137/060657005.

[25] Ming Cao, A. Stephen Morse, and Brian D. O. Anderson. "Reaching a Consensus in a Dynamically Changing Environment: Convergence Rates, Measurement Delays, and Asynchronous Events". In: *SIAM Journal on Control and Optimization* 47.2 (Jan. 2008), pp. 601–623. DOI: 10.1137/060657029.

[26] Arnaud Casteigts, Paola Flocchini, Emmanuel Godard, Nicola Santoro, and Masafumi Yamashita. "On the Expressivity of Time-Varying Graphs". In: *Theoretical Computer Science*. Fundamentals of Computation Theory 590 (July 26, 2015), pp. 27–37. DOI: 10.1016/j.tcs.2015.04.004.

[27] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. "Time-Varying Graphs and Dynamic Networks". In: *International Journal of Parallel, Emergent and Distributed Systems* 27.5 (Oct. 1, 2012), pp. 387–408. DOI: 10.1080/17445760.2012.668546.

[28] Bernadette Charron-Bost. *Orientation and Connectivity Based Criteria for Asymptotic Consensus*. Mar. 8, 2013. arXiv: 1303.2043 [cs].

[29] Bernadette Charron-Bost. *Geometric Bounds for Convergence Rates of Averaging Algorithms*. July 9, 2020. arXiv: 2007.04837 [cs.MA].

[30] Bernadette Charron-Bost, Matthias Függer, and Thomas Nowak. "Approximate Consensus in Highly Dynamic Networks: The Role of Averaging Algorithms". In: (Nov. 12, 2014). arXiv: 1408.0620 [cs].

[31] Bernadette Charron-Bost, Matthias Függer, and Thomas Nowak. "Approximate Consensus in Highly Dynamic Networks: The Role of Averaging Algorithms". In: *Automata, Languages, and Programming. ICALP 2015*. Ed. by Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann. Vol. 9135. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2015, pp. 528–539. DOI: 10.1007/978-3-662-47666-6_42.

[32] Bernadette Charron-Bost and Patrick Lambein-Monette. *Randomization and Quantization for Average Consensus*. Apr. 29, 2018. arXiv: 1804.10919 [cs.MA].

[33] Bernadette Charron-Bost and Patrick Lambein-Monette. "Randomization and Quantization for Average Consensus". In: *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, Dec. 2018, pp. 3716–3721. DOI: 10.1109/CDC.2018.8619817.

[34] Bernadette Charron-Bost and Patrick Lambein-Monette. *Average Consensus: A Little Learning Goes A Long Way*. Oct. 12, 2020. arXiv: 2010.05675 [cs].

[35] Bernadette Charron-Bost and Shlomo Moran. *MinMax Algorithms for Stabilizing Consensus*. June 21, 2019. arXiv: 1906.09073 [cs].

[36] Bernadette Charron-Bost and Shlomo Moran. "The Firing Squad Problem Revisited". In: *Theoretical Computer Science* 793 (Nov. 2019), pp. 100–112. DOI: 10.1016/j.tcs.2019.07.023.

[37] Bernadette Charron-Bost and André Schiper. "The Heard-Of Model: Computing in Distributed Systems with Benign Faults". In: *Distributed Computing* 22.1 (Apr. 2009), pp. 49–71. DOI: 10.1007/s00446-009-0084-6.

[38] Samprit Chatterjee. "Reaching a Consensus: Some Limit Theorems". In: *Proceedings of the 40th Session of the International Statistical Institute, Warsaw, Poland, 1975*. Vol. 3. 1975, pp. 156–160.

[39] Samprit Chatterjee and Eugene Seneta. "Towards Consensus: Some Convergence Theorems on Repeated Averaging". In: *Journal of Applied Probability* 14.1 (Mar. 1977), pp. 89–97. DOI: 10.2307/3213262. JSTOR: 3213262.

[40] Bernard Chazelle. "The Total S-Energy of a Multiagent System". In: *SIAM Journal on Control and Optimization* 49.4 (Jan. 2011), pp. 1680–1706. DOI: 10.1137/100791671.

[41] Felipe Cucker and Steve Smale. "Emergent Behavior in Flocks". In: *IEEE Transactions on Automatic Control* 52.5 (May 2007), pp. 852–862. DOI: 10.1109/TAC.2007.895842.

[42] George Cybenko. "Dynamic Load Balancing for Distributed Memory Multiprocessors". In: *Journal of Parallel and Distributed Computing* 7.2 (Oct. 1989), pp. 279–301. DOI: 10.1016/0743-7315(89)90021-X.

[43] Ariel Daliot, Danny Dolev, and Hanna Parnas. "Self-Stabilizing Pulse Synchronization Inspired by Biological Pacemaker Networks". In: *Self-Stabilizing Systems*. Ed. by Shing-Tsaan Huang and Ted Herman. Red. by Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen. Vol. 2704. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, pp. 32–48. DOI: 10.1007/3-540-45032-7_3.

[44] Charles Darwin. *On the Origin of Species By Means of Natural Selection Or, the Preservation of Favoured Races in the Struggle for Life*. First. John Murray, 1859. 502 pp. URL: https://www.gutenberg.org/ebooks/1228.

[45] Steve Deering and R. Hinden. *Internet Protocol, Version 6 (IPv6) Specification*. STD 86. RFC Editor, July 2017. DOI: 10.17487/RFC8200.

[46] Morris H. DeGroot. "Reaching a Consensus". In: *Journal of the American Statistical Association* 69.345 (Mar. 1974), pp. 118–121. DOI: 10.2307/2285509. JSTOR: 2285509.

[47] Michael Dinitz, Jeremy T. Fineman, Seth Gilbert, and Calvin Newport. "Smoothed Analysis of Dynamic Networks". In: *Distributed Computing* 31.4 (Aug. 2018), pp. 273–287. DOI: 10.1007/s00446-017-0300-8.

[48] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. "Impossibility of Distributed Consensus with One Faulty Process". In: *Journal of the ACM* 32.2 (Apr. 1, 1985), pp. 374–382. DOI: 10.1145/3149.214121.

[49] Matthias Függer, Thomas Nowak, and Kyrill Winkler. "On the Radius of Nonsplit Graphs and Information Dissemination in Dynamic Networks". In: *Discrete Applied Mathematics* 282 (Aug. 15, 2020), pp. 257–264. DOI: 10.1016/j.dam.2020.02.013.

[50] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. "Algorand: Scaling Byzantine Agreements for Cryptocurrencies". In: *Proceedings of the 26th Symposium on Operating Systems Principles*. SOSP '17. Association for Computing Machinery, Oct. 14, 2017, pp. 51–68. DOI: 10.1145/3132747.3132757.

[51] Jim N. Gray. "Notes on Data Base Operating Systems". In: *Operating Systems*. Ed. by R. Bayer, R. M. Graham, and G. Seegmüller. Red. by G. Goos, J. Hartmanis, P. Brinch Hansen, D. Gries, C. Moler, G. Seegmüller, J. Stoer, and N. Wirth. Vol. 60. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1978, pp. 393–481. DOI: 10.1007/3-540-08755-9_9.

[52] Frank Harary and Ronald C. Read. "Is the Null-Graph a Pointless Concept?" In: *Graphs and Combinatorics*. Ed. by Ruth A. Bari and Frank Harary. Red. by A. Dold and B. Eckmann. Vol. 406. Lecture Notes in Mathematics. Springer Berlin Heidelberg, 1974, pp. 37–44. DOI: 10.1007/BFb0066433.

[53] Wilfred Keith Hastings. "Monte Carlo Sampling Methods Using Markov Chains and Their Applications". In: *Biometrika* 57 (1 Apr. 1, 1970), pp. 97–109. DOI: 10.1093/biomet/57.1.97.

[54] Rainer Hegselmann and Ulrich Krause. "Opinion Dynamics and Bounded Confidence Models, Analysis and Simulation". In: *Journal of Artificial Societies and Social Simulation* 5.3 (June 30, 2002). URL: https://econpapers.repec.org/article/jasjasssj/2002-5-2.htm.

[55] 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*. July 2006, pp. 1229–1240. URL: http://hdl.handle.net/2078.1/91888.

[56] Julien M. Hendrickx, Alex Olshevsky, and John N. Tsitsiklis. "Distributed Anonymous Discrete Function Computation". In: *IEEE Transactions on Automatic Control* 56.10 (Oct. 2011), pp. 2276–2289. DOI: 10.1109/TAC.2011.2163874.

[57] Julien M. Hendrickx and John N. Tsitsiklis. "Fundamental Limitations for Anonymous Distributed Systems with Broadcast Communications". In: *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, Sept. 2015, pp. 9–16. DOI: 10.1109/ALLERTON.2015.7446980.

[58] Juho Hirvonen and Jukka Suomela. *Distributed Algorithms 2020*. 2020. 221 pp. URL: https://jukkasuomela.fi/da2020.

[59] John D. Hunter. "Matplotlib: A 2D Graphics Environment". In: *Computing in Science Engineering* 9.3 (May 2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[60] Ali Jadbabaie, Jie Lin, and A. Stephen Morse. "Coordination of Groups of Mobile Autonomous Agents Using Nearest Neighbor Rules". In: *IEEE Transactions on Automatic Control* 48.6 (June 2003), pp. 988–1001. DOI: 10.1109/TAC.2003.812781.

[61] David Kempe, Alin Dobra, and Johannes Gehrke. "Gossip-Based Computation of Aggregate Information". In: *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.* IEEE Computer. Soc, 2003, pp. 482–491. DOI: 10.1109/SFCS.2003.1238221.

[62] David Kempe, Jon Kleinberg, and Amit Kumar. "Connectivity and Inference Problems for Temporal Networks". In: *Journal of Computer and System Sciences* 64.4 (June 1, 2002), pp. 820–842. DOI: 10.1006/jcss.2002.1829.

[63] Fabian Kuhn, Nancy A. Lynch, and Rotem Oshman. "Distributed Computation in Dynamic Networks". In: *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*. STOC '10. Association for Computing Machinery, June 5, 2010, pp. 513–522. DOI: 10.1145/1806689.1806760.

[64] Fabian Kuhn, Yoram Moses, and Rotem Oshman. "Coordinated Consensus in Dynamic Networks". In: *Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*. PODC '11. Association for Computing Machinery, June 6, 2011, pp. 1–10. DOI: 10.1145/1993806.1993808.

[65] Fabian Kuhn and Rotem Oshman. "Dynamic Networks: Models and Algorithms". In: *ACM SIGACT News* 42.1 (Mar. 21, 2011), p. 82. DOI: 10.1145/1959045.1959064.

[66] Leslie Lamport, Robert Shostak, and Marshall Pease. "The Byzantine Generals Problem". In: *ACM Transactions on Programming Languages and Systems* 4.3 (1982), p. 20.

[67] Matthieu Latapy, Tiphaine Viard, and Clémence Magnien. "Stream Graphs and Link Streams for the Modeling of Interactions over Time". In: *Social Network Analysis and Mining* 8.1 (Oct. 3, 2018), p. 61. DOI: 10.1007/s13278-018-0537-7.

[68] Riccardo Lucchese and Damiano Varagnolo. "Average Consensus via Max Consensus". In: *IFAC-PapersOnLine*. 5th IFAC Workshop on Distributed Estimation and Control in Networked Systems NecSys 2015 48.22 (Jan. 1, 2015), pp. 58–63. DOI: 10.1016/j.ifacol.2015.10.307.

[69] Nancy A. Lynch. *Distributed Algorithms*. The Morgan Kaufmann Series in Data Management Systems. San Francisco, Calif: Morgan Kaufmann, 1997. 872 pp. ISBN: 978-1-55860-348-6.

[70] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. *Equation of State Calculations by Fast Computing Machines*. AECU-2435; LADC-1359. Los Alamos Scientific Lab., Los Alamos, NM (United States); Univ. of Chicago, IL (United States), Mar. 6, 1953. DOI: 10.2172/4390578.

[71] Renato E. Mirollo and Steven H. Strogatz. "Synchronization of Pulse-Coupled Biological Oscillators". In: *SIAM Journal on Applied Mathematics* 50.6 (Dec. 1990), pp. 1645–1662. DOI: 10.1137/0150098.

[72] Luc Moreau. "Stability of Multiagent Systems with Time-Dependent Communication Links". In: *IEEE Transactions on Automatic Control* 50.2 (Feb. 2005), pp. 169–182. DOI: 10.1109/TAC.2004.841888.

[73] Damon Mosk-Aoyama and Devavrat Shah. "Computing Separable Functions via Gossip". In: *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing*. PODC '06. Association for Computing Machinery, July 23, 2006, pp. 113–122. DOI: 10.1145/1146381.1146401.

[74] S. Muthukrishnan. "First- and Second-Order Diffusive Methods for Rapid, Coarse, Distributed Load Balancing". In: *Theory of Computing Systems* 31.4 (July 1, 1998), pp. 331–354. DOI: 10.1007/s002240000092.

[75] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2008-12.

[76] Angelia Nedić and Ji Liu. "On Convergence Rate of Weighted-Averaging Dynamics for Consensus Problems". In: *IEEE Transactions on Automatic Control* 62.2 (Feb. 2017), pp. 766–781. DOI: 10.1109/TAC.2016.2572004.

[77] Angelia Nedić, Alex Olshevsky, and Michael G. Rabbat. "Network Topology and Communication-Computation Tradeoffs in Decentralized Optimization". In: *Proceedings of the IEEE* 106.5 (May 2018), pp. 953–976. DOI: 10.1109/JPROC.2018.2817461.

[78] Angelia Nedić, Alexander Olshevsky, Asuman Ozdaglar, and John N. Tsitsiklis. "On Distributed Averaging Algorithms and Quantization Effects". In: *IEEE Transactions on Automatic Control* 54.11 (Nov. 2009), pp. 2506–2517. DOI: 10.1109/TAC.2009.2031203.

[79] Regina O'Dell and Roger Wattenhofer. "Information Dissemination in Highly Dynamic Graphs". In: *Proceedings of the 2005 Joint Workshop on Foundations of Mobile Computing*. DIALM-POMC '05. Association for Computing Machinery, Sept. 2, 2005, pp. 104–110. DOI: 10.1145/1080810.1080828.

[80] Alex Olshevsky. "Linear Time Average Consensus and Distributed Optimization on Fixed Graphs". In: *SIAM Journal on Control and Optimization* 55.6 (Jan. 2017), pp. 3990–4014. DOI: 10.1137/16M1076629.

[81] Alex Olshevsky and John N. Tsitsiklis. "Convergence Speed in Distributed Consensus and Averaging". In: *SIAM Journal on Control and Optimization* 48.1 (Jan. 2009), pp. 33–55. DOI: 10.1137/060678324.

[82] Alex Olshevsky and John N. Tsitsiklis. "A Lower Bound for Distributed Averaging Algorithms on the Line Graph". In: *IEEE Transactions on Automatic Control* 56.11 (Nov. 2011), pp. 2694–2698. DOI: 10.1109/TAC.2011.2159652.

[83] Alex Olshevsky and John N. Tsitsiklis. "Convergence Speed in Distributed Consensus and Averaging". In: *SIAM Review* 53.4 (Jan. 2011), pp. 747–772. DOI: 10.1137/110837462.

[84] Alex Olshevsky and John N. Tsitsiklis. "Degree Fluctuations and the Convergence Time of Consensus Algorithms". In: *IEEE Transactions on Automatic Control* 58.10 (Oct. 2013), pp. 2626–2631. DOI: 10.1109/TAC.2013.2257969.

[85] Rotem Oshman. "Distributed Computation in Wireless and Dynamic Networks". phd. Cambridge, Massachusetts, USA: Massachusetts Institute of Technology, Sept. 2012. 221 pp. URL: https://dspace.mit.edu/handle/1721.1/78456.

[86] David Pascucci, Maria Rubega, and Gijs Plomp. "Modeling Time-Varying Brain Networks with a Self-Tuning Optimized Kalman Filter". In: *PLOS Computational Biology* 16.8 (Aug. 17, 2020). Ed. by Francesco P. Battaglia, e1007566. DOI: 10.1371/journal.pcbi.1007566.

[87] Marshall Pease, Robert Shostak, and Leslie Lamport. "Reaching Agreement in the Presence of Faults". In: *Journal of the ACM (JACM)* 27.2 (Apr. 1980), pp. 228–234. DOI: 10.1145/322186.322188.

[88] David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. USA: Society for Industrial and Applied Mathematics, 2000. ISBN: 978-0-89871-464-7.

[89] Twitter Pundits. *"Bitcoin solves the Byzantine Generals' Problem"*. URL: https://twitter.com/search?q=%22bitcoin%20solves%20the%20byzantine%20generals%e2%80%99%20problem%22 (visited on 10/27/2020).

[90] Wei Ren. "Consensus Strategies for Cooperative Control of Vehicle Formations". In: *IET Control Theory & Applications* 1.2 (Mar. 1, 2007), pp. 505–512. DOI: 10.1049/iet-cta:20050401.

[91] Craig W. Reynolds. "Flocks, Herds, and Schools: A Distributed Behavioral Model". In: *Computer Graphics* 21.4 (July 1987), p. 10.

[92] Nicola Santoro and Peter Widmayer. "Time Is Not a Healer: Preliminary Version". In: *STACS 89*. Ed. by B. Monien and R. Cori. Red. by G. Goos, J. Hartmanis, D. Barstow, W. Brauer, P. Brinch Hansen, D. Gries, D. Luckham, C. Moler, A. Pnueli, G. Seegmüller, J. Stoer, and N. Wirth. Vol. 349. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1989, pp. 304–313. DOI: 10.1007/BFb0028994.

[93] Eugene Seneta. *Non-Negative Matrices and Markov Chains*. Springer Series in Statistics. New York, NY: Springer New York, 1981. ISBN: 978-0-387-29765-1. DOI: 10.1007/0-387-32792-4.

[94] Ichiro Suzuki and Masafumi Yamashita. "Distributed Anonymous Mobile Robots: Formation of Geometric Patterns". In: *SIAM Journal on Computing* 28.4 (Jan. 1999), pp. 1347–1363. DOI: 10.1137/S009753979628292X.

[95] Gerard Tel. *Introduction to Distributed Algorithms*. 2nd ed. Cambridge University Press, 2000. ISBN: 978-0-521-79483-1. DOI: 10.1017/CBO9781139168724.

[96] John N. Tsitsiklis, Dimitri P. Bertsekas, and Michael Athans. "Distributed Asynchronous Deterministic and Stochastic Gradient Optimization Algorithms". In: *IEEE Transactions on Automatic Control* 31.9 (Sept. 1986), pp. 803–812. DOI: 10.1109/TAC.1986.1104412.

[97] John Nikolaos Tsitsiklis. "Problems in Decentralized Decision Making and Computation". phd. Cambridge, Massachusetts, USA: Massachusetts Institute of Technology, Nov. 1984. 271 pp. URL: https://dspace.mit.edu/handle/1721.1/15254.

[98] Tamás Vicsek, András Czirók, Eshel Ben-Jacob, Inon Cohen, and Ofer Shochet. "Novel Type of Phase Transition in a System of Self-Driven Particles". In: *Physical Review Letters* 75.6 (Aug. 7, 1995), pp. 1226–1229. DOI: 10.1103/PhysRevLett.75.1226.

[99] Lin Xiao and Stephen Boyd. "Fast Linear Iterations for Distributed Averaging". In: *Systems & Control Letters* 53.5 (Sept. 2004), pp. 65–78. DOI: 10.1016/j.automatica.2013.02.015.

[100] Lin Xiao, Stephen Boyd, and Sanjay Lall. "A Scheme for Robust Distributed Sensor Fusion Based on Average Consensus". In: *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.* IEEE, 2005, pp. 63–70. DOI: 10.1109/IPSN.2005.1440896.

[101] Masafumi Yamashita and Tsunehiko Kameda. "Computing on Anonymous Networks". In: (1988), p. 14.

[102] Masafumi Yamashita and Tsunehiko Kameda. "Computing on Anonymous Networks: Part I-Characterizing the Solvable Cases". In: *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS* 7.1 (1996), p. 21.

[103] Masafumi Yamashita and Tsunehiko Kameda. "Computing on Anonymous Networks: Part II-Decision and Membership Problems". In: *IEEE Transactions on Parallel and Distributed Systems* 7 (Jan. 1996).

ECOLE
DOCTORALE

Titre : Consensus de moyenne dans les réseaux dynamiques anonymes

Mots clés : algorithmes distribués; réseaux dynamiques; consensus de moyenne; consensus asymptotique

Résumé : L'avènement de composants électroniques compacts et bon marché présage d'une diversification rapide d'applications dans lesquelles des agents autonomes en réseau travaillent à réaliser un objectif commun. Ces tâches complexes, en dépit de leur diversité, dépendent de la maîtrise d'un petit nombre de primitives de coordination, dont l'implémentation programmatique par des agents à faible puissance et capacité calculatoire constitue l'un des enjeux majeurs du développement de telles applications réparties.
Parmi ces dernières, citons par exemple la coordination du mouvement de réseaux mobiles et véhiculaires, l'aggrégation et le traitement distribué de mesures relevées par des réseaux de capteurs, et le a répartition de charge en temps réel au sein d'un réseau fournissant un service à grande échelle. L'implémentation distribuée de telles primitives se doit de répondre à différentes contraintes, qui ne résultent pas toutes de la nature numérique des entités constitutives du réseau ; en conséquence, l'étude théorique de ces primitives s'applique à la modélisation de comportements complexes de systèmes étudiés par les sciences naturelles, tels que les mouvements collectifs animaliers ou le système nerveux.
Cette monographie traite spécifiquement d'algorithmes distribués qui réalisent le calcul asymptotique de la moyenne de valeurs initialement détenues par les agents d'un réseau dont les liens de communication sont amenés à changer au cours du temps, ceci en l'absence de coordination centralisée. Ces algorithmes doivent être implémentables localement, en n'exploitant que l'information qui peut être collectée par les agents lors de leurs interactions sur le réseau, et en l'absence de mécanisme particulier pour marquer le départ, tel qu'un signal global ou un agent initiateur.
Nous développons des algorithmes qui réalisent un tel consensus de moyenne sur des réseaux dynamiques présentant certaines propriétés locales. Ces algorithmes sont simples à décrire, légers à implémenter, et opèrent en temps polynomial en le nombre d'agents.
Sur des réseaux présentant des interactions bidirectionnelles, nous fournissons un algorithme déterministe qui réalise le calcul asymptotique de la moyenne dès lors que le réseau ne se sépare jamais de façon permanente. Pour le cas plus général d'interactions asymétriques, nous présentons un algorithme Monte Carlo stabilisant qui est efficace en termes de complexité spatiale et opère en temps linéaire. Ce dernier algorithme admet une extension dont les exécutions terminent en tolérant un départ asynchrone des agents.
Nos algorithmes sont à considérer en regard de résultats et de méthodes qui reposent sur une information globale fournie externalement aux agents, sur des hypothèses de brisure initiale de symétrie, ou qui exploitent une topologie particulière et ne se généralisent pas à des réseaux quelconques. Dans ce contexte, nous contribuons des algorithmes dont les conditions de validité sont purement locales dans le temps et l'espace : pour le modèle d'interactions bidirectionnelles, nous montrons que le calcul asymptotique de la moyenne est réalisable par des agents déterministes, là où pour le modèle général nous fournissons des algorithmes randomisés dont les performances asymptotiques sont bien meilleures que celles de protocoles à information complète et robustes aux départs asynchrones.
Par-delà l'intérêt immédiat à l'obtention d'algorithmes efficaces implémentables, notre étude s'inscrit dans un effort de cartographie des limites que la localité des interactions impose aux applications réparties.

Title : Average consensus in anonymous dynamic networks

Keywords : distributed algorithms; dynamic networks; average consensus; asymptotic consensus

Abstract : Compact and cheap electronic components announce the near-future development of applications in which networked systems of autonomous agents are made to carry over complex tasks. These, in turn, depend on a small number of coordination primitives, which need to be programmatically implemented into potentially low-powered, and computationally limited, agents.
Such applications include for example the coordination of the collective motion of mobile and vehicular networks, the distributed aggregation and processing of data measured locally in sensor networks, and the on-line repartition of processing load in the computer farms powering wide-scale services. As they address constraints that are not specific to the digital nature of the network such primitives also serve to model complex behavior of natural systems, such as flocks and neural networks.
This monograph focuses on providing distributed algorithms that asymptotically compute the average of initial values, initially present at each agent of a networked system with time-varying communication links and in the absence of centralized control. Additionally, we consider the weaker problem of getting the agents to asymptotically agree on any value within the initial bounds. We focus on locally implementable algorithms, which leverage no information beyond what the agents can acquire by themselves, and which need no bootstrapping mechanism like a global start signal or a leader agent.
We provide distributed average consensus algorithms that operate over dynamic networks given different local assumptions. These algorithms are computationally simple and operate in polynomial time in the number of agents.
For bidirectional communications, we give a deterministic algorithm which asymptotically computes the average as long as the network never becomes permanently disconnected. For the general case of asymmetric communications, we provide a stabilizing Monte Carlo algorithm that is efficient in bandwidth and memory and operates in linear time, along with an extension by which the algorithm can be made to uniformly terminate over any connected network in which agents may start asynchronously.
This contrasts with a plethora of results and techniques in which agents are provided external information – the size of the system, a bound over their degree, – helped with exogenous symmetry breaking – a leader agent, unique identifiers, – or where the network is expected to conform to a specific shape – a ring, a a complete network, a regular graph. Indeed, because very different networks may look alike to the agents, they are limited in what they can learn locally, and many functions are impossible to compute in a fully distributed manner without assuming some structure in the network or additional symmetry-breaking device. Given these stringent constraints, our contribution is to offer algorithms whose validity depends uniquely on local and instantaneous conditions. In the bidirectional model, we show that anonymous deterministic agents can asymptotically compute the average in polynomial time. For the general model of directed interactions, we allow agents to consult random oracles. Under those conditions, full information protocols are capable of solving any problem, and so we focus on the spatial complexity and tolerance to a lack of initial coordination in the agents, while offering stronger termination guarantees than in the bidirectional case.
Beyond the fact that locally implementable algorithms are eminently desirable, our study contributes to mapping the limits that local interactions impose on networks.

Institut Polytechnique de Paris
91120 Palaiseau, France