université
PARIS-SACLAY

Agreement Problems in Dynamic Networks

Habilitation à diriger des recherches de l'Université Paris-Saclay

présentée et soutenue à Gif-sur-Yvette, le 24/1/2022, par

Thomas NOWAK

Composition du jury

Arnaud CASTEIGTS Maître de conférences, U. Bordeaux	Rapporteur
Petr KUZNETSOV Professeur, Télécom Paris	Rapporteur
Jennifer WELCH Professeure, Texas A&M University	Rapportrice
Joffroy BEAUQUIER Professeur émérite, U. Paris-Saclay	Examinateur
Patricia BOUYER-DECITRE Directrice de recherche, CNRS	Examinatrice
Pierre FRAIGNIAUD Directeur de recherche, CNRS	Examinateur

Habilitation à diriger des recherches

Contents

Chapter 1

Introduction

The present manuscript contains a summary of a part of my research activities after completion of my PhD thesis. Its goal is not to present the technical details of the work, but rather the underlying ideas and the context.

It contains a selection of connected research topics that revolve around a common theme: reaching agreement in networks that are marked by a high dynamicity in their communication capabilities. More specifically, the manuscript treats nine research papers published in the time frame from 2015 to 2020, which are listed as [P1]–[P9] at the end of this section. The choice of which papers to include in the manuscript is due to their coherence and synergy. Deliberately excluded is my work on VLSI design [71, 68, 75, 97, 67], clock synchronization [70, 80, 18], and discrete-event systems [30, 99, 90], as well as unpublished preprints [41, 33].

Dynamic networks are studied in a variety of practical and theoretical applications. They naturally occur in wireless networking [16, 17], where communication links go up and down depending on a number of external factors. These external factors (noise, interference, obstacles, mobility, *etc.*) can either be modeled explicitly or abstracted away. Due to their complexity, the feasibility of mathematical reasoning in models that include such factors is limited, however. An alternative to explicit modeling is to only focus on whether a process can successfully send a message to another process at a given point in time. This end-to-end capability of communicating entails favorable conditions regarding noise, interference, mobility, scheduling, available battery charge, *etc.*

In such an abstract model, the dynamicity of the communication capabilities is expressed as a time-varying graph [38]. The vertices of the graph are the processes involved in the distributed computation. The graph contains an edge from one process to another at a given time if message transmission is possible. An environment, or adversary, in the abstract model is described by a set of admissible time-varying graphs. The abstract model cannot capture all phenomena observable in a fine-grained physical model that takes

into account the various external factors contributing to successful message transmission. Thus, to be conservative, the scenarios described by the time-varying graphs of the abstract model will be an over-approximation of the important scenarios of a fine-grained model.

The usefulness of dynamic-network models exceeds applications in wireless networking. They are also applicable for other modes of computation with uncertain communication, such as wired—even on-chip—fault-tolerant computing [95, 14, 74, 110], the Internet [57], epidemics and sociological phenomena [83, 53], and communication between macro- and microbiological entities [29, 111]. In particular, it turns out that a wide range of classical models from theoretical fault-tolerant distributed computing can be expressed as dynamic networks [42]. In many cases, even uncertainty about processes failing can be reformulated as uncertainty purely about communication.

Even though dynamic networks have a very wide applicability and generality, historically they have not been thoroughly studied. In fact, targeted formal investigations have begun only relatively recently [91, 37]. Although there is growing interest in the community lately [15, 3, 48, 125], much work is left to be done to lift the state of the art on dynamic networks to a satisfactory level. A large part of my research, in particular the work presented in this manuscript, is an attempt to move closer to this goal.

Agreement problems come in different variants. From a theoretical perspective, they serve as a benchmark for the strength of distributed-computing models. From a practical perspective, they are often identified as necessary or useful sub-tasks for the implementation of a given system.

The most-studied agreement problem in distributed computing is likely the consensus problem [93]. Here, all processes seek to agree on a single value. The strong agreement guaranteed by solutions to the consensus problem make it useful for applications like database and state-machine replication [118]. For other applications, having exact consensus among all processes is not necessary, making the overhead to solve the consensus problem excessive. It also turns out that exact consensus is impossible to achieve under relatively mild fault assumptions [65].

A large part of this manuscript is devoted to asymptotic consensus, which is a weaker but still useful problem. Chapter 3 treats the content of papers [P1]–[P5]. Therein we give a quite complete analysis of the asymptotic and approximate consensus problems in the class of *oblivious* network models. We present both lower-bound proofs and matching upper bounds via explicit algorithms for solvability and time complexity. We also discuss space complexity and the case of higher-dimensional and structured data.

Chapter 4 contains the results of paper [P6] about the exact consensus problem. We present the first characterization of solvability of exact consensus in general dynamic network models. The characterization is in terms of topological properties of the space of executions. For this, we introduce a new topology on the execution space. Our point-set-topological approach is

distinct to established approaches for solvability characterizations in various distributed-computing models that use combinatorial topology [84].

In Chapter 5 we discuss paper [P7], which treats a fundamental question in dynamic networks. It presents an improved bound on the number of rooted communication graphs necessary to generate a non-split communication graph. Both classes of communication graphs appear in the analysis of asymptotic and exact consensus.

Chapters 6 and 7 work in a different model for dynamic networks than the earlier chapters. Chapter 6 treats paper [P8] on data gathering in population protocols. It is one of the only works on population protocols with a *non-uniform* random scheduler. The analysis requires new and non-standard methods. Chapter 7 discusses paper [P9], which treats the problem of reaching majority consensus in populations in which all agents continually duplicate. The motivation comes from computation with synthetically modified bacteria. The results are formalized in the standard model for chemical reaction networks.

[P1] Bernadette Charron-Bost, Matthias Függer, and Thomas Nowak. Approximate consensus in highly dynamic networks: the role of averaging algorithms. In: Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann (eds.) *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP 2015)*, pp. 528–539. Springer, Heidelberg, 2015.

[P2] Bernadette Charron-Bost, Matthias Függer, and Thomas Nowak. Fast, robust, quantizable approximate consensus. In: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi (eds.) *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, pp. 137:1–137:14. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, 2016.

[P3] Matthias Függer, Thomas Nowak, and Manfred Schwarz. Tight bounds for asymptotic and approximate consensus. In: Idit Keidar (ed.) *Proceedings of the 37th ACM Symposium on Principles of Distributed Computing (PODC 2018)*, pp. 325–334. Association for Computing Machinery, New York, 2018.

[P4] Matthias Függer and Thomas Nowak. Fast multidimensional asymptotic and approximate consensus. In: Ulrich Schmid (ed.) *Proceedings of the 32nd International Symposium on Distributed Computing (DISC 2018)*, pp. 27:1–27:16. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, 2018.

[P5] Thomas Nowak and Joel Rybicki. Byzantine approximate agreement on graphs. In: Jukka Suomela (ed.) *Proceedings of the 33rd Interna-*

tional Symposium on Distributed Computing (DISC 2019), pp. 29:1–29:17. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, 2019.

[P6] Thomas Nowak, Ulrich Schmid, and Kyrill Winkler. Topological characterization of consensus under general message adversaries. In: Faith Ellen (ed.) *Proceedings of the 38th ACM Symposium on Principles of Distributed Computing (PODC 2019)*, pp. 218–227. Association for Computing Machinery, New York, 2019.

[P7] Matthias Függer, Thomas Nowak, and Kyrill Winkler. On the radius of nonsplit graphs and information dissemination in dynamic networks. *Discrete Applied Mathematics* 282:257–264, 2020.

[P8] Chuan Xu, Joffroy Beauquier, Janna Burman, Shay Kutten, and Thomas Nowak. Data collection in population protocols with non-uniformly random scheduler. *Theoretical Computer Science* 806:516–530, 2020.

[P9] Da-Jung Cho, Matthias Függer, Corbin Hopper, Manish Kushwaha, Thomas Nowak, and Quentin Soubeyran. Distributed computation with continual population growth. In: Hagit Attiya (ed.) *Proceedings of the 34th International Symposium on Distributed Computing (DISC 2020)*, pp. 7:1–7:17. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, 2020.

Chapter 2

Dynamic Network Models

In this manuscript we use a general synchronous message-passing model with communication links that can change over time. The set of participating processes is assumed to remain the same over the course of the execution, but no *a priori* restrictions are placed on the evolution of the communication links. In particular, no stability assumptions of any kind are necessarily in place. Of course, in order to solve any non-trivial problem, some restrictions on the communication links have to be made. Questions of interest are the restrictions necessary to solve a given problem and the restrictions that allow it to be solved efficiently.

This kind of dynamic synchronous model was used early on by Santoro and Widmayer [115]. They showed that the consensus problem is unsolvable in some synchronous message-passing models, namely in those in which up to $n-1$ arbitrary messages can be lost each round. This first step was expanded upon considerably in the intervening years. In fact, it was shown that the general dynamic synchronous message-passing model can model not only link faults in synchronous systems [91], but also asynchronous message-passing models, as well as shared-memory models. These works include the round-by-round fault detectors (RRFD) [76], the perception-based fault model [116], the General Round-based Algorithm Framework (GIRAF) [88], and the Heard-Of model [42]. The model coincides with the time-varying graph (TVG) model [38] with the discrete temporal domain $\mathbb{N} = \{1, 2, 3, \dots\}$ and constant latency function $\zeta(e, t) = 1$ for all edges e and all times t. Recent publications often employ the term message adversary (MA) [3] to denote the model.

2.1 Communication Graphs

The letter n will denote the number of processes throughout the manuscript. We thus write $[n] = \{1, 2, \dots, n\}$ for the set of processes.

A *communication graph* is a digraph[1] with vertex set $V = [n]$ that contains all self-loops (p, p) for $p \in [n]$ as edges. The fact that communication graphs contain self-loops is tantamount to processes not forgetting their state. For the most part we will not be concerned with the issue of memory complexity or message sizes. Algorithms are thus free to encode their complete history into all messages they send. We will, however, make use of algorithms that encode much less information.

It is often useful to talk about indirect, or transitive, communication between processes. This is captured by the notion of the product of communication graphs defined by setting $G \circ H$ to contain those edges that can be obtained by choosing one edge in G followed by one edge in H, *i.e.*,

$$(p, q) \in E(G \circ H) \iff \exists r \in [n] \colon (p, r) \in E(G) \wedge (r, q) \in E(H) \ .$$

In particular, since communication graphs contain self-loops, both G and H are sub-digraphs of $G \circ H$. The product of two communication graphs is thus also a communication graph because self-loops are preserved.

The product operation is associative but not commutative, not even for undirected graphs. Moreover, the product of two undirected graphs need not be undirected. The identity element is the communication graph that contains all self-loops but no other edges. The set of communication graphs with the operation $\circ$ thus forms a non-commutative monoid.

2.2 Communication Patterns and Models

Because the communication links between processes can change over time, we will study sequences of communication graphs when studying executions of algorithms. We will refer to infinite sequences $(G_t)_{t \in \mathbb{N}}$ of communication graphs as *communication patterns*.

The notion of communication pattern thus corresponds to what is called a schedule in some other models. Indeed, the exact choice of communication pattern will be in the hands of the adversary. The adversary is constrained, however, by the specific *model*, which is defined as a set of admissible communication patterns.

A particular, simpler, way of defining a model is by specifying the set of communication graphs that are allowed. The admissible communication patterns are then those that contain only the allowed communication graphs, *i.e.*, given a set $\mathcal{G}$ of communication graphs, the model generated by $\mathcal{G}$ is equal to:

$$\mathcal{M}(\mathcal{G}) = \{(G_t)_{t \in \mathbb{N}} \mid \forall t \in \mathbb{N} \colon G_t \in \mathcal{G}\}$$

Of course, not all models are defined by such a specific safety property. We will refer to them as *oblivious* models.

[1] directed graph

2.3 Broadcastable Models

The most stringent restriction to put on a communication graph is for it to be complete, *i.e.*, to include all possible edges between pairs of processes. One step down from this requirement is for the communication graph to contain a *broadcaster*: a process that has outgoing edges to all other processes. We will call a communication graph *broadcastable* if it contains a broadcaster. Likewise, we will call a model *broadcastable* if all of its communication graphs are.

For certain problems, like exact consensus or explicit leader election, it is necessary for the initial value or identifier of one process to be relayed to all others. For these problems at least, some form of (transitive) broadcastability is inevitable.

2.4 Non-Split Models

A communication graph $G = ([n], E)$ is *non-split* if every pair of processes has a common incoming neighbor, *i.e.*, if

$$\forall p, q \in [n]\ \exists r \in [n]\colon\ (r, p) \in E \wedge (r, q) \in E\ .$$

A model is *non-split* if all of its communication graphs are.

Although the definition is deceptively simple, few structural results about non-split graphs exist. One of the few exceptions is the result by Charron-Bost and Schiper [42, Lemma 1] who showed that any product of at least $\lceil \log_2 n \rceil$ non-split communication graphs is broadcastable. This allows processes to simulate broadcastable communication graphs in a non-split model by constructing macro-rounds of length $O(\log n)$. By doing this, processes can execute an algorithm that was designed for broadcastable models in a non-split model with a slowdown of a logarithmic factor. Recently, we improved this bound from $O(\log n)$ to $O(\log \log n)$. This improvement is discussed in Chapter 5.

Interestingly, many communication graphs that originate from classical distributed computing models turn out to be non-split. In particular, this is the case for asynchronous message passing with a minority of crash-faulty processes: If every process receives messages from $n - f > n/2$ processes, then every pair of processes has a received message in common by the pigeonhole principle.

2.5 Rooted Models

A *rooted* communication graph is one that contains a rooted directed spanning tree. That is, there is a process that has directed paths to all other processes. A model is *rooted* if all of its communication graphs are.

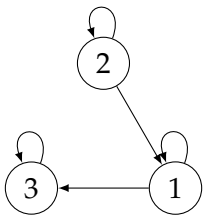

Figure 2.1: Rooted communication graph that is not non-split: processes $p = 2$ and $q = 3$ do not have a common incoming neighbor, although all other pairs of processes do.

Every non-split communication graph is rooted, but the converse is not true; see Figure 2.1 for an example. Due to self-loops, every communication graph that has an edge in either direction between any pair of processes is non-split. Because of this, rootedness and non-splitness are equivalent for $n = 2$ processes.

Just as it is possible to go from non-split to broadcastable communication graphs by considering sufficiently long products, it is also possible to go from rooted to non-split communication graphs. Put together, it is thus possible to go from rooted to broadcastable communication graphs via sufficiently long products.

Specifically, it suffices to build macro-rounds of length $n - 1$ to simulate non-split communication graphs is rooted models. This bound is tight. It was independently discovered by several authors [126, 35, 39]. The following lemma formalizes the result.

Lemma 2.1. *The product of $n - 1$ rooted communication graphs is non-split.*

Chapter 3

Asymptotic Consensus

In the problem of asymptotic consensus, processes are required to continually output a value in each time step such that all local output-value sequences converge to a common limit, subject to a validity condition. The problem is weaker than exact consensus, in which processes have to agree on a single value in finite time and irrevocably decide on that value. While exact consensus is used for applications like providing consistency among replicated databases or state-machines, it is not always required. Specifically, asymptotic consensus and its deciding variant approximate consensus has been used to study distributed control, sensor fusion, clock synchronization, formation control, rendezvous in space, robot gathering, load balancing, bird flocking, firefly synchronization, and opinion dynamics. These applications often come paired with limited computational power, local storage, and communication abilities.

In this chapter, we study the solvability and complexity of asymptotic consensus in oblivious network models. More details can be found in our papers at ICALP 2015 [39], ICALP 2016 [40], PODC 2018 [72], DISC 2018 [69], and DISC 2019 [105].

3.1 Problem Definition

In the asymptotic consensus problem, every process p starts with an input value x_p in some common Euclidean space $\mathbb{R}^d$ and continually outputs a value in $\mathbb{R}^d$. Write $x_p(0) = x_p$ and $x_p(t)$ for the output value of process p at the end of round t. For an algorithm to solve asymptotic consensus, each of its executions should satisfy:

- For all processes $p, q \in [n]$, if the sequences $\big(x_p(t)\big)_{t\geq 0}$ and $\big(x_q(t)\big)_{t\geq 0}$ converge, then their limits coincide. *(Agreement)*
- For all processes $p \in [n]$, if the sequence $\big(x_p(t)\big)_{t\geq 0}$ converges, then its limit lies in the convex hull of input values. *(Convex Validity)*

- For all processes $p \in [n]$, the sequence $\left(x_p(t)\right)_{t\geq 0}$ converges. *(Convergence)*

In the above definition, we prefaced the Agreement and Convex Validity condition with the phrase "if the sequences converge" to make the three conditions formally independent. We denote by $\text{hull}(X)$ the convex hull of a set $X \subseteq \mathbb{R}^d$, *i.e.*, the smallest convex set that contains X.[2] With this definition the conjunction of Agreement, Convex Validity, and Convergence can be expressed more compactly as:

$$\exists x^* \in \text{hull}\left(\{x_p(0) \mid p \in [n]\}\right) \forall p \in [n]\colon \quad \lim_{t\to\infty} x_p(t) = x^*$$

3.2 State of the Art

The problem of asymptotic consensus in dynamic networks has been extensively studied in distributed computing and control theory [58, 101, 44, 10, 36, 28]. The question of guaranteed convergence speeds and decision times of the corresponding approximate consensus problems, naturally arise in this context. Algorithms with convergence times exponential in the number of agents have been proposed. In particular, Cao *et al.* [36, Equation (26)] proved that the Equal Neighbor algorithm, which updates its value to the unweighted average of received values, has a contraction ratio of at most $\sqrt[n-1]{1-1/n^{n-1}}$ subject to rooted message adversaries, which leads to an exponential upper bound on the convergence time.

Olshevsky and Tsitsiklis [108], proposed an algorithm with polynomial convergence time in bidirectional networks with certain stability assumptions on the occurring communication graphs. The bounds on convergence times were later on refined by Nedic *et al.* [102]. Chazelle [44] proposed a convex-combination algorithm with polynomial convergence time, which is guaranteed to work for any bidirectional connected message adversary.

To speed up convergence times, algorithms where agents set their output based on values that have been received in rounds prior to the previous round have also been considered in literature: Olshevsky [107] proposed a linear convergence time algorithm that uses messages from two rounds, restricted to a fixed bidirectional communication graph, however. Yuan *et al.* [129] proposed a linear convergence-time algorithm for a possibly non-bidirectional fixed topology. It requires storing all received values.

To the best of our knowledge, the only study of lower bounds in dynamic networks has been done by Cao *et al.* [34]: the authors identified $1-1/n$ as the worst-case scrambling constant of the Equal Neighbor algorithm in non-split communication graphs, and $1-1/n^{n-1}$ for that of the product of $n-1$

[2] In more concrete terms, the convex hull of X is equal to the set of linear combinations $\sum_{x\in X} \lambda_x x$ where all but finitely many λ_x are zero, $\lambda_x \geq 0$ for all $x \in X$, and $\sum_{x\in X} \lambda_x = 1$.

rooted communication graphs. While scrambling constants can be used to prove upper bounds on the contraction ratio of asymptotic consensus algorithms, a lower bound on the scrambling constant does not in general imply a similar lower bound on the contraction ratio.

In the context of classical distributed computing failure scenarios, Dolev *et al.* [54] studied the approximate consensus problem: they considered fully-connected synchronous distributed systems with up to f Byzantine agents, and its asynchronous variant. The two presented algorithms require $n \geq 3f + 1$ for the synchronous and $n \geq 5f + 1$ for the asynchronous distributed system, the first of which is optimal in terms of resilience [64]. The latter result was improved to $n \geq 3f + 1$ by Abraham *et al.* [1]. Both papers also address the question of optimal contraction ratio in such systems. Since, however, in synchronous systems with $n \geq 3f + 1$ exact consensus is solvable, leading to a contraction ratio of 0, the authors consider bounds for round-by-round contraction ratios. Dolev *et al.* [54] showed that the achieved round-by-round contraction ratio of $\frac{1}{2}$ is actually tight for a certain class of algorithms that repeatedly set their output to the image of a so-called cautious function applied to the multiset of received values. A lower bound for arbitrary algorithms, however, remained an open problem. In higher dimensions, *i.e.*, for any $d \geq 1$, Mendes *et al.* [98] proposed algorithms with decision time of $d \cdot \lceil \log_2 \frac{\sqrt{d}\Delta}{\varepsilon} \rceil$ under the optimal resiliency condition $n \geq f \cdot (d + 2) + 1$.

Fekete [59] also studied round-by-round contraction ratios for several failure scenarios in which exact consensus is solvable. He proved asymptotically tight lower bounds for synchronous distributed systems in presence of crashes, omission, and Byzantine agents. The bounds hold for approximate consensus algorithms that potentially take into account information from all previous rounds. Fekete [60] later presented an algorithm for asynchronous message-passing systems with a minority of crashes, also proving a tight lower bound on the contraction ratio of any algorithm operating in asynchronous rounds for such systems. In the nonuniform iterated immediate snapshot (NIIS) model, Hoest and Shavit [86, Theorem 5.2] proved tight upper and lower bounds on the round complexity for the approximate consensus problem.

Herlihy *et al.* [85] studied generalizations of the approximate consensus problem in terms of combinatorial topology with round-based communication objects.

3.3 Averaging Algorithms

Averaging algorithms have a very simple structure. Each process p maintains a local variable $x_p \in \mathbb{R}^d$, initially equal to its input value. It then sends the value of x_p to its neighbors in each round and updates it to a (weighted)

average of the received values. Different averaging algorithms differ in the choice of the weights.

We will denote the value of the variable x_p at the end of round t by $x_p(t)$. In particular, process p's input value is equal to $x_p(0)$. Since we follow a send–receive–compute round structure, the value sent by process p in round t is equal to $x_p(t-1)$. The value $x_p(t)$ is thus an average of the previous-round values $x_q(t-1)$. Naturally, any value that was not received by p in round t has to have zero weight in the average. In the algorithm and in the sequel, we write $\mathrm{In}_p(t)$ for the set of incoming neighbors of p in the round-t communication graph G_t.

Algorithm 1 General structure of averaging algorithms

Initialization:

1: $x_p \in \mathbb{R}^d$

In round $t \geq 1$ **do:**

2: send x_p to other processes and receive x_q for $q \in \mathrm{In}_p(t)$

3: determine weights $\alpha_{p,q}(t)$

4: $x_p \leftarrow \sum_{q \in \mathrm{In}_p(t)} \alpha_{p,q}(t) x_q$

The weights $\alpha_{p,q}(t)$ have to be non-negative, they have to sum up to 1, *i.e.*, $\sum_{q \in [n]} \alpha_{p,q}(t) = 1$, and they need to respect the information flow, *i.e.*, $\alpha_{p,q}(t) = 0$ if $q \notin \mathrm{In}_p(t)$. Then, introducing the $n \times n$ matrices $A(t)$ whose coefficients are equal to the weights $\alpha_{p,q}(t)$, we have the non-homogeneous linear recurrence $x(t) = A(t)x(t-1)$. This observation gives rise to the possibility of using methods from linear algebra—spectral theory in particular—for the analysis of averaging algorithms.

One of the simplest averaging algorithms is the Equal Neighbor algorithm, which assigns the same weight to every incoming value. That is, it chooses $\alpha_{p,q}(t) = 1/|\mathrm{In}_p(t)|$ for every $q \in \mathrm{In}_p(t)$ in the notation of Algorithm 1. While it turns out that the Equal Neighbor algorithm is not time-optimal, it solves asymptotic consensus whenever it is solvable. Even more, any of a large class of averaging algorithms can be used to solve asymptotic consensus whenever it is solvable.

3.4 Characterization of Solvability in Oblivious Models

In this section, we present a solvability characterization for asymptotic consensus in oblivious models. No characterization for general models is known to date. Asymptotic consensus is solvable in an oblivious model if and only if it is rooted. Moreover, we get explicit upper bounds on the time until ε-agreement. This characterization confirms the fact that asymptotic consensus is solvable in asynchronous message passing with f crash faults if and only

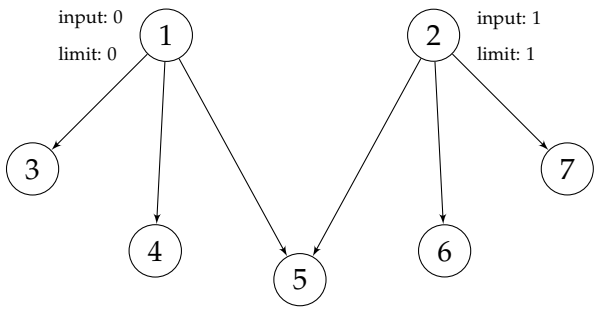

Figure 3.1: Unsolvability of asymptotic consensus in non-rooted models: By Convex Validity, the outputs of both processes $p = 1$ and $q = 2$ have to converge to their respective input values. If these input values are different, then Agreement is violated.

if there is a majority of correct processes, *i.e.*, if $f < n/2$. For the message-omission model with up to k omissions per round, it implies that asymptotic consensus is solvable if and only if $k \leq 2n - 3$. This fault-tolerance threshold is higher than the one for exact consensus [115], which is solvable if and only if $k \leq n - 2$

We start with the impossibility result:

Lemma 3.1. *Asymptotic consensus is unsolvable in oblivious models that are not rooted.*

Figure 3.1 illustrates the reason why asymptotic consensus in unsolvable in oblivious non-rooted models. Indeed, if a single non-rooted communication graph is repeated in all rounds of an execution and all processes start with different input values, then no algorithm can solve asymptotic consensus. Similar arguments also show that approximate consensus and stabilizing consensus are unsolvable in oblivious non-rooted models.

In order to show solvability in rooted models, we introduce the notion of confidence-bounded averaging algorithms. This class of algorithms assigns each received value a confidence lower-bounded by a positive constant. Let $\rho > 0$ be a positive real. We call an averaging algorithm *ρ-confidence-bounded* if it assigns a weight of at least ρ to every received value. In symbolic terms, $q \in \mathrm{In}_p(t) \implies \alpha_{p,q}(t) \geq \rho$.

The definition of non-split communication graphs turns out to be the right one for characterizing the possibility to reduce the diameter of values in a single round. Indeed, in the case $d = 1$ of scalar values, one can show that the distance of the values of two given processes contracts by a factor of $1 - \rho$ in a non-split round of a ρ-confidence-bounded averaging algorithm. Formally, we use the Dobrushin semi-norm [52] $\delta(x) = \max_{p,q \in [n]} |x_p - x_q|$

and its extension to matrices $\delta(A) = \sup_{x:\,\delta(x)\neq 0} \delta(Ax)/\delta(x)$. This is the central ingredient to showing the following lemma.

We will give explicit time bounds only for the scalar case $d = 1$. The multidimensional case will be discussed in Section 3.8.

Lemma 3.2. *Let $\rho > 0$. Every ρ-confidence-bounded averaging algorithm solves asymptotic consensus in any non-split model.*

Moreover, for $d = 1$, the time until ε-agreement is upper-bounded by

$$T(\varepsilon) \leq \left\lceil \log_{1/(1-\rho)} \frac{\Delta}{\varepsilon} \right\rceil \leq \left\lceil \frac{1}{\rho} \log \frac{\Delta}{\varepsilon} \right\rceil$$

where Δ is the diameter of the set of input values.

Since the Equal Neighbor algorithm is $\frac{1}{n}$-confidence-bounded, a consequence of Lemma 3.2 is that it achieves ε-agreement in $O(n \log \frac{\Delta}{\varepsilon})$ rounds in non-split models.

Due to Lemma 2.1, *i.e.*, the fact that macro-rounds of $n-1$ rooted communication graphs generate non-split rounds, it is possible to extend the possibility result of Lemma 3.2 from non-split to rooted models. There are two ways to do this reduction: via transitive weights or via value relaying.

The approach via transitive weights does not need a change in the algorithm, but is rather an analysis technique. Due to the linearity of the evolution of the processes' values, the aggregate effect of rounds t and $t+1$ is described by the product matrix $A(t+1)A(t)$ applied to the values of round $t-1$. Indeed, $x(t+1) = A(t+1)x(t) = A(t+1)A(t)x(t-1)$. The values $x_p(t+1)$ hence are weighted averages of the $x_q(t-1)$. The analysis technique now consists in viewing macro-rounds of length $n-1$ as a single round of another averaging algorithm, this time operating in a non-split model. Due to the multiplication of weights in the matrix products, in general it can only be guaranteed that the resulting algorithm operating in non-split rounds is ρ^{n-1}-confidence-bounded if the original algorithm is ρ-confidence-bounded. This technique is used to prove Lemma 3.3 below.

The second approach via value relaying consists in explicitly forwarding the received values during $n-1$ rounds, and then making a single averaging step at the end of the $n-1$ rounds. This needs a change in the algorithm and is the subject of Section 3.5 where we discuss time-optimal algorithms.

Lemma 3.3. *Let $\rho > 0$. Every ρ-confidence-bounded averaging algorithm solves asymptotic consensus in any rooted model.*

Moreover, for $d = 1$, the time until ε-agreement is upper-bounded by

$$T(\varepsilon) \leq (n-1) \left\lceil \frac{1}{\rho^{n-1}} \log \frac{\Delta}{\varepsilon} \right\rceil$$

where Δ is the diameter of the set of input values.

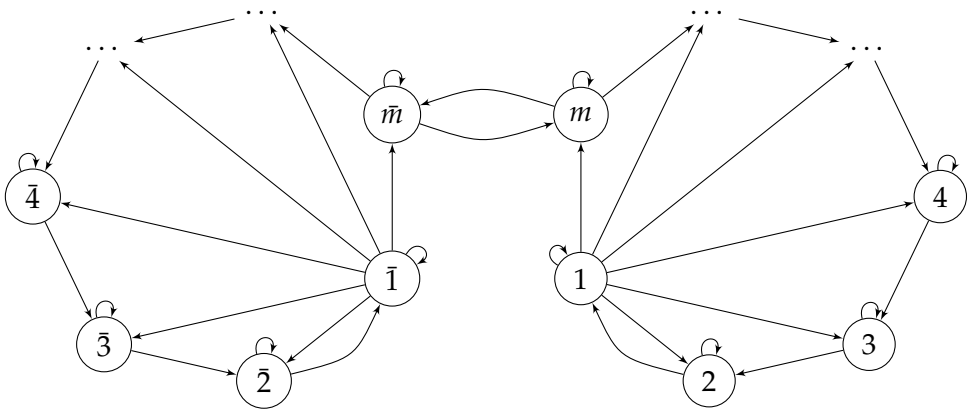

Figure 3.2: The butterfly communication graph with exponential ε-agreement time of the Equal Neighbor algorithm

This means that the Equal Neighbor algorithm achieves ε-agreement in $O(n^n \log \frac{\Delta}{\varepsilon})$ rounds in rooted models. This is an exponential upper bound on the time until ε-agreement. It turns out that the Equal Neighbor algorithm indeed shows exponential time behavior, even in a constant communication graph. We will see, however, that sub-exponential, even linear-time, algorithms exist.

The *butterfly* communication graph is depicted in Figure 3.2. The proof of the time lower bound for the Equal Neighbor algorithm is based on a spectral-gap argument. It can be interpreted in terms of an unbiased random walk with edge directions inverted with respect to Figure 3.2: To go from vertex 1 to vertex m, the walk has to follow the vertices $1, 2, \ldots, m$ in order. Since there is a constant probability at each of these vertices to drop down to vertex 1, the expected time to cross from vertex 1 on the right side to any vertex on the left side is exponentially large. In particular, the mixing time of the random walk is exponential as well.

Lemma 3.4. *In the model consisting exclusively of the butterfly communication graph, the worst-case time until ε-agreement of the Equal Neighbor algorithm is lower-bounded by*

$$T(\varepsilon) = \Omega\left(2^{n/3} \log \frac{\Delta}{\varepsilon}\right) \quad .$$

Theorem 3.5. *Let $\mathcal{M}$ be an oblivious model. Asymptotic consensus is solvable in $\mathcal{M}$ if and only if $\mathcal{M}$ is rooted.*

The same characterization—with essentially the same arguments—holds for approximate and stabilizing consensus as well.

3.5 Time-Optimal Algorithms for Scalar Values

In this section, we focus on the case $d = 1$ of scalar values. We present algorithms that are time-optimal in non-split models and nearly time-optimal in rooted models. The analysis technique used to derive them is based on the notion of α-safe averaging algorithms. With the help of this notion, we are able to push the algorithms to the maximal safeness parameter, which turns out to lead to time-optimality in non-split models. For rooted models, we introduce the notion of amortized averaging algorithms, which lead to near-optimal algorithms. We present the matching lower bounds in Section 3.6. Nearly time-optimal algorithms for higher-dimensional values are discussed in Section 3.8.

An averaging algorithm in dimension $d = 1$ is *ρ-safe* if its new value in a round keeps at least an ρ-fraction distance from the borders of the convex hull of received values. Formally, setting $m_p(t-1)$ to be the minimal and $M_p(t-1)$ the maximal value received by process p in round t, we must have:

$$(1-\rho)m_p(t-1) + \rho M_p(t-1) \leq x_p(t) \leq \rho m_p(t-1) + (1-\rho)M_p(t-1)$$

It is impossible for any algorithm to be ρ-safe if $\rho > 1/2$ since the interval of allowed values can be empty in this case. We can verify that the Equal Neighbor algorithm is $\frac{1}{n}$-safe. A more general result holds:

Lemma 3.6. *Every ρ-confidence-bounded averaging algorithm is ρ-safe.*

Generalizing Lemma 3.2 with the notion of ρ-safeness, we get:

Lemma 3.7. *Let $0 < \rho \leq 1/2$. Every ρ-safe averaging algorithm solves asymptotic consensus in any non-split model.*

Moreover, the time until ε-agreement is upper-bounded by

$$T(\varepsilon) \leq \left\lceil \log_{1/(1-\rho)} \frac{\Delta}{\varepsilon} \right\rceil \leq \left\lceil \frac{1}{\rho} \log \frac{\Delta}{\varepsilon} \right\rceil$$

where Δ is the diameter of the set of input values.

Having a larger safeness parameter leads to a smaller time bound in Lemma 3.7. This leads to the definition of the Midpoint algorithm, which achieves the maximal safeness parameter $\rho = 1/2$ by setting its value to the midpoint of the convex hull of received values, *i.e.*:

$$x_p(t) = \frac{m_p(t-1) + M_p(t-1)}{2}$$

The notion of safeness thus gives an explanation of why the midpoint algorithm has been used as a building block in previous algorithms [124].

The Midpoint algorithm takes $\lceil \log_2 \frac{\Delta}{\varepsilon} \rceil$ non-split rounds until it achieves ε-agreement. This is by a factor of n smaller than the bound of $O(n \log \frac{\Delta}{\varepsilon})$ for the Equal Neighbor algorithm.

In fact, the Midpoint algorithm is time-optimal, with a round-to-round contraction of $1/2$, in non-split models when $n \neq 2$. For $n = 2$, there is an algorithm that is faster. Intuitively, the reasoning for the midpoint algorithm is that one cannot be sure whether the minimum or the maximum received value will move in the current round, which is why equidistance to both is a safe bet. If there are only two processes, then each of the processes has either the minimum or the maximum value. Each process can thus locally decide to move the minimum or the maximum. This observation leads to Algorithm 2, which has a round-by-round contraction of $1/3$ and time until ε-agreement of $\lceil \log_3 \frac{\Delta}{\varepsilon} \rceil$ in non-split models. This is also optimal.

Algorithm 2 Time-optimal algorithm for two processes

Initialization:

1: $x_p \in \mathbb{R}^d$

In round $t \geq 1$ **do:**

2: send x_p to other process and receive x_q if $q \in \mathrm{In}_p(t)$
3: **if** x_q was received **then**
4: $\quad x_p \leftarrow x_p/3 + 2x_q/3$
5: **end if**

Since constructing macro-rounds of length $n - 1$ can transform rooted models into non-split models (Lemma 2.1), it is possible to transfer the time-optimal algorithms for non-split models to rooted models. Of course, *a priori* this does not mean that the resulting algorithms are time-efficient, let alone time-optimal.

There are multiple different ways of transferring non-split algorithms to rooted models. Lemma 3.3 used the concept of transitive weights in the combined weight matrix for a macro-round. This approach leads to an exponential time complexity for many averaging algorithms.

A different approach that does not explicitly rely on the linearity of the value updates is to explicitly construct the product graph of a macro-round by relaying messages. To properly detect duplicate messages, in general, it is necessary to tag messages with process identifiers, even if the underlying non-split algorithm does not use identifiers.

In some cases, however, it is not necessary. Namely whenever the underlying non-split algorithm only depends on the set of received values, but not their origin nor their multiplicity. For the Midpoint algorithm, for instance, only two values need to be relayed: the minimum and the maximum.

We call these algorithms the *amortized* versions of their underlying non-split algorithms. As an example, Algorithm 3 contains the amortized Equal Neighbor algorithm.

Algorithm 3 Amortized Equal Neighbor algorithm

Initialization:

$x_p \in \mathbb{R}^d$ and $W_p \leftarrow \{(p, x_p)\}$

In round $t \geq 1$ **do:**

send W_p to other processes and receive W_q from all processes $q \in \mathrm{In}_p(t)$

$W_p \leftarrow \bigcup_{q \in \mathrm{In}_p(t)} W_q$

if $t \equiv 0 \mod (n-1)$ **then**

  $x_p \leftarrow \frac{1}{|W_p|} \sum_{(q,v) \in W_p} v$

  $W_p \leftarrow \{(p, x_p)\}$

end if

The time complexity of amortized algorithms can be upper-bounded by the time complexity of the underlying non-split algorithms, multiplied with the length of the macro-rounds:

Lemma 3.8. *The amortized version of an algorithm with time to ε-agreement at most $\tilde{T}(\varepsilon)$ in non-split models achieves ε-agreement in rooted models in time*

$$T(\varepsilon) \leq (n-1)\tilde{T}(\varepsilon) \ .$$

In particular, for the Midpoint algorithm, we get:

Theorem 3.9. *The amortized Midpoint algorithm solves asymptotic consensus in any rooted model.*

Moreover, the time until ε-agreement is upper-bounded by

$$T(\varepsilon) \leq (n-1) \left\lceil \log_2 \frac{\Delta}{\varepsilon} \right\rceil$$

where Δ is the diameter of the set of input values.

It is not known whether this time complexity is tight in rooted models, but we will see that it is almost tight, up to a factor of $1 + O(1/n)$, in the next section (Corollary 3.16).

3.6 Time Lower Bounds in Oblivious Models

In this section, we study lower bounds on the time complexity of asymptotic consensus algorithms in oblivious models. From the results in Section 3.4, we know that this question only makes sense in rooted models as asymptotic consensus is not solvable in other models. As for the upper bounds in Section 3.5, we will separate the analysis of non-split and rooted models. For non-split models we are able to exactly identify the worst-case time complexity, while for rooted models we are able to identify it asymptotically.

The central vehicle to prove these lower bounds is the notion of valency. It is defined as the set of output limits reachable from a given configuration. As such, is it a generalization of the notion for exact consensus algorithms. A central difference is that the number of different output limits in asymptotic consensus is typically infinite, even uncountable, while it is typically finite for exact consensus. The formal definition of valency is one of main reasons for us to study asymptotic rather than approximate consensus. Both problems are closely related, but approximate consensus allows multiple output values in a single execution, whereas asymptotic consensus has a single output limit.

Given a model $\mathcal{M}$ and an asymptotic consensus algorithm $\mathcal{A}$, the *valency* of a configuration C is defined as the set

$$Y^*_{\mathcal{M},\mathcal{A}}(C) = \{y^*(\gamma) \mid C \text{ occurs in } \gamma \in \mathcal{E}_{\mathcal{M},\mathcal{A}}\}$$

of possible common limits starting from configuration C. If the model and the algorithm are clear from the context, we will omit the subscript. Over the course of an execution $\gamma = C_0, C_1, C_2, \ldots$, the valency is non-increasing: $Y^*(C_t) \supseteq Y^*(C_{t+1})$. The limit of this monotonous sequence of sets is a singleton:

$$\lim_{t\to\infty} Y^*(C_t) = \bigcap_{t=0}^{\infty} Y^*(C_t) = \{y^*(\gamma)\}$$

To measure speed of convergence, we use the diameter of the valencies of the configurations in an execution. We say that configuration C is *ε-valent* if the diameter of its valency is at most ε.

A configuration is *ε-agreeing* if the set of output values in the configuration has a diameter of at most ε. In general neither of the two notions, ε-valence and ε-agreement, is stronger than the other. Since the convex hull of the output values is non-increasing in every execution of an averaging algorithm, we have the following result:

Lemma 3.10. *Let C be a configuration of an averaging algorithm and let $\varepsilon > 0$. If C is ε-agreeing, then C is ε-valent.*

For this reason, all time upper bounds from Section 3.5 not only hold for ε-agreement but also for ε-valence. In the rest of the section, we establish lower bounds on the time until ε-valence. For approximate consensus, the fact that processes have to decide on output values at a finite time necessitates the stability of ε-valence.

The proofs of the lower bounds follow the same general strategy:

1. For an arbitrary $\Delta \geq 0$, find an initial configuration C_0 that satisfies $\operatorname{diam} Y^*(C_0) \geq \alpha \cdot \Delta(C_0)$ and $\Delta(C_0) = \Delta$ where $\Delta(C_0)$ denotes the diameter of the set of values in C_0.

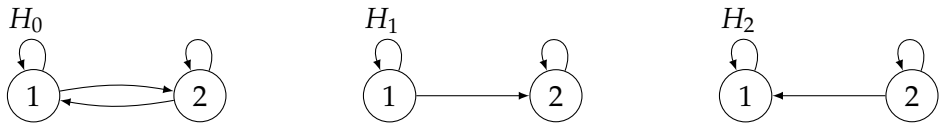

Figure 3.3: The rooted communication graphs H_0, H_1, and H_2 for $n = 2$

2. For an arbitrary execution $\gamma = C_0, C_1, C_2, \ldots$ starting from the initial configuration C_0, show that $\operatorname{diam} Y^*(C_t) \geq \beta \cdot \operatorname{diam} Y^*(C_{t-1})$ for all rounds $t \geq 1$.

3. Conclude that $\operatorname{diam} Y^*(C_t) \geq \alpha \cdot \beta^t \cdot \Delta$ for all $t \geq 0$.

4. The smallest T for which $\operatorname{diam} Y^*(C_T) \leq \varepsilon$ thus fulfills $T \geq \log_{1/\beta} \frac{\alpha\Delta}{\varepsilon}$.

The first lower bound we present is for two processes, for any oblivious model that includes the lossy-link model. Figure 3.3 depicts the three communication graphs in the model: H_0 contains both directed edges, H_1 contains only the edge from process 1 to process 2, and H_2 contains only the edge from process 2 to process 1. In terms of the general proof strategy, we choose the parameters $\alpha = 1$ and $\beta = 1/3$.

Theorem 3.11. *Let $n = 2$. In an oblivious model that contains the communication graphs H_0, H_1, and H_2, the time until ε-valence is lower-bounded by $\lceil \log_3 \frac{\Delta}{\varepsilon} \rceil$ where Δ is the diameter of the set of initial values.*

This shows that Algorithm 2 is time-optimal in general oblivious models with two processes:

Corollary 3.12. *The worst-case time until ε-valence in oblivious non-split models with $n = 2$ processes is equal to $\lceil \log_3 \frac{\Delta}{\varepsilon} \rceil$ where Δ is the diameter of the set of initial values.*

For models with more than two processes, we distinguish non-split and rooted models. This distinction is not necessary for two processes since the two notions coincide in that case. For non-split models, we identify the tight bound on the time until ε-valence. For rooted models, we identify an almost-tight bound, up to a factor of $\frac{n-1}{n-2} \sim 1$.

The bound for non-split models is based on the notion of *deaf* graphs. For any communication graph G, the set $\operatorname{deaf}(G)$ contains all communication graphs derived from G by removing all incoming edges from some process p.

Theorem 3.13. *Let $n \geq 3$. In an oblivious model that contains the communication graphs in $\operatorname{deaf}(G)$, the time until ε-valence is lower-bounded by $\lceil \log_2 \frac{\Delta}{\varepsilon} \rceil$ where Δ is the diameter of the set of initial values.*

This shows that the Midpoint algorithm is time-optimal in general non-split oblivious models with $n \geq 3$ processes:

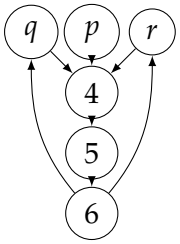

Figure 3.4: Rooted communication graph Ψ_p for $n = 6$

Corollary 3.14. *The worst-case time until ε-valence in oblivious non-split models with $n \geq 3$ processes is equal to $\lceil \log_2 \frac{\Delta}{\varepsilon} \rceil$ where Δ is the diameter of the set of initial values.*

For rooted models, we introduce the class of Ψ-graphs. We distinguish processes $1, 2, 3$ and the remaining processes. For a process $p \in \{1, 2, 3\}$, we define the graph as follows: We arrange the processes $4, 5, 6 \ldots, n$ in a directed path, all processes in $\{1, 2, 3\}$ have an edge to process 4, and process n has an edge to the two processes in $\{1, 2, 3\} \setminus \{p\}$. Figure 3.4 shows the construction in visual form.

With these graphs, we construct blocks of $n - 2$ rounds, which do not allow process n to quickly decide which of the Ψ-graphs was chosen.

Theorem 3.15. *Let $n \geq 3$. In an oblivious model that contains the Ψ-graphs, the time until ε-valence is lower-bounded by*

$$(n-2) \left\lceil \log_2 \frac{\Delta}{\varepsilon} \right\rceil - (n-3)$$

where Δ is the diameter of the set of initial values.

This is almost tight, but the block size of $n - 2$ used in the lower bound is slightly smaller than the block size of $n - 1$ used by the amortized Midpoint algorithm. It is an open question whether another lower-bound construction can prove strict optimality of the amortized Midpoint algorithm.

Corollary 3.16. *The worst-case time $T(\varepsilon)$ until ε-valence in oblivious rooted models with $n \geq 3$ processes is bounded by*

$$(n-2) \left\lceil \log_2 \frac{\Delta}{\varepsilon} \right\rceil - (n-3) \leq T(\varepsilon) \leq (n-1) \left\lceil \log_2 \frac{\Delta}{\varepsilon} \right\rceil$$

where Δ is the diameter of the set of initial values.

Using the relations α^* and β on communication graphs introduced by Coulouma *et al.* [48] for characterizing the solvability of exact consensus in oblivious models as a proof ingredient, we can relate the solvability of exact and asymptotic consensus in terms of the valency set of initial configurations:

Theorem 3.17. *Let $\mathcal{M}$ be an oblivious model. Exact consensus is solvable in $\mathcal{M}$ if and only if there exists an asymptotic consensus algorithm $\mathcal{A}$ for $\mathcal{M}$ such that the initial valency $Y^*_{\mathcal{M}',\mathcal{A}}(C_0)$ is either a singleton or disconnected for all sub-models $\mathcal{M}' \subseteq \mathcal{M}$ and all initial configurations C_0 of $\mathcal{A}$.*

Furthermore, we can introduce the notion of α-diameter, derived from the relation α^* to prove time lower bounds in some more classes of oblivious models:

Theorem 3.18. *In an oblivious model in which exact consensus is not solvable, the time until ε-valence is lower-bounded by*

$$\left\lceil \log_{D+1} \frac{\delta}{\varepsilon} \right\rceil$$

where $\delta = \operatorname{diam} Y^*(C_0)$ is the diameter of the initial configuration's valency.

This generalizes the bounds for two processes and for non-split models since the α-diameters are $D = 2$ and $D = 1$, respectively, in these cases.

Theorem 3.18 allows to reason about round-based algorithms in asynchronous message passing with f crash faults. It can be shown that the α-diameter of the resulting set of communication graphs is bounded by $D \geq \lceil n/f \rceil$. The following lower bound is stated in terms of contraction ratio

$$\rho = \sup_{\gamma=(C_t)_t} \limsup_{t\to\infty} \sqrt[t]{\operatorname{diam} Y^*(C_t)}$$

which is used to assess asymptotic and approximate consensus algorithms in the literature. The contraction ratio of Algorithm 2 is $\rho = 1/3$ and that of the Midpoint is $\rho = 1/2$ in non-split models.

Theorem 3.19. *The contraction ratio of round-based algorithms in asynchronous message passing with up to f crash faults is lower-bounded by $\frac{1}{\lceil n/f \rceil + 1}$.*

An algorithm by Fekete [60] shows that this lower bound is almost tight:

Corollary 3.20. *The contraction ratio ρ of round-based algorithms in asynchronous message passing with up to f crash faults is bounded by*

$$\frac{1}{\lceil n/f \rceil + 1} \leq \rho \leq \frac{1}{\lceil n/f \rceil - 1} \ .$$

3.7 Quantization and Rounding

A natural question when studying averaging algorithms is whether it is necessary to send and receive real, *i.e.*, continuous, values. Since space and message complexity is usually measured in bits, these measures are moot when

dealing with averaging algorithms. Simply receiving the input value of the asymptotic consensus problem already requires infinite bit complexity.

One could, however, ask the question of bit complexity if the input values are taken from a finite set of possible values. In the one-dimensional case $d = 1$, this value quantization can be done, *e.g.*, by restricting the initial values to be integer multiples of some quantization parameter $1/Q$ in the bounded interval $[0, 1]$. If the requested precision ε is strictly smaller than the distance $1/Q$ of two neighboring quantized values, then it is necessary to exactly agree on a single value. If the requested precision ε is at least $1/Q$, then multiple different output vales are possible. In any case, the output values of each process have to stabilize to satisfy Convergence. As we will see, this one-dimensional quantized version of asymptotic consensus is solvable in any rooted model if $\varepsilon \geq 1/Q$. In particular, choosing $\varepsilon = 1/Q$, it is possible to convergence, and decide, onto two neighboring values k/Q and $(k+1)/Q$. This solves a version of 2-set consensus [50].

In higher dimensions this is not possible in general. Take, for instance, the two-dimensional grid with integer coordinates. There exist arbitrarily far pairs of points on the grid that do not contain any other grid point in their convex hull. One would thus need to exactly agree on one of the two input values if any precision $\varepsilon < \Delta$ is requested.

It turns out that the algorithm to solve quantized asymptotic consensus is a simple variant of the Midpoint algorithm: after calculating the midpoint, round it down to the nearest multiple of $1/Q$. The space and per-message complexity is thus $\lceil \log_2 Q \rceil$ bits.

The next theorem summarizes the analysis of the Midpoint algorithm when applying the rounding rule. The time until $\frac{1}{Q}$-agreement is $O(\log Q)$ in non-split models, which is consistent with the $O(\log \frac{1}{\varepsilon})$-bound of the vanilla Midpoint algorithm with bounded input values.

Its proof is based on directly using the contraction property of the Midpoint algorithm until $\frac{1}{Q-2}$-agreement, and then showing that two additional rounds suffice to achieve $\frac{1}{Q}$-agreement.

Theorem 3.21. *The quantized Midpoint algorithm solves asymptotic consensus for $\varepsilon \geq 1/Q$ in any non-split model.*

Moreover, the time until $\frac{1}{Q}$-agreement is upper-bounded by $\lfloor \log_2(Q-2) \rfloor + 2$.

Using the amortization technique, Theorem 3.21 is easily generalized to rooted models as well. In this case, the time-complexity bound is multiplied by $(n-1)$.

While parts of the proof of Theorem 3.21 are applicable to any ρ-safe algorithm, it is not obvious how to adapt some other key parts to the case of non-optimal safeness parameters $\rho > 1/2$. This not only impacts time complexity, but also solvability for all $\varepsilon \geq 1/Q$. The result thus seems specific to the Midpoint algorithm.

3.8 Multidimensional Values

Although the lower bounds we presented are applicable to values from arbitrary Euclidean spaces $\mathbb{R}^d$, some of the algorithms we gave, in particular the Midpoint algorithm, are specific to the case $d = 1$ of scalar values, and not applicable to higher dimensions. Other algorithms are applicable to higher-dimensional values, like the Equal Neighbor algorithm, but their time complexity is not optimal, and depends on n, even in non-split models.

In this section, we present an algorithm that works for values of arbitrary dimension[3] and whose contraction ratio, and thus the time until ε-agreement, and ε-valence, in non-split models does not depend on n or the dimension of the values. This algorithm are the first with this property. It is an averaging algorithm.

Previous algorithms include the algorithms by Mendes *et al.* [98], who presented two algorithms, one with contraction ratio at most $1/\sqrt[d]{2}$ and one with contraction ratio at most $1 - 1/n$. Their algorithms were conceived for asynchronous message passing with f Byzantine processes. They also identified the solvability threshold as $n > (d+2)f$. The Centroid algorithm [41] was designed for non-split models and has a contraction ratio of at most $d/(d+1)$.

The algorithm presented in this section is the MidExtremes algorithm, which is a specific generalization of the Midpoint algorithm to multiple dimensions. The algorithm selects two values it received with maximum distance and updates its value to the midpoint between the two selected values. Its pseudo-code is detailed in Algorithm 4.

Algorithm 4 MidExtremes algorithm for process p

Initialization:

1: y_p is the initial value in V

In round $t \geq 1$ **do:**

2: broadcast y_p

3: $\mathrm{Rcv}_p \leftarrow$ set of received values

4: $(a, b) \leftarrow \operatorname{argmax}_{(a,b)\in Rcv_p^2} \|a - b\|$

5: $y_p \leftarrow \dfrac{a+b}{2}$

Being a generalization of the Midpoint algorithm for scalar values, it is time-optimal in oblivious non-split models if $n \geq 3$. The analysis in higher dimensions relies on elementary linear-algebraic facts. Figure 3.5 shows the involved tetrahedron when estimating the distance of two arbitrary midpoints formed by the MidExtremes algorithm. The fact that one only ever

[3] The algorithm, and its performance bounds, even work in infinite dimensions, provided that the values are taken from an inner-product space.

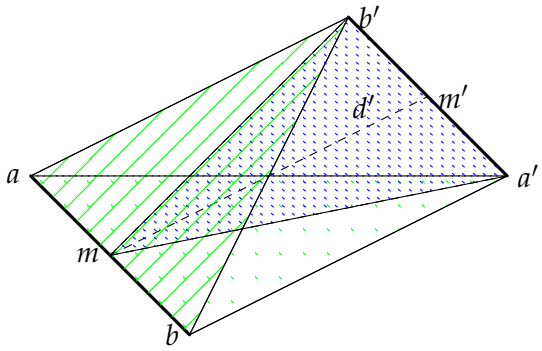

Figure 3.5: Tetrahedron formed by extreme points a and b of processes p and extreme points a' and b' of process q. The distance between the new values m and m' is d'.

needs to inspect pairwise distances, and thus only ever four extreme points, limits the analysis to a three-dimensional space. The resulting bound does not grow with the dimension. It turns out that the contraction ratio is at most $\rho \leq \sqrt{7/8}$:

Theorem 3.22. *The MidExtremes algorithm solves asymptotic consensus in any non-split model.*

Moreover, the time until ε-agreement is upper-bounded by

$$T(\varepsilon) \leq \left\lceil \log_{\sqrt{8/7}} \frac{\Delta}{\varepsilon} \right\rceil$$

where Δ is the diameter of the set of input values.

It can also be shown that, using the framework of Mendes *et al.* [98], the MidExtremes algorithm can be applied to the asynchronous message passing with Byzantine faults with the same time complexity. This improves the time complexity from $\Omega(d \log \frac{d\Delta}{\varepsilon})$ using their algorithm to $O(\log \frac{\Delta}{\varepsilon})$ using ours, eliminating all terms that depend on the dimension d.

3.9 Relation to Approximate Consensus

Approximate consensus is a variant of asymptotic consensus with a finite-time decision. Since asymptotic consensus is oftentimes employed in models in which exact consensus is not solvable, requiring decision values to agree exactly is not always useful. Decision values in approximate consensus are thus allowed to disagree, but only within a maximal distance of $\varepsilon > 0$.

Formally, every process p starts with an input value $x_p \in \mathbb{R}^d$, the parameter $\varepsilon > 0$, and an upper bound Δ on the diameter of the set of initial values, and has a write-once decision variable y_p. For an algorithm to solve approximate consensus, each of its executions should satisfy:

- For all processes $p, q \in [n]$, if both p and q have decided, then their decision values satisfy $\|y_p - y_q\|_2 \leq \varepsilon$. *(Approximate Agreement)*
- For all processes $p \in [n]$, if p has decided, then y_p lies in the convex hull of input values. *(Convex Validity)*
- Every process $p \in [n]$ eventually decides. *(Termination)*

In oblivious models, the solvability of approximate consensus is equivalent to that of asymptotic consensus with error estimation. Their time complexities are also similar.

Formally, for asymptotic consensus with error estimation, we have every process output not only its current value $x_p(t)$, but also an estimate $\varepsilon_p(t)$ on its distance to the limit value. Of course, we would need to add the requirement that $\varepsilon_p(t) \to 0$ as $t \to \infty$ in every execution. A strong estimate would be one that guarantees

$$x^* \in \left[x_p(t) - \varepsilon_p(t)\,,\, x_p(t) + \varepsilon_p(t)\right]$$

for the distance to the common limit x^* at all times t. All our algorithms in this chapter solve asymptotic consensus with error estimation if an upper bound Δ on the diameter of the set of initial values is known.

Given an algorithm that solves asymptotic consensus with error estimation, we can transform it into an algorithm for approximate consensus just by deciding the current output value at the earliest time t such that $\varepsilon_p(t) \leq \varepsilon/2$. In particular, most of the time lower bounds of Section 3.6 are also valid for approximate consensus.

To reduce asymptotic consensus to approximate consensus, we repeatedly execute approximate consensus with ever smaller parameters $\varepsilon_k = 1/e^k$. Because all processes can decide in the same time step due to the compactness of oblivious models, this temporal composition is immediately possible. The time complexity is almost preserved, modulo a rounding operation:

Lemma 3.23. *Let $\mathcal{M}$ be an oblivious model. If there is an algorithm that solves approximate consensus in time $A \log \frac{\Delta}{\varepsilon}$ in $\mathcal{M}$, then there exists an algorithm that solves asymptotic consensus whose time until ε-agreement is upper-bounded by*

$$T(\varepsilon) \leq A + A \log \frac{\Delta}{\varepsilon}$$

where Δ is the diameter of the set of input values.

3.10 Relation to Stabilizing Consensus

In the stabilizing consensus problem, processes have the same input and output capabilities as in the asymptotic consensus problem, *i.e.*, an initial value

$x_p \in \mathbb{R}^d$ and a continual output $x_p(t) \in \mathbb{R}^d$. For an algorithm to solve stabilizing consensus, each of its executions should satisfy the Agreement condition of asymptotic consensus and also:

- For all processes $p \in [n]$ and all rounds $t \in \mathbb{N}$, the output value $x_p(t)$ is the input value of some process. *(Validity)*
- For all processes $p \in [n]$, the sequence $\big(x_p(t)\big)_{t\geq 0}$ stabilizes. *(Stabilization)*

Since every stabilizing sequence converges and every input value is in the convex hull of input values, it is clear that every algorithm that solves stabilizing consensus also solves asymptotic consensus. The converse is not true. However, stabilizing consensus can be reduced to asymptotic consensus if an estimation on the distance to the common limit value is available, as we will demonstrate below. Such an estimation is necessary in almost all applications. It can be argued that requiring the existence of such an estimation gives a more robust[4] and practically useful problem definition.

In fact, there is a local reduction from the problem of scalar asymptotic consensus with eventual distance estimates to stabilizing consensus. An algorithm that uses the value estimates $x_p(t)$ and distance estimates $\varepsilon_p(t)$ sets its output value $y_p(t)$ for stabilizing consensus with the following rule:

$$y_p(t) = \begin{cases} 0 & \text{if } x_p(t) + \varepsilon_p(t) < 1/2 \\ 1 & \text{else} \end{cases} \tag{3.1}$$

In other words, the algorithm sets $y_p(t) = 0$ if it can determine, based on the current estimates, that the common limit x^* of the asymptotic consensus algorithm is strictly smaller than $1/2$. Otherwise it sets $y_p(t) = 1$. Indeed, if $x^* < 1/2$, then $y_p(t) = 0$ for almost all t since the distance estimates $\varepsilon_p(t)$ converge to zero and the estimates are eventually correct. On the other hand, if $x^* \geq 1/2$, then no correct estimate allows to deduce that $x^* < 1/2$. Thus we have $y_p(t) = 1$ for almost all t in this case since estimates are eventually correct. Validity for stabilizing consensus directly follows from validity for asymptotic consensus.

We cannot give an *a priori* upper bound on the time until stabilization, however. If we could, it would be possible to solve consensus by deciding the current output value after the upper bound. But consensus is not always solvable when asymptotic consensus with distance estimates is. Worse yet, in compact models, stabilizing consensus with uniform stabilization-time upper bound is solvable if and only if consensus is.

[4] If an error estimation is available, then the common limit in asymptotic consensus is a continuous function of the communication pattern.

For the algorithm given in (3.1), even if we have an *a priori* upper bound on the distance estimate $\varepsilon_p(t)$ that converges to zero, we do not get an upper bound on the stabilization time because the limit x^* can be arbitrarily close to $1/2$.

3.11 Approximate Consensus on Graphs

We have seen in Section 3.7 that it is possible to define and solve asymptotic and approximate consensus with a discrete value domain. The specific domain in that section was the discretized unit interval. It is possible, however, to work with much more general discrete value domains. In this section, we present two examples: graphs and semi-latices.

We restrict ourselves to approximate consensus since all convergent sequences in discrete spaces eventually stabilize and it turns out that it is possible to make an irrevocable decision.

While it is possible to study the problem in more generality, in asynchronous Byzantine systems with values from abstract convexity spaces, we stick with our dynamic network models as defined in Chapter 2 for simplicity of exposition.

The notion of distance between two vertices in a graph is straightforward: the length of the shortest path connecting the two vertices. This makes the Approximate Agreement property well-defined in graphs. The Validity property, *i.e.*, the notion of convexity is less obvious. Indeed, multiple definitions of convexity in graphs exist. For our purposes we chose the *monophonic* convexity: A set U of vertices is convex if all vertices on all chordless paths between vertices in U are contained in U. A path is chordless if no pair of non-consecutive vertices on the path form an edge.

A graph is called chordless if it does not contain any induced cycle of length larger than three. Trees are chordless, but cycle graphs of size larger than three are not.

Solving approximate agreement on trees is not very different from solving it in the discretized unit interval. This can be extended to general chordless graphs via clique-tree decomposition:

Theorem 3.24. *Approximate consensus with values in a chordless graph $G = (V, E)$ is solvable in any non-split model in time $O(\log|V|)$.*

The result of Theorem 3.24 can be transferred to semi-lattices. A semi-lattice $L = (V, \oplus)$ is described by a finite set V and a binary *join operator* $\oplus : V \times V \to V$. Convex subsets of V are those that are closed under the join operator $\oplus$. The join operator induces a partial order $\leq$ on V by setting $u \leq v \iff u \oplus v = v$. This partial order, in turn, induces the *comparability graph* $G = (V, E)$ that contains an edge $\{u, v\}$ if and only if u and v are comparable

with respect to the partial order $\leq$. A semi-lattice is called cycle-free if its comparability graph is chordless.

Similarly to Theorem 3.24, we can prove:

Theorem 3.25. *Approximate consensus with values in a cycle-free semi-lattice* $L = (V, \oplus)$ *is solvable in any non-split model in time* $O(\log|V|)$.

3.12 Conclusion

While the results presented in this chapter push the state of the art to a quite complete picture of asymptotic and approximate consensus in several setting, in particular dynamic networks, many related questions remain open. Two important and timely related subjects include distributed optimization and machine learning.

Chapter 4

Exact Consensus

In this chapter, we provide a topological characterization of the solvability of exact consensus in arbitrary models.[5] Contrary to earlier work on topological methods in distributed computing [84], we do not use combinatorial topology but point-set topology.

This approach has first been taken by Alpern and Schneider [8] to characterize safety and liveness properties, but was not followed up on for a long time. They defined a topology on the set of executions, and showed that safety properties exactly correspond to closed sets and that liveness properties exactly correspond to dense sets. The fact that every property is an intersection of a safety and a liveness property then translates to the fact that every subset of a topological space is the intersection of a closed and a dense set.

We consider a different topology on the set of executions (and the set of communication patterns) than Alpern and Schneider. Succinctly put, we focus on the *last* process to see a difference between two executions rather than the first. It is this change to the original topology that allows us to exactly characterize the decision problem that is consensus. Figure 4.1 illustrates the conceptual difference between the two topological approaches.

Additional details on the results of this chapter can be found in our PODC 2019 paper [106].

4.1 Problem Definition

In the exact consensus problem, every process p starts with an input value x_p from some set $\mathcal{V}$ of possible values, and has a write-once decision variable y_p. For an algorithm to solve exact consensus, each of its executions should satisfy:

[5] In fact, the characterization is not limited to models that are described by message adversaries, but can be extended to more general distributed computing models. Also those that contain the notion of a faulty process.

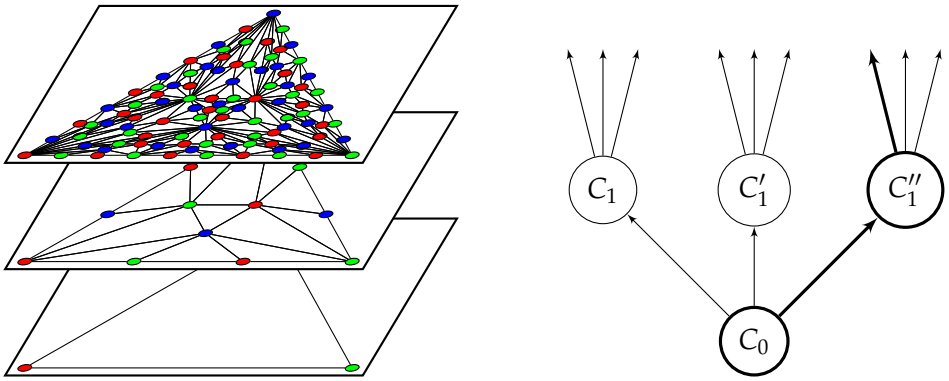

Figure 4.1: Comparison of the combinatorial-topology approach and the point-set-topology approach: The combinatorial-topology approach (left) studies sequences of increasingly refined spaces in which the objects of interest are simplices (corresponding to configurations). The point-set-topology approach (right) studies a single space in which the objects of interest are executions (*i.e.*, infinite sequences of configurations).

- For all processes $p, q \in [n]$, if both p and q have decided, then their decision values are equal: $y_p = y_q$. *(Agreement)*
- If all processes have the same initial value v, *i.e.*, $\forall p \in [n]\colon x_p = v$, then v is the only possible decision value. *(Validity)*
- Every process $p \in [n]$ eventually decides. *(Termination)*

The problem as defined here is also denoted *uniform* consensus, due to the fact that all processes have to decide. In the non-uniform consensus problem, only non-faulty processes have to decide. As the notion of a faulty process does not exist in the system model as we defined it in Chapter 2, the two definitions coincide in our case. It is, however, possible to generalize the characterization we give to non-uniform consensus, in models in which the notion exists. In that case, we need to modify the definition of the topology to consider the last non-faulty process that can distinguish two executions.

4.2 State of the Art

Santoro and Widmayer [115] provided the first comprehensive characterization of consensus solvability in synchronous distributed systems prone to communication errors. Using bivalence arguments [65], they proved that consensus is impossible if up to $n - 1$ messages may be lost (by the same process) in each round. Schmid *et al.* [117] showed that consensus can even

be solved when a quadratic number of messages is lost per round, provided these losses do not isolate the processes.

The above results were substantially refined by Coulouma *et al.* [48]. They identified a property of an equivalence relation defined on the set of communication graphs, which exactly captures consensus solvability in the oblivious setting. A universal consensus algorithm was also given, which, *e.g.*, allows to solve consensus when the set of possible graphs for $n = 2$ consists of $\{\leftarrow, \rightarrow\}$.

The situation is considerably more complex for non-oblivious message adversaries, where the set of possible graphs may change over time. In sharp contrast to oblivious message adversaries, it need not be the case that the set of graph sequences is limit-closed, *i.e.*, the model does not need to be compact [92]. For example, eventually stabilizing message adversaries like the vertex-stable source component (VSSC) message adversaries [26, 125] guarantee that some rounds with "good" communication graphs (a VSSC, which allows a consensus algorithm to terminate) occur eventually. However, limits of these sequences, in which the VSSC would never appear, are of course not in the message adversary.

For the special case of $n = 2$, Fevat and Godard [63] provided a complete characterization of consensus solvability for non-oblivious message adversaries as well: Using a bivalence argument, they showed that certain graph sequences (a "fair sequence" or a special pair of "unfair sequences") must not be in the message adversary to render consensus solvable, and provided a universal algorithm for this case. However, a complete characterization of consensus solvability for arbitrary system sizes did not exist until now.

4.3 Point-Set Topology on Executions

We now define the topology used for the consensus characterization. For that, we use the notion of *full-information* executions. These executions occur when every process continually sends all the information it gathered (*i.e.*, message receptions, initial values) to the other processes. We thus consider the class of algorithms that does not place any restrictions on the maximum size of messages or local memory. In the message-adversary model as introduced in Chapter 2, full-information executions can be captured by process-time graphs [23]. The formalization we give here is more general, however.

Formally, a full-information execution is a sequence of full-information configurations. For every process p, there is an equivalence relation $\sim_p$ on the set $\mathcal{C}$ of full-information configurations. We have $C \sim_p D$ if p cannot distinguish the two configurations C and D. That is, process p is in the same round, has the same initial value, and received the same messages in both configurations. In the message-adversary model, two executions that are indistinguishable for all processes are equal. This is not necessarily the case

in models whose configurations include some state of the environment not accessible to any process.

These indistinguishability relations can be expressed via the following distance function on the set $\mathcal{C}$ of configurations:

$$d_p(C, D) = \begin{cases} 0 & \text{if } C \sim_p D \\ 1 & \text{else} \end{cases}$$

In the topology induced by this distance function, it is not possible to separate configurations that are indistinguishable for process p.

The extension to executions, *i.e.*, sequences of configurations, can be done by using the product topology of the infinite product $\mathcal{C}^\omega = \mathcal{C} \times \mathcal{C} \times \mathcal{C} \times \cdots$ where every copy of $\mathcal{C}$ is equipped with the topology induced by d_p. This product topology is induced by the distance function

$$d_p(\gamma, \delta) = 2^{-\inf\{t \geq 0 | C_t \not\sim_p D_t\}}$$

where $\gamma = (C_t)_{t\geq 0}$ and $\delta = (D_t)_{t\geq 0}$. The round number $\inf\{t \geq 0 \mid C_t \not\sim_p D_t\}$ is the first in which process p can distinguish executions γ and δ. Two executions that p can never distinguish have zero distance.

To arrive at a single distance function that captures consensus solvability, we take the minimum over all individual distance functions:

$$d(\gamma, \delta) = \min_{p\in[n]} d_p(\gamma, \delta) \tag{4.1}$$

That is, we identify the time that the *last* process sees a difference in two executions. Historically, the topology that is induced by the *maximum* of the d_p has been considered [8, 94, 104, 77]. This definition has the advantage that the resulting distance function is a pseudo-metric, often even a metric. That is, the triangle inequality holds. Semantically, however, it focuses on the time that the *first* process sees a difference in two executions. This can be used to show impossibility results [94, 104], but cannot characterize solvability of exact consensus since another process that does not yet see a difference can force the decision value.

It is true that the triangle inequality does not hold for our distance function in (4.1). Nonetheless, it induces a topology. In fact, a very general result can be shown:

Lemma 4.1. *Let X be a nonempty set and let $d : X \times X \to \mathbb{R}$ be a function. Then*

$$\mathcal{T}_d = \{O \subseteq X \mid \forall x \in O\ \exists \varepsilon > 0 \colon B_\varepsilon(x) \subseteq O\}$$

is a topology on X where

$$B_\varepsilon(x) = \{y \in X \mid d(x, y) < \varepsilon\} \quad .$$

In the sequel, we equip the set Σ of full-information executions with the topology induced by the distance function (4.1).

4.4 Characterization of Consensus Solvability

Since we consider the space of full-information executions, a decision algorithm is simply a collection of functions $\delta_p : \mathcal{C} \to \mathcal{V} \cup \{\perp\}$ such that $\Delta_p(C) = \Delta_p(D)$ if $C \sim_p D$ and $\Delta_p(C') = \Delta_p(C)$ if C' is reachable from C and $\Delta_p(C) \neq \perp$. Here, $\Delta_p(C) = \perp$ represents the fact that process p has not yet decided in configuration C. In other words, decisions only depend on local information and are irrevocable. We can extend the decision functions to executions by setting $\Delta_p(\gamma) = \lim_{t\to\infty} \Delta_p(C_t)$ where $\gamma = (C_t)_{t\geq 0}$. In the case of exact consensus, we can also define a global decision function $\Delta(\gamma) = \Delta_p(\gamma)$ for an arbitrary process p. It is guaranteed that $\Delta(\gamma) \neq \perp$.

It turns out that the decision function is continuous in our topology. In fact, it is locally constant, *i.e.*, every executions has an ε-neighborhood on which the decision function is constant. This can be seen by choosing $\varepsilon = 2^{-T}$ where T is a round number by which all processes have already decided.

Lemma 4.2. *Let $\Delta : \Sigma \to \mathcal{V}$ be the decision function of an algorithm that solves exact consensus. Then Δ is continuous.*

The defining property of the decision function Δ being continuous is that the inverse image of every open set in $\mathcal{V}$ is open in Σ. The discrete topology being placed on the value set $\mathcal{V}$, every subset of $\mathcal{V}$ is open, in particular the singleton sets $\{v\}$. This means that the set Σ_v of executions for which a correct consensus algorithm decides v is an open subset of Σ. To complete the solvability characterization, one needs to also prove the inverse: Each partition of Σ into open subsets Σ_v ($v \in \mathcal{V}$) can be locally recognized by all processes. The time until a process can recognize in which Σ_v the current execution lies in can vary from process to process, and from execution to execution.

Theorem 4.3. *Exact consensus is solvable if and only if there exists a partition of the set Σ of full-information executions into sets Σ_v ($v \in \mathcal{V}$) such that the following holds:*

1. *Every Σ_v is an open subset of Σ.*
2. *If all processes have the same initial value v in execution $\gamma \in \Sigma$, then $\gamma \in \Sigma_v$.*

This gives rise to the following meta-procedure to determine whether consensus is solvable, and constructing an algorithm if possible. It requires knowledge of the connected components of the execution space Σ.

1. Initially, start with an empty set Σ_v for all values $v \in \mathcal{V}$.
2. Add to Σ_v every connected component that includes an executions in which all processes have initial value v.
3. Add every remaining connected component to an arbitrarily chosen Σ_v.

4. If the sets Σ_v are pairwise disjoint, then exact consensus is solvable. In this case, each set Σ_v is equal to the inverse image $\Delta^{-1}[\{v\}]$ of a canonically constructed algorithm that solves exact consensus. If the Σ_v are not pairwise disjoint, then exact consensus is not solvable.

4.5 Conclusion

The use of point-set topology allowed a characterization of the solvability of a widely used decision problem in a very general class of models—in particular non-compact models—for the first time. It is thus a promising direction for future investigation. One possible extension is to generalize the consensus characterization to other models, *e.g.*, Byzantine processes, randomized algorithms, message-size restrictions, *etc*. Another important extension is, of course, to other decision problems.

Chapter 5

Improved Bound from Non-Splitness to Broadcastability

Few structural results are known about the class of non-split graphs. In this section, we present the result that any product of at least $\Omega(\log \log n)$ non-split communication graphs is broadcastable. This is an exponential improvement over the previously best bound of at least $\Omega(\log n)$ communication graphs [42]. Combined with the reduction from rooted to non-split models (Lemma 2.1), this also shows that the radius of at least $\Omega(n \log \log n)$ rooted communication graphs is broadcastable, which is smaller than the $\Omega(n \log n)$ bound of previous results.

More details can be found in our article in the journal *Discrete Applied Mathematics* [73].

5.1 State of the Art

Information dissemination has been studied in a variety of settings [66, 81, 82, 87, 46]. Only some of these works have considered dynamic networks.

Kuhn *et al.* [91] considered the same framework of dynamic networks as introduced in Chapter 2. They assumed, however, that all communication graphs are bi-directional and connected. In this case, they showed a lower bound $\Omega(n \log n)$ and an upper bound of $O(n^2)$ for the problem of all-to-all token dissemination, even if the number n of processes is unknown.

Charron-Bost and Schiper [42] showed that $O(\log n)$ non-split rounds are sufficient for broadcastability.

Zeiner *et al.* [130] showed an upper bound of $O(n)$ rooted rounds until broadcastability if every communication graph is a tree with a constant number of leaves and inner vertices.

5.2 A Lower Bound

No non-constant lower bound on the length of broadcastable products of non-split graphs is known. Figure 5.1 shows a non-trivial lower bound of 3 on this length. It contains a single non-split graph. For products of multiple copies of the same graph, the question about the length of products until broadcastability is the question of finding the graph's radius. Preliminary computer search found a non-split graph of radius 4.

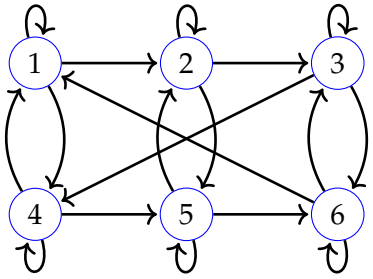

Figure 5.1: Non-split graph with radius 3

5.3 Upper Bound

We now give a sketch of the proof of the $O(\log \log n)$ upper bound on the length of products of non-split graphs required for broadcastability.

The best previously known bound was $O(\log n)$. Its proof iteratively uses the definition of non-splitness: In the first step, one shows that $\lceil n/2 \rceil$ processes suffice to reach all n processes via a single non-split graph, *i.e.*, an increment of 1 to the length of the product. In the second step, one uses the same argument again to show that $\approx n/4$ processes suffice to reach all n processes via a sequence of any two non-split graphs. Repeating this $\Theta(\log n)$ times, a single process suffices to reach all n processes via a sequence of any $O(\log n)$ non-split graphs.

For the $O(\log \log n)$ upper bound, we re-use the same argument multiple times, but supplement it with a counting argument at depth $\Theta(\log \log n)$. The core technical result is formalized in the following lemma:

Lemma 5.1. *Let U and W be finite sets with $|U| = k$ and $|W| = n$, and let $f : \binom{U}{\lfloor \log n \rfloor} \to W$ be a function. If $n \geq 8$, then there exists some $w \in W$ such that $\left|\bigcup f^{-1}[\{w\}]\right| \geq k/e^4$.*

Using the basic iterative argument, we show that any process set of size $O(\log n)$ can be reached by a single process via a sequence of $O(\log \log n)$ non-split graphs. We use Lemma 5.1 to show that there is a set of $O(\log n)$ processes that can reach all n processes via $O(\log \log n)$ non-split graphs.

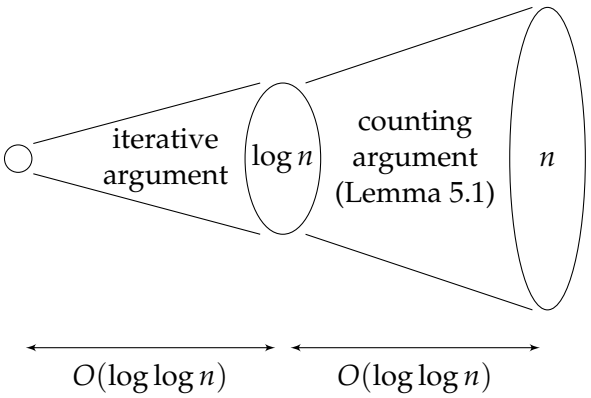

Figure 5.2: Proof of the $O(\log\log n)$ upper bound

The function f in the lemma statement is chosen such that $f(A)$ is any process that covers all processes in the set A of size $\Theta(\log n)$ via a sequence of $\Theta(\log\log n)$ non-split graphs. Again, the existence of $f(A)$ follows from the basic iterative argument. Combining both, we get the existence of a single process that reaches all n processes via a sequence of $O(\log\log n)$ non-split graphs. Figure 5.2 shows this proof structure.

5.4 Conclusion

The structure of non-split graphs is still poorly understood. The results of this chapter are but a modest step in the direction of a more complete understanding. A deeper investigation of the class of non-split graphs is interesting, not only due to the seeming simplicity of its definition, but also due to their occurrence in classical models of distributed computing.

Chapter 6

Non-Uniform Population Protocols

In this chapter, we present a study of population protocols with non-uniform random schedulers. Classically, when a random scheduler is considered in population protocols, it is assumed to be uniform, *i.e.*, at every time step, every pair of agents[6] has the same probability of being activated. Such a uniformity assumption is often made when no specific knowledge of the distributions of activations is available; it is also often made for mathematical convenience. It is, however, not always a realistic assumption.

An immediate example is a population of mobile agents in which agents move at different speeds, with an interaction occurring when two agents meet. In this population, faster agents have a higher rate of activation than slower agents. In a population in which all agents have different speeds, every pair of agents will have a different probability of being activated in a given time step.

Another example are chemical reaction networks. Here, agents are partitioned into different species. A set of reactions determines the evolution of the population. A reaction removes a set of agents, one of each of the reactant species, and produces a set of agents, one of each of the product species. The rate of a reaction is proportional to its rate constant and the counts of each of the reactant species. We use chemical reaction networks to model growing population of bacteria in Chapter 7.

For more details on the results of this chapter, see our papers at Algosensors 2017 [20] and in *Theoretical Computer Science* [128].

[6] We follow the established terminology for population protocols and refer to processes as agents in this chapter.

6.1 Problem and Model Definition

As in the standard population protocol model, we assume a fixed set of n agents. We do require the existence of a unique distinguished agent—the base station (BST). For convenience, we assign the number 0 as the base station's identifier. The remaining agents, which we call mobile agents, are identified by the numbers $1, 2, \ldots, n-1$.

The random sequence of interactions is determined by the probability matrix $\left(P_{i,j}\right)$. At each time step, the scheduler chooses the pair (i, j) of agents to interaction with probability $P_{i,j}$, independently of the choice in other time steps. In this case, agent i takes the role of initiator in the interaction and agent j takes the role of responder. No agent is chosen to interaction with itself, that is, we have $P_{i,i} = 0$ for all agents i. Since there is exactly one pair of agents chosen to interact, we have $\sum_{i,j} P_{i,j} = 1$. We allow asymmetric interactions in the model. The usual uniform scheduler has $P_{i,j} = 1/n(n-1)$ for $i \neq j$. The combination of an interaction schedule, a protocol, *i.e.*, a state-transition function for pairs of agents, and an initial configuration defines an execution, *i.e.*, a sequence of configurations.

We study the problem of data collection. Every mobile agent initially carries some data. The goal is to collect all data items at the base station. For that, an agent may transfer its data items to another agent during an interaction. This entails the removal of the data item from the first agent. It is assumed that an agent can hold an arbitrary amount of data items.

The measure of time complexity that we employ is that of step complexity, *i.e.*, the number of time steps until completion of the task. When studying the uniform random scheduler, it is customary to measure time complexity in terms of *parallel* time. Parallel time is defined as the number of time steps divided by the number n of agents. This definition is motivated by a lower-level system model in which each agent is equipped with a local Poisson clock with rate 1. When the local clock is activated, the agent takes part in an interaction together with another agents chosen uniformly at random. The expected time until the next interaction is thus equal to $1/n$. With a non-uniform scheduler, the lower-level system model would have to be adapted, with non-uniform clock rates and non-uniform distributions for choosing the other agent in the interaction. This means that the expected time until the next interaction is not necessarily equal to $1/n$. We do not develop the non-uniform lower-level system model here. Rather, we stick with the discrete time measurement of counting the number of time steps.

For a non-uniform random scheduler, we can define the cover time cv_i for every agent i. It is defined as the expected time until agent i interacts at least once with all other agents. We have the formula

$$\mathrm{cv}_i = \int_0^\infty 1 - \prod_{j \in [n] \setminus \{i\}} \left(1 - e^{-(P_{i,j} + P_{j,i})t}\right) \, dt \ .$$

For the uniform scheduler, we recover the well-known number of $\Theta(n^2 \log n)$ steps until agent i interacts with all other agents. Agents do not have access to their cover time. However, we do assume that two interacting agents can compare their cover times, *i.e.*, determine which of the two agents, if any, has a lower cover time than the other.

6.2 State of the Art

Data collection in population protocols was studied by Beauquier *et al.* [19] in the context of adversarial scheduling. They also considered agents of different speeds, although not with a random scheduler.

Xu *et al.* [127] presented an energy model for population protocols, which we adapt for our purposes The energy complexity is evaluated by the expected total energy spent for establishing the interactions until convergence.

The uniform random scheduler has been introduced in the context of population protocols by Angluin *et al.* [11]. Common problems that are studied in this uniform model are leader election [6, 56, 79] and exact majority [7, 78, 24]. The performance of the solutions to these problems are evaluated by the parallel expected convergence time and by the number of states available at each agent (space complexity). Trade-offs between time and space complexities of protocols solving these problems have been studied [4]. Any leader election or majority protocol converges in $\Omega(n/\text{polylog } n)$ expected time using $\frac{1}{2} \log \log n$ states. In [5], authors show that by employing "phase clocks", both problems can be solved in $O(\log^2 n)$ expected time, using $O(\log n)$ states. Recently, these performances are further improved. Leader election can be achieved in $O(\log n)$ time w.h.p. using $O(\log \log n)$ states [25]. Exact majority can be solved in $O(\log^{5/3} n)$ time in expectation and w.h.p. using $\Theta(\log n)$ states [24]. Other interesting problems, such as counting [13, 55], community detection [21], proportion computation [100], function computation [22], proportion and plurality consensus [47, 78], have been also studied under the similar uniform scheduler model. Besides the uniformly random scheduling independent of the agents' states, there are works assuming a scheduling depending on the states of agents, like the *transition function* scheduler [43] or the scheduling of reactions in chemical reaction networks according to the model of stochastic chemical kinetics [49].

The randomized gossip algorithm of Boyd *et al.* [31] was designed for the problem of averaging in an arbitrary connected networks. There, each agent runs an independent Poisson clock (asynchronous time model), and at each clock tick, the agent randomly selects a neighbor, with the probability given by the algorithm. Then, it averages its value with the chosen neighbor. Observe that this algorithm can be seen as a population protocol with a non-uniform random scheduler, in which two interacting agents average their values.

6.3 Time Lower Bounds

We start by giving three non-trivial lower bounds on the expected convergence time of protocols that solve the data collection problem.

The first is based on a generalization of the coupon collector's problem. Each agent has to take part in at least one interaction for data collection to be solved. Contrary to the classical coupon collector's problem, each interaction covers two agents instead of a single one and the activation probabilities are not uniform. This generalization of the classical coupon collector's problem has been studied, among others, by Stadje [120], Adler and Ross [2], and Ferrante and Saltalamacchia [62]. By showing that any coupon collector's process with n coupons in which every step covers g coupons takes an expected number of $\frac{n}{g} \log \frac{n}{g}$ steps, setting $g = 2$, we get:

Theorem 6.1. *The expected convergence time of any protocol solving data collection is* $\Omega(n \log n)$.

An even more naïve approach directly bounds the time until the slowest agent is activated at least once:

Theorem 6.2. *The expected convergence time of any protocol solving data collection is* $\Omega\left(\max_{i\in[n]} \; 1/\sum_{j=1}^{n} \left(P_{i,j} + P_{j,i}\right)\right)$.

In particular, the protocol that has mobile agents transfer their data items only directly to the base station has expected convergence time at least in the order of $\Omega\left(\max_{i\in[n]} \; (1/P_{i,\mathrm{BST}} + P_{\mathrm{BST},i})\right)$. If $\min_i P_{i,\mathrm{BST}} + P_{\mathrm{BST},i} \ll 1/n^2$, this translates into a large convergence time of this naïve protocol.

Since all data items have to be transferred to the base station, the base station has to interact at least once with a mobile agent:

Theorem 6.3. *The expected convergence time of any protocol solving data collection is* $\Omega\left(\min_{i\in[n]} \; 1/\left(P_{i,\mathrm{BST}} + P_{\mathrm{BST},i}\right)\right)$.

For the special case of the uniform scheduler, Theorem 6.3 gives a lower bound of $\Omega(n^2)$ on the expected convergence time.

6.4 Time Upper Bounds

We start with an analysis of the transfer-to-faster (TTF) protocol. In this protocol, a mobile agent transfers all its data items to another mobile agent if the other agent is faster, *i.e.*, has a smaller cover time. A mobile agent always transfers its data items to the base station when possible.

At least from an external-analysis perspective, we can *a priori* partition the set of mobile agents into different categories according to their cover time. We thus map every agent i to its cover-time category ca_i such that $\mathrm{ca}_i < \mathrm{ca}_j$ if and only if $\mathrm{cv}_i < \mathrm{cv}_j$. We denote the number of different cover-time categories by m. In the worst case we can have $m = n$. For the uniform scheduler we have $m = 1$.

For the following analysis we associate a vector of nonnegative integers $x \in \mathbb{N}_0^n$ with every configuration of the TTF protocol. Each entry of the vector contains the number of data items at an agent. Since data items are only moved, never copied, we have $\sum_i^n x_i = n - 1$. Without loss of generality we fix the identifier of the base station to $\mathrm{BST} = 1$. For the TTF protocol, it is sufficient to study the vectors associated to configurations.

Since every mobile agent initially holds one data item, we start at $x_{\mathrm{init}} = 1 - e_1$ where 1 denotes the vector of all ones and e_i denotes the i^{th} standard-basis vector. The terminal vector is $x_{\mathrm{end}} = (n-1)e_1$, when the base station has collected all $n - 1$ data items.

We denote by x^t the vector after t time steps. This random sequence of vectors evolves as $x^{t+1} = W(t)x^t$ with the random matrix satisfying $W(t) = I + e_i \cdot {}^t e_j - e_j \cdot {}^t e_i$ if $\mathrm{ca}_i < \mathrm{ca}_j$, *i.e.*, agent j transfers all its data items to agent i according to the TTF protocol.

To measure the distance of the current to the terminal configuration we use the following measure:

$$d_\gamma(x) = \|(x - x_{\mathrm{end}}) \circ \gamma\|_2$$

where $x \circ y$ denotes the component-wise product of x and y. The vector γ is chosen appropriately such that $d_\gamma(x^{t+1}) \leq d_\gamma(x^t)$ for all times t, with a strict inequality whenever a data transfer occurs. Specifically, we choose $\gamma_{\mathrm{BST}} = 0$, $\gamma_i = 1$ for all i in the fastest category, and $\gamma_j = \gamma_i\sqrt{2n}$ whenever is i is faster than j and no other category is between ca_i and ca_j.

We now state our main upper bound on the convergence time of the TTF protocol. Without loss of generality, we assume that $\mathrm{ca}_2 \leq \mathrm{ca}_3 \leq \cdots \leq \mathrm{ca}_n$. In particular we have $\gamma_n = (2n)^{(m-1)/2}$.

Theorem 6.4. *The convergence time of the TTF protocol is at most* $\frac{m \log n}{\log \lambda_2^{-1}(\tilde{W})}$ *in expected value and with high probability, where* γ *is defined above,* $\Gamma_{i,j} = \gamma_i/\gamma_j$, $\tilde{W} = \sum_{i<j \wedge \mathrm{ca}_i < \mathrm{ca}_j} (P_{i,j} + P_{j,i}) W_{ij}^{\Gamma^2} + \sum_{i<j \wedge \mathrm{ca}_i = \mathrm{ca}_j} (P_{i,j} + P_{j,i}) I$, $W_{ij}^{\Gamma^2} = I + \Gamma_{i,j}(e_i \cdot {}^t e_j + e_j \cdot {}^t e_i) + (\Gamma_{i,j}^2 - 1) e_j \cdot {}^t e_j$, *and* $\lambda_2(A)$ *denotes the modulus of the second largest eigenvalue of matrix* A.

The proof of the theorem is based on a spectral analysis of the involved matrices when passing from one vector to the next in the random execution.

For the special case of a uniform scheduler, Theorem 6.4 yields an upper bound of $O(n^2 \log n)$ since then $\lambda_2(\tilde{W}) = 1 - \frac{2}{n \cdot (n-1)}$. The lower bound of

Theorem 6.3 is equal to $\Omega(n^2)$ in this case. A direct analysis of the TTF protocol with the uniform scheduler confirms that it takes $\Theta(n^2 \log n)$ interactions, *i.e.*, the upper bound of Theorem 6.4 turns out to be tight.

6.5 Energy Consumption

We now turn to the question of energy consumption of the TTF protocol. For that, we introduce a variant, the *lazy* TTF protocol, which we show to have lower energy consumption, at least in some cases.

In the lazy TTF protocol, each interaction has a certain probability of not having any effect, *i.e.*, of no data items being transferred, irrespective of the available items and the agents' cover times. More specifically, every agent i has a probability p_i that an interaction in which i is the initiator follows the TTF protocol. The original TTF protocol corresponds to the case of $p_i = 1$ for all agents i.

In terms of the probability matrix P of a non-uniform scheduler, the lazy TTF protocol with scheduler P is equivalent to the original TTF protocol with scheduler $P' = P \circ (p \cdot {}^t1)$ where $p = (p_i)_{i \in [n]}$ is the vector of probabilities p_i. The matrix P', however, is not a probability matrix in general, as the sum of its elements can be strictly smaller than 0. This requires a slight generalization of the system model such that, if no pair is chosen to interact in a given time step, no interaction takes place and the configuration remains the same.

Using the same arguments as for Theorem 6.4, we can prove:

Theorem 6.5. *The convergence time of the lazy TTF protocol is at most* $\frac{m \log 2n}{\log \lambda_2^{-1}(\tilde{W})}$ *with high probability*

$$\text{where } \tilde{W} = \sum_{cv_i < cv_j} (P_{i,j} p_i + P_{j,i} p_j) W_{ij}^{\Gamma^2} + \sum_{cv_i < cv_j} (P_{i,j}(1 - p_i) + P_{j,i}(1 - p_j) I$$
$$+ \sum_{cv_i = cv_j} (P_{i,j} + P_{j_i}) I, \text{and } W_{ij}^{\Gamma^2} = I + \Gamma_{i,j}(e_i \cdot {}^t e_j + e_j \cdot {}^t e_i) + (\Gamma_{i,j}^2 - 1) e_j \cdot {}^t e_j.$$

To minimize the time complexity of lazy TTF, one can try to minimize its upper bound in Theorem 6.5. We investigated this approach by randomly generating a set of scheduler matrices and initiation probabilities, and comparing the upper bound of Theorem 6.5 with the observed w.h.p. convergence time. Figure 6.1 shows an approximately linear relationship between the two numbers. It is hence not unreasonable to employ the heuristic of minimizing the upper bound of Theorem 6.5 to minimize the time complexity of lazy TTF.

Minimizing the upper bound of Theorem 6.5 means solving the following optimization problem.

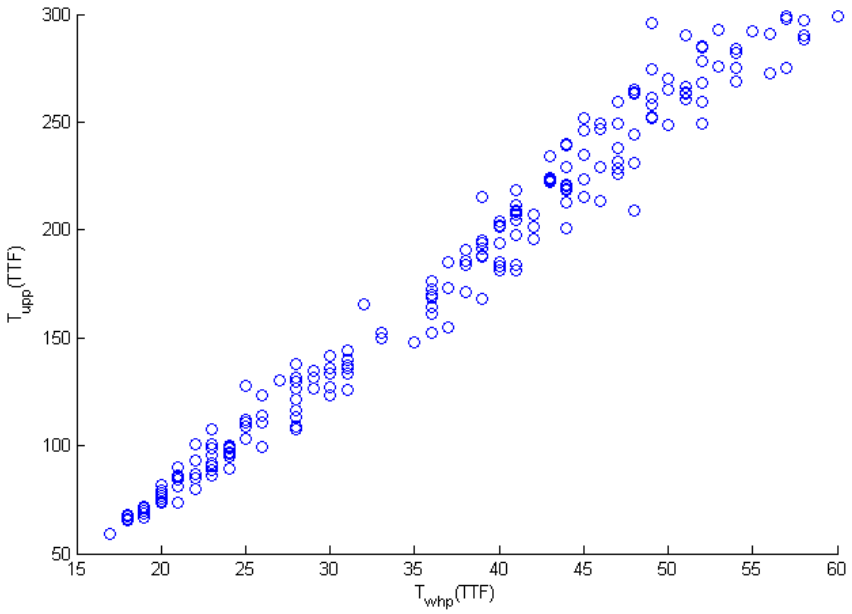

Figure 6.1: Relation between the w.h.p. convergence time T_{whp} and the upper bound T_{upp} of Theorem 6.5

$$\begin{aligned} \text{OP}_1: \quad & \min_{p \in \mathbb{R}^n} && \lambda_2(\tilde{W}) \\ & s.t && \forall i \in \{1, ..., n\}: \quad 0 \leq p_i \leq 1 \end{aligned}$$

The optimization problem OP_1 is *a priori* not convex. It can, however, be transformed into the following convex optimization problem:

$$\begin{aligned} \text{OP}_2: \quad & \min_{\substack{p \in \mathbb{R}^n \\ s \in \mathbb{R}^n}} && s \\ & s.t && sI - \tilde{W} \succeq 0 \\ & && \forall i \in \{1, ..., n\}: \quad 0 \leq p_i \leq 1 \end{aligned}$$

It turns out that minimizing the upper bound of Theorem 6.5 also tends to minimize the protocol's energy consumption. Here, we assume that the dominant factor of energy consumption is the number of steps in which the initiator agent decides to have the interaction follow the TTF protocol. For energy consumption, we thus count the number of steps in which the lazy TTF protocol follows the TTF protocol.

We ran several simulations in randomly chosen systems of sizes 4–8 to test our hypothesis. As can be seen in Table 6.1, the time complexity tends

increase while the energy consumption tends to decrease when solving optimization problem OP_2.

n	time gap	energy gap
4	11.60%	-15.32%
5	17.10%	-23.60%
6	22.04%	-30.79%
7	26.31%	-36.99%
8	27.41%	-39.07%

Table 6.1: Average relative gaps on time and energy between the original TTF protocol and the lazy TTF protocol using the optimum of OP_2

6.6 Conclusion

The results of the present chapter are among the first to consider non-uniform random schedulers. Virtually all other investigations of probabilistic behavior of population protocols are based on the uniform scheduler. The mathematical analysis of non-uniform schedulers tends to be much more involved, which could be one explanation for the lack of works in this direction. The gain in generality, on the other hand can make for a more useful model in specific application settings. It is thus worthwhile to continue this research direction.

When the non-uniformity of rates of interaction originates from a relatively small number of different categories of interaction rates, then the stochastic dynamics are likely to be reasonably captured by the chemical reaction network model, which is studied in the next chapter. If the non-uniformity originates from individual differences between the agents, however, then a more specific treatment is necessary. Our techniques using spectral theory appear to be limited to the data-collection problem. It seems likely that new techniques have to be developed for studying other problems with non-uniform schedulers.

Chapter 7

Computation with Continual Population Growth

In this chapter, we turn to the problem of computing with bacterial colonies.

The behavior of naturally occurring cells, like bacteria, can be influenced by adding external DNA into the cells. This synthetic biology approach is used in several applications, including the production of metabolic compounds of interest [109], bio-remediation of toxic environments [122], sensing of disease bio-markers [119], and therapeutic intervention by targeted effector delivery [9].

Adding too many processes to a cell leads to stress due to resource limitation [111, 121]. One solution is to distribute the processes to multiple cell types. This, of course, leads to further challenges like orthogonality of communication signals, the bandwidth of communication channels, cellular growth and its effect on signal amplification or dissipation, and effect of cross-talk between different signals.

In this chapter, we will introduce and analyze a distributed amplification protocol, the *A-B protocol*, and a distributed NAND-gate protocol for populations of duplicating agents. This is a first step towards an accurate mathematical modeling of interacting bacteria.

From a mathematical perspective, we model populations of interacting bacteria in the language of chemical reaction networks. For a species X, we will write $X(t)$ for the count of X at time t. The random function $X\colon \mathbb{R}_+ \to \mathbb{N}_0$ is always right-continuous.

In this chapter, we will use the term "with high probability" with respect to the total initial population. That is, if the the sum of the counts of species at time $t = 0$ is n, then an event happens with high probability if its probability is at least $1 - O(1/n^c)$ for some $c > 0$.

More details can be found in our DISC 2020 paper [45]. For more background on the similarities and the differences of bacterial and electronic circuit design, see, *e.g.*, our book chapter in *Advances in Synthetic Biology* [67].

7.1 State of the Art

Angluin *et al.* [12] introduced and analyzed a three-state population protocol that has transitions denoted "duplication" and "cancellation". Like our A-B protocol, it has states denotes A and B. However, it being a population protocol, a "duplication" does not actually change the overall population count. It rather denotes a transition when an A agent meets a "blank" agent to both agents being in state A. Similarly, a "cancellation" does not decrease the population count, but rather transforms an A or a B agent into a "blank" agent. While the three-state protocol bears similarity to our A-B protocol, the population protocol model is suitable for our purposes due to the requirement of a population count that remains constant over the course of the execution.

Birth-death processes, which track a population with "birth" and "death" events over time, have been studied in the mathematics literature for a long time [123, 61]. They processes have been used to model competition, predation, or infection in evolutionary biology, ecology, genetics, and queuing theory [103, 114]. General analytical results, however, are often limited to single-species processes [32]. Some extensions for multiples species, with applications to genetic mutations, are found in the literature on competition and branching processes [27, 89, 112]. The closest to our work is by Ridler-Rowe [113] who considered a similar stochastic process between two competing species. The results, however, were limited in scope. We present a more precise analysis here.

7.2 Majority Consensus

In this section, we present the *A-B protocol* for (approximate) majority consensus. It contains two species, A and B, and a single non-duplication reaction. This reaction has one A and one B annihilate each other when they meet. The complete set of reactions thus is

$$A \xrightarrow{\gamma} 2A \qquad B \xrightarrow{\gamma} 2B \qquad A + B \xrightarrow{\delta} \emptyset$$

where $\gamma > 0$ is the duplication rate and $\delta > 0$ is a rate constant.

It turns out that this simple protocol solves the majority consensus problem fast and with high probability if the initial gap between the two species is sufficiently large. Majority consensus is reached if the initial minority species has reached count zero. If both species start with the same count, then majority consensus is reached if one of the species has reached count zero. We say that consensus is reached if one of the species has reached count zero (no matter which one).

It turns out that the A-B protocol reaches majority consensus in constant time $O(1)$. This can be shown via a coupling between the continuous-time Markov chain of the A-B protocol and that of a one-species process that can

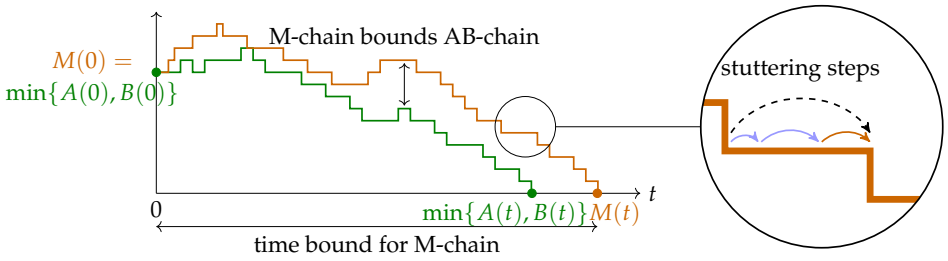

Figure 7.1: Proof overview for time bound: Construction of a continuous-time coupling of the A-B protocol and the single species birth-death M-chain. The M-chain is always an upper-bound for the minimum in the AB-chain. Stuttering steps occur but create the same continuous-time chain.

be shown to go to extinction fast and that is always an upper-bound on the minority species in the execution of the A-B protocol. Figure 7.1 summarizes the proof structure. An easy repetition of Bernoulli trials extends this result to time $O(\log n)$ with high probability.

Theorem 7.1. *The A-B protocol reaches consensus in expected time $O(1)$ and in time $O(\log n)$ with high probability.*

As to the probability with which majority consensus is reached, we employ a *discrete-time* coupling of the protocol's jump chain, *i.e.*, the Markov chain in which only the sequence of state changes is recorded, but not their timing. The second discrete-time process for the coupling is an urn process [96]: We repeatedly draw from an urn containing white and black balls, and put back two of whatever kind we have drawn. This urn process evolves as the discrete-time version of two parallel birth-only processes. The probability of ever having the same number of white and black balls in the urn is proportional to the incomplete beta function $I_{1/2}(W_0, B_0)$ where W_0 and B_0 are the initial number of white and black balls, respectively. The coupling is done in such a way to show that the probability of having the same number of white and black balls is higher than that of having the same number of species A and B. An analysis of the occurring incomplete beta function finally shows that an initial gap roughly in the order of $\sqrt{n}$ suffices to have majority consensus with high probability. In particular, it is not necessary to have a linear initial gap, *e.g.*, to require that the majority represent at least 51% of the initial population.

Theorem 7.2. *The A-B protocol reaches majority consensus with probability $1 - e^{-\Omega(\Delta^2/n)}$ where $\Delta = |A(0) - B(0)|$ is the initial gap.*

In particular, it reaches majority consensus with high probability if the initial gap is lower-bounded by $\Delta = \Omega(\sqrt{n \log n})$.

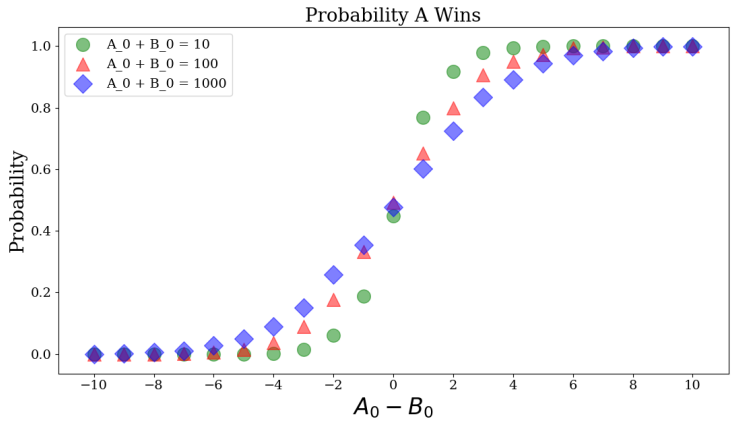

Figure 7.2: The probability that species A survives while species B goes extinct is sharply dependent on their initial gap The sharpness of the transition is inversely proportional to the total initial population size.

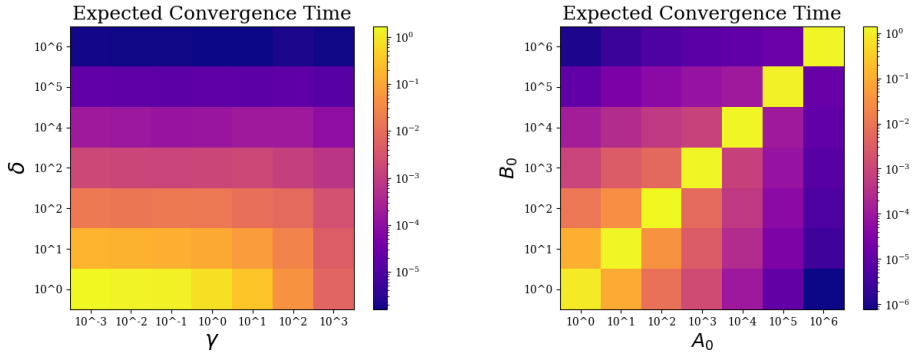

Figure 7.3: Log-scaled expected convergence time of the A-B protocol is represented by color. Corresponding values are shown on the adjacent vertical bar. **Left:** rate constants γ and δ with $A_0 = B_0 = 100$. **Right:** initial populations sizes with $\gamma = 0.01$ and $\delta = 1$.

We further analyzed the A-B protocol with simulations. In Figure 7.2, we set both rate constants γ and δ to 1 and plotted the probability of majority consensus as a function of the initial gap for several total initial population sizes. This behavior indeed qualitatively matches the bound in Theorem 7.2.

In Figure 7.3, we explored the dependence of the expected time until consensus over a range of rate constants and initial populations. The convergence time is more dependent on the death rate constant δ than on the birth rate constant γ. Convergence time sharply increases if the initial gap is small. The off-diagonal initial populations convergence faster for larger populations.

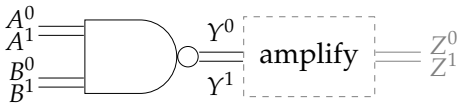

Figure 7.4: Dual-rail NAND-gate with input signals A and B an output signal Y. Successive amplification of Y to signal Z shown in gray.

7.3 Boolean Gates

The A-B protocol can be seen as a differential amplifier. This makes it useful for any system that produces some signal with pronounced noise, and even adversarial behavior.

In this section, we present a building block that produces a computational signal, subject to some noise. We take the example of a NAND-gate that takes two inputs and produces one output. Here, every signal (input and output) is represented by *two* species: one that signifies that the signal is logical 1 and one that signifies that the signal is logical 0. As such, we use a dual-rail encoding. The logical gate can thus be combined with the A-B protocol to perform differential amplification of the output signal, see Figure 7.4.

The mathematical analysis of this section uses some of the same techniques as that for the A-B protocol, but refines some of them to deal with two concurrent duplicating input signals and a duplicating output signal. We skip a more detailed description of the proof structure in this overview.

The NAND-gate has the reactions

$$A^a + B^b \xrightarrow{\alpha} A^a + B^b + Y^y$$

where $y = \neg(a \wedge b)$ and $\alpha > 0$ is the gate's rate constant. Since all species are permanently replicating, we further have the obligatory duplication reactions $A^i \xrightarrow{\gamma} 2A^i$, $B^i \xrightarrow{\gamma} 2B^i$, and $Y^i \xrightarrow{\gamma} 2Y^i$.

One can show that the NAND-gate fulfills the following input–output guarantees. Here, say that a signal X is initially (n, Δ)-correct with value x if $X(0) \geq n$ and $X^{\neg x}(0) \leq (n - \Delta)/2$.

Theorem 7.3. *Assume that the NAND-gate's input signals A, B are dual-rail encoded signals, and that they are initially (n, Δ)-correct with values $a, b \in \{0, 1\}$, respectively, where $n \in \mathbb{N}^+$ and $\Delta \geq 0.62 \cdot \max\{A(0), B(0)\}$. Then with high probability, there exists some time $t = O(1)$ such that $Y(t) = n$ and $Y^y(t) - Y^{\neg y}(t) = \Omega(n)$ for the output signal Y where $y = \neg(a \wedge b)$ is the correct NAND output based on the initial values a, b of signals A and B, respectively.*

When paired with the A-B protocol for amplification at its output, with realistic parameters for a biological implementation, the NAND-gate indeed correctly computes its Boolean function in a large majority of output bacteria, even when subjected to significant input noise. Figure 7.5 contains the corresponding simulations.

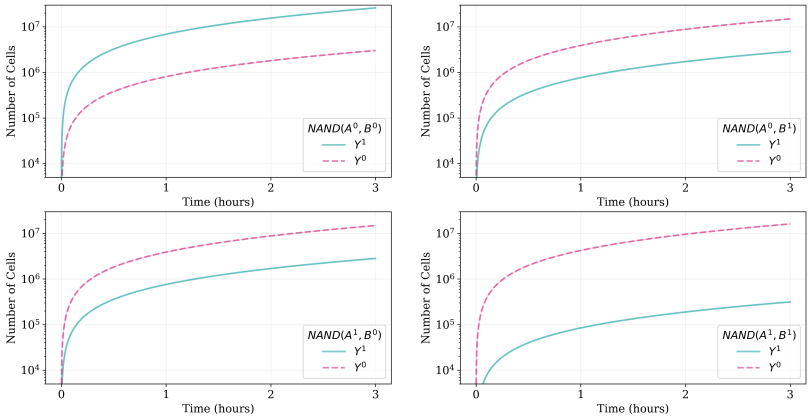

Figure 7.5: A biologically plausible implementation of a NAND-gate with amplifiers on inputs and outputs. Initial population size is 5×10^8, carrying capacity of 10^9 cells. Reaction rate constants were set to $\gamma = 0.016$ and $\delta = 10^{-11}$ [51]. The output species is shown for each choice of inputs. The initial input error is 10%. For all inputs, at least 80% of the output species are correct at all times. Confidence intervals from 30 sample simulations are smaller than the width of the lines.

7.4 Conclusion

Having identified and proven correct an amplification and a Boolean-gate protocol in birth systems is an important first step in two concurrent directions. Firstly, the mathematical analysis of the two protocols opens up new possibilities for a more accurate modeling of biological behavior. Appropriate next targets are individual death reactions, nutrient limitation, and non-uniform birth/death rates. Secondly, the identification of new simple, robust, biologically implementable building blocks for distributed bacterial circuits seems to be a promising route to breaking current complexity barriers that exist due to the fact that whole circuits are implemented in single cells.

Future work will thus focus not only on *in vivo* biological implementation of the two protocols presented in this chapter, but also on the development and implementation of new building blocks and circuits.

Bibliography

[1] Ittai Abraham, Yonatan Amit, and Danny Dolev. Optimal resilience asynchronous approximate agreement. In Teruo Higashino, editor, *Proceedings of the 8th International Conference On Principles Of Distributed Systems (OPODIS 2004)*, pages 229–239, Heidelberg, 2004. Springer.

[2] Ilan Adler and Sheldon M. Ross. The coupon subset collection problem. *Journal of Applied Probability*, 38:737–746, 2001.

[3] Yehuda Afek and Eli Gafni. Asynchrony from synchrony. In Davide Frey, Michel Raynal, Saswati Sarkar, RudrapatnaK. Shyamasundar, and Prasun Sinha, editors, *Distributed Computing and Networking*, volume 7730 of *Lecture Notes in Computer Science*, pages 225–239. Springer Berlin Heidelberg, 2013.

[4] Dan Alistarh, James Aspnes, David Eisenstat, Rati Gelashvili, and Ronald L. Rivest. Time-space trade-offs in population protocols. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 2560–2579, 2017.

[5] Dan Alistarh, James Aspnes, and Rati Gelashvili. Space-optimal majority in population protocols. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 2221–2239, 2018.

[6] Dan Alistarh and Rati Gelashvili. Polylogarithmic-time leader election in population protocols. In *International Colloquium on Automata, Languages, and Programming*, pages 479–491. Springer, 2015.

[7] Dan Alistarh, Rati Gelashvili, and Milan Vojnović. Fast and exact majority in population protocols. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, pages 47–56. ACM, 2015.

[8] Bowen Alpern and Fred B. Schneider. Defining liveness. *Information Processing Letters*, 21(4):181–185, 1985.

[9] J. Christopher Anderson, Elizabeth J. Clarke, Adam P. Arkin, and Christopher A. Voigt. Environmentally controlled invasion of cancer

cells by engineered bacteria. *Journal of Molecular Biology*, 355(4):619–627, January 2006.

[10] David Angeli and Pierre-Alexandre Bliman. Stability of leaderless discrete-time multi-agent systems. *Mathematics of Control, Signals, and Systems*, 18(4):293–322, July 2006.

[11] Dana Angluin, James Aspnes, Zoe Diamadi, Michael J. Fischer, and Rene Peralta. Computation in networks of passively mobile finite-state sensors. In *Proceedings of the Twenty-third Annual ACM Symposium on Principles of Distributed Computing*, pages 290–299, 2004.

[12] Dana Angluin, James Aspnes, and David Eisenstat. A simple population protocol for fast robust approximate majority. *Distributed Computing*, 21(2):87–102, March 2008.

[13] James Aspnes, Joffroy Beauquier, Janna Burman, and Devan Sohier. Time and space optimal counting in population protocols. In *OPODIS 2016*, pages 13:1–13:17, 2016.

[14] Hagit Attiya and Jennifer Welch. *Distributed Computing*. John Wiley & Sons, Inc., Hoboken, April 2004.

[15] John Augustine, Gopal Pandurangan, Peter Robinson, and Eli Upfal. *Towards Robust and Efficient Computation in Dynamic Peer-to-Peer Networks*, pages 551–569. 2012.

[16] F. Baccelli and B. Błaszczyszyn. Stochastic geometry and wireless networks: Volume I: Theory, foundation, and trends. *Networking*, 3(3–4):249–449, 2009.

[17] F. Baccelli and B. Błaszczyszyn. Stochastic geometry and wireless networks: Volume II: Applications, foundation, and trends. *Networking*, 4(1–2):1–312, 2009.

[18] S. A. Bakura, A. Lambert, and T. Nowak. Clock synchronization with exponential smoothing for dynamic networks. In *2020 54th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6, 2020.

[19] Joffroy Beauquier, Janna Burman, Julien Clement, and Shay Kutten. On utilizing speed in networks of mobile agents. In *Proceedings of the 29th ACM symposium on Principles of distributed computing, PODC 2010*, pages 305–314, 2010.

[20] Joffroy Beauquier, Janna Burman, Shay Kutten, Thomas Nowak, and Chuan Xu. Data Collection in Population Protocols with Non-uniformly Random Scheduler. In *Algorithms for Sensor Systems - 13th*

International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2017, Vienna, Austria, September 7-8, 2017, Revised Selected Papers, pages 13–25, 2017.

[21] Luca Becchetti, Andrea E. F. Clementi, Emanuele Natale, Francesco Pasquale, and Luca Trevisan. Find your place: Simple distributed algorithms for community detection. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 940–959, 2017.

[22] Amanda Belleville, David Doty, and David Soloveichik. Hardness of computing and approximating predicates and functions with leaderless population protocols. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017*, pages 141:1–141:14, 2017.

[23] Ido Ben-Zvi and Yoram Moses. Beyond Lamport's happened-before: On time bounds and the ordering of events in distributed systems. *Journal of the ACM*, 61(2):13:1–13:26, April 2014.

[24] Petra Berenbrink, Robert Elsässer, Tom Friedetzky, Dominik Kaaser, Peter Kling, and Tomasz Radzik. A population protocol for exact majority with $O(\log 5/3n)$ stabilization time and $\Theta(\log n)$ states. In *32nd International Symposium on Distributed Computing, DISC 2018*, pages 10:1–10:18, 2018.

[25] Petra Berenbrink, George Giakkoupis, and Peter Kling. Optimal time and space leader election in population protocols. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC 2020)*, page 119–129, New York, 2020. ACM Press.

[26] Martin Biely, Peter Robinson, Ulrich Schmid, Manfred Schwarz, and Kyrill Winkler. Gracefully degrading consensus and k-set agreement in directed dynamic networks. *Theoretical Computer Science*, 726:41–77, 2018.

[27] L Billard. Competition between two species. *Stochastic Processes and their Applications*, 2(4):391–398, 1974.

[28] Pierre-Alexandre Bliman, Angelia Nedic, and Asuman E. Ozdaglar. Rate of convergence for consensus with delays. In *Proceedings of the 47th IEEE Conference on Decision and Control, and the European Control Conference (CDC-ECC 2008)*, pages 2226–2231. IEEE, New York, 2008.

[29] Vincent D. Blondel, Julien M. Hendrickx, Alex Olshevsky, and John N. Tsitsiklis. Convergence in multiagent coordination, consensus, and flocking. In *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference (CDC-ECC)*, pages 2996–3000. IEEE, New York, NY, 2005.

[30] Anne Bouillard and Thomas Nowak. Fast symbolic computation of the worst-case delay in tandem networks and applications. *Performance Evaluation*, 91:270–285, 2015. Special Issue: Performance 2015.

[31] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE/ACM Transactions on Networking (TON)*, 14(SI):2508–2530, 2006.

[32] Pierre Brémaud. *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*. Springer, Heidelberg, 1999.

[33] Janna Burman, Ho-Lin Chen, Hsueh-Ping Chen, David Doty, Thomas Nowak, Eric Severson, and Chuan Xu. Time-optimal self-stabilizing leader election in population protocols, 2021.

[34] M. Cao, D.A. Spielman, and A.S. Morse. A lower bound on convergence of a distributed network consensus algorithm. In Hannes Frey, Xu Li, and Stefan Rührup, editors, *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 2356–2361. IEEE, 2005.

[35] Ming Cao, A. Stephen Morse, and Brian D. O. Anderson. Reaching a consensus in a dynamically changing environment: A graphical approach. *SIAM Journal on Control and Optimization*, 47(2):575–600, January 2008.

[36] 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. *SIAM Journal on Control and Optimization*, 47(2):601–623, January 2008.

[37] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-Varying Graphs and Dynamic Networks. In Hannes Frey, Xu Li, and Stefan Rührup, editors, *ADHOC-NOW*, volume 6811 of *Lecture Notes in Computer Science*, pages 346–359. Springer, 2011.

[38] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, October 2012.

[39] Bernadette Charron-Bost, Matthias Függer, and Thomas Nowak. Approximate consensus in highly dynamic networks: The role of averaging algorithms. In *Automata, Languages, and Programming*, pages 528–539. Springer Berlin Heidelberg, Heidelberg, 2015.

[40] Bernadette Charron-Bost, Matthias Függer, and Thomas Nowak. Fast, robust, quantizable approximate consensus. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi,

editors, *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, volume 55 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 137:1–137:14, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[41] Bernadette Charron-Bost, Matthias Függer, and Thomas Nowak. Multidimensional asymptotic consensus in dynamic networks. Preprint, arXiv:1611.02496 [cs.DC], 2016.

[42] Bernadette Charron-Bost and André Schiper. The Heard-Of model: computing in distributed systems with benign faults. *Distributed Computing*, 22(1):49–71, 2009.

[43] Ioannis Chatzigiannakis, Shlomi Dolev, Sándor Fekete, Othon Michail, and Paul Spirakis. On the fairness of probabilistic schedulers for population protocols. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2010.

[44] Bernard Chazelle. The total s-energy of a multiagent system. *SIAM Journal on Control and Optimization*, 49(4):1680–1706, January 2011.

[45] Da-Jung Cho, Matthias Függer, Corbin Hopper, Manish Kushwaha, Thomas Nowak, and Quentin Soubeyran. Distributed computation with continual population growth. In Hagit Attiya, editor, *34th International Symposium on Distributed Computing (DISC 2020)*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2020. To appear.

[46] Andrea Clementi, Pierluigi Crescenzi, Carola Doerr, Pierre Fraigniaud, Marco Isopi, Alessandro Panconesi, Francesco Pasquale, and Riccardo Silvestri. Rumor spreading in random evolving graphs. In *European Symposium on Algorithms*, pages 325–336. Springer, 2013.

[47] Gennaro Cordasco and Luisa Gargano. Space-optimal proportion consensus with population protocols. In *Stabilization, Safety, and Security of Distributed Systems - 19th International Symposium, SSS 2017*, pages 384–398, 2017.

[48] Étienne Coulouma, Emmanuel Godard, and Joseph Peters. A characterization of oblivious message adversaries for which consensus is solvable. *Theoretical Computer Science*, 584:80–90, June 2015.

[49] Rachel Cummings, David Doty, and David Soloveichik. Probability 1 computation with chemical reaction networks. *Natural Computing*, 15(2):245–261, 2016.

[50] Roberto De Prisco, Dahlia Malkhi, and Michael K. Reiter. On *k*-set consensus problems in asynchronous systems. In *18th ACM Symposium on Principles of Distributed Computing (PODC 1999)*, pages 257–265. ACM, New York City, 1999.

[51] Tatiana Dimitriu, Chantal Lotton, Julien Bénard-Capelle, Dusan Misevic, Sam P Brown, Ariel B Lindner, and François Taddei. Genetic information transfer promotes cooperation in bacteria. *Proceedings of the National Academy of Sciences*, 111(30):11103–11108, 2014.

[52] Roland L. Dobrushin. Central limit theorem for non-stationary Markov chains I. *Theor. Probab. Appl.+*, 1(1):65–80, 1956.

[53] Benjamin Doerr, Mahmoud Fouz, and Tobias Friedrich. Why rumors spread so quickly in social networks. *Communications of the ACM*, 55(6):70–75, 2012.

[54] Danny Dolev, Nancy A. Lynch, Shlomit S. Pinter, Eugene W. Stark, and William E. Weihl. Reaching approximate agreement in the presence of faults. *Journal of the ACM*, 33(3):499–516, May 1986.

[55] David Doty, Mahsa Eftekhari, Othon Michail, Paul G. Spirakis, and Michail Theofilatos. Brief announcement: Exact size counting in uniform population protocols in nearly logarithmic time. In *32nd International Symposium on Distributed Computing, DISC 2018*, pages 46:1–46:3, 2018.

[56] David Doty and David Soloveichik. Stable leader election in population protocols requires linear time. *Distributed Computing*, 9363(8):1–15, 2015.

[57] R. Ekwall, P. Urban, and A. Schiper. Robust tcp connections for fault tolerant computing. In *Ninth International Conference on Parallel and Distributed Systems, 2002. Proceedings.*, pages 501–508, 2002.

[58] Seyed Rasoul Etesami and Tamer Başar. Convergence time for unbiased quantized consensus. *IEEE Transactions on Automatic Control*, 61(2):443–455, 2016.

[59] A. D. Fekete. Asymptotically optimal algorithms for approximate agreement. *Distributed Computing*, 4(1):9–29, March 1990.

[60] Alan D. Fekete. Asynchronous approximate agreement. *Information and Computation*, 115(1):95–124, 1994.

[61] William Feller. Die Grundlagen der volterraschen Theorie des Kampfes ums Dasein in wahrscheinlichkeitstheoretischer Behandlung (1939). In *Selected Papers I*, pages 441–470. Springer, 2015.

[62] Marco Ferrante and Monica Saltalamacchia. The coupon collector's problem. *Materials matemàtics*, 2014(2), 2014.

[63] Tristan Fevat and Emmanuel Godard. Minimal obstructions for the coordinated attack problem and beyond. In *25th IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2011, Anchorage, Alaska, USA, 16-20 May, 2011 - Conference Proceedings*, pages 1001–1011, 2011.

[64] Michael J Fischer, Nancy A Lynch, and Michael Merritt. Easy impossibility proofs for distributed consensus problems. *Distributed Computing*, 1(1):26–39, 1986.

[65] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, April 1985.

[66] Pierre Fraigniaud and Emmanuel Lazard. Methods and problems of communication in usual networks. *Discrete Applied Mathematics*, 53(1):79 – 133, 1994.

[67] Matthias Függer, Manish Kushwaha, and Thomas Nowak. Digital circuit design for biological and silicon computers. In *Advances in Synthetic Biology*, pages 153–171. Springer, 2020.

[68] Matthias Függer, Robert Najvirt, Thomas Nowak, and Ulrich Schmid. A faithful binary circuit model. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1–1, 2019.

[69] Matthias Függer and Thomas Nowak. Fast multidimensional asymptotic and approximate consensus. In Ulrich Schmid and Josef Widder, editors, *32nd International Symposium on Distributed Computing (DISC 2018)*, volume 121 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 27:1–27:16, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[70] Matthias Függer, Thomas Nowak, and Bernadette Charron-Bost. Diffusive clock synchronization in highly dynamic networks. In *2015 49th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6. IEEE, IEEE, March 2015.

[71] Matthias Függer, Thomas Nowak, and Ulrich Schmid. Unfaithful glitch propagation in existing binary circuit models. *IEEE Transactions on Computers*, 65(3):964–978, March 2016.

[72] Matthias Függer, Thomas Nowak, and Manfred Schwarz. Tight bounds for asymptotic and approximate consensus. In *Proceedings of*

the 2018 ACM Symposium on Principles of Distributed Computing - PODC '18. ACM Press, 2018.

[73] Matthias Függer, Thomas Nowak, and Kyrill Winkler. On the radius of nonsplit graphs and information dissemination in dynamic networks. *Discrete Applied Mathematics*, 282:257–264, 2020.

[74] Matthias Függer and Ulrich Schmid. Reconciling fault-tolerant distributed computing and systems-on-chip. *Distributed Computing*, 24(6):323–355, December 2011.

[75] M. Függer, J. Maier, R. Najvirt, T. Nowak, and U. Schmid. A faithful binary circuit model with adversarial noise. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1327–1332, 2018.

[76] Eli Gafni. Round-by-round fault detectors (extended abstract): unifying synchrony and asynchrony. In *Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing*, pages 143–152, New York, NY, USA, 1998. ACM Press.

[77] Eli Gafni, Petr Kuznetsov, and Ciprian Manolescu. A generalized asynchronous computability theorem. In *Proceedings of the 33rd ACM Symposium on Principles of Distributed Computing (PODC 2014)*, pages 222–231. ACM, New York, 2014.

[78] Leszek Gasieniec, David D. Hamilton, Russell Martin, Paul G. Spirakis, and Grzegorz Stachowiak. Deterministic population protocols for exact majority and plurality. In *20th International Conference on Principles of Distributed Systems, OPODIS 2016*, pages 14:1–14:14, 2016.

[79] Leszek Gąsieniec and Grzegorz Stachowiak. Fast space optimal leader election in population protocols. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 2653–2667, 2018.

[80] C. Han, T. Nowak, and A. Lambert. Pulse synchronization for vehicular networks. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1125–1130, 2018.

[81] H. A. Harutyunyan, A. L. Liestman, J. Peters, and D. Richards. Broadcasting and gossiping). In J. Gross and P. Zhang, editors, *The Handbook of Graph Theory*, pages 1477–1494. Chapman and Hall/CRC, 2013.

[82] Sandra M. Hedetniemi, Stephen T. Hedetniemi, and Arthur L. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18(4):319–349, 1988.

[83] R. Hegselmann and U. Krause. Opinion dynamics and bounded confidence models, analysis, and simulation. *Journal of artificial societies and social simulation*, 5(3):1–33, 2002.

[84] Maurice Herlihy, Dmitry N. Kozlov, and Sergio Rajsbaum. *Distributed Computing Through Combinatorial Topology*. Morgan Kaufmann, 2013.

[85] Maurice Herlihy, Sergio Rajsbaum, Michel Raynal, and Julien Stainer. From wait-free to arbitrary concurrent solo executions in colorless distributed computing. *Theoretical Computer Science*, 683:1–21, 2017.

[86] Gunnar Hoest and Nir Shavit. Toward a topological characterization of asynchronous complexity. *SIAM Journal on Computing*, 36(2):457–497, 2006.

[87] Juraj Hromkovič, Ralf Klasing, Burkhard Monien, and Regine Peine. Dissemination of information in interconnection networks (broadcasting & gossiping). In Ding-Zhu Du and D. Frank Hsu, editors, *Combinatorial Network Theory*, pages 125–212. Springer US, Boston, MA, 1996.

[88] Idit Keidar and Alex Shraer. Timeliness, failure detectors, and consensus performance. In *Proceedings of the twenty-fifth annual ACM SIGACT-SIGOPS symposium on Principles of Distributed Computing (PODC'06)*, pages 169–178, New York, NY, USA, 2006. ACM Press.

[89] David G Kendall. Branching processes since 1873. *Journal of the London Mathematical Society*, 1(1):385–406, 1966.

[90] Arthur Kennedy-Cochran-Patrick, Glenn Merlet, Thomas Nowak, and Sergeĭ Sergeev. New bounds on the periodicity transient of the powers of a tropical matrix: Using cyclicity and factor rank. *Linear Algebra and its Applications*, 611:279–309, 2021.

[91] Fabian Kuhn, Nancy Lynch, and Rotem Oshman. Distributed computation in dynamic networks. In *Proceedings of the 42nd ACM symposium on Theory of computing - STOC '10*, pages 513–522. ACM, ACM Press, 2010.

[92] Petr Kuznetsov, Thibault Rieutord, and Yuan He. An asynchronous computability theorem for fair adversaries. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*, pages 387–396, 2018.

[93] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.

[94] Ronit Lubitch and Shlomo Moran. Closed schedulers: A novel technique for analyzing asynchronous protocols. *Distrib. Comput.*, 8(4):203–210, June 1995.

[95] Nancy A Lynch. *Distributed algorithms*. Morgan Kaufmann, 1996.

[96] Hosam M. Mahmoud. *Pólya Urn Models*. CRC Press, Boca Raton, 2009.

[97] J. Maier, M. Függer, T. Nowak, and U. Schmid. Transistor-level analysis of dynamic delay models. In *2019 25th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pages 76–85, 2019.

[98] Hammurabi Mendes, Maurice Herlihy, Nitin Vaidya, and Vijay K. Garg. Multidimensional agreement in Byzantine systems. *Distributed Computing*, 28:423–441, 2015.

[99] Glenn Merlet, Thomas Nowak, and Sergeĭ Sergeev. On the tightness of bounds for transients of weak csr expansions and periodicity transients of critical rows and columns of tropical matrix powers. *Linear and Multilinear Algebra*, 2021. To appear.

[100] Yves Mocquard, Emmanuelle Anceaume, and Bruno Sericola. Optimal proportion computation with population protocols. In *IEEE International Symposium on Network Computing and Applications*, pages 216–223, 2016.

[101] L. Moreau. Stability of multiagent systems with time-dependent communication links. *IEEE Transactions on Automatic Control*, 50(2):169–182, February 2005.

[102] Angelia Nedic, Alexander Olshevsky, Asuman E. Ozdaglar, and John N. Tsitsiklis. On distributed averaging algorithms and quantization effects. *IEEE Transactions on Automatic Control*, 54(11):2506–2517, 2009.

[103] Artem S Novozhilov, Georgy P Karev, and Eugene V Koonin. Biological applications of the theory of birth-and-death processes. *Briefings in bioinformatics*, 7(1):70–85, 2006.

[104] Thomas Nowak. Topology in distributed computing. Master thesis, March 2010.

[105] Thomas Nowak and Joel Rybicki. Byzantine approximate agreement on graphs. In Jukka Suomela, editor, *33rd International Symposium on Distributed Computing (DISC 2019)*, volume 146 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 29:1–29:17, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[106] Thomas Nowak, Ulrich Schmid, and Kyrill Winkler. Topological characterization of consensus under general message adversaries. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing - PODC '19*, PODC '19, pages 218–227, New York, NY, USA, 2019. ACM Press.

[107] Alex Olshevsky. Linear time average consensus on fixed graphs. *IFAC-PapersOnLine*, 48(22):94–99, 2015.

[108] Alex Olshevsky and John N. Tsitsiklis. Convergence speed in distributed consensus and averaging. *SIAM Review*, 53(4):747–772, January 2011.

[109] C.J. Paddon, P.J. Westfall, D.J. Pitera, K. Benjamin, K. Fisher, D. McPhee, M.D. Leavell, A. Tai, A. Main, D. Eng, D.R. Polichuk, K.H. Teoh, D.W. Reed, T. Treynor, J. Lenihan, H. Jiang, M. Fleck, S. Bajad, G. Dang, D. Dengrove, D. Diola, G. Dorin, K.W. Ellens, S. Fickes, J. Galazzo, S.P. Gaucher, T. Geistlinger, R. Henry, M. Hepp, T. Horning, T. Iqbal, L. Kizer, B. Lieu, D. Melis, N. Moss, R. Regentin, S. Secrest, H. Tsuruta, R. Vazquez, L.F. Westblade, L. Xu, M. Yu, Y. Zhang, L. Zhao, J. Lievense, P.S. Covello, J.D. Keasling, K.K. Reiling, N.S. Renninger, and J.D. Newman. High-level semi-synthetic production of the potent antimalarial artemisinin. *Nature*, 496(7446):528–532, April 2013.

[110] Martin Radetzki, Chaochao Feng, Xueqian Zhao, and Axel Jantsch. Methods for fault tolerance in networks-on-chip. *ACM Computing Surveys*, 46(1), 2013.

[111] Sergi Regot, Javier Macia, Núria Conde, Kentaro Furukawa, Jimmy Kjellén, Tom Peeters, Stefan Hohmann, Eulàlia de Nadal, Francesc Posas, and Ricard Solé. Distributed biological computation with multicellular engineered networks. *Nature*, 469(7329):207–211, December 2010.

[112] GEH Reuter. Competition processes. In *Proc. 4th Berkeley Symp. Math. Statist. Prob*, volume 2, pages 421–430, 1961.

[113] CJ Ridler-Rowe. On competition between two species. *Journal of Applied Probability*, 15(3):457–465, 1978.

[114] Thomas L Saaty. *Elements of queueing theory: with applications*, volume 34203. McGraw-Hill New York, 1961.

[115] Nicola Santoro and Peter Widmayer. Time is not a healer. In B. Monien and R. Cori, editors, *STACS 89*, volume 349 of *LNCS*, pages 304–313. Springer-Verlag, Heidelberg, 1989.

[116] Ulrich Schmid. How to model link failures: A perception-based fault model. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN'01)*, pages 57–66, Göteborg, Sweden, 2001.

[117] Ulrich Schmid, Bettina Weiss, and Idit Keidar. Impossibility results and lower bounds for consensus under link failures. *SIAM Journal on Computing*, 38(5):1912–1951, January 2009.

[118] Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys*, 22(4):299–319, 1990.

[119] Shimyn Slomovic, Keith Pardee, and James J Collins. Synthetic biology devices for in vitro and in vivo diagnostics. *Proceedings of the National Academy of Sciences*, 112(47):14429–14435, 2015.

[120] Wolfgang Stadje. The collector's problem with group drawings. *Advances in Applied Probability*, 22(4):866–882, 1990.

[121] Alvin Tamsir, Jeffrey J. Tabor, and Christopher A. Voigt. Robust multicellular computing using genetically encoded NOR gates and chemical 'wires'. *Nature*, 469(7329):212–215, December 2010.

[122] Pei Kun R Tay, Peter Q Nguyen, and Neel S Joshi. A synthetic circuit for mercury bioremediation using self-assembling functional amyloids. *ACS synthetic biology*, 6(10):1841–1850, 2017.

[123] Vito Volterra. Leçons sur la theorie mathematique de la lutte pour la vie. *Gauthier-Villars, Paris*, 1931.

[124] Jennifer Lundelius Welch and Nancy Lynch. A new fault-tolerant algorithm for clock synchronization. *Information and Computation*, 77(1):1–36, April 1988.

[125] Kyrill Winkler, Manfred Schwarz, and Ulrich Schmid. Consensus in directed dynamic networks with short-lived stability. *Distributed Computing*, 2019.

[126] Chai Wah Wu. Synchronization and convergence of linear dynamics in random directed networks. *IEEE Transactions on Automatic Control*, 51(7), 2006.

[127] C. Xu, J. Burman, and J. Beauquier. Power-aware population protocols. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 2067–2074, June 2017.

[128] Chuan Xu, Joffroy Beauquier, Janna Burman, Shay Kutten, and Thomas Nowak. Data collection in population protocols with non-uniformly random scheduler. *Theoretical Computer Science*, August 2019.

[129] Y. Yuan, G.-B. Stan, L. Shi, M. Barahona, and J. Goncalves. Decentralized minimum-time consensus. *Automatica*, 49(5):1227–1235, 2013.

[130] Martin Zeiner, Manfred Schwarz, and Ulrich Schmid. On linear-time data dissemination in dynamic rooted trees. *Discrete Applied Mathematics*, 255:307 – 319, 2019.

université
PARIS-SACLAY

Titre : Problèmes d'accord dans des réseaux dynamiques

Mots clés : réseaux dynamiques, consensus, protocoles de population, réseau de réaction chimique

Résumé : Les réseaux dynamiques sont étudiés dans une variété d'applications pratiques et théoriques. Ils apparaissent naturellement dans les réseaux sans fil, où les liens de communication apparaissent et disparaissent en fonction d'un certain nombre de facteurs externes. L'utilité des modèles de réseaux dynamiques dépasse toutefois les applications dans les réseaux sans fil. Ils sont également applicables à d'autres modes de calcul avec une communication incertaine, tels que l'informatique tolérante aux pannes, câblée ou sur puce, l'Internet, les phénomènes épidémiques et sociologiques, et la communication entre des entités macro et microbiologiques. Bien que les réseaux dynamiques aient une applicabilité et une généralité très larges, ils n'ont historiquement pas été étudiés en profondeur. En fait, les investigations formelles ciblées n'ont commencé que relativement récemment.

Les problèmes d'accord se présentent sous différentes variantes. D'un point de vue théorique, ils servent de référence pour évaluer la force des modèles de calcul distribué. D'un point de vue pratique, ils sont souvent identifiés comme des sous-tâches nécessaires ou utiles pour l'implémentation d'un système donné. Une grande partie de ce manuscrit est consacrée au consensus asymptotique, pour lequel nous donnons une caractérisation assez complète dans la classe des modèles de réseaux oblivious. Nous présentons également la première caractérisation de la solvabilité du consensus exact dans les modèles de réseaux dynamiques généraux. Nous étudions également le problème de l'obtention d'un consensus majoritaire dans des populations dans lesquelles tous les agents se dupliquent continuellement. Cette étude est motivée par des calculs effectués avec des bactéries synthétiquement modifiées.

Title : Agreement Problems in Dynamic Networks

Keywords : dynamic networks, consensus, population protocols, chemical reaction networks

Abstract : Dynamic networks are studied in a variety of practical and theoretical applications. They naturally occur in wireless networking, where communication links go up and down depending on a number of external factors. The usefulness of dynamic-network models exceeds applications in wireless networking, however. They are also applicable for other modes of computation with uncertain communication, such as wired—even on-chip—fault-tolerant computing, the Internet, epidemic and sociological phenomena, and communication between macro- and microbiological entities. Even though dynamic networks have a very wide applicability and generality, historically they have not been thoroughly studied. In fact, targeted formal investigations have begun only relatively recently.

Agreement problems come in different variants. From a theoretical perspective, they serve as a benchmark for the strength of distributed-computing models. From a practical perspective, they are often identified as necessary or useful sub-tasks for the implementation of a given system. A large part of this manuscript is devoted to asymptotic consensus, for which we give a quite complete in the class of oblivious network models. We further present the first characterization of solvability of exact consensus in general dynamic network models. We also study the problem of reaching majority consensus in populations in which all agents continually duplicate. The motivation for this comes from computation with synthetically modified bacteria.