Deterministic Population Protocols for Exact Majority and Plurality*

Leszek Gąsieniec[1], David Hamilton[1], Russell Martin[1], Paul G. Spirakis[1] and Grzegorz Stachowiak[2]

1 **Department of Computer Science, University of Liverpool, UK**
{L.A.Gasieniec,D.D.Hamilton,Russell.Martin,P.Spirakis}@liverpool.ac.uk
2 **Instytut Informatyki, Uniwersytet Wrocławski, Poland**
gst@cs.uni.wroc.pl

Abstract

In this paper we study space-efficient deterministic population protocols for several variants of the *majority* problem including *plurality consensus*. We focus on space efficient majority protocols in populations with an arbitrary number of colours C represented by k-bit labels, where $k = \lceil \log C \rceil$. In particular, we present asymptotically space-optimal (with respect to the adopted k-bit representation of colours) protocols for (1) the *absolute majority* problem, i.e., a protocol which decides whether a single colour dominates all other colours considered together, and (2) the *relative majority problem*, also known in the literature as *plurality consensus*, in which colours declare their volume superiority versus other individual colours.

The new population protocols proposed in this paper rely on a *dynamic formulation* of the majority problem in which the colours originally present in the population can be changed by an *external force* during the communication process. The considered dynamic formulation is based on the concepts studied in [4] and [24] about *stabilizing inputs* and *composition of population protocols*. Also, the protocols presented in this paper use a composition of some known protocols for static and dynamic majority.

1998 ACM Subject Classification G.2.1 Combinatorial Algorithms, G.2.2 Network Problems

Keywords and phrases Deterministic population protocols, majority, plurality consenus

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2016.XX

Contact author: Leszek Gąsieniec, **Email:** L.A.Gasieniec@liverpool.ac.uk

* This work is sponsored in part by the University of Liverpool initiative Networks Sciences and Technologies (NeST) and by the Polish National Science Centre grant DEC-2012/06/M/ST6/00459.

1 Introduction

The model of *population protocols* adopted in this paper was proposed first in the seminal paper by Angluin *et al.* [5] and popularised later in [7]. Their model provides a suitable theoretical framework for studying pairwise interactions within a large collection of anonymous (indistinguishable) *entities*, also referred to as *agents*, equipped with little computational power. The entities are modelled as *finite state machines*. When two entities engage in interaction they mutually access their local states and, on the conclusion of the encounter, their states get updated according to the global (shared) *transition function*. In the *asynchronous model*, also adopted in this paper, the order of interactions in consecutive rounds is unpredictable but fair, i.e., none of the pairs of entities can be starved from interaction. In this model, the main emphasis is on feasibility of the solution, subject to the limit on the number of states available to the entities. In the *probabilistic model*, in each round the *random scheduler* picks a pair of entities uniformly at random. In the presence of the *random scheduler*, on the top of space restrictions, one is also interested in the time complexity of a specific distributed task. A population protocol terminates if all participating entities eventually agree on some value represented by dedicated states, independently of the order of interactions. This value can reflect the colour or the size of selected majority [6, 7, 9, 23], the identity of the leader [2, 1, 19], but also completion of more complex tasks such as network formation [25], counting [26], and others.

In this paper, the adopted computation model encompasses a population A of n fault-free entities, each equipped with a $O(k)$-bit memory, where 2^k is the bound on the number C of colours present in the population. This is in contrast to the majority settings considered earlier in [6, 7, 23, 9] where only two original colours were permitted. Here each entity is coloured with exactly one of C available colours and a k-bit label representing this colour is kept in the entity's memory.

As indicated before, the entities communicate in pairs in an *asynchronous manner*. The main task in the *majority problem* is to identify the most frequent colour in the population. Due to presence of more than two colours in the population, we distinguish between the *absolute majority*, i.e., where one colour dominates all others taken together, and the *relative majority*, also known in the literature as *plurality consensus*, where the population is expected to agree on (one of) the most frequent colour(s). We also distinguish between the *static majority* in which the original colours of entities cannot be altered in time - the assumption used in the past work on majority protocols [6, 7, 23, 9], and the *dynamic majority* in which the original colours of entities can be changed in due course by an *external force*, and by doing so may alter the outcome of the majority protocol. This is the main reason why in our model the entities must store their original colour, which could be altered at any time but only by the external force, in addition to $O(k)$ memory bits required during interactions and to report the majority on the conclusion of the computation process.

The model with the external force adopted in this paper was considered earlier in [24] under the name *computing with stabilizing inputs*. Note that the dynamic protocol described in Section 3 is a special variant of self-stabilization, as state alterations done by the external force are permitted only between certain (colour indicating) states. We would also like to emphasise that protocols for absolute majority presented in Section 4 and the relative majority in Section 5 refer to earlier work on *composition of population protocols* from [4].

In our model, entities interact using a classical population protocol, i.e., via global grammars mapping pairs of states to pairs of states. In particular, no exchange of local memories happens during pairwise interactions. The entities use their local memory in order to or-

ganise the sub-protocols executed and in order to draw local conclusions. Thus, if we count the states needed for entities' interactions, we require only $\Theta(k)$ states in our algorithms for absolute and relative majority, and only a constant number of states for protocols computing static and dynamic majority of two colours. In addition, we need only $O(k)$ bits of local memory per entity in our absolute and relative majority protocols in order to handle up to 2^k colours, which is optimal in terms of space requirements.

1.1 Related Work

The population protocol model was initially introduced to simulate behaviour of animal populations [5, 6]. In [5] we can find a formal definition of computations in populations where pairwise interactions of finite-state agents advance the computation. The authors showed a fundamental result that any predicate which is semi-liner can be stably computed by such protocols. In the introduction of their paper, they present a protocol for majority which is exactly the same as the protocol in Section 2 of this paper. In [24] the authors present several models of population protocols including protocols in which each entity of the population is allowed to have some memory, and they discuss several classes of computable predicates in those models. In the first pages, they present a protocol for majority as in [5], which is almost the first protocol presented here. We have included this first protocol in this paper because we add a detailed explanation about reporting ties. Self-stabilizing population protocols were defined in [8] and properties of such protocols were demonstrated. Stabilizing population protocols in the presence of faults were considered in [16].

In due course, population protocols proved to be a useful abstraction in diverse environments including, e.g., wireless sensor networks [4, 27, 20], chemical reaction networks [14], and gene regulatory networks [13]. A large portion of work devoted to population protocols refers to the majority problem. In particular, in [7] the authors study populations with entities governed by 3 states and propose a probabilistic population protocol for *approximate majority*, i.e., where the initial difference between the volumes of the two colours does not fall below $\omega(\sqrt{n}\log n)$. The algorithm stabilises in $O(n\log n)$ rounds with high probability. It also tolerates groups of $o(\sqrt{n})$ entities expressing Byzantine behaviour. Further analysis of this protocol and its 4-state amendment leading to the first efficient exact majority protocol can be found in [23]. Another aspect referring to the parallelism of majority population protocols in the presence of a random scheduler has been studied by Alistarh et al. in [3]. They proposed a poly-logarithmic time majority protocol for entities equipped with memories of size $O(1/\varepsilon + \log n \log 1/\varepsilon)$, for any $\varepsilon > 0$. They also study the respective lower bounds. In a very recent work [1] Alistarh et al. consider a wide spectrum of time and space trade-offs for population protocols and they propose a fast Split-Join majority algorithm stabilising in $O(\log^3 n)$ parallel rounds with high probability. An interesting extension of population protocols to the *random walk* model can be found in [9]. Please note that neither of the majority algorithms discussed above is able to report the tie.

The relative majority variant considered in this paper is well known in the literature under the name of *plurality consensus*. In contrast to the deterministic sequential model adopted in this paper, so far plurality consensus was considered solely in the *gossiping model*. In this model, in a sequence of synchronous rounds each entity contacts a random neighbour simultaneously. Moreover, the protocols converge under the assumption that the number of entities supporting the winning colour must exceed those supporting any other colour by a sufficiently large bias. In this model one explores parallelism of connections aiming at protocols stabilising rapidly with high probability. Doerr et al. [17] explored the *power of two choices* in complete graphs, proposing a stabilisation protocol in the binary

case requiring constant memory and message size. Their protocol converges in $O(\log n)$ rounds assuming a bias of size $\Omega(\sqrt{n \log n})$. A more rigorous analysis of this protocol can be found in [15], also in networks modeled by regular graphs, for which the authors provide tight bounds on convergence time as a function of the second-largest eigenvalue of the graph. In [10] Bechetti et al. consider a plurality consensus protocol based on a sequence of local majority agreements with three randomly chosen neighbours during each round requiring bias $\Omega(\sqrt{Cn \cdot \log n})$. The protocol converges in $\Theta(\min\{C, n^{1/3}\} \cdot \log n)$ rounds using $\Theta(\log C)$ memory and message size, where C refers to the number of original opinions. In later work [11] the authors solve general plurality consensus in complete graphs via *undecided-state dynamics* using an extra state to accommodate intermediate disagreements. They propose the notion of *monochromatic distance* which reflects on the difference between the initial colour configuration from the closest monochromatic solution. Their plurality protocol converges with a logarithmic overhead on the top of the monochromatic distance. A more recent study on plurality consensus in noisy communication channels can be found in [21].

There is also growing interest in exact-space complexity in probabilistic plurality consensus. In particular, in [12] Berenbrink et al. proposed a plurality consensus protocol converging in $O(\log C \cdot \log \log n)$ synchronous rounds using only $\log C + (\log \log C)$ bits of local memory. They also show a slightly slower solution converging in $O(\log n \cdot \log \log n)$ rounds using only $\log C + 4$ bits of local memory. This disproves a conjecture by Becchetti et al. [11] implying that any protocol with local memory $\log C + O(1)$ has the worst-case running time $\Omega(k)$. In [22] Ghaffari and Parter propose an alternative algorithm converging in $O(\log C \log n)$ rounds while having message and local memory sizes based on $\log C + O(1)$ bits. In addition to the above, some work on the application of the random walk in plurality consenus protocols can be found in [11, 9].

1.2 Our results and organisation of the paper

In this paper we study space-optimal population protocols for several variants of the majority problem. The paper presents space-efficient algorithms for majority with many colours, and these algorithms are obtained by using a combination of known protocols for simple majority. In Section 2 we discus an amendment allowing majority protocols to report a tie (equality) if neither of the two original colours dominates the other. In Section 3 we discuss a solution to the dynamic version of the majority problem in which the original colours assigned to the entities can be changed by an external force. Such a solution is a special case of self-stabilizing population protocols which were considered in [8]. We discuss it here to prepare the ground for our space-optimal protocols for many colours.

We consider space-efficient majority protocols in populations with an arbitrary number C of colours represented by k-bit labels, where $k = \lceil \log C \rceil$. In Section 4 we present an asymptotically space-optimal $O(k)$-bit protocol for the *absolute majority*, i.e., a protocol which answers the question whether one colour dominates all others taken together. In Section 5 we propose a multistage $O(k)$-bit protocol for *relative majority*, where all most frequent colours eventually become aware of their dominance, and all nodes learn about the most frequent colour with the largest label. In Section 6 we conclude with final comments and leave the reader with a list of open problems.

2 Population protocol for static majority with equality

This section reformulates the algorithm for majority presented in [5].

Initially each entity $a \in A$ obtains its original colour c_a, being one of the three available denoted by integers $-1, 0$, and 1. Thus, the main goal in our reformulation of majority protocols is to determine whether there are more 1's than (-1)'s (green domination), more (-1)'s than 1's (red domination), or whether there is a tie between the two. In other words, our majority protocols aim at determining the sign of the expression:

$$\sum_{a \in A} c_a.$$

If this sign is positive, there are more 1's, if negative, there are more (-1)'s, and if the sum is 0, we report equivalence between the two competing colours. During the communication process each entity $a \in A$ has an attributed *state* s_a. In due course we will also use the notion of *knowledge* of entities, which includes information about the state and the original colour of the entity.

Throughout the computation process the entities can be in one of the three *strong states* $[-1], [0]$, and $[1]$ or the three weak states $\langle -1\rangle, \langle 0\rangle, \langle 1\rangle$. In the begining, each entity $a \in A$ with attributed colour $c_a = x$ is in state $[x]$. With each state s we associate a weight $w(s)$ such that $w([x]) = x$ and $w(\langle x\rangle) = 0$. This association is illustrated by the table in Fig. 1.

state s	weight $w(s)$
$[-1]$	-1
$[0], \langle -1\rangle, \langle 0\rangle, \langle 1\rangle$	0
$[1]$	1

Figure 1 The states and their weights.

In due course, when two entities $a, b \in A$ interact the shared *transition function* determines their resulting states. And, in particular, if an entity in a strong state $[x]$ meets another in a weak state $\langle y\rangle$, the weak state becomes $\langle x\rangle$ and the strong state remains unchanged. If during a meeting a strong state $[x]$, for $x \neq 0$, meets $[0]$ then only state $[0]$ is changed to $\langle x\rangle$. Finally, if $[1]$ interacts with $[-1]$ both states are changed to $[0]$. Other type of encounters does not change the states of entities. The respective shared transition function is illustrated by the table in Fig. 2.

$s_a \backslash s_b$	$[-1]$	$[0]$	$[1]$	$\langle -1\rangle$	$\langle 0\rangle$	$\langle 1\rangle$
$[-1]$	$([-1],[-1])$	$([-1],\langle -1\rangle)$	$([0],[0])$	$([-1],\langle -1\rangle)$	$([-1],\langle -1\rangle)$	$([-1],\langle -1\rangle)$
$[0]$	$(\langle -1\rangle,[-1])$	$([0],[0])$	$(\langle 1\rangle,[1])$	$([0],\langle 0\rangle)$	$([0],\langle 0\rangle)$	$([0],\langle 0\rangle)$
$[1]$	$([0],[0])$	$([1],\langle 1\rangle)$	$([1],[1])$	$([1],\langle 1\rangle)$	$([1],\langle 1\rangle)$	$([1],\langle 1\rangle)$
$\langle -1\rangle$	$(\langle -1\rangle,[-1])$	$(\langle 0\rangle,[0])$	$(\langle 1\rangle,[1])$	$(\langle -1\rangle,\langle -1\rangle)$	$(\langle -1\rangle,\langle 0\rangle)$	$(\langle -1\rangle,\langle 1\rangle)$
$\langle 0\rangle$	$(\langle -1\rangle,[-1])$	$(\langle 0\rangle,[0])$	$(\langle 1\rangle,[1])$	$(\langle 0\rangle,\langle -1\rangle)$	$(\langle 0\rangle,\langle 0\rangle)$	$(\langle 0\rangle,\langle 1\rangle)$
$\langle 1\rangle$	$(\langle -1\rangle,[-1])$	$(\langle 0\rangle,[0])$	$(\langle 1\rangle,[1])$	$(\langle 1\rangle,\langle -1\rangle)$	$(\langle 1\rangle,\langle 0\rangle)$	$(\langle 1\rangle,\langle 1\rangle)$

Figure 2 The transition table for static majority protocol with ties.

▶ **Lemma 1** (Invariant 1). *Initially, the sum* $S = \sum_{a \in A} w(s_a)$ *equals to* $\sum_{a \in A} c_a$*, and its value remains unchanged during the computation process.*

Proof. Follows directly from the definition of the transition function. ◀

Observation If the sum S is negative, it declares majority of reds (denoted by -1), positive S indicates majority of greens (denoted by 1), otherwise S refers to the tie.

▶ **Lemma 2** (Invariant 2). *The value of the sum $R = \sum_{a \in A} |w(s_a)|$ decreases monotonically throughout the communication process and it stabilises eventually on the value $R_{\text{fin}} = |S|$.*

Proof. At any stage of the algorithm R represents the number of strong states $[-1]$ and $[1]$ still present in the population. According to the transition function the number of such states can only decrease when two states $[1]$ and $[-1]$ annihilate one another during a direct interaction. Thus, eventually the sum R stabilises on the original difference between the number of strong states $|S|$. ◀

We conclude this section with a theorem.

▶ **Theorem 3.** *The population protocol presented in this section computes majority and returns equality if neither of the colours dominates the other.*

Proof. According to the observation and the two lemmas, if a majority exists, the remaining entities in strong states of the dominating colour will recolour all entities accordingly. Otherwise, the annihilation of the last pair of states $([1], [-1])$ results in obtaining two entities with states $[0]$ which in due course will change states in all other entities to $\langle 0 \rangle$. Finally, if neither of the states $[1]$ or $[-1]$ is initially present in the population all entities remain in the neutral state $[0]$. ◀

3 Population protocol for dynamic majority with equality

In this section we consider a variant of population protocols in which the original colours (attributes) of entities could be altered by an *external force* for some unspecified, however limited, period of time. After this initial period, the relevant population protocol is expected to eventually stabilize. The model of changing inputs from [4] and the concept of composing several population protocols as described in [24] are the inspirations for our approach here. In essence, we reformulate the majority algorithm from [4] and show how to modify this reformulation so that it can be used as a subprotocol for our next section.

We assume that an entity is aware when its original colour changes, and is able to modify its current state as a result, but such a change is again governed by common state transition rules for all entities. We also assume that it is not possible for the external force to alter the original colour of an entity while it is simultaneously interacting with another entity.

We use the protocol we propose here as a subroutine in more structurally complex population protocols for the absolute majority in Section 4, and for the relative majority in Section 5.

The population protocol presented below determines whether there are more original 1's, more (-1)'s, or there is a tie after the last intervention of the external force. For the purpose of our protocol each entity $a \in A$ must store its original colour $c_a \in \{-1, 0, 1\}$, and this stored colour can be altered only by the external force at any time. Besides the colour, the entity maintains a state s_a governed by the shared transition function. More formally, an entity's knowledge refers to the pair (c_a, s_a). We define five strong states:$[-2], [-1], [0], [1], [2]$, and three weak states $\langle -1 \rangle, \langle 0 \rangle, \langle 1 \rangle$. Before the protocol is initiated, if $c_a = x$ we set $s_a = [x]$. On the conclusion all entities are in state

- $[1], [2]$ or $\langle 1 \rangle$ if there are more 1's than (-1)'s,
- $[-1], [-2]$ or $\langle -1 \rangle$ if there are less 1's than (-1)'s, and
- $[0]$ or $\langle 0 \rangle$ when there is a tie.

We define the weight function, $w(s)$, on a state s as $w([x]) = x$ and $w(\langle x \rangle) = 0$, see the table in Fig. 3.

During execution of the majority protocol we maintain two invariants:

s	$w(s)$
$[-2]$	-2
$[-1]$	-1
$[0], \langle -1\rangle, \langle 0\rangle, \langle 1\rangle$	0
$[1]$	1
$[2]$	2

s_a, $c_a = 1$ changes to $c'_a = -1$	s'_a
$[0], \langle -1\rangle, \langle 0\rangle, \langle 1\rangle$	$[-2]$
$[1]$	$[-1]$
$[2]$	$[0]$
s_a, $c_a = -1$ changes to $c'_a = 1$	s'_a
$[0], \langle -1\rangle, \langle 0\rangle, \langle 1\rangle$	$[2]$
$[-1]$	$[1]$
$[-2]$	$[0]$

Figure 3 The weight function $w(s)$ and the state transition rules when recolouring occurs by an external force.

1. $\sum_{a\in A} c_a = \sum_{a\in A} w(s_a)$, and
2. for each $a \in A$, $|w(s_a) - c_a| \leq 1$.

The two invariants are preserved thanks to carefully crafted state transition rules and counterparting alterations of an entity's state caused by changes of the original colour c_a imposed by the external force. When the colour c_a is changed to $c'_a = c_a + \delta$, the state is changed from s_a to $s'_a = [w(s_a) + \delta]$. Note that this rule preserves both invariants 1 and 2. This is illustrated by the table to the right in Fig. 3 describing how states are changed when $c_a = 1$ is changed to $c'_a = -1$, or vice-versa. In this table we do not consider, for example, combinations of states $s_a = [-1], [-2]$ with colour $c_a = 1$ because of the invariant 2.

In what follows we describe what happens to the states when two entities $a, b \in A$ interact. If a strong state $[1]$ or $[2]$ meets a weak state $\langle y\rangle$ or $[0]$, then this second state becomes $\langle 1\rangle$. If a strong state $[-1]$ or $[-2]$ meets a weak state $\langle y\rangle$ or $[0]$, then the latter state becomes $\langle -1\rangle$. If a strong state $[0]$ meets a weak state, the weak state is changed to $\langle 0\rangle$. If $[1]$ meets $[-1]$ or $[2]$ meets $[-2]$, they are both changed to $[0]$. If $[2]$ meets $[-1]$, they are changed to $[1]$ and $[0]$ respectively. If $[-2]$ meets $[1]$, they are changed to $[-1]$ and $[0]$ respectively. Other encounters do not result in state alteration. This is illustrated by the table in Fig. 4 which does not take into account encounters between entities where both are in weak states, because they do not result in state alteration.

$s_a \backslash s_b$	$[-2]$	$[-1]$	$[0]$	$[1]$	$[2]$
$[-2]$	$([-2],[-2])$	$([-2],[-1])$	$([-2],\langle -1\rangle)$	$([-1],\langle -1\rangle)$	$([0],[0])$
$[-1]$	$([-1],[-2])$	$([-1],[-1])$	$([-1],\langle -1\rangle)$	$([0],[0])$	$(\langle 1\rangle,[1])$
$[0]$	$(\langle -1\rangle,[-2])$	$(\langle -1\rangle,[-1])$	$([0],[0])$	$(\langle 1\rangle,[1])$	$(\langle 1\rangle,[2])$
$[1]$	$(\langle -1\rangle,[-1])$	$([0],[0])$	$(\langle 1\rangle,[1])$	$([1],[1])$	$([1],[2])$
$[2]$	$([0],[0])$	$([1],\langle 1\rangle)$	$([2],\langle 1\rangle)$	$([2],[1])$	$([2],[2])$
weak	$(\langle -1\rangle,[-2])$	$(\langle -1\rangle,[-1])$	$(\langle 0\rangle,[0])$	$(\langle 1\rangle,[1])$	$(\langle 1\rangle,[2])$

Figure 4 The state transition table for interacting entities for dynamic majority.

▶ **Lemma 4.** *The invariants 1 and 2 are preserved during execution of the majority protocol.*

Proof. First, we consider interactions between pairs of entities.

Invariant 1 is preserved, because for any state transition, if the weight of one entity is reduced, then the weight of the other is increased by the same (absolute) value. Also, if colour c_a is changed, then the weight $w(s_a)$ is changed too by the same value.

Invariant 2 is preserved because during every interaction of entities $|w(s_a)|$ can only decrease and $w(s_a)$ does not change its sign. So if $c_a = 1$, then s_a is initially in the interval

$[0, 2]$ and it remains in this interval. The reasoning in the remaining cases when $c_a = 0$ or -1 is analogous.

Now we consider the invariants when the external force changes the colour of an entity. Suppose that an entity is coloured $c_a = 1$ and its colour is changed to $c'_a = -1$ (the other case will be similar).

Invariant 1 is preserved by the choice of the transitions shown in the table in the right of Fig. 3. The left hand side of the equation in invariant 1 decreases by 2 (since the colour changes from 1 to -1). If the state of the entity was $s_a \in \{[0], \langle -1\rangle, \langle 0\rangle, \langle 1\rangle\}$, the new state is $s'_a = [w(s_a) - 2] = [-2]$. Hence the corresponding weight changes from $w(s_a) = 0$ to $w(s'_a) = -2$, so the right hand side of invariant 1 also decreases by 2 (i.e., preserving the invariant). Similarly, if $s_a = [1]$, then the new state is $s'_a = [-1]$, hence the contribution to the right hand side of invariant 1 from the entity changes from $w(s_a) = 1$ to $w(s'_a) = -1$, again a decrease by 2. We can check the remaining case, where $s_a = [2]$, in an analogous manner.

Invariant 2 is also maintained by the rules that govern how the entity's state is updated when its colour is changed by an external force. E.g., $c_a = 1$ changing to $c'_a = -1$ means that the new weight $w(s'_a) \in \{0, -1, -2\}$ from the rules in Fig. 3. ◀

▶ **Lemma 5.** *The value of $R = \sum_a |w(s_a)|$ does not increase after the last intervention of the external force. Moreover the value of R stabilises when eventually there are no two entities $a, b \in A$ such that $w(s_a) > 0$ and $w(s_b) < 0$.*

Due to Lemma 5 the majority process stabilises in three possibile configurations with respect to $C_{\text{fin}} \stackrel{def}{=} \sum_{a \in A} c_a$ (where c_a is referring to the *final* colour of the entity a, after any external forces have stopped changing the colours of entities). If on the conclusion $C_{\text{fin}} > 0$, there must be some entities in states $[1]$ or $[2]$ which would earlier ensure that all weak states and the state $[0]$ are switched to $\langle 1\rangle$. If $C_{\text{fin}} < 0$, there must be some entities in states $[-1]$ or $[-2]$ which would earlier ensure that all weak states and the state $[0]$ are switched to $\langle -1\rangle$. However, if on the conclusion $C_{\text{fin}} = 0$, there are no entities in states $[x]$ with $x \neq 0$ and the last entity that reached state $[0]$ will have a chance to alter all weak states to $\langle 0\rangle$.

4 Absolute majority

In the remaining part of the paper we work under the assumption that the population is coloured with an arbitrary number C of colours, where $2^{k-1} < C \leq 2^k$, for some integer $k \geq 1$ that is known to all entities. Each colour is denoted by a k-bit label $l[0..k-1]$, and single labels are attributed to entities with the relevant colours. As in previous sections, we interpret the individual bits $l[i]$ in this label as -1 or 1, rather than more standard 0 or 1. Each entity is assumed to own an extra $O(k)$ bits used to support the computation process, including interaction with other entities in the population.

In this section we present an asymptotically optimal $O(k)$-bit population protocol computing *absolute majority*, i.e., answering whether there exists a colour which dominates all the remaining colours in the population taken together. The absolute majority algorithm presented here is a combination of the static majority protocol introduced in Section 2, and later referred to as P_1, as well as the dynamic majority protocol from Section 3, from now on referred to as P_2. We recall that protocol P_2 assumes full knowledge of entities and it is using two types of state transitions: (1) imposed by the *external force* and altering original colours associated with entities, and (2) caused by the interaction with other entities in the

population.

Memory organisation Each entity uses $O(k)$ bits of memory to accommodate:
1. The k-bit label $l[0..k-1]$ representing the original colour of the entity,
2. An array $s[0..k-1]$ representing k independent instances of protocol P_1, and
3. An instance of protocol P_2 with the *external force* based on k instances of P_1.

For the purpose of our algorithm we define k independent instances of static majority protocols $P_1(i)$, for $i = 0, \ldots, k-1$, such that colours competing in $P_1(i)$ refer to the bits $l[i]$ drawn from each entity in the population. Assume $l^*[0..k-1]$ is a k-bit label of the colour of the absolute majority in the population. One can observe that when the majority protocols stabilise, for all $i = 0, \ldots, k-1$, each bit $l^*[i]$ must be in majority reported by $P_1(i)$ via entry $s[i]$. Thus, if the absolute majority exists, one can run k static majority protocols $P_1(i)$ to determine the majority colour. However, if there is no absolute majority the protocol proposed above may still return a false positive "winner". This can happen, e.g., if no entity has a colour with the label in which all bits are set to 1s but the majority of bits $l[i]$, for all $i = 0, .., k-1$ for all entities are 1's. In such case, the non-existing colour with the label filled with 1s would be wrongly recognised by the entities as the absolute majority. In order to overcome this clear deficiency of the protocol, an extra (final) test is performed with the help of protocol P_2 to decide whether the returned colour is in the absolute majority.

4.1 Algorithm Absolute-Majority

Initialisation Stage

1. Before execution of the algorithm, each entity $a \in A$ sets for itself $s[i] = [1]$ if $l[i] = 1$ and $[-1]$ otherwise, for all $i = 0, .., k-1$. This choice refers to the belief that its original colour c_a is in majority. And, indeed, each entity initially adopts an extra colour 1 (denoting membership in the majority) for the purpose of protocol P_2.
2. Later, during pairwise interactions between entities, the current states in $s[i]$ get updated by the relevant majority protocols $P_1(i)$, for each $i = 0, \ldots, k-1$ independently. And if at any time the contents of $s[i]$ and $l[i]$ do not reflect its initial setting, the belief of the entity changes to -1. However, this belief becomes 1 again as soon as the consistency between bits in $s[0..k-1]$ and $l[0..k-1]$ is restored. This consistency measure determines actions of the *external force* in protocol P_2.

Stabilisation Stage
1. At first, the majority algorithm stabilises on all protocols $P_1(i)$, for $i = 0, \ldots, k-1$, which allows each entity to establish the final relationship between the corresponding bits in $s[0..k-1]$ and $l[0..k-1]$. This, in turn, determines the extra colour (1 or -1) of the entity adopted for the purpose of protocol P_2.
2. When eventually protocol P_2 also terminates and concludes with colour 1 in majority, all entities receive confirmation that the final states in $s[0..k-1]$ refer to the absolute majority colour $l^*[0..k-1]$. Otherwise, the entities learn that none of the colours is in the absolute majority.

Note that all protocols described above run simultaneously right from the beginning, and, in particular, protocol P_2 works at least for some time on unstable data. Nevertheless, as the bits generated by protocols P_1 eventually stabilise, thanks to protocol P_2's tolerance of dynamic changes, the absolute majority (if such exists) is confirmed. We conclude with this theorem.

▶ **Theorem 6.** *Algorithm Absolute-Majority computes absolute majority on populations with at most 2^k colours with the help of $O(k)$ memory bits in each entity.*

Proof. If an absolute majority colour exists (represented as a k-bit label $l[0..k-1]$) then, when the k independent instances of P stabilize, each $P_1(i)$ stabilizes in the bit $l(i)$. In fact, each bit of the label of the colour of the absolute majority is then reported by $P_1(i)$ via its entry $s[i]$. However, the population still needs to verify this since, in case of no absolute majority colour, the above protocol may return a false positive "winner" . This can happen if for each i there is an absolute majority bit but the whole tuple of these bits does not correspond to a colour in the population. In order for this case not to be wrongly understood as the absolute majority , we need a verifying step. This is exactly what protocol P_2 does. In fact, $P(2)$ always runs a test to decide whether the returned supposed absolute majority colour is indeed the absolute majority. Protocol P_2 works for some time on unstable data. However, after a time t by which all $P_1(i)$ have stabiized, protocol P_2 shall also stabilize either by concluding that the assumed majority colour (indicated by colour 1 in the algorithm) is indeed an absolute majority, or it shall stabilize reporting nonexistence of the absolute majority colour to all entities. Note that each time P_2 has to check only one supposed majority colour against all others, treated as a single colour -1 in the algorithm. The above proof works due to the established fact that P_2 tolerates dynamic changes in the input colours. ◀

5 Relative majority

As in Section 4, in this section we assume that the population is attributed with an arbitrary number C of colours, where $2^{k-1} < C \leq 2^k$, for some integer $k \geq 1$ that is known to all entities. Each colour is denoted by a k-bit label $l[0..k-1]$, where $l[i] \in \{-1, 1\}$. Each entity is assumed to have extra $O(k)$ bits used to support the computation process, including communication with other entities in the population. The *relative majority* problem refers to the task of finding the most frequent colour in the population. Note that there can be more than one colour that is the most frequent. In such case the colour with the latest in the lexicographical order label $l^*[0..k-1]$ is declared as the *winner*.

Computing relative majority is a more complex task, comparing to the absolute majority, as here one needs to collect evidence confirming that the winning colour beats any other colour in the population. At first we describe a protocol for the relative majority which only finds the winner $l^*[0..k-1]$. This is done by marking all entities possessing this colour with the winning label. In this setting, the colour in the relative majority always exists. The case in which the uniqueness of the majority colour is required is commented later in Section 5.2.

In the relative majority protocol, instead of engaging in the total comparison (via majority computation) in pairs formed of any two colours, which would require $O(k^2)$-bit memories, we propose a solution similar to finding maximal elements in parallel stages based on duels. In each stage the winning colours perform pairwise duels via majority protocols to reduce the number of winners by half. This multi-stage computation is made feasible thanks to pipelining of dynamic majority protocols P_2 which gradually stabilise starting from the lowest stage and finishing at the highest stage of the dueling process.

Stages are enumerated by descending numbers from the lowest stage $k-1$ to the highest 0. In stage i, for all $i = k-1, .., 0$, two colours are in the same group if their k-bit labels $l[0..k-1]$ share i-bit prefix $l[0..i-1]$ (in stage 0 all labels form one group). In this stage agents in one group aim at finding the majority colour label in each group.

Memory organisation. Each entity $a \in A$ uses $O(k)$ bits of memory to accommodate:

1. The k-bit label $l[0..k-1]$ representing the original colour of the entity. This colour is fixed (never changed) throughout the computation process.
2. The k-bit label $c[0..k-1]$ represents current colours $c[i]$ of the entity in each consecutive stage i, with the decreasing index $i = k-1, .., 0$. On the conclusion of stage i, if label $l[0..k-1]$ is declared as the winner in the group of labels with prefix $l[0..i]$, the value $c[i]$ equals to ± 1, otherwise $c[i] = 0$. All entities with the winning colour $l[0..k-1]$ in its group in higher stage $i-1$ have the value $c[i]$ set to $l[i]$. Before the stabilisation of $P_2(i-1)$ the value $c[i]$ reflects the current belief of the entity about this value.
3. An array $s[0..k-1]$ representing states $s[i]$ in k independent instances of protocol $P_2(i)$ associated with colours $c[i]$. The computations with respect to $P_2(i)$ are performed only if the two interacting entities have the same label prefix $l[0..i-1]$. Otherwise protocol $P_2(i)$ is not executed. We emphasise here that computations in $P_2(i)$ can change values $c[i-1]$ whose change in turn cause alteration of states $s[i-1]$. Also, changes in $c[i]$ can change $c[i-1]$.

5.1 Algorithm Relative-Majority

Initialisation Stage Before execution of the algorithm, each entity sets $c[i] = l[i]$ and $s[i] = [1]$ if $c[i] = 1$ and $[-1]$ otherwise, for all $i = 0, .., k-1$.

Stabilisation Stage

1. The algorithm stabilises first on protocol $P_1(k-1)$, as at the beginning of the pipeline there is no external force, and then subsequently on protocols $P_2(k-2)$, $P_2(k-3)$, ..., $P_2(0)$.
2. An entity believes that its colour wins on stage i if, either $c[i] = -1$ and $s[i] \in \{[-1], [-2], \langle -1\rangle\}$, or $c[i] = 1$ and $s[i] \in \{[0], [1], [2], \langle 0\rangle, \langle 1\rangle\}$. The states $[0], \langle 0\rangle$ correspond to a tie and in this case the lexicographically larger label becomes the winner. If the entity believes its label $l[0..k-1]$ is the winner in stage i, it sets $c[i-1] = l[i-1]$ and adjusts $s[i-1]$ as specified in protocol $P_2(i-1)$ if $c[i-1]$ gets changed. If, to the contrary, the entity believes it did not win, it sets $c[i-1] = 0$ and also adjusts $s[i-1]$ should change occur in $c[i-1]$. Note, that in both cases changes in $c[i-1]$ are propagated to $c[i-2]$ and further on.
3. Eventually protocol $P_2(0)$ stabilizes. At that time entities that win in stage 0 hold the winning majority colour.

▶ **Theorem 7.** *Algorithm Relative-Majority computes relative majority on population with at most 2^k colours with the help of $O(k)$ memory bits in each entity.*

Proof. The memory requirement follows directly from the formulation of the protocol. In order to prove correctness, we proceed by induction on stage numbers i taken in reverse order. The colours $c[k-1]$ do not change during the protocol so in some moment t_{k-1} protocols $P_2(k-1)$ stabilize and states $s[k-1]$ stop being changed. These states determine unique winning k-bit colours in groups corresponding to all possible prefixes $l[0..k-2]$.

Now let $i > k-1$ be a stage number. By inductive hypothesis in some time t_{i+1}, protocols $P_2(i+1)$ stabilize and states $s[i+1]$ stop being changed. They indicate unique winning k-bit colours in groups corresponding to each prefix $l[0..i]$. So, since t_{i+1} colours $c[i]$ are ± 1 for these winners, 0 for others and do not change anymore. Thus, in some later time t_i, protocols $P_2(i)$ stabilize and states $s[i]$ cease being changed. From the formulation of the

protocol these final states $s[i]$ determine the winning k-bit colours in groups corresponding to prefixes $l[0..i-1]$.

Finally ,at some time t_0, protocols $P_2(0)$ stabilize and all entities compute states $s[0]$ corresponding to the unique winning k-bit colour amongst all of them. ◀

5.2 Uniqueness in relative majority

As indicated at the beginning of Section 5, one may want to report only unique relative majority colours, i.e., when there is exactly one, the most frequent colour. And indeed if the winning colour l^* is not unique, there must exist some other colours which lost to l^* in a tie at some stage. The purpose of the mechanism presented below is to encounter such ties (if they exist) and to distribute this information to all entities in the population. This can be done by performing an additional *dissemination protocol* with the help of an extra bit c' drawn from the set $\{0, 1\}$. This dissemination protocol is run by each entity in conjunction with the relative majority protocol described above, and its actions are governed by the current belief of the entity whether it is a winner or not and by encountered or not ties in duels. The following four rules govern values of the extra bit c'.

Initially, (1) in each entity the extra bit c' is set to 0 to denote that the entity does not carry any information about ties between the winners. This value can be changed to 1 if (2) the colour of the entity is still a potential winner (did not lose any duel yet in the most recent climb through the stages) and at some stage its duel ends up in a tie; or if (3) the colour of the entity is already deemed as the loser and it meets another entity with the colour still being a potential winner and its extra bit $c' = 1$. And (4) the extra bit c' can be changed back to 0 if and only if the colour of its owner is deemed as loser and it meets another entity with the colour still being a potential winner and its extra bit $c' = 0$.

In due course the values of each extra bits c' can be altered several times according to the rules 1, 2 or 3. However, when eventually the relative majority protocol determines the winning colour l^* in stage 0, only entities coloured with l^* are able to change values of extra bits in other entities. Now, if the extra bit associated with entities coloured by l^* is 0, i.e., the winning colour has never experienced a tie, all other entities are eventually informed accordingly by rule 4. And, if the extra bit associated with entities coloured by l^* is 1, i.e., the winning colour has encountered a tie in the past, all other entities are eventually informed accordingly by rule 3.

6 Conclusion

In this paper we presented memory-efficient population protocols for several variants of the majority problem.

In Section 2 we show how to amend majority protocols to report ties. The proposed protocol relies on a relatively large number of states used by entities. One can show a more space-efficient solution limited to six states. Also in a wider context, in our solutions the emphasis was on asymptotic space optimality. One open problem, however, is to determine more exact bounds on the number of states required to compute the considered types of majorities for a given number of colours C. Another interesting problem refers to the time complexity and parallelism of considered majority problems in the presence of a random scheduler. Finally, one can ask what other computations are possible through a composition of several "partially self-stabilizing" (sub)protools.

Acknowledgements

The authors wish to thank Yukiko Yamauchi for the valuable referral to [18] about fair composition of self-stabilising algorithms, and for discussions related to the work in this paper.

We also wish to thank the anonymous referees and the shepherd of OPODIS 2016 for their help in preparation of the final version of this paper.

References

1 D. Alistarh, J. Aspnes, D. Eisenstat, R. Gelashvili, and R.L. Rivest. Time-space trade-offs in population protocols. CoRR abs/1602.08032, 2016.

2 D. Alistarh and R. Gelashvili. Polylogarithmic-time leader election in population protocols. In *Proc. 42nd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 479–491, 2015.

3 D. Alistarh, R. Gelashvili, and M. Vojnovic. Fast and exact majority in population protocols. In *Proc. 33rd Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 47–56, 2015.

4 D. Angluin, J. Aspnes, M. Chan, M.J. Fischer, H. Jiang, and R. Peralta. Stably computable properties of network graphs. In *Proc. 1st IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 63–74, 2005.

5 D. Angluin, J. Aspnes, Z. Diamadi, M.J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. In *Proc. 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 290–299, 2004.

6 D. Angluin, J. Aspnes, Z. Diamadi, M.J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4):235–253, 2006.

7 D. Angluin, J. Aspnes, and D. Eisenstat. A simple population protocol for fast robust approximate majority. *Distributed Computing*, 21(2):87–102, 2008.

8 D. Angluin, J. Aspnes, M.J. Fischer, and H. Jiang. Self-stabilizing population protocols. *ACM Trans. Auton. Adapt. Syst.*, 3(4):1–28, 2008.

9 L. Gąsieniec, D.D. Hamilton, R. Martin, and P.G. Spirakis. The match-maker: Constant-space distributed majority via random walks. In *Proc. 17th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 67–80, 2015.

10 L. Becchetti, A. Clementi, E. Natale, F. Pasquale, R. Silvestri, and L. Trevisan. Simple dynamics for plurality consensus. In *Proc. 26th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 247–256, 2014.

11 L. Becchetti, A.E.F. Clementi, E. Natale, F. Pasquale, and R. Silvestri. Plurality consensus in the gossip model. In *Proc. 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 371–390, 2015.

12 P. Berenbrink, T. Friedetzky, G. Giakkoupis, and P. Kling. Efficient plurality consensus, or: The benefits of cleaning up from time to time. In *Proc. 43rd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 1–14, 2016.

13 J.M. Bower and H. Bolouri. *Computational modeling of genetic and biochemical networks.* MIT Press, 2004.

14 H.-L. Chen, R. Cummings, D. Doty, and D. Soloveichik. Speed faults in computation by chemical reaction networks. *Distributed Computing*, pages 16–30, 2014.

15 C. Cooper, R. Elsaesser, and T. Radzik. The power of two choices in distributed voting. In *Proc. 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 435–446, 2014.

16 C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, and E. Ruppert. When birds die: Making population protocols fault-tolerant. In *IEEE 2nd Intl. Conference on Distributed Computing in Sensor Systems (DCOSS)*, volume 4026 of *Lecture Notes in Computer Science*, pages 51–56. Springer-Verlag, 2006.

17 B. Doerr, L.A. Goldberg, L. Minder, T. Sauerwald, , and C. Scheideler. Stabilizing consensus with the power of two choices. In *Proc. 23rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 149–158, 2011.

18 S. Dolev. *Self Stabilization*. MIT Press, 2000.

19 D. Doty and D. Soloveichik. Stable leader election in population protocols requires linear time. In *Proc. 29th International Symposium on Distributed Computing (DISC)*, pages 602–616, 2015.

20 M. Draief and M. Vojnovic. Convergence speed of binary interval consensus. *SIAM Journal on Control and Optimization*, 50(3):1087–1109, 2012.

21 P. Fraigniaud and E. Natale. Noisy rumor spreading and plurality consensus. In *Proc. 34th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 127–136, 2016.

22 M. Ghaffari and M. Parter. A polylogarithmic gossip algorithm for plurality consensus. In *Proc. 34th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 117–126, 2016.

23 G.B. Mertzios, S.E. Nikoletseas, C. Raptopoulos, and P.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)*, pages 871–882, 2014.

24 O. Michail, I. Chatzigiannakis, and P.G. Spirakis. *New Models for Population Protocols*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2011.

25 O. Michail and P.G. Spirakis. Simple and efficient local codes for distributed stable network construction. In *Proc. 32nd Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 76–85, 2014.

26 Y. Mocquard, E. Anceaume, J. Aspnes, Y. Busnel, and B. Sericola. Counting with population protocols. In *Proc. 2015 IEEE 14th International Symposium on Network Computing and Applications (NCA)*, pages 35–42, 2015.

27 E. Perron, D. Vasudevan, and M. Vojnovic. Using three states for binary consensus on complete graphs. In *Proc. 28th Conference on Computer Communications (INFOCOM)*, pages 2527–2535, 2009.