Fast and Exact Majority in Population Protocols

Dan Alistarh
Microsoft Research
daalista@microsoft.com

Rati Gelashvili*
MIT
gelash@mit.edu

Milan Vojnović
Microsoft Research
milanv@microsoft.com

ABSTRACT

Population protocols, roughly defined as systems consisting of large numbers of simple identical agents, interacting at random and updating their state following simple rules, are an important research topic at the intersection of distributed computing and biology. One of the fundamental tasks that a population protocol may solve is *majority*: each node starts in one of two states; the goal is for all nodes to reach a correct consensus on which of the two states was initially the majority. Despite considerable research effort, known protocols for this problem are either exact but slow (taking linear parallel time to converge), or fast but approximate (with non-zero probability of error).

In this paper, we show that this trade-off between precision and speed is not inherent. We present a new protocol called *Average and Conquer (AVC)* that solves majority *exactly* in expected parallel convergence time $O(\log n/(s\epsilon) + \log n \log s)$, where n is the number of nodes, ϵn is the initial node advantage of the majority state, and $s = \Omega(\log n \log\log n)$ is the number of states the protocol employs. This shows that the majority problem can be solved exactly in time poly-logarithmic in n, provided that the memory per node is $s = \Omega(1/\epsilon + \log n \log 1/\epsilon)$. On the negative side, we establish a lower bound of $\Omega(1/\epsilon)$ on the expected parallel convergence time for the case of four memory states per node, and a lower bound of $\Omega(\log n)$ parallel time for protocols using any number of memory states per node.

Categories and Subject Descriptors

F.1.1 [**Computation by Abstract Devices**]: Models of Computation; F.1.1 [**Computation by Abstract Devices**]: --*Parallelism and concurrency*

Keywords

Population protocols, majority computation, randomized algorithms

*Work performed in part while an intern with Microsoft Research.

PODC'15, July 21–23, 2015, Donostia-San Sebastián, Spain.

DOI: http://dx.doi.org/10.1145/2767386.2767429..

1. INTRODUCTION

Population protocols [AAD+06], roughly defined as systems consisting of large numbers of randomly interacting identical agents, which update their state using simple rules, have recently become an important topic of research in distributed computing. In particular, considerable research effort has gone into determining the set of tasks solvable by population protocols, e.g., [AAD+06,AAER07], and into understanding the convergence bounds achievable by these algorithms in the case of fundamental tasks, e.g., [AAE08, PVV09, DV12, MNRS14]. Recently, research in biology and nanotechnology has shown that such algorithms can in fact be implemented at the level of molecules [CDS+13], and that there are non-trivial connections between population protocols and the computational tasks which biological cells must solve in order to function correctly [CCN12].

One fundamental task in the context of population protocols is *majority computation*: given a set of n nodes, starting in one of two distinct states (A or B), the task requires the nodes to reach consensus on a value associated to one of the two initial states. Crucially, the final value the nodes converge on should be associated to the *majority initial state*.

Evidence suggests that the cell cycle switch in eukaryotic cells solves an approximate version of majority [CCN12]; a three-state population protocol for approximate majority was empirically studied as a model of epigenetic cell memory by nucleosome modification [DMST07]; also, this task is a key component when simulating register machines via population protocols [AAE08]. The majority task has been studied by different research communities by considering population protocols in computation models defined in discrete and continuous time. In the discrete model, e.g., [AAE08], in each *round*, a pair of nodes is drawn at random to interact. Alternatively, other sources, e.g., [PVV09, DV12] assume that pairs of nodes interact at instances of a Poisson process in continuous time. The two models are essentially equivalent. Possible interactions are usually specified as a graph on n nodes, where the complete graph is an interesting special case, especially in the context of biological computations. In both models, we are interested in the amount of time until the algorithm converges to the correct state. In the discrete-time model, the metric of interest is *parallel time*, defined as number of rounds to convergence divided by the number of nodes n, which corresponds to "real-time" until convergence in the continuous-time model.

The complexity landscape of the majority task is quite intriguing. Assuming a system of n nodes, where one of the initial states (say, A) has an initial relative advantage over the other state (B) of ϵn nodes, there exists an elegant four-state protocol that can solve the problem in expected $O(\log n/\epsilon)$ parallel time [DV12, MNRS14]. This al-

gorithm solves *exact* majority, i.e. has 0 probability of error; however, its running time is polynomial in n for small initial discrepancy between the two states, e.g., $\epsilon n = \Theta(n^\alpha)$, for $\alpha < 1$. At the same time, there exists an even simpler *three-state* protocol which can compute *approximate majority* in $O(\log n)$ parallel time, with high probability in n, assuming that the initial relative advantage is $\epsilon n = \omega(\sqrt{n}\log n)$ [AAE08, PVV09]. For small ϵ, this can be exponentially faster than the four-state protocol, but the price of fast convergence is *approximation*: the failure probability of the three-state protocol is $\exp(-c\epsilon^2 n)$ for constant $c > 0$ and small ϵ [PVV09], which becomes constant for small ϵ. Furthermore, it has recently been shown that it is impossible to solve majority *exactly* if each node only has three memory states [MNRS14].

Given this apparent trade-off between correctness and performance, it is natural to ask: is there a population protocol for majority which is both *fast* (converging in logarithmic or poly-logarithmic time for any ϵ), and *exact* (with 0 probability of error)?

In this paper, we answer this question in the affirmative. We present a new protocol, called AVC (for *Average-and-Conquer*), which solves *exact* majority in expected parallel time $O(\log n/(m\epsilon) + \log n \log m)$, for any parameter $m \leq n$ where n is the number of nodes, ϵ is the margin, and $s = \Theta(m + \log n \log m)$ is the number of *states* employed by the protocol. Specifically, for number of states $s = \Omega(1/\epsilon + \log n \log 1/\epsilon)$, the expected convergence time of the protocol is always poly-logarithmic in n. (Similar bounds hold with high probability in n.) In other words, AVC uses $O(\log(1/\epsilon) + \log\log n)$ local bits to solve exact majority in poly-logarithmic time. Our protocol exhibits a new trade-off between the algorithm's error probability and convergence time, versus the number of states s employed by each node, and the discrepancy ϵ.

The AVC protocol roughly works as follows. Let $m \geq 1$ be an odd integer parameter. Each state in the protocol will correspond to an integer value between $-m$ and m, which will be either odd, or 0. The *sign* of the value corresponds to the node's current opinion (by convention, + corresponds to initial majority A), and its absolute value corresponds will be the *weight* of this opinion. Therefore, in the following, we will use the terms *state* and *value* interchangeably. Initially, nodes with initial state A start with initial value $+m$, and nodes with initial value B start with initial value $-m$.

At a high level, the protocol combines two components: an *averaging* reaction, by which nodes with values > 1 average their initial values whenever interacting, and a *neutralization* reaction, by which nodes with absolute value 1 and opposite signs become neutralized and stop influencing the decision value. More precisely, whenever a node with weight > 1 interacts with a node with weight > 0, their updated states will correspond the *average* of their initial values, rounded to odd integers. (For example, input states 5 and -1 will yield output states 1 and 3.) Nodes with the same weight but different signs do not average directly to 0 states, but to intermediate states corresponding to values $+1$ and -1, respectively. States corresponding to value 0 (with positive or negative sign) are created by a *neutralization* reaction, when nodes in states $+1$ and -1 interact. Zero states are *weak*, in that they will not influence the *weight* of their interaction partner, and automatically adopt its *opinion*. (For example, input states 3 and -0 will yield output states 3 and 0.)

While the protocol is relatively simple, its analysis is non-trivial. To illustrate, take $m = 1$, and notice that in this special case the protocol would be identical to the four-state algorithm of [DV12, MNRS14]: nodes start in states $+1$ or -1; whenever two such nodes meet, they are downgraded to weak states $+0$ and -0, respectively. Weak 0 nodes shift their opinion whenever meeting a node with weight > 0, and preserve their weight. The process dynamics on a clique are relatively simple: given an initial state with discrepancy ϵ in favor of state A $(+1)$, the minority state B (-1) will eventually be depleted (downgraded to state -0) in expected $O(\log n/\epsilon)$ parallel time [DV12]. At this point, there still exist ϵn nodes in state $+1$, and it will take additional $O(\log n/\epsilon)$ expected parallel time to shift all the -0 states to the correct opinion $+0$.

Our algorithm leverages additional states to speed up *both* components of this dynamics, and overcome the linear dependence on $1/\epsilon$. This works for two main reasons: first, initial states, corresponding to large initial weight $\pm m$, will always be depleted fast, since reactions of the type $+m$ meets $-m$ are frequent at first, while, as the algorithm progresses, high values will start interacting often with small values of the opposite sign, again decreasing their weight.

Further, our algorithm is designed to keep the *total sum of values in the system invariant*. This is initially ϵmn (assuming that A is the initial majority by a margin of ϵn nodes). In particular, as the absolute values of states in the majority decrease, their *number* must increase to maintain the invariant. Our analysis formalizes this augmentation in the number of majority nodes to prove the expected convergence time bounds. The protocol's correctness follows by a variation on the total sum invariant.

It is interesting to contrast AVC with a trivial algorithm in which each node maintains an array of n initial values, one for each other possible interaction partner, and nodes merge their arrays on each interaction, deciding when they have all input values. Such an algorithm would converge in $O(\log n)$ parallel time, by standard gossip analysis. However, there are two issues: first, nodes would need *distinct identifiers* for this to work, which is not standard in population protocols. Moreover, even with identifiers, the trivial protocol would require an exponential number of states per process, to cover all possible input combinations.

We complement our upper bound by two lower bounds showing that the trade-off between the number of states and convergence time is tight within a log factor at its two extremes. The first lower bound uses a complex case analysis to prove that any dynamics based on agents with four states must have parallel convergence time $\Omega(1/\epsilon)$. At the other extreme, we leverage an information-propagation argument to prove that any population protocol solving exact majority converges in $\Omega(\log n)$ parallel time, irrespective of the number of states it employs.

Finally, we provide an implementation of the AVC protocol, and compare its empirical performance both with the four-state protocol of [DV12, MNRS14], and with the three-state algorithm of [AAE08, PVV09]. The simulation results confirm the tightness of our theoretical analysis, and show that AVC provides significant speedup over the four-state protocol by increasing the number of memory states per node, with the convergence time being competitive to that of the three-state protocol for large enough number of memory states per node.

Related Work: The framework of population protocols was first formally introduced by [AAD+06]. A two-state population protocol was studied by [HP99] in a discrete-time model such that in each round every node picks a neighbor node with probability of selection allowed to be specific to this pair of nodes, and adopts the observed opinion of the

selected neighbor. For the complete-graph pairwise interactions and continuous-time model where each node picks a neighbor at instances of a Poisson process, this corresponds to *the voter model* studied in the context of interacting particle systems, see e.g. Chapter 5 in [Lig85]. In particular, a subset of the results in [HP99] show that, for the case of complete graph pairwise interactions, the error probability of the two-state protocol is equal to the portion of nodes in the initial minority state $(1-\epsilon)/2$, and that the expected parallel convergence time is $\Omega(n)$.

Reference [AAE08] studied a three-state population protocol for computing approximate majority in a discrete-time computation model. They established the parallel time convergence bound $O(\log n)$ with high probability given that the initial margin is $\epsilon n = \omega(\sqrt{n \log n})$. The same three-state protocol was empirically studied as a model of epigenetic cell memory by nucleosome modification [DMST07]. Independently, the same three-state population protocol for approximate majority was studied by [PVV09] in a continuous-time computation model. They established a tight bound on the probability of error for convergence to the correct terminal state $\exp(-D((1+\epsilon)/2||1/2)n)$ where $D(p||q)$ is the Kullback-Leibler divergence between two Bernoulli distributions with parameters p and q. This implies the asymptotic bound on the error probability $\exp(-c\epsilon^2 n)$ for constant $c > 0$ and small ϵ. Moreover, they established a parallel convergence time bound $O(\log(1/\epsilon) + \log n)$ for the limit system dynamics described by a system of ordinary differential equations as the number of nodes n grows large. [CCN12] showed that the dynamics of this three-state population protocol is under certain initial conditions equivalent of the cell cycle switch and other dynamics.

[DV12] first proposed a four-state protocol for exact majority computation and studied it for the case of arbitrary rates of pairwise interactions, which are such that the underlying graph G of pairwise interactions is connected. They showed an expected parallel time convergence bound of $(\log n + 1)/\delta(G, \epsilon)$ where $\delta(G, \epsilon) > 0$ is the smallest eigenvalue gap for a family of pairwise interaction rate matrices. For the case of a complete graph, this yields the expected parallel time bound of $O(\log n/\epsilon)$. A similar four-state protocol was also studied by [MNRS14] in a discrete-time computation model, who established that four memory states are necessary for exact majority computation.

A random process similar to the averaging stage of our algorithm was studied by Sauerwald and Sun [SS12], for arbitrary graphs. Their goal is to bound the necessary time for every node's value to converge within a constant, whereas we are interested in the time it takes for all minority values to reach absolute value ≤ 1, after which we bound the remaining time until all nodes with a minority opinion are eliminated. Another similar process, where nodes store values and update according to a dynamics that resembles averaging, is studied in [KDG03].

Related Work: The framework of population protocols was first formally introduced by [AAD+06]. A two-state population protocol was studied by [HP99] in a discrete-time model such that in each round every node picks a neighbor node with probability of selection allowed to be specific to this pair of nodes, and adopts the observed opinion of the selected neighbor. For the complete-graph pairwise interactions and continuous-time model where each node picks a neighbor at instances of a Poisson process, this corresponds to *the voter model* studied in the context of interacting particle systems, see e.g. Chapter 5 in [Lig85]. In particular, for the complete graph pairwise interactions, [HP99] showed that the error probability of the two-state protocol is equal to the portion of nodes in the initial minority state $(1-\epsilon)/2$, and that the expected parallel convergence time is $\Omega(n)$.

[AAER07] studied a three-state population protocol for computing approximate majority in a discrete-time computation model. They established the parallel time convergence bound $O(\log n)$ with high probability given that the initial margin is $\epsilon n = \omega(\sqrt{n \log n})$. The same three-state protocol was empirically studied as a model of epigenetic cell memory by nucleosome modification [DMST07]. Independently, the same three-state population protocol for approximate majority was studied by [PVV09] in a continuous-time computation model. They established a tight bound on the probability of error for convergence to the correct terminal state $\exp(-D((1+\epsilon)/2||1/2)n)$ where $D(p||q)$ is the Kullback-Leibler divergence between two Bernoulli distributions with parameters p and q. This implies the asymptotic bound on the error probability $\exp(-c\epsilon^2 n)$ for constant $c > 0$ and small ϵ. Moreover, they established a parallel convergence time bound $O(\log(1/\epsilon) + \log n)$ for the limit system dynamics described by a system of ordinary differential equations as the number of nodes n grows large. [CCN12] showed that the dynamics of this three-state population protocol is under certain initial conditions equivalent of the cell cycle switch and other dynamics. [FHK14] have provided an algorithm in a push gossip model with one-bit messages that can be flipped with small probability. Similar to the three-state algorithm, for a sufficiently large margin, this algorithm also converges to the correct majority w.h.p. within time logarithmic in n. The main difference besided model is resiliency against message corruptions.

[DV12] first proposed a four-state protocol for exact majority computation and studied it for the case of arbitrary rates of pairwise interactions, which are such that the underlying graph G of pairwise interactions is connected. They showed an expected parallel time convergence bound of $(\log n + 1)/\delta(G, \epsilon)$ where $\delta(G, \epsilon) > 0$ is the smallest eigenvalue gap for a family of pairwise interaction rate matrices. For the case of a clique, this yields the expected parallel time convergence bound $O(\log n/\epsilon)$. A similar four-state protocol was also studied by [MNRS14] in a discrete-time computation model, who established that four memory states are necessary for exact majority computation.

In population protocols, it is known that increasing the number of states can lead to faster algorithms. For example, with constant states, leader election has been shown to require at least linear time [DS15]. However there is an algorithm using polylogarithmic number of states that converges in polylogarithmic time [AG15].

A random process similar to the averaging stage of our algorithm was studied by Sauerwald and Sun [SS12], for arbitrary graphs. Their goal is to bound the necessary time for every node's value to converge within a constant, whereas we are interested in the time it takes for all minority values to reach absolute value ≤ 1, after which we bound the remaining time until all nodes with a minority opinion are eliminated. Another process where nodes store values and update according to a dynamics that resembles averaging is studied in [KDG03], and its analysis also critically relies on the sum being invariant. However, the values can be arbitrary requiring an infinite state space in our model.

2. PRELIMINARIES

Population Protocols: We assume a population consisting of n agents, or cells, each executing as a deterministic

state machine with states from a finite set Q, with a finite set of input symbols $X \subseteq Q$, a finite set of output symbols Y, a transition function $\delta : Q \times Q \to Q \times Q$, and an output function $\gamma : Q \to Y$. Initially, each agent starts with an input from the set X, and proceeds to update its state following interactions with other agents, according to the transition function δ. For simplicity of exposition, we assume that agents have identifiers from the set $V = \{1, 2, \ldots, n\}$, although these identifiers are not known to agents, and not used by the protocol.

The agents' interactions proceed according to a directed *interaction graph* G without self-loops, whose edges indicate possible agent interactions. Usually, the graph G is considered to be the complete graph on n vertices, a convention we also adopt in this work.

The execution proceeds in *steps*, where in each step a new edge (u, w) is chosen uniformly at random from the set of edges of G. Each of the two chosen agents updates its state according to the function δ.

We say that a population protocol computes a function $g : X^V \to Y$ within ℓ steps with probability $1 - \phi$ if for all $x \in g^{-1}(Y)$, the configuration $c : V \to Q$ reached by the protocol after ℓ steps satisfies the following two properties with probability $1 - \phi$:

1. For all $v \in V$, $g(x) = \gamma(c(v))$. Specifically, all agents have the correct output in c.

2. For every configuration c' reachable from c, and for all $v \in V$, $g(x) = \gamma(c'(v))$.

Parallel Time: The above setup considers sequential interactions; however, in general, interactions between pairs of distinct agents are independent, and are usually considered as occurring in parallel. In particular, it is customary to define one unit of *parallel time* as n consecutive steps of the protocol.

The Majority Problem: In the *majority problem*, agents start with arbitrary initial states in the input set $X = \{A, B\}$. Let a be the number of agents starting in state A, and b be the number of agents starting in state B, and let $\epsilon = |a - b|/n$ denote the relative advantage of an initial state. The output set is $Y = \{0, 1\}$.

A population protocol solves the majority problem within ℓ steps with probability $1 - \phi$, if, for any configuration $c : V \to Q$ reachable by the protocol after $\geq \ell$ steps, it holds with probability $1 - \phi$ that (1) If $a > b$ for the given input, then for any agent i, $\gamma(c(i)) = 1$, and, conversely, (2) If $b > a$ for the given input, then for any agent i, $\gamma(c(i)) = 0$.

3. THE AVC ALGORITHM

In this section, we describe the *Average and Conquer* (AVC) algorithm, which solves the majority problem in population protocols *exactly*, using s states. Throughout the rest of the exposition, we fix an odd integer parameter $m \geq 1$, which will determine the number of states of the protocol and its running time.

Overview: The intuition behind the protocol is as follows. Each node state is associated with a *sign* (+ or −) and a *weight* (an non-negative odd integer between 0 and m), whose product determines the *value* corresponding to each state. Throughout the exposition, we will identify the node *state* with corresponding *integer value*, making the distinction where appropriate. (The formal mapping is given for each state in lines 1–3 of the pseudocode in Figure 2.)

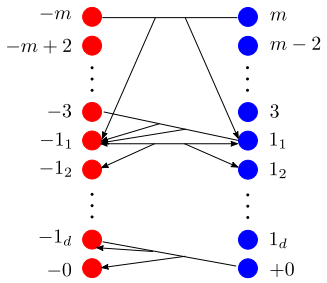

Figure 1: Structure of the states, and some reaction examples. Initially, nodes are in state $-m$ or $+m$. Each line corresponds to a reaction between two states; the two arrows point to the resulting states. For example, states m and $-m$ react to produce states -1_1 and 1_1.

Nodes start in one of two distinguished states. By convention, nodes starting in state A have initial value (state) $+m$, while nodes starting in state B have initial value (state) $-m$. The protocol is based on the following idea: whenever two nodes holding non-zero values interact, they *average* their current values. (Since states only represent odd values, the algorithm rounds the average if needed.) States $+0$ and -0 are *weak*: a node in this state does not influence other values in interactions, and will change its state's sign to match the that of the interaction partner. Also, for technical reasons, value 1 corresponds to d distinct *intermediate states*, which we denote by $1_1, 1_2, \ldots, 1_d$ (and, symmetrically, states $-1_1, -1_2, \ldots, -1_d$ for value -1). A node in such a state must interact several times with other nodes of opposite sign and intermediate value in order to eventually downgrade to 0.

Nodes continue to interact according to these simple rules. We prove that all nodes will converge to values of the same sign as the initial majority, and that convergence is fast with high probability. This solves the majority problem since we can define γ as mapping each state to the output based on its sign. We present the algorithm in detail below.

State Parameters: Each state in the protocol corresponds to an integer value, composed by a *sign* (+ or −) and a and a *weight*, which is either 0, or an odd positive integer between 1 and m. Intuitively, a node's sign gives its tentative output, whereas its *weight* gives the "confidence" in that specific output. *Strong* states (with weight ≥ 3) can drive weaker states, of lower absolute value, to change sign. *Weak* states (absolute value 0) will not influence other states' sign. States with weight 1 (denoted by $\pm 1_j$, where j is a positive integer denoting the state's level) are *intermediate*, and are interpreted as having weight 1, and sign ± 1. (Intuitively, the reason why we grade values of 1 is to prevent nodes from becoming weak too quickly.) The total number of states in the protocol is $s = m + 2d + 1$.

Nodes start in one of two initial states: state A, which we identify as having *sign* $= +1$ and *weight* $= m$, and state B, which corresponds to *sign* $= -1$ and *weight* $= m$. The algorithm, specified in Figure 2, consists of a set of simple deterministic update rules for the node state, depending on the *sign* and *weight* of the interacting node, and will ensure that, after some time, all nodes converge to the correct (majority) opinion. In the pseudocode, the node states before an interaction are denoted by x and y, while their new states are given by x' and y'. The algorithm is entirely symmetric, requiring no order on the agents in the interaction.

Parameters:
m, an odd integer > 0, and d, integer > 0
State Space:
$StrongStates = \{-m, -m+2, \ldots, -3, 3, 5, \ldots, m-2, m\}$,
$IntermediateStates = \{-1_1, -1_2, \ldots, -1_d, 1_d, 1_{d-1}, \ldots, 1_1\}$,
$WeakStates = \{-0, +0\}$
Input: States of two nodes, x and y
Output: Updated states x' and y'
Auxiliary Procedures:
`/* Functions mapping states to integers */`

1 $weight(x) = \begin{cases} |x| & \text{if } x \in StrongStates \text{ or } x \in WeakStates; \\ 1 & \text{if } x \in IntermediateStates. \end{cases}$

2 $sgn(x) = \begin{cases} 1 & \text{if } x \in \{+0, 1_d, \ldots, 1_1, 3, 5, \ldots, m\}; \\ -1 & \text{otherwise.} \end{cases}$

3 $value(x) = sgn(x) \cdot weight(x)$

`/* Functions for rounding state interactions */`

4 $\phi(x) = -1_1$ if $x = -1; 1_1$ if $x = 1; x$, otherwise

5 $R_\downarrow(k) = \phi(k$ if k odd integer, $k-1$ if k even)

6 $R_\uparrow(k) = \phi(k$ if k odd integer, $k+1$ if k even)

7 $Shift\text{-}to\text{-}Zero(x) = \begin{cases} -1_{j+1} & \text{if } x = -1_j \text{ for some index } j < d \\ 1_{j+1} & \text{if } x = 1_j \text{ for some index } j < d \\ x & \text{otherwise.} \end{cases}$

8 $Sign\text{-}to\text{-}Zero(x) = \begin{cases} +0 & \text{if } sgn(x) > 0 \\ -0 & \text{oherwise.} \end{cases}$

9 **procedure** update$\langle x, y \rangle$

10 **if** $(weight(x) > 0$ *and* $weight(y) > 1)$ ***or*** $(weight(y) > 0$ *and* $weight(x) > 1)$ **then**

11 $x' \leftarrow R_\downarrow\left(\frac{value(x)+value(y)}{2}\right)$ and $y' \leftarrow R_\uparrow\left(\frac{value(x)+value(y)}{2}\right)$

12 **else if** $weight(x) \cdot weight(y) = 0$ *and* $value(x) + value(y) > 0$ **then**

13 **if** $weight(x) \neq 0$ **then** $x' \leftarrow Shift\text{-}to\text{-}Zero(x)$ and $y' \leftarrow Sign\text{-}to\text{-}Zero(x)$

14 **else** $y' \leftarrow Shift\text{-}to\text{-}Zero(y)$ and $x' \leftarrow Sign\text{-}to\text{-}Zero(y)$

15 **else if** $(x \in \{-1_d, +1_d\}$ *and* $weight(y) = 1$ *and* $sgn(x) \neq sgn(y))$ ***or***

16 $(y \in \{-1_d, +1_d\}$ *and* $weight(x) = 1$ *and* $sgn(y) \neq sgn(x))$ **then**

17 $x' \leftarrow -0$ and $y' \leftarrow +0$

18 **else**

19 $x' \leftarrow Shift\text{-}to\text{-}Zero(x)$ and $y' \leftarrow Shift\text{-}to\text{-}Zero(y)$

Figure 2: The state update rules for the majority computation.

Interactions: The core of the update rule is the *averaging* of nodes' values after each interaction, where for each state x, $value(x) = sgn(x) \cdot weight(x)$, where states $\pm 1_j$ have weight 1. Intuitively, the values of the two participants after an interaction will equal the average of their previous values, where we round values up and down to the closest odd numbers. Note that this rule always preserves the total sum of values in the system. Values 0 and 1 are special, and therefore we consider them separately in the update rules. We have three cases for the update rules: *strong meets non-zero* (line 11), *zero meets non-zero* (lines 12—14), and *intermediate meets intermediate* (lines 15—19), with a special case if one of the nodes is 1_d (lines 15—17). (The *zero meets zero* reaction does not change the participants' states.) Figure 1 gives examples of reactions.

If a node in a strong state (with weight > 1) meets a node in strong or intermediate state (with weight > 0), their new values will correspond to the average of their initial value (line 11). Specifically, if the average of the two node values is even, the updated states will encode values $average - 1$ and $average + 1$. If the average is odd, then both states encode $average$. When a node reaches value 1 or -1, its state becomes either 1_1 or -1_1. This rounding is performed through the functions $R_\uparrow$ and $R_\downarrow$.

Second, if a node with 0 weight meets a node with non-zero weight, then it will adopt the sign of the interaction partner via the *Sign-to-Zero* function, whose weight remains unchanged (lines 12—14). Nodes $\pm 1_j$ with $j < d$ are a slight exception, as they become downgraded to $\pm 1_{j+1}$. Thus, the ordering of states corresponding to value 1 is such that 1_d is the weakest / closest to value 0.

Third, if the participants have different signs, weight 1, and one of them is in state 1_d or -1_d, then both are downgraded to value 0, albeit with different signs (lines 15—17).

In the last case unless both values are zero (and nothing changes) both participants have weight 1 but none of them is in state 1_d or -1_d. Then, both keep weight 1, but are downgraded to the next level of state 1 via the *Shift-to-Zero* function. For example, a node in a state 1_j with $j < d$ now moves to state 1_{j+1}.

4. ANALYSIS OF THE AVC ALGORITHM

In this section, we provide a complete analysis of the AVC algorithm. We first prove correctness, showing that the algorithm always converges to a correct answer, and also characterize the distribution of its convergence time.

Notation: Throughout this proof, we denote the set of nodes executing the protocol by V. We measure execution time in discrete steps (rounds), where each step corresponds to an interaction. The *configuration* at a given time t is a function $c : V \to Q$, where $c(v)$ is the state of the node v at time t. (We omit the explicit time t when clear from the context.) We use the quantities $sgn(c(v))$, $weight(c(v))$ and $value(c(v))$ associated with the state $c(v)$ of the node

v in configuration c, as defined on Figure 2 lines 1-3. We will refer to them as the sign, weight and value of the node, respectively.

Analysis Overview: The analysis first considers correctness, and then speed of convergence. For correctness, we prove that nodes never converge to the sign of the initial minority (safety), and that they eventually converge to the sign of the initial majority (termination). The first statement follows since the algorithm is designed to keep the total sum of values in the system *invariant*: for instance, if the initial sum is positive, then positive values must always exist in the system. Further, we prove that, eventually, all minority values are eliminated, implying termination.

The convergence bound is more involved. We first focus on the *extremal* weights in the system, showing that, roughly, either their weight is halved every $\log n$ parallel time steps, or the number of nodes in strong states is depleted for one of the signs. Further, this halving process, which takes expected $O(\log n \log m)$ parallel time, still preserves the initial system sum: therefore, instead of having a few deciding nodes with high values, after the halving completes, we will have many deciding nodes of small value. We then show that these nodes sway the remaining minority states to the correct decision within expected $O(\log n/(\epsilon m))$ parallel time, completing the bound. Our main result is the following theorem.

THEOREM 1. *The* AVC *algorithm solves majority problem with probability* 1. *There is a setting of parameters, such that it uses* $s = m + O(\log n \log m)$ *states for parameter* $m \leq n$ *and has parallel convergence time* $O(\log n/(m\epsilon) + \log n \log m)$ *in expectation and* $O(\log^2 n/(m\epsilon) + \log^2 n)$ *with high probability.*

An interesting setting $m = 1/\epsilon$ gives the following.

COROLLARY 1. *There is a setting of parameters, such that the* AVC *algorithm uses* $s = 1/\epsilon + O(\log n \log 1/\epsilon)$ *states, and has parallel convergence time* $O(\log n \log 1/\epsilon)$ *in expectation and* $O(\log^2 n)$ *with high probability.*

Correctness: We observe that, given the interaction rules of the algorithm, the sum of the encoded values stays constant as the algorithm progresses. The proof follows by the structure of the algorithm. In particular, the averaging reaction is designed to maintain the sum. Same holds for all other types of interactions.

INVARIANT 2. *The sum* $\sum_{v \in V} value(c(v))$ *never changes, for all reachable configurations* c *of the protocol.*

This invariant implies that the algorithm may never converge to a wrong decision value, as at least a node with the sign of the initial majority must exist in the system. We therefore only need to show that the algorithm converges to a state where all nodes have the same sign, which we do via a rough convergence bound, assuming an arbitrary starting configuration.

LEMMA 1. *Let* c *be an arbitrary starting configuration for the the* AVC *algorithm and define* $S := \sum_{v \in V} value(c(v)) \neq 0$. *With probability* 1, *the algorithm will reach a configuration* f *such that* $sgn(f(v)) = \mathrm{sgn}(S)$ *for all nodes* $v \in V$. *Moreover, in all later configurations* e *reachable from* f, *no node can ever have a different sign, i.e.* $\forall v \in V$: $sgn(e(v)) = sgn(f(v))$. *The convergence time to* f *is at most* $n^3(m + d + 1)$ *expected communication rounds, i.e. parallel time* $n^2(m + d + 1)$.

PROOF. Assume without loss of generality that the sum S is positive. At any time, given the current configuration is r, let us define $P := \sum_{v \in V \mid sgn(r(v))>0} weight(r(v))$ and $N := \sum_{v \in V \mid sgn(r(v))<0} weight(r(v))$. Then, by Invariant 2, $P - N = S > 0$ holds throughout the execution. Also, $N \leq nm$, as each of the at most n contributing weights to the sum is at most m.

We estimate the expected convergence time by splitting the execution into three phases. The first phase starts at the beginning of the execution, and lasts until either i) N becomes 0, *or* ii) until all nodes are in states that encode values in $\{-1, 0, 1\}$. Therefore, throughout this phase, there are always nodes that encode non-zero values with different signs and at least one node in a strong state (with weight > 1).

Consequently, in any communication round in the first phase, with at least $1/n^2$ probability this node of large weight interacts with a node with opposite sign and non-zero value, and therefore N (and P) both decrease by at least 1. At the same time, because of the update rules of the algorithm, N can never increase. In expected n^2 rounds, N will decrease by at least 1. By $N < nm$ and the linearity of expectation, expected number of rounds until N becomes 0, or until no nodes remain in a strong state (maximum weight of nodes ≤ 1), is at most $n^3 m$. This upper bounds the expected number of rounds in the first phase.

The second phase starts immediately after the first, and ends when N becomes 0. Note that if the first phase ended because of $N = 0$, the second phase is trivially empty. Now consider the case when all nodes are in states with values -1, 0 and 1 at the beginning of the second phase. There are clearly N nodes with value -1, P nodes with value 1 ($N < P < n$) and the remaining nodes must be in weak states (weight 0).

Under these circumstances, because of the update rules, no node will ever be in a strong state (weight > 1) again in any future configuration. Also, the number of nodes in intermediate states (weight 1) can only decrease. In each round, with probability at least $1/n^2$, two nodes with weights 1_i and 1_j and opposite signs meet. Either $\max(i, j) = d$ and both new values become zero, meaning N (and P) decreases by 1, or the weights become 1_{i+1} and 1_{j+1} without affecting the signs. But this can only happen nd times, since there can be at most n nodes encoding value -1, and each becomes 0 after at most d meetings with a nodes of opposite sign. Therefore, the expected number of rounds in the second phase is at most $n^3 d$.

After the second phase, $P = S$ holds and all nodes with negative sign ought to have weight 0. From this time, all rounds belong to the final, third phase, throughout which N clearly remains equal to zero. The third phase lasts until the system converges, that is, until all nodes with weight 0 get a positive sign. Since $S \geq 1$ there is at least one node with a positive sign and non-zero weight, and there are at most $n - 1$ *conflicting* nodes with a negative sign, all with value zero as $N = 0$. Thus, independently in each round, with probability at least $1/n^2$, a conflicting node meets a node with strictly positive value and the number of conflicting nodes decreases by one. The number of conflicting nodes never increases because of the update rules and when it becomes zero, the system has converged to the desired configuration f. Therefore, the expected number of rounds in the third phase is at most n^3.

Combining the results, the total expected number of rounds is at most $n^3(m+d+1)$. The expectation being finite implies that the algorithm converges with probability 1. Finally,

when two nodes with positive sign meet, they both remain positive, so any configuration e reachable from f also contains every node with the correct sign. □

Convergence Time Bounds: We now focus on bounding the time until all nodes converge to the correct sign. We begin by establishing some notation. We call a round a *negative-round* if, in the configuration c at the beginning of the round, at least a *third* of the nodes encode a strictly negative value, i.e. $|\{v \in V : value(c(v)) < 0\}| \geq |V|/3$. Analogously, we call a round a *positive-round* if a third of the nodes encode a strictly positive value. A round can be simultaneously negative *and* positive. Our first claim bounds the speed at which the maximum weight present in the system decreases. The proof follows by bounding the probability that a node with the maximum weight meets a node with non-zero value and opposite sign in a round. This decreases the weight of the node, and we use Chernoff and Union Bounds to bound the probability that the node avoids such a meeting for logarithmic parallel time.

Claim 1. *Let $w > 1$ be the maximum weight among the nodes with a negative (resp., positive) sign. For constant $\beta \geq 54$, after $\beta n \log n$ positive-rounds (resp., negative-rounds) the maximum weight among the nodes with a negative (resp., positive) sign will be at most $\lfloor w/2 \rfloor$ with probability at least $1 - \log n / n^{\beta/54}$.*

Proof. We will prove the claim for nodes with strictly negative values. (The converse claim follows analogously.) Fix a round r, and recall that w is the maximum weight of a node with a negative value at the beginning of the round. Let U be the set of nodes encoding negative values between $-\lceil w/2 \rceil$ and $-w$, with $w > 1$, at the beginning of the round, and let $u = |U|$. We call these nodes *target nodes*.

By the structure of the AVC algorithm, the number of target nodes never increases, and decreases by one in every *eliminating* round where a target node meets a node with a strictly positive value. Consider a set of αn consecutive positive-rounds after r, for $\alpha \geq 18$. In each round, if there are still at least $\lceil u/2 \rceil$ target nodes, then the probability of this round being eliminating is at least $\lceil u/2 \rceil / 3n$ (recall that by definition of a positive round at least third of the nodes have strictly positive value). Let us describe the process by considering a random variable $X \sim \text{Bin}(\alpha n, \frac{\lceil u/2 \rceil}{3n})$, where each success event corresponds to an eliminating round. By a Chernoff Bound, the probability of having αn iterations ($\alpha \geq 18$) with at most $\lceil u/2 \rceil$ eliminations is at most:

$$\begin{aligned}\mathbf{Pr}\left[X \leq \lceil u/2 \rceil\right] = \quad & \mathbf{Pr}\left[X \leq \frac{\alpha \lceil u/2 \rceil}{3}\left(1 - \frac{\alpha - 3}{\alpha}\right)\right] \\ \leq \quad & \exp\left(-\frac{\alpha \lceil u/2 \rceil (\alpha-3)^2}{6\alpha^2}\right) \leq 2^{-\frac{u\alpha}{18}}.\end{aligned}$$

For $u \geq \log n$, the probability of this event is at most $\frac{1}{n^{\alpha/18}}$ for αn positive-rounds. Applying the same rationale iteratively as long as $u \geq \log n$, we obtain by using a Union Bound that the number of target nodes will become less than $\log n$ within $\alpha n(\log n - \log\log n)$ positive-rounds, with probability at least $1 - (\log n - \log\log n)/n^{\alpha/18}$.

Finally, we wish to upper bound the remaining number of positive-rounds until no target node remains. When $u < \log n$, we get from the same argument as above that the number of target nodes is reduced to $\lfloor u/2 \rfloor$ within $\frac{\alpha n \log n}{u}$ consecutive positive-rounds with probability $1/n^{\alpha/18}$. So we consider increasing numbers of consecutive positive-rounds, and obtain that no target nodes will be left after at most $\alpha n + 2\alpha n + \ldots + \alpha n \log n \leq 2\alpha n \log n$ positive-rounds, with probability at least $1 - \log\log n / n^{\alpha/18}$, where we have taken the union bound over $\log\log n$ events. The original claim follows by setting $\beta = 3\alpha$, and taking the Union Bound over the above two events ($u \geq \log n$ and $u < \log n$). □

In the following, we consider the AVC algorithm with parameters $\log n \log\log n \leq m \leq n$ and $d = 1000 \log m \log n$, for the total of $s = m + O(\log m \log n)$ states. We first prove that in this setting, no node will reach value 0 within the first $432n \log m \log n$ rounds, w.h.p. The proof follows from Chernoff Bound by showing that the probability that a node gets selected at least d times (necessary for it to downgrade to 0) during this interval is extremely low.

Claim 2. *The probability that any node gets weight 0 during the first $432n \log m \log n$ rounds is $\leq 1/n^4$.*

Proof. In each round, the probability that a particular node is selected is $2/n$. Let us describe the number of times a given node is selected in $432n \log m \log n$ rounds by considering a random variable $Z \sim \text{Bin}(432n \log m \log n, 2/n)$. Given that $d = 1000 \log m \log n$, by the Chernoff Bound, the probability that the node gets selected more than d times in these rounds is at most:

$$\begin{aligned}\mathbf{Pr}\left[Z \geq d\right] &= \mathbf{Pr}\left[Z \geq (864 \log m \log n) \cdot \left(1 + \frac{136}{864}\right)\right] \\ &\leq \exp\left(-\frac{864 \cdot 136^2}{3 \cdot 864^2} \log m \log n\right) \leq \frac{1}{(mn)^5}.\end{aligned}$$

A node cannot get weight 0 during the first $432n \log m \log n$ rounds unless it is selected for interactions at least d times in total. Thus, the probability that a given node gets value zero is at most $1/(mn)^5$. Applying the Union Bound over all nodes, we conclude that no node gets *weight* $= 0$ during the first $432n \log m \log n$ rounds, with probability at least $1 - 1/(m^5 n^4) \geq 1 - 1/n^4$. □

Claim 1 gave a condition for halving the maximum positive and the minimum negative values in the system with high probability. The initial maximum weight in the system is m, which can only be halved $\log m$ times for each sign. This motivates the following claim:

Claim 3. *During the first $432n \log m \log n$ rounds, if no node gets value 0, then with probability at least $1 - (2 \log n \log m)/n^4$, one of the following three events occurs at some point during these rounds:*

1. *Nodes only have values in $\{-1, 1\}$;*
2. *There are less than $n/3$ nodes with negative values, all with value -1,*
3. *There are less than $n/3$ nodes with positive values, all with value 1.*

Proof. Since no node gets value 0 during the first $432n \log m \log n$ rounds, each round during this interval is either a negative-round, a positive-round, or both. Unless the maximum positive value in the system is 1, by Claim 1, a stretch of $216n \log n$ negative-rounds halves the maximum positive value available, with probability at least $1 - \log n / n^4$. The same occurs for stretches of $216n \log n$ positive-rounds and the minimum negative value.

Assume that none of the three events hold at any time during the first $432n \log m \log n$ rounds. In that case, each

round can be classified as either a negative round where the maximum positive value is strictly larger than 1, or a positive round where the minimum negative value is strictly smaller than -1. Thus, each round contributes to at least one of the stretches of $216n \log n$ rounds that halve the maximum positive of minimum negative value, w.h.p. However, this may happen at most $2 \log m$ times. By applying Claim 1 $2 \log m$ times (using $\beta = 216$) and the Union Bound we get that after the first $432n \log m \log n$ rounds, with probability at least $1 - 2 \log n \log m / n^4$ only values -1 and 1 will remain.

However, this is the same as the first event above. Thus, the probability that none of these events happen is at most $2 \log n \log m / n^4$. □

Returning to the proof, recall that we assumed without loss of generality that the initial majority of nodes was in A (positive) state, i.e. $a > b$. The following claim provides the last piece of the puzzle.

Claim 4. *Consider a configuration after the first $432n \log m \log n + 4dn$ rounds. With probability at least $1 - (2 \log n \log m + 2)/n^4$, all nodes with a strictly negative value will be in state -1_d while at least $n \cdot \min(1/3, \epsilon m)$ more nodes will encode a strictly positive value than the nodes in state -1_d.*

Proof. By our assumption about the initial majority and Invariant 2, $\sum_{v \in V} value(c(v)) = \epsilon nm$ holds in every reachable configuration c. Let us focus on the high probability events in Claim 2 and Claim 3: no node gets a value 0 within the first $432n \log m \log n$ rounds, and during these rounds the execution reaches a configuration where one of the three events from Claim 3 holds. Consider this point T in the execution.

Since no node has value 0, the third event is impossible, because the total sum would be negative. Second event implies that there are at least $n/3$ more strictly positive values than strictly negative values. In the first event, the total sum is $\geq \epsilon nm$ and values are all -1 and 1, so there must be at least ϵmn more strictly positive than negative values. Hence, at point T during the first $432n \log n \log m$ rounds there are at least $n \cdot \min(1/3, \epsilon m)$ more strictly positive than strictly negative values.

In both cases, -1's are the only strictly negative values of the nodes at point T, and this will be the case for the rest of the execution because of the update rules. Also, for any level $i < d$, every time one of the nodes in state -1_i interacts with another node with value not equal to -1 (has to be value ≥ 0), either both nodes get values ≥ 0, or the node moves to state -1_{i+1} via the *Shift-to-Zero* function.

In each round, the probability that a given node with value -1 interacts with a node with value ≥ 0 is at least $\min(2/3, (1 + \epsilon m)/2)/n \geq 1/(2n)$. Let us describe the number of times the given node is selected in for such an interaction during the at least $4dn$ rounds after T by a random variable $Y \sim \text{Bin}(4dn, 1/2n)$. Then, by Chernoff Bound, the probability that the node is selected less than d times is at most:

$$\begin{aligned} \mathbf{Pr}[Y \leq d] &= \mathbf{Pr}\left[Y \leq 2d\left(1 - \frac{1}{2}\right)\right] \leq \\ \exp\left(-\frac{2d}{2 \cdot 2^2}\right) &\leq 2^{-d/4} < \frac{1}{n^5}, \end{aligned}$$

where we have used that d is set to $1000 \log m \log n$. Union Bound over all the $\leq n$ nodes that had value -1 at point T, we get that after $4dn$ more rounds, with probability $1 - 1/n^4$, all these nodes interacted at least d times with nodes with non-negative values. Hence, they either have value ≥ 0 or are in state -1_d.

After T, the number of nodes with strictly positive values only decreases in an interaction with a node in state -1_i, in which case the number of nodes with strictly negative values also decreases by one. Hence there must still be at least $n \cdot \min(1/3, \epsilon m)$ more nodes with strictly positive that strictly negative values.

A Union Bound over Claim 2, Claim 3 and the above argument completes the proof. □

By Claim 4, with high probability after $432n \log m \log n + 4dn$ rounds we reach a configuration, from which the convergence bares many similarities to the four state protocol of [DV12] and can be analyzed similarly. We describe its convergence below.

Claim 5. *Consider a configuration where all nodes with strictly negative values are in state -1_d, while at least $n\delta$ more nodes encode a strictly positive value than the nodes in state -1_d. The number of rounds until convergence is $O(\frac{n \log n}{\delta})$ in expectation and $O(\frac{n \log^2 n}{\delta})$ w.h.p.*

Proof. In any configuration, let us call *conflicting* any node that is in state -1_d, and *target* node any node that has a strictly positive value. Because of the structure of the algorithm, and that in configuration c the only nodes with negative sign are in states -1_d or -0, in all configurations reachable from c nodes with negative values will also only be in states -1_d or -0. Moreover, the number of conflicting nodes can never increase after an interaction. As in Claim 4, observe that the number of target nodes can only decrease after an interaction where a node with value 1 meets a node in state -1_d, in which case the number of conflicting nodes also decreases by one. There are $n\delta$ more target nodes than conflicting nodes in c, therefore, in every later configuration, there must always be at least $n\delta$ target nodes.

Every time a conflicting node meets a target node, both nodes get value ≥ 0, so the number of conflicting nodes decreases by one. Let us estimate the number of rounds until each conflicting node has interacted with a target node, at which point no more conflicting nodes may exist. Let us say there were x conflicting nodes in configuration c. The expected number of rounds until the first conflicting node meets a target node is at most $\frac{n}{x\delta}$, since the probability of such an interaction happening in each round is at least $\frac{x}{n} \cdot \frac{n\delta}{n}$. The expected number of rounds for the second node is then $\frac{n}{(x-1)\delta}$, and so on. By linearity of expectation, the expected number of rounds until all conflicting nodes are eliminated is $O(\frac{n \log x}{\delta}) \leq O(\frac{n \log n}{\delta})$.

At this point, all nodes that have a negative sign must be in state -0. If we redefine *conflicting* to describe these nodes, it is still true that an interaction of a conflicting node with a target node brings the conflicting node to state $+0$, thus decreasing the number of conflicting nodes. As we discussed at least $n\delta$ target nodes are still permanently present in the system. By the structure of the algorithm no interaction can increase the number of nodes with negative sign, the system will converge when all conflicting nodes are eliminated. But this takes expected $O(\frac{n \log n}{\delta})$ rounds by exactly the same argument as above.

To get the high probability claim, simply observe that when there are x conflicting nodes in the system, a conflicting node will interact with a target node within $\frac{nO(\log n)}{x\delta}$ rounds, with high probability. The same applies for the next conflicting node, etc. Taking Union Bound over these events gives the desired result. □

Thus, after the first $432n \log m \log n + 4dn$ rounds, in the high probability case, plugging $\delta = \min(1/3, \epsilon m)$ in Claim 5,

we get that the remaining number of rounds is at most $O(n \log n + (n \log n)/(m\epsilon))$ in expectation and $O(n \log^2 n + (n \log^2 n)(m\epsilon))$ w.h.p. The high probability statement immediately follows by the Union Bound.

We only have to bound expectation in the remaining low probability event. With probability at most $(2 \log n \log m + 2)/n^4$, the expected remaining number of rounds is at most $O(n^3(m+d))$ by Lemma 1 and since in our parameter setting $m+d = O(n)$, it is negligible compared to the previous case.

5. LOWER BOUNDS

5.1 Convergence Time with Four States

We prove an $\Omega(1/\epsilon)$ lower bound on the convergence time of any four-state protocol for exact majority. The proof extends the approach introduced by [MNRS14] to show impossibility of exact three-state majority protocols, in that it explores all possible state transitions and output mappings. However, there are considerably more possibilities to consider, among which, unlike the three-state case, some yield correct four-state protocols. We combine indistinguishability arguments to demonstrate that certain algorithms could converge to the incorrect output, with potential-based arguments to show that certain algorithms may never converge, and with convergence analysis of the correct algorithms. Due to space limitations, the proof is deferred to the full version of this paper [AGV15].

We slightly alter notation in this section. We call the initial states S_1 (corresponding to A) and S_0 (corresponding to B). The output (formally defined by the mapping γ) should be $i \in \{0, 1\}$ if a majority of the nodes start in state S_i. We define C_i as a set of all *absorbing* configurations c for output i: for all configurations c_i reachable from $c \in C_i$, we must have $\forall v \in V : \gamma(c_i(v)) = i$.

THEOREM 3. *Consider any majority algorithm using four states, providing the following correctness properties for any $|V| = n \geq 1$:*

- *For each possible outcome $i \in \{0, 1\}$, the set C_i of absorbing configurations is non-empty.*
- *The system never converges to the wrong decision: if in the initial configuration the majority of nodes are in S_i for $i \in \{0, 1\}$, it must be impossible to reach any configuration $c \in C_{1-i}$ under* any *schedule of interactions.*
- *It is always possible to converge to the correct solution: if a majority of nodes start in S_i for $i \in \{0, 1\}$, from every reachable configuration there is a sequence (schedule) of interactions leading to a configuration $c \in C_i$.*

Then, if in the initial configuration ϵn more nodes start in state S_i than in S_{1-i} for $i \in \{0, 1\}$, the expected parallel convergence time to reach some configuration $c \in C_i$ is $\Omega(1/\epsilon)$.

5.2 Arbitrary Number of States

We now prove that any algorithm solving exact majority must take expected $\Omega(\log n)$ parallel time until convergence on a worst-case input, irrespective of the number of states it uses. The structure is as follows: we start from an input state where the outcome is decided by a *constant* number of nodes (the initial states of other nodes are balanced). We then bound expected parallel time for the initial states of these nodes to propagate to every node by $\Omega(\log n)$ by a standard information-based argument, e.g. [CDR86, Jay98].

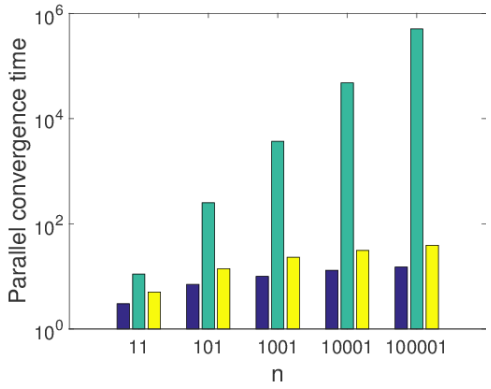

Figure 3: Comparison of performance of different protocols (left bar) 3-state protocol, (middle bar) 4-state protocol, and (right bar) n-state protocol.

We then argue that any node that does not have this decisive information may only have converged with *constant probability*, completing the proof. Due to space limitations, we defer the proof to the full version of our paper.

THEOREM 4. *Any algorithm $\mathcal{A}$ for exact majority takes expected $\Omega(\log n)$ parallel time until convergence under a worst-case input.*

6. DISCUSSION

We have tested the performance of our protocol through implementation, and compared it with the three-state protocol of [AAE08, PVV09], which has non-zero probability of error, and with the exact four-state protocol of [DV12, MNRS14].

For $\epsilon = 1/n$, the n-state version of AVC is orders of magnitude faster than the four-state protocol, and has comparable running time with the three-state protocol, noting that the latter has constant failure probability in this case (see Figure 3). Experiments varying parameters s and ϵ support the claim that the running time of the protocol is linear in these parameters, and that the constant factors are small (see Figure 4).

We have given the first sub-linear time population protocol solving exact majority, which highlights a new trade-off between the number of memory states of the algorithm (alternatively, number of local bits), the convergence time, and the error probability of the protocol. We have also shown that this trade-off is tight up to logarithmic factors when considered at its extremal points in s.

The above experiments were performed setting $d = 1$, i.e., with a single version of the 1 states. This suggests that this variant is also correct and efficient. The multiple levels of 1 and -1 are necessary in the analysis; however, setting $d > 1$ does not significantly affect the running time of the protocol in the experiments. It is therefore an interesting question whether the protocol can be further simplified.

The main open question left by our work is that of matching lower bounds. It would also be interesting to explore whether the average-and-conquer technique would also be useful in the context of other problems in population protocols. Further, we highlight the question of obtaining a fast *approximate* majority protocol which is correct with high probability for all possible values of the discrepancy ϵ.

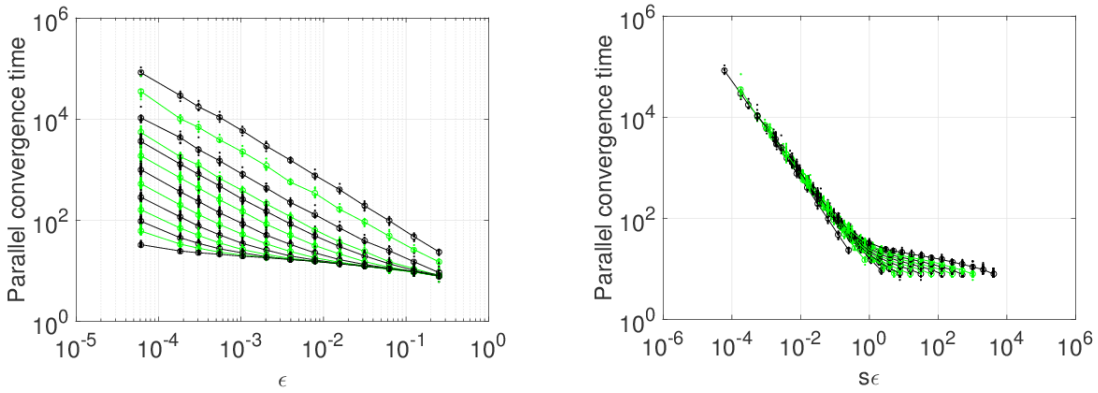

Figure 4: (left) Convergence time vs. parameter ϵ for for the number of states per node $s = 4, 6, 12, 24, 34, 66, 130, 258, 514, 1026, 2050, 4098, 16340$ for the respective curves in the graph from the top to bottom. (right) Convergence time vs. the product $s\epsilon$.

7. REFERENCES

[AAD+06] 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.

[AAE08] Dana Angluin, James Aspnes, and David Eisenstat. A simple population protocol for fast robust approximate majority. *Distributed Computing*, 21(2):87–102, July 2008.

[AAER07] Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, November 2007.

[AG15] Dan Alistarh and Rati Gelashvili. Polylogarithmic-time leader election in population protocols. In *Proceedings of ICALP 2015*, 2015.

[AGV15] Dan Alistarh, Rati Gelashvili, and Milan Vojnovic. Fast and exact majority in population protocols. Technical Report MSR-TR-2015-13, February 2015.

[CCN12] Luca Cardelli and Attila Csikasz-Nagy. The cell cycle switch computes approximate majority. *Nature Scientific Reports*, 2:656, 2012.

[CDR86] Stephen Cook, Cynthia Dwork, and Ruediger Reischuk. Upper and lower time bounds for parallel random access machines without simultaneous writes. *SIAM J. Comput.*, 15(1):87–97, February 1986.

[CDS+13] Yuan-Jyue Chen, Neil Dalchau, Niranjan Srnivas, Andrew Phillips, Luca Cardelli, David Soloveichik, and Georg Seelig. Programmable chemical controllers made from dna. *Nature Nanotechnology*, 8(10):755–762, 2013.

[DMST07] Ian B. Dodd, A. M. Micheelsen, Kim Sneppen, and Geneviéve Thon. Theoretical analysis of epigenetic cell memory by nucleosome modification. *Cell*, 129(4):813–822, 2007.

[DS15] David Doty and David Soloveichik. Stable leader election in population protocols requires linear time. ArXiv preprint. http://arxiv.org/abs/1502.04246, 2015.

[DV12] Moez Draief and Milan Vojnovic. Convergence speed of binary interval consensus. *SIAM Journal on Control and Optimization*, 50(3):1087–1109, 2012.

[FHK14] Ofer Feinerman, Bernhard Haeupler, and Amos Korman. Breathe before speaking: efficient information dissemination despite noisy, limited and anonymous communication. In *Proceedings of PODC 2014*, pages 114–123. ACM, 2014.

[HP99] Yehuda Hassin and David Peleg. Distributed probabilistic polling and applications to proportionate agreement. In *Proceedings of ICALP 1999*, pages 402–411, London, UK, UK, 1999. Springer-Verlag.

[Jay98] Prasad Jayanti. A time complexity lower bound for randomized implementations of some shared objects. In *Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '98, pages 201–210, New York, NY, USA, 1998. ACM.

[KDG03] David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pages 482–491. IEEE, 2003.

[Lig85] Thomas M. Liggett. *Interacting Particle Systems*. Springer, 1985.

[MNRS14] 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 *Proceedings of ICALP 2014*, pages 871–882, 2014.

[PVV09] Etienne Perron, Dinkar Vasudevan, and Milan Vojnovic. Using three states for binary consensus on complete graphs. In *INFOCOM 2009, IEEE*, pages 2527–2535. IEEE, 2009.

[SS12] Thomas Sauerwald and He Sun. Tight bounds for randomized load balancing on arbitrary network topologies. In *Proceedings of FOCS 2012*, pages 341–350, 2012.