Fast Graphical Population Protocols

Dan Alistarh
IST Austria, Klosterneuburg, Austria

Rati Gelashvili
Novi Research, Menlo Park, CA, USA

Joel Rybicki
IST Austria, Klosterneuburg, Austria

Abstract

Let G be a graph on n nodes. In the stochastic population protocol model, a collection of n indistinguishable, resource-limited nodes collectively solve tasks via pairwise interactions. In each interaction, two randomly chosen neighbors first read each other's states, and then update their local states. A rich line of research has established tight upper and lower bounds on the complexity of fundamental tasks, such as majority and leader election, in this model, when G is a *clique*. Specifically, in the clique, these tasks can be solved *fast*, i.e., in $n \operatorname{polylog} n$ pairwise interactions, with high probability, using at most $\operatorname{polylog} n$ states per node.

In this work, we consider the more general setting where G is an arbitrary regular graph, and present a technique for simulating protocols designed for fully-connected networks in any connected regular graph. Our main result is a simulation that is *efficient* on many interesting graph families: roughly, the simulation overhead is polylogarithmic in the number of nodes, and quadratic in the conductance of the graph. As a sample application, we show that, in any regular graph with conductance φ, both leader election and exact majority can be solved in $\varphi^{-2} \cdot n \operatorname{polylog} n$ pairwise interactions, with high probability, using at most $\varphi^{-2} \cdot \operatorname{polylog} n$ states per node. This shows that there are fast and space-efficient population protocols for leader election and exact majority on graphs with good expansion properties. We believe our results will prove generally useful, as they allow efficient technology transfer between the well-mixed (clique) case, and the under-explored spatial setting.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases population protocols, leader election, exact majority, graphs

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2021.14

Related Version *Full Version*: `https://arxiv.org/abs/2102.08808` [6]

Funding *Dan Alistarh*: This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 805223 ScaleML).
Joel Rybicki: This project has received from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 840605.

Acknowledgements We grateful to Giorgi Nadiradze for pointing out a generalisation of the phase clock construction to non-regular graphs. We also thank anonymous reviewers for their useful comments on earlier versions of this manuscript.

1 Introduction

Since the early days of computer science, there has been significant interest in developing an algorithmic theory of molecular and biological systems [49]. In distributed computing, *population protocols* [8] have become a popular model for investigating the collective computational power of large collections of communication- and computationally-bounded agents. This model consists of n identical agents, seen as finite state machines, and computation

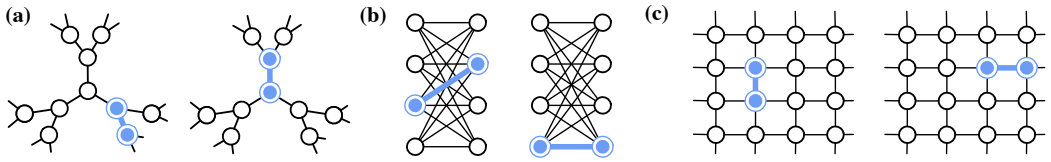

Figure 1 The graphical population protocol model. In each step, a random edge $\{u, v\}$ is selected and the nodes u and v interact (blue nodes). Examples of graph classes covered by our construction: (a) regular high-girth expanders, (b) bipartite complete graphs, (c) toroidal grids.

proceeds via pairwise interactions which trigger local state transitions. The sequence of interactions is provided by a scheduler, which picks pairs of agents to interact. The goal is to have the system reach a configuration satisfying a given predicate, while minimising the number of interactions (time complexity) and the number of states per node (space complexity) required by the protocol.

Early work on population protocols focused on the computational power of the model under various interaction graphs [8, 11]. More recently, the focus has shifted to understanding complexity thresholds, often in the form of fundamental complexity trade-offs between time and space complexity, e.g. [10, 7, 32, 35, 4, 16, 19, 31]; for recent surveys please see [34, 5].

The second line of work almost focuses mainly on the *uniform* stochastic scheduler, where each interaction pair is chosen uniformly at random *among all pairs* of agents in the population, and the time complexity of a protocol is measured by the number of interactions needed to solve a task. This is analogous to having a large *well-mixed* solution of interacting particles when modelling chemical reactions. However, many natural systems exhibit spatial structure and this structure can significantly influence the system dynamics.

Indeed, there is a separation in terms of computational power for population protocols in the clique versus other interaction graphs: connected interaction graphs can simulate adversarial interactions on the clique graph by shuffling the states of the nodes [8] and population protocols on some interaction graphs can compute a strictly larger set of predicates than protocols on the clique; see e.g. [13] for a survey of computability results.

By comparison, surprisingly little is known about the *complexity* of basic tasks in general interaction graphs under the stochastic scheduler. So far, only a handful of protocols have been analysed on general graphs. Existing analyses tend to be complex, and specialised to specific algorithms on limited graph classes [33, 27, 42, 43, 17]. This is natural: given the intricate dependencies which arise due to the underlying graph structure, the design and analysis of protocols in the spatial setting is understood to be challenging.

Contributions. In this work, we provide a general approach showing that standard problems in population protocols can be solved *efficiently* under *graphical* stochastic schedulers, by leveraging solutions designed for complete graphs. Our results are as follows:

1. We give a general framework for simulating a large class of *synchronous* protocols designed for *fully-connected networks*, in the graphical stochastic population protocol model (see Figure 1). Thus, the user can design efficient (and simple to analyse) synchronous algorithms on a clique model, and transport the analysis automatically to the population protocol model on a large class of interaction graphs. For instance, on any d-regular graph with edge expansion $\beta > 0$, the resulting overhead in parallel time and state complexity is in the order of $(d/\beta)^2 \cdot \operatorname{polylog} n$.

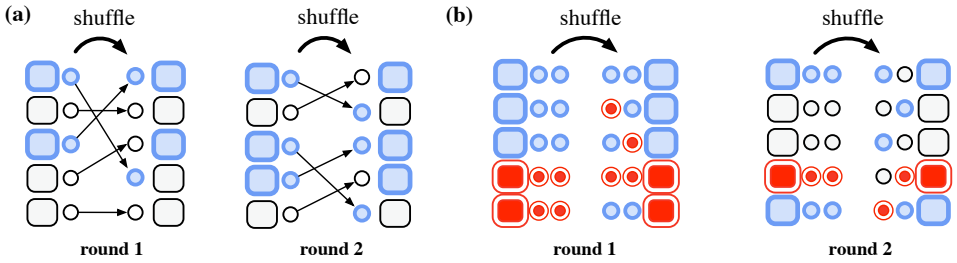

Figure 2 The synchronous k-token shuffling model with 5 nodes for $k = 1$ and $k = 2$. Rectangles are nodes and the small circles are tokens. In each round, nodes generate k tokens based on their current state. Then all nk tokens are shuffled randomly. After this, nodes update their state based on the vector of k tokens they hold. (a) An execution of a protocol in the 1-token shuffling model. The arrows between tokens represent the random permutation used to shuffle tokens. (b) An execution of a protocol for $k = 2$. Each node sends and receives two tokens.

2. As concrete applications, we show that for any d-regular graph with edge expansion $\beta > 0$, there exist protocols for leader election and exact majority that stabilise both in expectation and with high probability in $(d/\beta)^2 \cdot \operatorname{polylog} n$ parallel time, using $(d/\beta)^2 \cdot \operatorname{polylog} n$ states.
3. To complement the results following from the simulation, we also show that, on any graph G with diameter $\operatorname{diam}(G)$ and m edges, leader election can be solved both in expectation and with high probability in $O(\operatorname{diam}(G) \cdot mn^2 \log n)$ parallel time, by analysing the time complexity of the constant-state protocol by Beauquier et al. [15].

Technical Overview. Our reduction framework combines several techniques from different areas, and can be distilled down to the following ingredients.

We start by defining a simple *synchronous, fully-connected* model of communication for the n nodes, called the *k-token shuffling model*. This is the model in which the algorithm should be designed and analysed, and is similar, and in some ways simpler, relative to the standard population model. Specifically, nodes proceed in *synchronous* rounds, in which every node v first generates k tokens based on its current state. Tokens are then shuffled uniformly at random among the nodes. At the end of a round, every node v updates its local state based on its current state, and the tokens it received in the round. Figure 2 illustrates the model. This simple model is quite powerful, as it can simulate both *pairwise* and *one-way* interactions between all sets of agents, for well-chosen settings of the parameter k.

Our key technical result is that any algorithm specified in this round-synchronous k-token shuffling model can be *efficiently* simulated in the graphical population model. Although intuitive, formally proving this result, and in particular obtaining bounds on the efficiency of the simulation, is non-trivial. First, to show that simulating *a single round* of the k-token shuffling model can be done efficiently, we introduce new type of *card shuffling process* [28, 50, 23, 38], which we call the k-stack interchange process, and analyse its mixing time by linking it to random walks on the symmetric group.

Second, to allow correct and efficient asynchronous simulation of the synchronous token shuffling model, we introduce two new gadgets: (1) a *graphical* version of *decentralised phase clocks* [4, 36, 35], combined with (2) an *asynchronous* token shuffling protocol, which simulates the k-token interchange process in a graphical population protocol. The latter ingredient is our main technical result, as it requires both efficiently combining the above components, and carefully bounding the probability bias induced by simulating a synchronous model under asynchronous pairwise-random interactions.

Table 1 Protocols for exact majority (EM) and leader election (LE) for different graph classes. The state complexity is the number of states used by the protocol. The parallel time column gives the expected parallel time (expected number of interaction steps divided by n) to stabilise. (*) In [33], the running time of the protocol is bounded by the initial discrepancy in the inputs and the spectral properties of the contact rate matrix; bounds in terms of n are only given for select graph classes (paths, cycles, stars, random graphs and cliques). No sublinear in n bounds on parallel time are given in [33]. Protocols marked with ($\star$) stabilise also in non-regular graphs in poly(n) time.

Graph class	Task	States	Parallel time	Note
cliques	EM	4	$O(n \log n)$	[33]
	EM	$O(\log n)$	$\Theta(\log n)$	[31]
	LE	2	$\Theta(n)$	[32]
	LE	$\Theta(\log \log n)$	$\Theta(\log n)$	[19]
connected	EM	4	poly(n)	[33, 17], (*)
	LE	6	$O(\mathrm{diam}(G) \cdot mn^2 \log n)$	**new** analysis of [15]
d-regular	EM	$(d/\beta)^2 \cdot \mathrm{polylog}\, n$	$(d/\beta)^2 \cdot \mathrm{polylog}\, n$	**new**, ($\star$)
	LE	$(d/\beta)^2 \cdot \mathrm{polylog}\, n$	$(d/\beta)^2 \cdot \mathrm{polylog}\, n$	**new**, ($\star$)

Finally, we instantiate this framework to solve exact majority and leader election in the graphical setting. We provide simple token-shuffling protocols for these problems, as well as backup protocols to ensure their correctness in all executions.

Implications. Our results imply new and improved upper bounds on the time and state complexity of majority and leader election for a wide range of graph families. In some cases, they improve upon the best known upper bounds for these problems. Please see Table 1 for a systematic comparison. Specifically, our results show that:

- In *sparse* graphs with good expansion properties, such as constant-degree graphs with constant edge expansion (Figure 1a), our simulation has polylogarithmic time and state complexity overhead, relative to clique-based algorithms. Thus, good expanders admit fast protocols using polylogarithmic states, despite being sparser than the clique.
- In *dense* graphs, we obtain similar bounds whenever $d/\beta \in \mathrm{polylog}\, n$ holds. This is the case for instance in d-dimensional hypercubes with $n = 2^d$ nodes, but also in highly-dense clique-like graphs, such as regular complete multipartite graphs (Figure 1b), where the degree and expansion are both $\Theta(n)$.
- In D-dimensional toroidal grids, we get algorithms with $n^{2/D}\, \mathrm{polylog}\, n$ parallel time and state complexity. These graphs include cycles (1-dimensional toroidal grids), two-dimensional grids (Figure 1c), three-dimensional lattices, and so on.

While our protocols guarantee *fast* stabilisation in regular graphs with high expansion, they will stabilise in polynomial expected time in *any connected graph*. The results can be carried over to certain classes of *non-regular* graphs provided that they are not highly irregular and have high expansion; we discuss this in Section 8, and provide examples in the Appendix.

It is known that, in the clique setting, constant-state protocols are necessarily slower than protocols with super-constant states [32, 4]. Our results suggest the existence of a similar complexity gap in the graphical setting. Specifically, on d-regular graphs with good expansion, such that $d/\beta \in \mathrm{polylog}\, n$, we provide polylogarithmic-time protocols for both leader election and exact majority. This opens a significant complexity gap relative to known constant-state protocols on graphs. For instance, the 4-state exact majority protocol for general graphs [33] requires $\Omega(n)$ parallel time even in regular graphs with high expansion, if

node degrees are $\Theta(n)$. (A simple example is the complete bipartite graph given in Figure 1b.) Yet, our protocols guarantee stabilisation in only $\operatorname{polylog} n$ parallel time in both low and high degree graphs, as long as d/β is at most $\operatorname{polylog} n$.

Due to space constraints, in the following we focus on the simulation framework and its properties; the correctness proofs for our framework and the algorithmic applications are given in the full version of the paper [6].

2 Related Work

Computability for Graphical Population Protocols. A variant of the graphical setting was already considered in the foundational work of Angluin et al. [8], which also uses a state shuffling approach. However, the resulting line of work focused on *computational power* in the case where the number of states per node is constant [8, 9, 12, 11, 21]. A key difference is that we aim to simulate pairwise interactions under the uniform stochastic scheduler, as fast protocols in the clique require that pairwise interactions are uniformly random [34]. Thus, one of the main technical challenges is to devise an *efficient* shuffling procedure that guarantees that the simulated interactions are (almost) uniform.

In addition, *self-stabilising* population protocols on graphs have been investigated particularly in the context of leader election [12, 15, 51, 47, 24, 25, 39, 48]. This considers more stringent *transient fault* models than ours: we will thus be able to obtain better bounds, but our results will not directly transfer to self-stabilising protocols. This is natural, since self-stabilising leader election is not solvable on all graph families [12].

Beauquier et al. [15] noted that without the requirement of self-stabilisation, leader election can be solved on every connected graph by a constant-state protocol. We provide the first running-time upper bounds for this protocol; please see Table 1 for a summary of the known bounds. Concurrent work by Sudo et al. [48] on self-stabilising leader election on general graphs uses a similar approach to our analysis of the leader election protocol, presented in the full version [6].

Complexity in the Clique Model. A parallel line of work has focused on determining the fundamental space-time trade-offs for key tasks, such as majority and leader election, when the interaction graph is a *clique* [32, 33, 42, 3, 4, 20, 16, 19, 31]. In this case, tight or almost-tight complexity trade-offs are now known for these problems [19, 37, 4].

The vast majority of the work on complexity has focused on the clique case [34, 5]. Two natural justifications for this choice are that: (1) the clique is a good approximation for well-mixed solutions, and (2) the analysis of population protocols can be difficult enough even without additional complications due to graph structure. Bounds on non-complete graphs have been studied for exact [33] and approximate majority [42, 43], with some recent work considering *plurality consensus* [26, 27, 17] in a related model. The recent survey of [34] points out that running time on general graphs is poorly understood, and sets this as an open question. We take a first step towards addressing this gap.

Interacting Particle Systems. Another related line of work investigated dynamics of interacting particle systems on graphs, e.g. [2]. However, in this context dynamics are often assumed to be round-synchronous, which allows the use of more powerful techniques, related to independent random walks on graphs. Cooper et al. [26] analysed the coalescence time of independent random walks on a graph in terms of the expansion properties of the graph, where each node initially holds a unique particle, and in each step particles randomly move to

another node. Whenever, two particles meet, they coalesce into a single one, which continues its walk. We also employ token-based protocols on graphs, but in our case tokens are shuffled between nodes instead of coalescing.

Token-based processes have also been used to implement efficient, randomised rumour spreading protocols. For example, Berenbrink et al. [18] analysed the cover time of a synchronous coalescing-branching random walk on regular graphs. Similarly to our work, they use conductance to bound the behaviour of this process in regular graphs. In this work, we use token-based population protocols on graphs, where the tokens are shuffled between nodes during an interaction and the tokens instead of coalescing, may also interact in other ways.

Plurality Consensus on Expanders. In plurality consensus, there are $k > 1$ opinions and the task is the agree on opinion supported by the most nodes. Berenbrink et al. [17] present a protocol for the plurality consensus problem in a synchronous pull-based interaction model. Their protocol also circulates tokens, and samples their count periodically (after mixing) to estimate opinion counts, running into the issue that the token movements are correlated. The authors provide a generalisation of a result by Sauerwald and Sun [45] in order to show that the joint token distribution is negatively correlated, and therefore the token counting mechanism concentrates.

In this work, we also employ a token exchange protocol, and encounter non-trivial correlation issues. However, we resolve these issues differently: we characterise the distribution of the token interactions using the k-stack interchange process, and bound its total variation distance relative to the uniform distribution, showing that the two distributions are indistinguishable in polynomial time with high probability. More generally, the goal of our construction is different, as we aim to provide a general framework to efficiently simulate pairwise random node interactions.

Shuffling Processes. Our results also connect to the work on card shuffling processes, and in particular, the interchange process, which has a long and rich history, e.g. [29, 1, 28, 50, 38]. While many of these processes are simple to describe, they are often surprisingly challenging to analyse. In the classic interchange process, a card is placed on every node of a graph and the shuffling is performed by randomly exchanging cards between adjacent nodes.

Diaconis and Shahshahani [29] gave sharp bounds on the mixing time of the random transpositions shuffle, i.e., interchange process on the clique. Diaconis and Saloff-Coste [28] developed a powerful comparison technique for upper bounding the mixing time of a random walk on a finite group. This is one the key techniques for upper bounding mixing times of the interchange process.

Later, Wilson [50] developed a general technique for proving lower bounds for many shuffling processes. For example, he showed that the mixing time of the interchange process on the two-dimensional $\sqrt{n} \times \sqrt{n}$ grid is $\Theta(n^2 \log n)$ and $\Omega(n \log^2 n)$ on the hypercube. Subsequently, Jonasson [38] gave additional upper and lower bounds on the interchange process on various graphs.

3 Preliminaries

Graphs. A graph $G = (V, E)$ is d-regular if every node $v \in V$ is adjacent to exactly d other nodes. The edge boundary of a set $S \subseteq V$ is the set $\partial S \subseteq E$ of edges with exactly one endpoint in S. The *edge expansion* of the graph G is defined as $\beta = \min\{|\partial S|/|S| : S \subseteq V, |S| \leq n/2\}$. If G is regular, its *conductance* is β/d. Unless otherwise mentioned, all graphs are assumed to be regular and connected.

Probability distributions. Let E be a finite set. We say $\mu\colon E \to [0,1]$ is a probability distribution on E if $\sum_{x\in E}\mu(x) = 1$ holds. For $A \subseteq E$ we write $\mu(A) = \sum_{x\in A}\mu(x)$. The *uniform distribution* on E is the distribution ν defined by $\nu(x) = 1/|E|$. The *support* of μ is the set $\{x : \mu(x) > 0\}$. The *total variation distance* between distributions μ_1 and μ_2 on E is

$$\|\mu_1 - \mu_2\|_{\mathrm{TV}} = \frac{1}{2}\sum_{x\in E}|\mu_1(x) - \mu_2(x)| = \max_{A\subseteq E}|\mu_1(A) - \mu_2(A)|.$$

We say that μ is ε-uniform on E if $\|\mu - \nu\|_{\mathrm{TV}} \leq \varepsilon$.

Permutations and the symmetric group. Let $N > 0$ be a positive integer and $[N] = \{0,\ldots,N-1\}$. A permutation on $[N]$ is a bijection from $[N]$ to $[N]$. The symmetric group S_N over $[N]$ is the group consisting of the set of all permutations on $[N]$ with function composition as the group operation and identity element id defined by $\mathrm{id}(i) = i$. The inverse x^{-1} of an element $x \in S_N$ is the map satisfying $x^{-1}\cdot x = x\cdot x^{-1} = \mathrm{id}$. A *transposition* $(i\ j) \in S_N$ of i and j is the permutation that swaps the elements i and j, but leaves other elements in place. We say that a set $H \subseteq S_N$ generates S_N if every element of S_N can be expressed as a finite product of elements in H and their inverses. We use $\cdot$ and $\circ$ interchangeably to denote function composition.

Let μ be a symmetric probability distribution on S_N, i.e., $\mu(x) = \mu(x^{-1})$. The *random walk on* S_N with increment distribution μ is a discrete time Markov chain with state space S_N. In each step, a random element x is sampled according μ and the chain moves from state y to state xy. Thus, the probability of transitioning from state x to state yx is $\mu(y)$. The holding probability of the random walk is $\alpha = \mu(\mathrm{id})$. The following remark summarises some useful properties of such random walks; see e.g. [41] for proofs.

▶ Remark 1. Let μ be an increment distribution for a random walk on S_N.

1. The uniform distribution ν on S_N is a stationary distribution for the random walk.
2. The random walk is reversible if and only if μ is symmetric.
3. The random walk is irreducible if and only if the support of μ generates S_N.
4. If $\mu(\mathrm{id}) > 0$, then the random walk is aperiodic.

Mixing times. Let ν be the uniform distribution on S_N and be $p^{(t)}$ be the probability distribution over states of the chain after t steps. Following [28], we define the ℓ^s-norm and the normalised ℓ^s-distance to stationarity for $s > 0$ as:

$$\|\mu\|_s = \left(\sum_x |\mu(x)|^s\right)^{1/s} \quad \text{and} \quad d_s(t) = |S_N|^{1-1/s}\cdot\|p^{(t)} - \nu\|_s.$$

The total variation distance and the normalised distances satisfy $2\left\|p^{(t)} - \nu\right\|_{\mathrm{TV}} = d_1(t) \leq d_2(t)$, where the latter inequality follows from the Cauchy-Schwarz inequality. We define the ε-mixing time as $\tau(\varepsilon) = \min\{t : d_1(t) \leq 2\varepsilon\}$. We refer to the value $\tau_{\mathrm{mix}} = \tau(1/2)$ as the *mixing time* of the walk. Note that $\tau(\varepsilon) \leq \lceil\log_2\varepsilon^{-1}\rceil\cdot\tau_{\mathrm{mix}}$.

Tasks. Let Σ and Γ be nonempty finite sets of input and output labels, respectively. A task Π on a set V of n nodes is a function Π that maps any input labelling $z\colon V \to \Sigma$ to a set $\Pi(z) \subseteq \Gamma^V$ of feasible output labellings. If $\Pi(z) = \emptyset$, then we say that z is an infeasible input. We focus on two tasks:

- In leader election, the input is the constant function $z(v) = 1$ and the output labelling z' is feasible iff there exists $v \in V$ such that $z'(v) = 1$ and $z'(u) = 0$ for all $u \neq v$. That is, exactly one node should output 1 and all others should output 0.
- In the majority task, the inputs are given by $z\colon V \to \{0,1\}$ and $z' \in \Pi(z)$ if $z'(v) = b$, where b is the input value held by the majority of the nodes. As conventional, the input with equally many zeros and ones is taken to be infeasible.

Graphical stochastic population protocols. Let $G = (V, E)$ be a graph. In the graphical stochastic population model, the computation proceeds *asynchronously*, where in each time step $t > 0$:

1. a stochastic scheduler picks uniformly at random a pair $e_t = (u, v)$ of neighbouring nodes,
2. the nodes u and v read each other's states and update their local states.

As is common in population protocols, we assume that the node pairs are *ordered*, which will allow us to distinguish the two nodes: node u is called the *initiator* and v is the *responder*. We assume that nodes have access to independent and uniform random bits. Specifically, upon each interaction, both u and v are provided with a single random bit each. We note that this assumption is common in the context of population protocols, e.g. [35], and can be justified practically by the fact that chemical reaction network (CRN) implementations can directly obtain random bits given the structure of their interactions [22].

Formally, a protocol for a task Π is a tuple $\mathbf{A} = (f, \ell_{\text{in}}, \ell_{\text{out}})$, where $f\colon S \times \{0,1\} \times S \times \{0,1\} \to S \times S$ is the state transition function and S is the set of states, $\ell_{\text{in}}\colon \Sigma \to S$ maps inputs to initial states, and $\ell_{\text{out}}\colon S \to \Gamma$ maps states to outputs. A configuration is a map $x\colon V \to S$ and $x_0 = \ell_{\text{in}} \circ z$ is the initial configuration on input z. An asynchronous schedule is a random sequence $(e_t)_{t \geq 1}$ of the interaction pairs. An execution is the sequence $(x_t)_{t \geq 0}$ of configurations given by

$$x_{t+1}(u), x_{t+1}(v) = f\left(x_t(u), q_{t+1}(u), x_t(v), q_{t+1}(v)\right) \text{ and } x_{t+1}(w) = x_t(w) \text{ for } w \in V \backslash \{u, v\},$$

where $(u, v) = e_{t+1}$ and $q_{t+1}(u) \in \{0,1\}$ is the random bit provided to the node u during the interaction. The output of the protocol at step t is given by $z'_t = \ell_{\text{out}} \circ x_t$.

We say that $\mathbf{A}$ stabilises on input z by step T if $z'_{t+1} = z'_t$ and $z'_t \in \Pi(z)$ holds for all $t \geq T$. Moreover, $\mathbf{A}$ solves the task Π with probability at least p in $T(\mathbf{A})$ steps if the protocol stabilises by step $T(\mathbf{A})$ on any feasible input with probability at least p. The state complexity of the protocol is $S(\mathbf{A}) = |S|$, i.e., the number of states used by the protocol.

Synchronous token protocols. In the synchronous k-token shuffling model, we assume that there are n agents which communicate in a round-based fashion using *tokens*. In each round,

1. every node v generates exactly k tokens based on its current state,
2. all nk tokens are shuffled uniformly at random so that each node gets exactly k tokens,
3. every node v updates its local state based on its current state and the k tokens it received.

Let X be the set of states a node can take and Y be a set of distinct token types. An algorithm in the token shuffling model is a tuple $\mathbf{B} = (f, g, \ell_{\text{in}}, \ell_{\text{out}})$. The map $f\colon X \times Y^k \to X$ is a state transition function, and $g\colon X \to Y^k$ determines which tokens each node creates at the start of each round. As before, $\ell_{\text{in}}\colon \Sigma \to X$ maps input values to initial states and $\ell_{\text{out}}\colon X \to \Gamma$ maps the state of a node onto an output value. The initial configuration on input z is $x_0 = \ell_{\text{out}} \circ z$.

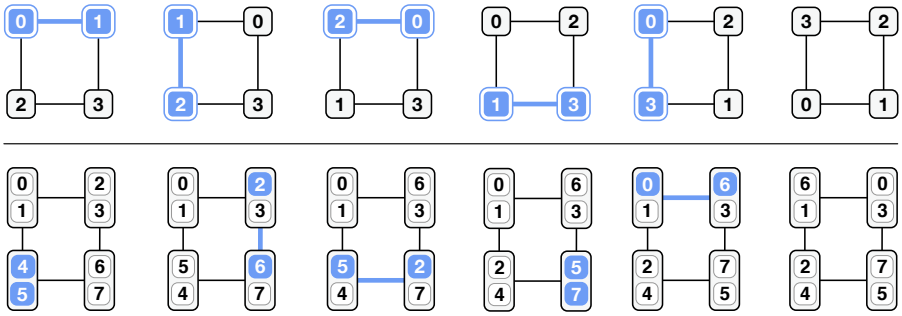

Figure 3 Interchange dynamics on a 4-cycle. In each step, blue cards are swapped. Top row: The 1-stack interchange process. Bottom row: The 2-stack interchange process. In each step, a randomly selected node either moves its top card to the bottom of its stack or exchanges it with the top card of a randomly selected neighbour.

A *synchronous schedule* is a sequence $(\sigma_r)_{r\geq 1}$, where the permutation $\sigma_r \in S_{nk}$ describes how the tokens are shuffled in round r. For any $y\colon [nk] \to Y$, we let $y(v_0,\ldots,v_{k-1}) = (y(v_0),\ldots,y(v_{k-1}))$. A synchronous execution induced by $(\sigma_r)_{r\geq 1}$ on input z is defined by

$$y_{r+1}(v_0,\ldots,v_{k-1}) = (g\circ x_r)(v) \quad \text{and} \quad x_{r+1}(v) = f\left(x_r(v), (y_{r+1}\circ\sigma_{r+1})(v_0,\ldots,v_{k-1})\right),$$

where $y_r(v_0,\ldots,v_{k-1})$ and $(y_r\circ\sigma_{r+1})(v_0,\ldots,v_{k-1})$, respectively, are the k tokens generated and received by node v during round r.

We assume the uniform synchronous scheduler, which picks each permutation σ_r independently and uniformly at random from the set of all permutations S_{nk}. The output of node v at the end of round r is $z'_r(v) = (\ell_{\text{out}}\circ x_r)(v)$. The synchronous algorithm $\mathbf{B}$ stabilises on input z in R rounds if $z_{r+1} = z'_r$ and $z'_r \in \Pi(z)$ holds for all $r \geq R$. The algorithm solves the problem Π if it stabilises in R rounds on any feasible input with probability at least p.

4 Shuffling states on graphs: the k-stack interchange process

We now describe a shuffling process on graphs, which we call the *k-stack interchange process*. This process will be useful in our analysis, and is a variant of the classic graph interchange process, e.g. [30, 38]. We analyse its mixing time using the path comparison method of Diaconis and Saloff-Coste [28], leveraging a classical flow result of Leighton and Rao [40].

The k-stack interchange process. Let $G = (V, E)$ a graph with n vertices $\{0,\ldots,n-1\}$ and $N = kn$ for $k > 0$. Assume each node of G holds a stack of exactly k cards, and consider the shuffling process where, in every time step, one of the following actions is taken:

1. with probability $1/2$, move the top card of a random node to the bottom of its stack,
2. with probability $1/4$, choose a random edge $\{u, v\}$ and swap the top cards of u and v,
3. with probability $1/4$, do nothing.

We refer to this process as the *k-stack interchange process on G*. The special case of $k = 1$ is the classic interchange process on G with holding probability $3/4$, as the first rule does not do anything on stacks of size 1. For $k > 1$, the holding probability will be $1/4$. Instances of the process for $k = 1$ and $k = 2$ are illustrated in Figure 3.

▶ **Theorem 2.** *Let G be a d-regular graph with edge expansion $\beta > 0$. For any constant $k > 0$, the mixing time of the k-stack interchange process on G is $O\left((d/\beta)^2\, n \log^3 n\right)$.*

We prove this theorem in the full version of the paper [6]. In Section 6, we will show that this shuffling process can be implemented efficiently in the graphical population protocol model.

5 Decentralised *graphical* phase clocks

We now describe a *bounded phase clock* construction for the stochastic population protocol model over regular graphs. Interestingly, the construction can be generalised to *non-regular* graphs, assuming that node degrees do not deviate too much from the average degree; see the full version [6] for details. Our approach generalises that of Alistarh et al. [4], who built a leaderless phase clock on cliques leveraging the classic two-choice load balancing process [14, 44].

Phase clocks. Let $\phi > 0$ be an integer and consider a population protocol $\mathbf{C}$ with state variables $c(v) \in \{0, \dots, \phi - 1\}$ for each $v \in V$. The variable $c(v)$ represents the value of the clock at node v. Let $c(v, t)$ be the clock value node v has at the end of time step t (regardless of whether it was active during that step). We define the distance D between two clock values and the skew Δ of the clock at the end of step t, respectively, as follows:

$$D(x, y) = \min\{|x - y|, \phi - |x - y|\} \quad \text{and} \quad \Delta(t) = \max_{u,v \in V} D\left(c(u, t), c(v, t)\right).$$

We say that the protocol $\mathbf{C}$ implements a (ϕ, γ, κ)-clock if for all $t \geq 0$ the following hold:

1. $\Pr[\Delta(t) \geq \gamma] < t/n^{\kappa}$, and
2. $c(v, t + 1) = c(v, t) + 1 \bmod \phi$ for exactly one $v \in V$ and $c(u, t + 1) = c(u, t)$ for all $u \in V \setminus \{v\}$.

Intuitively, ϕ is the length of a phase, γ is the skew of the clock, and κ controls the failure probability. The above properties guarantee that the clocks (1) have a skew bounded by γ for polynomially many steps, w.h.p.; and (2) in each step, the clocks make progress (at some node). A clock protocol $\mathbf{C}$ *fails* at time t if $\Delta(t) \geq \gamma$ occurs. Several types of phase clocks have been proposed in the population protocol literature, e.g. [10, 35, 4, 46].

Bounded phase clocks via graphical load balancing. Let G be a graph and suppose that each node of G contains a bin, which is initially empty. Our phase clock is based on the classic graphical load-balancing process [44] where, in each step, a directed edge (u, v) is sampled uniformly at random and a ball is placed into the *least* loaded of bin among the two nodes connected by the edge (in case of ties, place the ball into bin u). Using this idea, we obtain *bounded* phase clocks in the graphical population protocol model. We note that this is the only place in our framework where the initiator/responder distinction is used. The proof of this result can be found in the full version [6].

▶ **Theorem 3.** *Let $G = (V, E)$ be a d-regular graph with n nodes and edge expansion $\beta > 0$ and let $\kappa > 1$ be a constant. There exists a constant $c(\kappa)$ such that for any γ and ϕ satisfying*

$$\gamma \geq c(\kappa)\frac{d}{\beta}\log n \quad \text{and} \quad \phi \geq 2\gamma$$

there exists (ϕ, γ, κ)-clock on G that uses ϕ states per node.

6 Simulating synchronous token shuffling protocols

In this section, we give our main technical result: synchronous protocols in the fully-connected token shuffling model can be simulated in the graphical, stochastic population protocol model.

▶ **Theorem 4.** *Let $k > 0$ be a constant and $\mathbf{A}$ be a synchronous k-token shuffling protocol on n nodes, where X is the set of local states and Y the set of token types used the protocol $\mathbf{A}$. If $\mathbf{A}$ solves the task Π with high probability in $R \in \mathrm{poly}(n)$ rounds, then there exists a stochastic population protocol $\mathbf{B}$ that also solves task Π with high probability on any n-node d-regular graph G with edge expansion $\beta > 0$. The step complexity $T(\mathbf{B})$ and state complexity $S(\mathbf{B})$ of the protocol $\mathbf{B}$ satisfy*

$$T(\mathbf{B}) \in O\left(R \cdot n \cdot \zeta\right) \quad \text{and} \quad S(\mathbf{B}) \in O\left(|X| \cdot |Y|^k \cdot \zeta\right) \quad \text{with} \quad \zeta = \log n \cdot \left(\frac{d}{\beta} + \frac{\tau_{mix}}{n}\right),$$

where τ_{mix} is the mixing time of the k-stack interchange process on G.

Notation. The rest of this section is dedicated to proving this theorem. Throughout, we fix $R = R(n) \in \mathrm{poly}(n)$ and $\varepsilon = 1/n^a < 1/(Rn^\lambda)$ for an arbitrary large constant $a > 0$. Let $G = (V, E)$ be d-regular n-node graph and $N = kn$. We use μ to denote the increment distribution of the k-stack interchange process on the graph G. The support of μ is the set $H \subseteq S_N$ and $\tau = \tau(\varepsilon)$ is the ε-mixing time of the k-stack interchange process.

6.1 The token shuffling protocol

We now give a stochastic population protocol that simulates uniform schedules of the synchronous token shuffling model. The protocol simulates the random walk made by the k-stack interchange process, synchronised by phase clocks.

Setting up the clock. We choose the parameter $\kappa > 0$ such that a (ϕ, γ, κ)-clock $\mathbf{C}$ with parameters given by

$$\gamma \in \Theta\left(\frac{d}{\beta} \log n\right) \qquad \phi = \gamma + \vartheta \qquad \vartheta = \frac{2\tau}{n} + 3\gamma \qquad t^* = (R\phi + \gamma)n$$

fails (i.e., the clock skew becomes γ or greater) with probability at most $1/n^\lambda$ during the first t^* steps. Since $\phi \geq 2\gamma$, $R \in \mathrm{poly}(n)$, and $t^* \in \mathrm{poly}(n)$ hold, such a protocol exists by Theorem 3 for any constant $\lambda > 0$ by choosing a sufficiently large κ. The fact that t^* is polynomially bounded follows from Theorem 2 and that $\beta \geq 1/n^2$ for any regular connected graph. Further, $\tau \leq \lceil \log 1/\varepsilon \rceil \cdot \tau_{\mathrm{mix}} \in \mathrm{poly}(n)$, and hence, $\phi, \gamma \in \mathrm{poly}(n)$.

The token shuffling protocol. The parameter ϑ is used as a special threshold value for the token shuffling protocol. We assume that each node v holds exactly k tokens, which are ordered from 0 to $k-1$, in the same manner as cards ordered are in the k-stack interchange process. We say that the first token is the *top token*. We say that node u is *receptive* when ever its clock satisfies $c(u) < \vartheta$ and that it is *suspended* otherwise. When nodes in $\{u, v\}$ interact, they apply the following rule:

1. If both are receptive, that is, $c(u) < \vartheta$ and $c(v) < \vartheta$ holds, then
   a. Let $q(u)$ and $q(v)$ be the random coin flips of u and v, respectively.
   b. If $q(u) = q(v) = 0$, then u and v swap their top tokens.
   c. If $q(u) < q(v)$, then v moves its top token to the bottom of its stack; u does nothing.
   d. If $q(u) = q(v) = 1$, then do nothing.
2. Otherwise, do nothing.

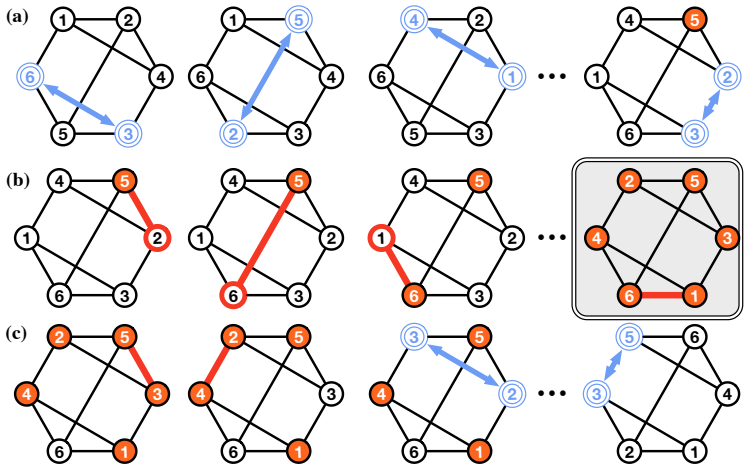

Figure 4 The dynamics of the shuffling protocol for $k = 1$. Circles filled with white and red denote receptive and suspended nodes, respectively. The blue arrows connect nodes who exchange their tokens in the given step. Red lines denote steps, where at least one of the interacting nodes is suspended, and thus, no swap is made. (a) Initially all nodes are receptive and swap tokens with their interaction partners. After sufficiently many interactions, nodes become suspended and refrain from swapping tokens. (b) Eventually all nodes are suspended. The highlighted panel shows the resulting permutation, which will act as the interaction pattern for the simulated round. (c) As the phase clocks reset back to 0, nodes become receptive again, and the tokens are shuffled once more.

The protocol uses at most one random bit per node per interaction and that this is the only part of our framework, where the random bits provided to the nodes are used. The interacting nodes exchange at most 4 bits (i.e., whether they receptive or not, and the result of their coin flip) in addition to the contents of the swapped tokens in Step (1b). Finally, observe that when all nodes are receptive, the tokens are shuffled according to the increment distribution μ of the k-stack interchange process on G. Figure 4 illustrates the dynamics of the shuffling protocol in the case $k = 1$.

6.2 Properties of the shuffling protocol

We now analyse the above shuffling protocol. Let $c(u, t)$ indicate the clock value of node u at the end of step t. Let $c(u, 0) = 0$ and $t(v, 0) = 0$. We say that the clock of node u resets at time step t if its value transitions from $\phi - 1$ to 0. For $r \geq 0$, define

- $t(v, r+1) = \min\{t > t(v, r) : c(v, t) = 0\}$; the step when v resets its clock for the rth time,
- $t_{\min}(r) = \min\{t(v, r) : v \in V\}$; the *earliest* step when some clock is reset for the rth time,
- $t_{\max}(r) = \max\{t(v, r) : v \in V\}$; the *latest* step when some clock is reset for the rth time.

Similarly, we define the times with respective to the events when the clocks reach the value ϑ:

- $s(v, r) = \min\{t > t(v, r) : c(v, t) = \vartheta\}$,
- $s_{\min}(r) = \min\{s(v, r) : v \in V\}$,
- $s_{\max}(r) = \max\{s(v, r) : v \in V\}$.

The following lemma captures the relationship between the timing of these events.

▶ **Lemma 5.** *With high probability, the following inequalities hold:*

1. $t_{\max}(R + 1) \leq t^* = (R\phi + \gamma)n$,
2. $s_{\min}(r) - t_{\max}(r) \geq \tau$ *for each* $1 \leq r \leq R$.
3. $t_{\max}(r) < s_{\max}(r) < t_{\min}(r + 1)$ *for each* $1 \leq r \leq R$.

Distribution of tokens. We now show that the distribution tokens mix to an ε-uniform distribution during the intervals $\{t_{\max}(r)+1,\dots,s_{\min}(r)\}$ for $1 \le r \le R$. Let $\pi_0 = \mathrm{id}$ and π_t denote the locations of the tokens after t steps of the shuffling protocol. Define $\sigma_0 = \mathrm{id}$ and

$$\sigma_r = \pi_{s_{\max}(r)} \text{ for } 1 \le r \le R.$$

Observe that $\sigma_r = \rho_3 \cdot \rho_2 \cdot \rho_1 \cdot \sigma_{r-1}$, where each ρ_i is product of elements from the support $H \subseteq S_N$ of the increment distribution μ of the k-stack interchange process, where

- $\rho_1 = x_{t_{\max}(r)} \cdots x_{t_{\min}(r-1)+1}$ (a subset of nodes have become receptive for the rth time),
- $\rho_2 = x_{s_{\min}(r)} \cdots x_{t_{\max}(r)+1}$ (all nodes are receptive),
- $\rho_3 = x_{s_{\max}(r)} \cdots x_{s_{\min}(r)+1}$ (a subset of nodes have become suspended for the rth time).

(Recall that permutations are applied from right to left.) Observe that while each x_i is a random element of H, only the elements ρ_2 are guaranteed to be distributed according to the increment distribution μ of the k-stack interchange process. The elements of ρ_1 and ρ_3 are skewed towards the identity permutation, as some nodes are suspended whenever their clock values are in $\{\vartheta,\dots,\phi-1\}$. The next lemma establishes that this does not interfere with the mixing behaviour.

▶ **Lemma 6.** *Let $0 \le r < R$. For any $A \subseteq S_N$, we have $|\Pr[\sigma_{r+1} \in A \mid \sigma_r] - \nu(A)| \le \varepsilon$.*

6.3 The simulation protocol

Using the shuffling protocol in the population protocol model, we can simulate an R-round algorithm $\mathbf{A}$ in the synchronous k-token shuffling model. Let $f\colon X \times Y^k \to X$ be the state transition function and $g\colon X \to Y^k$ be the token generation function of the algorithm $\mathbf{A}$. Recall that X and Y denote the sets of local states and token types, respectively.

The simulation protocol. Each node v maintains the following variables:

- $a(v) \in X$ to simulate the local state of the synchronous protocol $\mathbf{A}$,
- $b_0(v),\dots,b_{k-1}(v) \in Y$ to store the sent and received tokens, and
- $r(v) \in \{0,1,\dots,R\}$ to store the number of simulated rounds.

The variable $a(v)$ is initialised to the initial state $x_0(v)$ of node v in the algorithm $\mathbf{A}$ and $b_0(v),\dots,b_{k-1}(v)$ are initialised to the values given by $g(x_0(v))$. The variable $r(v)$ is initially set to 0. When node v interacts (in the asynchronous population protocol model), v updates its state according to the following rules:

1. Run the clock and the shuffling protocol using $b_0(v),\dots,b_{k-1}(v)$ to hold the k tokens.
2. If $c(v) = \vartheta$, then
   - update the round counter and set $r(v) \leftarrow \min\{r(v)+1, R\}$,
   - compute the new state $a(v) \leftarrow f\left(a(v), b_0(v),\dots,b_{k-1}(v)\right)$, and
   - generate new tokens $b_0(v),\dots,b_{k-1}(v) \leftarrow g(a(v))$.

As output value of the simulation, node v uses the output value algorithm $\mathbf{A}$ associates to state $a(v)$. The above algorithm simulates an execution of the synchronous algorithm $\mathbf{A}$ under the schedule $\sigma_1,\dots,\sigma_R$ given by the shuffling protocol. To this end, define $x_0(v) = a(v,0)$ and $x(r) = a(v, s(v,r))$ for all $1 \le r \le R$.

▶ **Lemma 7.** *With high probability, the sequence $(x_r)_{0\le r\le R}$ is an execution induced by the schedule $(\sigma_r)_{1\le r\le R}$.*

6.4 From almost-uniform schedules to uniform schedules

The schedules provided by the shuffling protocol are only ε-uniform, as the shuffling process is executed for finitely many steps. We now show that this does not matter: any synchronous protocol behaves statistically similarly under ε-uniform and uniform schedules.

To formalise this, let Φ be the distribution over sequences $(\sigma_1, \ldots, \sigma_R) \in S_N^R$ of permutations generated by the shuffling protocol under the assumption that the clock protocol works correctly for T time steps. Let $\nu^R = \nu \times \cdots \times \nu$ denote the distribution of a sequence of R independently and uniformly sampled random permutations from S_N. That is, ν^R is the distribution of the uniform R-round schedules. The following then holds:

▶ **Lemma 8.** *The total variation distance between Φ and ν^R satisfies $\left\|\Phi - \nu^R\right\|_{\mathrm{TV}} \leq \varepsilon R$.*

Together with the following lemma, we can show that protocols simulated under the ε-uniform schedules behave almost the same as under perfectly uniform schedules.

▶ **Lemma 9.** *Let μ and ν be probability distributions over a finite domain Ω. For any function $F\colon \Omega \to \Omega'$, the total variation distance satisfies $\|F(\mu) - F(\nu)\|_{\mathrm{TV}} \leq \|\mu - \nu\|_{\mathrm{TV}}$.*

6.5 The main simulation theorem

With all the pieces now in place, we can now state our simulation theorem.

▶ **Theorem 4.** *Let $k > 0$ be a constant and $\mathbf{A}$ be a synchronous k-token shuffling protocol on n nodes, where X is the set of local states and Y the set of token types used the protocol $\mathbf{A}$. If $\mathbf{A}$ solves the task Π with high probability in $R \in \mathrm{poly}(n)$ rounds, then there exists a stochastic population protocol $\mathbf{B}$ that also solves task Π with high probability on any n-node d-regular graph G with edge expansion $\beta > 0$. The step complexity $T(\mathbf{B})$ and state complexity $S(\mathbf{B})$ of the protocol $\mathbf{B}$ satisfy*

$$T(\mathbf{B}) \in O\left(R \cdot n \cdot \zeta\right) \quad \text{and} \quad S(\mathbf{B}) \in O\left(|X| \cdot |Y|^k \cdot \zeta\right) \quad \text{with} \quad \zeta = \log n \cdot \left(\frac{d}{\beta} + \frac{\tau_{mix}}{n}\right),$$

where τ_{mix} is the mixing time of the k-stack interchange process on G.

7 Applications: leader election and exact majority

Using Theorem 4, we can automatically transport algorithms from the *fully-connected synchronous token shuffling model* to the graphical, asynchronous population protocol model. We utilise this result to obtain fast protocols for leader election and exact majority in the graphical population protocol model.

The leader election protocol for the token shuffling model uses a one-way information dissemination protocol and a protocol for generating synthetic coins in the token shuffling model with $k > 1$. Specifically, we show the following result.

▶ **Theorem 10.** *There is a synchronous 2-token shuffling protocol for the leader election task that stabilises in $O(\log^2 n)$ rounds with high probability, uses $O(\log n)$ states per node and two token types.*

For exact majority in the token shuffling model, we give an algorithm that simulates two-way interactions in a population of $2n$ virtual agents. The algorithm uses the classic cancellation-doubling dynamics used in the clique model [13, 34], yielding the following result.

▶ **Theorem 11.** *There is a synchronous 2-token shuffling protocol for the exact majority task that stabilises in $O(\log^2 n)$ rounds with high probability, uses $O(\log n)$ states and five token types.*

8 Conclusions

In this work, we established a general framework for simulating clique-based protocols in arbitrary, connected regular graphs. We now conclude by briefly discussing some limitations of our approach and summarise key problems left open by this work.

First, we focused on regular interaction graphs. The justification for this assumption is two-fold. First, this assumption is only used once: in Section 5, to obtain clean bounds for the skew of the phase clock. However, upon close inspection, we notice that this regularity assumption can be relaxed in many cases if the minimum and maximum degrees do not deviate too much from the average degree of the graph [6]. Second, regular graphs give a natural extension of the notion of *parallel time*, since all nodes interact at the same rate.

The simulation overhead has a polylogarithmic dependency on n. We have made no particular effort to optimise the degree of this polylogarithmic dependency. The dependency can be improved by providing better bounds on the k-stack interchange process. Indeed, even in the case of the well-studied (1-stack) interchange process, exact bounds on mixing time have been – and still remain – an open question for many graph classes [38]. Improved bounds for these processes imply better running time bounds for our simulations.

Finally, our complexity bounds have a quadratic dependency on d/β. We suspect a polynomial dependency on the expansion properties is necessary for step complexity and leave the investigation of tight space-time trade-offs for population protocols in the general graphical setting as an intriguing open problem.

References

1 David Aldous. Random walks on finite groups and rapidly mixing Markov chains. In *Séminaire de Probabilités XVII 1981/82*, pages 243–297. Springer, 1983.

2 David Aldous and James Allen Fill. Reversible markov chains and random walks on graphs, 2002. Unfinished monograph, recompiled 2014, available at `http://www.stat.berkeley.edu/users/aldous/RWG/book.html`.

3 Dan Alistarh, James Aspnes, David Eisenstat, Rati Gelashvili, and Ronald L Rivest. Time-space trade-offs in population protocols. In *Proc. 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, pages 2560–2579, 2017.

4 Dan Alistarh, James Aspnes, and Rati Gelashvili. Space-optimal majority in population protocols. In *Proc. 29th ACM-SIAM Symposium on Discrete Algorithms (SODA 2018)*. SIAM, 2018. `doi:10.1137/1.9781611975031.144`.

5 Dan Alistarh and Rati Gelashvili. Recent algorithmic advances in population protocols. *SIGACT News*, 49(3):63–73, 2018. `doi:10.1145/3289137.3289150`.

6 Dan Alistarh, Rati Gelashvili, and Joel Rybicki. Fast graphical population protocols. Full version. `arXiv:2102.08808`.

7 Dan Alistarh, Rati Gelashvili, and Milan Vojnović. Fast and exact majority in population protocols. In *Proc. 34th ACM Symposium on Principles of Distributed Computing (PODC 2015)*, pages 47–56, 2015.

8 Dana Angluin, James Aspnes, Zoë Diamadi, Michael J Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed computing*, 18(4):235–253, 2006.

9 Dana Angluin, James Aspnes, and David Eisenstat. Stably computable predicates are semilinear. In *Proc. 25th ACM Symposium on Principles of distributed computing (PODC 2006)*, pages 292–299, 2006.

10 Dana Angluin, James Aspnes, and David Eisenstat. Fast computation by population protocols with a leader. *Distributed Computing*, 21(3):183–199, 2008. `doi:10.1007/s00446-008-0067-z`.

11 Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007.

12 Dana Angluin, James Aspnes, Michael J Fischer, and Hong Jiang. Self-stabilizing population protocols. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 3(4):1–28, 2008.

13 James Aspnes and Eric Ruppert. An introduction to population protocols. In *Middleware for Network Eccentric and Mobile Applications*, pages 97–120. Springer, 2009.

14 Yossi Azar, Andrei Z. Broder, Anna R. Karlin, and Eli Upfal. Balanced allocations. *SIAM Journal on Computing*, 29(1):180–200, 1999.

15 Joffroy Beauquier, Peva Blanchard, and Janna Burman. Self-stabilizing leader election in population protocols over arbitrary communication graphs. In *International Conference on Principles of Distributed Systems*, pages 38–52. Springer, 2013. URL: `https://hal.archives-ouvertes.fr/hal-00867287v2`.

16 Petra Berenbrink, Robert Elsässer, Tom Friedetzky, Dominik Kaaser, Peter Kling, and Tomasz Radzik. A population protocol for exact majority with $O(\log_{5/3} n)$ stabilization time and $\Theta(\log n)$ states. In *Proc. 32nd International Symposium on Distributed Computing (DISC 2018)*, pages 10:1–10:18, 2018. `doi:10.4230/LIPIcs.DISC.2018.10`.

17 Petra Berenbrink, Tom Friedetzky, Peter Kling, Frederik Mallmann-Trenn, and Chris Wastell. Plurality consensus in arbitrary graphs: Lessons learned from load balancing. In *Proc. 24th Annual European Symposium on Algorithms (ESA 2016)*, volume 57, pages 10:1–10:18, 2016. `doi:10.4230/LIPIcs.ESA.2016.10`.

18 Petra Berenbrink, George Giakkoupis, and Peter Kling. Tight bounds for coalescing-branching random walks on regular graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1715–1733. SIAM, 2018.

19 Petra Berenbrink, George Giakkoupis, and Peter Kling. Optimal time and space leader election in population protocols. In *Proc. 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC 2020)*, pages 119–129, 2020.

20 Petra Berenbrink, Dominik Kaaser, Peter Kling, and Lena Otterbach. Simple and efficient leader election. In *Proc. 1st Symposium on Simplicity in Algorithms (SOSA 2018)*, pages 9:1–9:11, 2018. `doi:10.4230/OASIcs.SOSA.2018.9`.

21 Michael Blondin, Javier Esparza, and Stefan Jaax. Large flocks of small birds: on the minimal size of population protocols. In *Proc. 35th Symposium on Theoretical Aspects of Computer Science (STACS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

22 Robert Brijder. Computing with chemical reaction networks: a tutorial. *Natural Computing*, 18(1):119–137, 2019.

23 Pietro Caputo, Thomas M. Liggett, and Thomas Richthammer. Proof of Aldous' spectral gap conjecture. *Journal of the American Mathematical Society*, 23(3):831–851, 2010.

24 Hsueh-Ping Chen and Ho-Lin Chen. Self-stabilizing leader election. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 53–59, 2019.

25 Hsueh-Ping Chen and Ho-Lin Chen. Self-stabilizing leader election in regular graphs. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, PODC '20, pages 210–217, New York, NY, USA, 2020. Association for Computing Machinery. `doi:10.1145/3382734.3405733`.

26 Colin Cooper, Robert Elsaßser, Hirotaka Ono, and Tomasz Radzik. Coalescing random walks and voting on connected graphs. *SIAM Journal on Discrete Mathematics*, 27(4):1748–1758, 2013. `doi:10.1137/120900368`.

27 Colin Cooper, Tomasz Radzik, Nicolás Rivera, and Takeharu Shiraga. Fast plurality consensus in regular expanders. In *Proc. 31st International Symposium on Distributed Computing (DISC 2017)*, pages 13:1–13:16, 2017. `doi:10.4230/LIPIcs.DISC.2017.13`.

28 Persi Diaconis and Laurent Saloff-Coste. Comparison techniques for random walk on finite groups. *The Annals of Probability*, 21(4):2131–2156, 1993.

29 Persi Diaconis and Mehrdad Shahshahani. Generating a random permutation with random transpositions. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 57(2):159–179, 1981.

30 AB Dieker. Interlacings for random walks on weighted graphs and the interchange process. *SIAM Journal on Discrete Mathematics*, 24(1):191–206, 2010.

31 David Doty, Mahsa Eftekhari, Leszek Gąsieniec, Eric Severson, Grzegorz Stachowiak, and Przemysław Uznański. A time and space optimal stable population protocol solving exact majority. *arXiv preprint*, 2021. `arXiv:2106.10201`.

32 David Doty and David Soloveichik. Stable leader election in population protocols requires linear time. *Distributed Computing*, 31(4):257–271, 2018.

33 Moez Draief and Milan Vojnović. Convergence speed of binary interval consensus. *SIAM Journal on Control and Optimization*, 50(3):1087–1109, 2012.

34 Robert Elsässer and Tomasz Radzik. Recent results in population protocols for exact majority and leader election. *Bulletin of the EATCS*, 126, 2018. URL: `http://bulletin.eatcs.org/index.php/beatcs/article/view/549/546`.

35 Leszek Gąsiniec and Grzegorz Stachowiak. Fast space optimal leader election in population protocols. In *Proc. 29th ACM-SIAM Symposium on Discrete Algorithms (SODA 2018)*, 2018. `doi:10.1137/1.9781611975031.169`.

36 Leszek Gąsiniec, Grzegorz Stachowiak, and Przemyslaw Uznański. Almost logarithmic-time space optimal leader election in population protocols. In *Proc. 31st ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2019)*, 2019. `doi:10.1145/3323165.3323178`.

37 Leszek Gąsiniec, Grzegorz Stachowiak, and Przemyslaw Uznański. Time and space optimal exact majority population protocols, 2020. `arXiv:2011.07392`.

38 Johan Jonasson. Mixing times for the interchange process. *Latin American Journal of Probability and Mathematical Statistics*, 9(2):667–683, 2012.

39 Anissa Lamani and Masafumi Yamashita. Realization of periodic functions by self-stabilizing population protocols with synchronous handshakes. In *International Conference on Theory and Practice of Natural Computing*, pages 21–33. Springer, 2016.

40 Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46(6):787–832, 1999.

41 David A. Levin and Yuval Peres. *Markov Chains and Mixing Times*. American Mathematical Society, 2 edition, 2017.

42 George B. Mertzios, Sotiris E. Nikoletseas, Christoforos Raptopoulos, and Paul G. Spirakis. Determining majority in networks with local interactions and very small local memory. In *Proc. 41st International Colloquium on Automata, Languages, and Programming (ICALP 2014)*, pages 871–882, 2014. `doi:10.1007/978-3-662-43948-7_72`.

43 George B Mertzios, Sotiris E Nikoletseas, Christoforos L Raptopoulos, and Paul G Spirakis. Determining majority in networks with local interactions and very small local memory. *Distributed Computing*, 30(1):1–16, 2017.

44 Yuval Peres, Kunal Talwar, and Udi Wieder. Graphical balanced allocations and the $(1+\beta)$-choice process. *Random Structures and Algorithms*, 47(4):760–775, 2014. `doi:10.1002/rsa.20558`.

45 Thomas Sauerwald and He Sun. Tight bounds for randomized load balancing on arbitrary network topologies. In *Proc. 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2012)*, pages 341–350, 2012. `doi:10.1109/FOCS.2012.86`.

46 Yuichi Sudo, Fukuhito Ooshita, Taisuke Izumi, Hirotsugu Kakugawa, and Toshimitsu Masuzawa. Logarithmic expected-time leader election in population protocol model. In *Proc. International Symposium on Stabilizing, Safety, and Security of Distributed Systems (SSS 2019)*, pages 323–337, 2019.

47 Yuichi Sudo, Fukuhito Ooshita, Hirotsugu Kakugawa, Toshimitsu Masuzawa, Ajoy K Datta, and Lawrence L Larmore. Loosely-stabilizing leader election for arbitrary graphs in population protocol model. *IEEE Transactions on Parallel and Distributed Systems*, 30(6):1359–1373, 2018.

48 Yuichi Sudo, Masahiro Shibata, Junya Nakamura, Yonghwan Kim, and Toshimitsu Masuzawa. Self-stabilizing population protocols with global knowledge. *IEEE Transactions on Parallel and Distributed Systems*, 2021.

49 Alan M. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London, Series B*, 237(641):37–72, 1952.

50 David B. Wilson. Mixing times of lozenge tiling and card shuffling Markov chains. *The Annals of Applied Probability*, 14(1):274–325, 2004.

51 Daisuke Yokota, Yuichi Sudo, and Toshimitsu Masuzawa. Time-optimal self-stabilizing leader election on rings in population protocols. In *International Symposium on Stabilizing, Safety, and Security of Distributed Systems*, pages 301–316. Springer, 2020.