Distrib. Comput. (2018) 31:455–469
https://doi.org/10.1007/s00446-017-0317-z

How many cooks spoil the soup?

Othon Michail[1] · **Paul G. Spirakis**[1,2]

Received: 24 February 2017 / Accepted: 14 October 2017 / Published online: 27 October 2017

Abstract In this work, we study the following basic question: *"How much parallelism does a distributed task permit?"* Our definition of *parallelism* (or *symmetry*) here is not in terms of speed, but in terms of identical *roles* that processes have at the same time in the execution. For example, we may ask: *"Can a given task be solved by a protocol that always has at least two processes in the same role at the same time?"* (i.e., by a protocol that never elects a *unique leader*). We choose to initiate this study in population protocols, a very simple model that not only allows for a straightforward definition of what a role is, but also encloses the challenge of isolating the properties that are due to the protocol from those that are due to the *adversary scheduler*, who controls the interactions between the processes. In particular, we define the role of a process at a given time to be equivalent to the *state* of the process at that time. Moreover, we isolate the symmetry that is due to the protocol (*inherent symmetry*) by focusing on those schedules that maximize symmetry for that protocol and observing how much symmetry breaking the protocol is forced to achieve in order to solve the problem. To allow for such *symmetry maximizing* schedules we consider *parallel schedulers* that in every step may select a whole collection of pairs of nodes (up to a perfect matching) to interact and not just a single pair. Based on these definitions of symmetric computation, we (i) give a *partial characterization* of the set of predicates on input assignments that can be *stably computed with maximum symmetry*, i.e., $\Theta(N_{min})$, where N_{min} is the minimum multiplicity of a state in the initial configuration, and (ii) we turn our attention to the remaining predicates (that have some essentially different properties) and prove a *strong impossibility result* for the *parity* predicate: the inherent symmetry of any protocol that stably computes it is *upper bounded by a constant that depends on the size of the protocol*. The latter immediately generalizes to a subset of the predicates that are *not closed under doubling*.

Keywords Coordinator · Parallelism · Symmetry · Symmetry breaking · Population protocol · Leader election · Majority · Parity

1 Introduction

George Washington said *"My observation on every employment in life is, that, wherever and whenever one person is found adequate to the discharge of a duty by close application thereto, it is worse executed by two persons, and scarcely done at all if three or more are employed therein"* [40]. The goal of the present paper is to investigate whether the analogue of this observation in simple distributed systems is true. In particular, we ask whether a task that can be solved when a single process has a crucial duty is still solvable when that (and any other) duty is assigned to more than one process. Moreover, we are interested in quantifying the degree of *parallelism* (also called *symmetry* in this paper) that a task is susceptible of.

Supported in part by the School of EEE/CS of the University of Liverpool, NeST initiative, and the EU IP FET-Proactive project MULTIPLEX under Contract No. 317532. A preliminary version of the results in this paper has appeared in [33].

✉ Othon Michail
Othon.Michail@liverpool.ac.uk

Paul G. Spirakis
P.Spirakis@liverpool.ac.uk

[1] Department of Computer Science, University of Liverpool, Ashton Street, Liverpool L69 3BX, UK

[2] Computer Technology Institute and Press "Diophantus" (CTI), Patras, Greece

Leader election is a task of outstanding importance for distributed algorithms. One of the oldest [8] and probably still one of the most commonly used approaches [3,11,22,25,27] for solving a distributed task in a given setting, is to execute a distributed algorithm that manages to elect a unique leader (or *coordinator*) in that setting and then compose this (either sequentially or in parallel) with a second algorithm that can solve the task by assuming the existence of a unique leader. Actually, it is quite typical, that the tasks of electing a leader and successfully setting up the composition enclose the difficulty of solving many other higher-level tasks in the given setting.

Due to its usefulness in solving other distributed tasks, the leader election problem has been extensively studied, in a great variety of distributed settings [6,11,21,22,27]. Still, there is an important point that is much less understood, concerning whether an election step is *necessary* for a given task and *to what extent* it can be avoided. Even if a task T can be solved in a given setting by first passing through a configuration with a unique leader, it is still valuable to know whether there is a correct algorithm for T that avoids this. In particular, such an algorithm succeeds without the need to ever have less than k processes in a given "role", and we are also interested in how large k can be without sacrificing solvability.

Depending on the application, there are several ways of defining what the "role" of a process at a given time in the execution is. In the typical approach of electing a unique leader, a process has the leader role if a *leader* variable in its local memory is set to true and it does not have it otherwise. In other cases, the role of a process could be defined as its complete local history. In such cases, we would consider that two processes have the same role after t steps iff both have the same local history after each one of them has completed t local steps. It could also be defined in terms of the external interface of a process, for example, by the messages that the process transmits, or it could even correspond to the branch of the program that the process executes. In this paper, as we shall see, we will define the role of a process at a given time in the execution, as the entire content of its local memory. So, in this paper, two processes u and v will be regarded to have the same role at a given time t iff, at that time, the local state of u is equal to the local state of v.

Another important issue has to do with the fact that, no matter which definition we choose, the present role of a given process usually depends not only on the algorithm executed but also on a number of adversarially determined factors. Actually, it is always the result of applying the local program to the initial state and the sequence of received messages, where the initial state can be arbitrarily selected from a set of possible initial states and the messages depend on the network (which may be even dynamic and adversarially controlled in some settings) and possibly also on other factors that are model-specific, e.g., faults. Given that in this work we are interested in characterizing the parallelism that is only due to the protocol, we have to agree on a way of isolating it from all these protocol-external factors.

Understanding the parallelism that a distributed task allows, is of fundamental importance for the following reasons. First of all, usually, the more parallelism a task allows, the more efficiently it can be solved. Moreover, the less symmetry a solution for a given problem has to achieve in order to succeed, the more vulnerable it is to faults. For an extreme example, if a distributed algorithm elects in every execution a unique leader in order to solve a problem, then a single crash failure (of the leader) can be fatal. Finally, understanding the inherent parallelism of distributed tasks, may enable the development of truly distributed algorithms for those tasks that allow much parallelism or to conclude that electing a unique process (or a few such processes) in a given role is necessary for solving the task.

1.1 Our approach

We have chosen to initiate the study of the above problem in a very minimal distributed setting, namely in Population Protocols of Angluin et al. [3] (see Sect. 1.2 for more details and references). One reason that makes population protocols convenient for the problem under consideration, is that the role of a process at a given step in the execution can be defined in a straightforward way as the state of the process at the beginning of that step. So, for example, if we are interested in an execution of a protocol that stabilizes to the correct answer without ever electing a unique leader, what we actually require is an execution that, up to stability, never goes through a configuration in which a state q is the state of a single node, which implies that, in every configuration of the execution, every state q is either absent or the state of at least two nodes. Then, it is straightforward to generalize this to any symmetry requirement k, by requiring that, in every configuration, every state q is either absent or the state of at least k nodes.

What is not straightforward in this model (and in any model with adversarially determined events), is how to isolate the symmetry that is *only* due to the protocol. For if we require the above condition on executions to be satisfied *for every* execution of a protocol, then most protocols will fail trivially, because of the power of the adversary scheduler. In particular, there is almost always a way for the scheduler to force the protocol to break symmetry maximally, for example, to make it reach a configuration in which some state is the state of a single node, even when the protocol does not have an *inherent* mechanism of electing a unique state. Moreover, though for computability questions it is sufficient to assume that the scheduler selects in every step a single pair of nodes to interact with each other, this type of a scheduler is

problematic for estimating the symmetry of protocols. The reason is that even fundamentally parallel operations necessarily pass through a highly-symmetry-breaking step. For example, consider the rule $(a, a) \rightarrow (b, b)$ and assume that an even number of nodes are initially in state a. The goal is here for the protocol to convert all as to bs. If the scheduler could pick a perfect matching between the as, then in one step all as would be converted to bs, and additionally the protocol would never pass trough a configuration in which a state is the state of fewer than n nodes. Now, observe that the sequential scheduler can only pick a single pair of nodes in each step, so in the very first step it yields a configuration in which state b is the state of only 2 nodes. Of course, there are turnarounds to this, for example by taking into account only equal-interaction configurations, consisting of the states of the processes after all processes have participated in an equal number of interactions, still we shall follow an alternative approach that simplifies the arguments and the analysis.

In particular, we will consider schedulers that can be maximally parallel. Such a scheduler, selects in every step a matching (of any possible size) of the complete interaction graph, so, in one extreme, it is still allowed to select only one interaction but, in the other extreme, it may also select a perfect matching in a single step. Observe that this scheduler is different both from the sequential scheduler traditionally used in the area of population protocols and from the fully parallel scheduler which assumes that $\Theta(n)$ interactions occur in parallel in every step.

Finally, in order to isolate the *inherent* symmetry, i.e., the symmetry that is only due to the protocol, we shall focus on those schedules[1] that achieve as high symmetry as possible for the given protocol. Such schedules may look into the protocol and exploit its structure so that the chosen interactions maximize parallelism. It is crucial to notice that this restriction does by no means affect correctness. Our protocols are still, as usual, required to stabilize to the correct answer in *any* fair execution (and, actually, in this paper against a more generic scheduler than the one traditionally assumed). The above restriction is only a convention for estimating the *inherent* symmetry of a protocol designed to operate in an adversarial setting. On the other hand, one does not expect this *measure of inherent symmetry* to be achieved by the majority of executions. If, instead, one is interested in some *measure of the observed symmetry*, then it would make more sense to study an *expected observed symmetry* under some probabilistic assumption for the scheduler. We leave this as an interesting direction for future research (see Sect. 5 for more details on this).

For a given initial configuration, we shall estimate the symmetry breaking performed by the protocol not in any possible execution but an execution in which the scheduler tries to maximize the symmetry. In particular, we shall define the symmetry of a configuration c as the minimum count of any state present in c. Then the symmetry of an execution will be the minimum symmetry of any configuration in it. Based on these, we shall define the symmetry of a protocol $\mathcal{A}$ from initial configuration c_0 as the maximum symmetry of any fair execution starting at c_0. So, in order to lower bound by k the symmetry of a protocol on a given c_0, it will be sufficient to present *a* schedule in which the protocol stabilizes without ever "electing" fewer than k nodes. On the other hand, to establish an upper bound of h on symmetry, we will have to show that *in every* schedule (on the given c_0) the protocol "elects" at most h nodes. Then we may define the symmetry of the protocol on a set of initial configurations as the minimum of its symmetries over those initial configurations. The symmetry of a protocol (as a whole) shall be defined as a function of the symmetry of the initial configuration and is deferred to Sect. 2.

Observation 1 *The above definition leads to very strong impossibility results, as these upper bounds are also upper bounds on the observed symmetry. In particular, if we establish that the symmetry of a protocol $\mathcal{A}$ is at most h then, it is clear that under* any scheduler *the symmetry of $\mathcal{A}$ is at most h.*

Section 2 brings together all definitions and basic facts that are used throughout the paper. In Sect. 3, we give a set of positive results. The main result here is a partial characterization, showing that a wide subclass of semilinear predicates is computed with symmetry $\Theta(N_{min})$, where N_{min} is the minimum multiplicity of a state in the initial configuration, which is asymptotically optimal. Then, in Sect. 4, we study some basic predicates that seem to require much symmetry breaking. In particular, we study the *majority* and the *parity* predicates. For majority we establish that for any constant k it can be computed with symmetry at least k. For parity we prove a strong impossibility result, stating that the symmetry of any protocol that stably computes it, is upper bounded by an integer depending only on the size of the protocol (i.e., a constant, compared to the size of the system). This excludes protocols that would solve parity with symmetry depending on the symmetry of the initial configuration, but does not exclude protocols that solve it with symmetry k, for any constant k. The negative result for parity immediately generalizes to a subset of the predicates that are *not closed under doubling*. Our impossibility result implies that there exist predicates which can *only* be computed by protocols that perform some sort of leader-election (not necessarily a unique leader but at most a constant number of nodes in a distinguished leader role). In Sect. 5, we give further research directions that are opened by our work.

[1] By "schedule" we mean an "execution" throughout.

1.2 Further related work

In contrast to static systems with unique identifiers (IDs) and dynamic systems, the role of symmetry in *static anonymous systems* has been deeply investigated [8,20,26,42]. *Similarity* as a way to compare and contrast different models of concurrent programming has been defined and studied in [24]. One (restricted) type of symmetry that has been recently studied in systems with IDs, is the existence of *homonyms*, i.e., processes that are initially assigned the same ID [16]. Moreover, there are several standard models of distributed computing that do not suffer from a necessity to break symmetry globally (e.g., to elect a leader) like Shared Memory with Atomic Snapshots [1,11], Quorums [31,37,39], and the LOCAL model [36,41].

Population Protocols were originally motivated by highly dynamic networks of simple sensor nodes that cannot control their mobility. The first papers focused on the computational capabilities of the model which have now been almost completely characterized. In particular, if the interaction network is complete (as is also the case in the present paper), i.e., one in which every pair of processes may interact, then the computational power of the model is equal to the class of the *semilinear predicates* (and the same holds for several variations) [5]. Interestingly, the generic protocol of [3] that computes all semilinear predicates, elects a unique leader in every execution and the same is true for the construction in [13]. Moreover, according to [6], all known generic constructions of semilinear predicates "fundamentally rely on the election of a single initial *leader* node, which coordinates phases of computation". Moreover, it has been recently proved that such a leader-election sub-procedure is necessarily slow [requiring $\Omega(n^2)$ interactions] [19], which further highlights the usefulness of protocols that avoid it. Other works have investigated what can be achieved in terms of performance if the initial configuration already provides a unique leader (e.g., via a preprocessing step). In a very early example, Angluin et al. [4] proved that with a pre-elected unique leader any semilinear predicate can be computed in a subquadratic expected number of interactions with high probability [(in particular, in $O(n \log^5 n)$ interactions]. Another recent example exploits a pre-elected leader to count the size of the population by a protocol that always terminates and is correct with high probability [30]. Semilinearity of population protocols persists up to $o(\log \log n)$ local space but not more than this [15]. If, additionally, the connections between processes can hold a state from a finite domain, then the computational power dramatically increases to the commutative subclass of **NSPACE**(n^2) [28].

Interestingly, population protocols are a special case of *chemical reaction networks* (CRNs), which model chemistry in a *well-mixed solution*. Specifically, they correspond to the special case in which all reactions have two reactants and two products, all rate constants are 1, and the volume (which is a parameter of the model of CRNs) is equal to the number of molecules. A slight difference that remains between the models is that the CRN model uses a continuous-time Markov chain, compared to the discrete-time Markov chain of population protocols, however, the probability of various executions is identical between the models (cf., e.g., [18,38]). Moreover, the recently proposed *Network Constructors* extension of population protocols [34] is capable of constructing arbitrarily complex stable networks. Czyzowicz et al. [14] have recently studied the relation of population protocols to antagonism of species, with dynamics modeled by discrete Lotka-Volterra equations. Finally, in [12], the authors highlighted the importance of executions that necessarily pass through a "bottleneck" transition (meaning a transition between two states that have only constant counts in the population, which requires $\Omega(n^2)$ expected number of steps to occur), by proving that protocols that avoid such transitions can only compute existence predicates.

To the best of our knowledge, our type of approach, of computing predicates stably without *ever* electing a unique leader, has not been followed before in this area (according to [6], "[17] proposes a leader-less framework for population computation", but this should not be confused with what we do in this paper, as it only concerns the achievement of dropping the requirement for a *pre-elected* unique leader that was assumed in all previous results for that problem). For introductory texts to population protocols, the interested reader is encouraged to consult [9,29,35].

2 Preliminaries

A *population protocol* (PP) is a 6-tuple (X, Y, Q, I, O, δ), where X, Y, and Q are all finite sets and X is the *input alphabet*, Y is the *output alphabet*, Q is the set of *states*, $I: X \to Q$ is the *input function*, $O: Q \to Y$ is the *output function*, and $\delta: Q \times Q \to Q \times Q$ is the *transition function*.

If $\delta(a, b) = (a', b')$, we call $(a, b) \to (a', b')$ a *transition*. A transition $(a, b) \to (a', b')$ is called *effective* if $x \neq x'$ for at least one $x \in \{a, b\}$ and *ineffective* otherwise. When we present the transition function of a protocol we only present the effective transitions. The system consists of a population V of n distributed *processes* (also called *nodes*). In the generic case, there is an underlying *interaction graph* $G = (V, E)$ specifying the permissible interactions between the nodes. Interactions in this model are always pairwise. In this work, G is a *complete directed interaction graph*.

Let Q be the set of states of a population protocol $\mathcal{A}$. A configuration c of $\mathcal{A}$ on n nodes is an element of $\mathbb{N}_{\geq 0}^{|Q|}$, such that, for all $q \in Q$, $c[q]$ is equal to the number of nodes that are in state q in configuration c and it holds that $\sum_{q \in Q} c[q] = n$. For example, if $Q = \{q_0, q_1, q_2, q_3\}$ and

$c = (7, 12, 52, 0)$, then, in c, 7 nodes of the $7+12+52+0 = 71$ in total, are in state q_0, 12 nodes in state q_1, and 52 nodes in state q_2.

Execution of the protocol proceeds in discrete steps and it is determined by an *adversary scheduler* who is allowed to be *parallel*, meaning that, in every step, it may select one or more pairwise interactions (up to a maximum matching) to occur at the same time. This is an important difference from classical population protocols where the scheduler could only select a single interaction per step. More formally, in every step, a non-empty matching $(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)$ from E is selected by the scheduler and, for all $1 \leq i \leq k$, the nodes u_i, v_i interact with each other and update their states according to the transition function δ. A *fairness condition* is imposed on the adversary to ensure the protocol makes progress. An infinite execution is *fair* if for every pair of configurations c and c' such that $c \rightarrow c'$ (i.e., c can go in one step to c'), if c occurs infinitely often in the execution then so does c'.

In population protocols, we are typically interested in computing predicates on the inputs, e.g., $N_a \geq 5$, being true whenever there are at least 5 as in the input.[2] Moreover, computations are *stabilizing* and not terminating, meaning that it suffices for the nodes to eventually converge to an output that is *correct* and *stable* (i.e., that output cannot change regardless of subsequent transitions). We call *stability* the earliest configuration c in an execution, such that the output of c is stable. We say that a protocol *stably computes* a predicate if, on any population size, any input assignment, and any fair execution on these, all nodes eventually stabilize their outputs to the value of the predicate on that input assignment.

We define the *symmetry* $s(c)$ *of a configuration* c as the *minimum multiplicity of a state that is present in* c (unless otherwise stated, in what follows by "symmetry" we shall always mean "inherent symmetry"). That is, $s(c) = \min_{q \in Q\,:\, c[q] \geq 1}\{c[q]\}$. For example, if $c = (0, 4, 12, 0, 52)$ then $s(c) = 4$, if $c = (1, \ldots)$ then $s(c) = 1$, which is the minimum possible value for symmetry, and if $c = (n, 0, 0, \ldots, 0)$ then $s(c) = n$ which is the maximum possible value for symmetry. So, the range of the symmetry of a configuration is $\{1, 2, \ldots, n\}$.

Let $\mathcal{C}_0(\mathcal{A})$ be the set of all *initial configurations* for a given protocol $\mathcal{A}$. Given an initial configuration $c_0 \in \mathcal{C}_0(\mathcal{A})$, denote by $\Gamma(c_0)$ the set of all fair executions of $\mathcal{A}$ that begin from c_0, each execution being truncated to its prefix *up to stability*.[3]

Given any initial configuration c_0 and any execution $\alpha \in \Gamma(c_0)$, define the *symmetry breaking of ca on* α as the difference between the symmetry of the initial configuration of α and the minimum symmetry of a configuration of α, that is, the *maximum drop in symmetry* during the execution. Formally, $b(\mathcal{A}, \alpha) = s(c_0) - \min_{c \in \alpha}\{s(c)\}$. Also define the *symmetry of* $\mathcal{A}$ on α as $s(\mathcal{A}, \alpha) = \min_{c \in \alpha}\{s(c)\}$. Of course, it holds that $s(\mathcal{A}, \alpha) = s(c_0) - b(\mathcal{A}, \alpha)$. Moreover, observe that, for all $\alpha \in \Gamma(c_0)$, $0 \leq b(\mathcal{A}, \alpha) \leq s(c_0) - 1$ and $1 \leq s(\mathcal{A}, \alpha) \leq s(c_0)$. In several cases we shall denote $s(c_0)$ by N_{min}.

The *symmetry breaking of a protocol* $\mathcal{A}$ *on an initial configuration* c_0 can now be defined as $b(\mathcal{A}, c_0) = \min_{\alpha \in \Gamma(c_0)}\{b(\mathcal{A}, \alpha)\}$ and:

Definition 1 We define the *symmetry of* $\mathcal{A}$ on c_0 as $s(\mathcal{A}, c_0) = \max_{\alpha \in \Gamma(c_0)}\{s(\mathcal{A}, \alpha)\}$.

Remark 1 To estimate the *inherent* symmetry with which a protocol computes a predicate on a c_0, we execute the protocol against an *imaginary* scheduler who is a *symmetry maximizer*.

Now, given the set $\mathcal{C}(N_{min})$ of all initial configurations c_0 such that $s(c_0) = N_{min}$, we define the *symmetry breaking of a protocol* $\mathcal{A}$ *on* $\mathcal{C}(N_{min})$ as $b(\mathcal{A}, N_{min}) = \max_{c_0 \in \mathcal{C}(N_{min})}\{b(\mathcal{A}, c_0)\}$ and:

Definition 2 We define the *symmetry of* $\mathcal{A}$ *on* $\mathcal{C}(N_{min})$ as $s(\mathcal{A}, N_{min}) = \min_{c_0 \in \mathcal{C}(N_{min})}\{s(\mathcal{A}, c_0)\}$.

Observe again that $s(\mathcal{A}, N_{min}) = N_{min} - b(\mathcal{A}, N_{min})$ and that $0 \leq b(\mathcal{A}, N_{min}) \leq N_{min} - 1$ and $1 \leq s(\mathcal{A}, N_{min}) \leq N_{min}$.

This means that, in order to establish that a protocol $\mathcal{A}$ is at least $g(N_{min})$ symmetric asymptotically (e.g., for $g(N_{min}) = \Theta(\log N_{min})$), we have to show that for every sufficiently large N_{min}, the symmetry breaking of $\mathcal{A}$ on $\mathcal{C}(N_{min})$ is at most $N_{min} - g(N_{min})$, that is, to show that for all initial configurations $c_0 \in \mathcal{C}(N_{min})$ there exists *an* execution on c_0 that drops the initial symmetry by at most $N_{min} - g(N_{min})$, e.g., by at most $N_{min} - \log N_{min}$ for $g(N_{min}) = \log N_{min}$, or that does not break symmetry at all in case $g(N_{min}) = N_{min}$. On the other hand, to establish that the symmetry is at most $g(N_{min})$, e.g., at most 1 which is the minimum possible value, one has to show a symmetry breaking of at least $N_{min} - g(N_{min})$ on infinitely many N_{min}s.

[2] We shall use throughout the paper $N_i(t)$, for $t \geq 0$, to denote the number of nodes in state (with input) i after the tth interaction (initially, respectively). In some cases, when states are indexed by integers, we use $N_i(t)$ to refer to the number of nodes in state q_i after the tth interaction. When time is clear from context, we shall shorten these to N_i. An exception is N_{min}, which is reserved for the minimum multiplicity of a state in the initial configuration (i.e., "min" is not an input/state, but stands for "minimum").

[3] In this work, we only require protocols to preserve their symmetry *up to stability*. This means that a protocol is allowed to break symmetry arbitrarily after stability, e.g., even elect a unique leader, without having to pay for it. We leave as an interesting open problem the comparison of this convention to the apparently harder requirement of maintaining symmetry forever.

Table 1 Summary of the main definitions

Notion	Definition
$c[q]$	Number of nodes that are in state q in configuration c
$\mathbf{x}$, $\mathbf{c}$	Vector notation of an input assignment x and a configuration c, respectively
Symmetry $s(c)$ of a configuration c	$s(c) = \min_{q \in Q\,:\, c[q] \geq 1}\{c[q]\}$
$\Gamma(c_0)$	Set of all fair executions of a protocol that begin from initial configuration c_0
Symmetry $s(\mathcal{A}, \alpha)$ *of protocol* $\mathcal{A}$ *on* $\alpha \in \Gamma(c_0)$	$s(\mathcal{A}, \alpha) = \min_{c \in \alpha}\{s(c)\}$
Symmetry $s(\mathcal{A}, c_0)$ *of protocol* $\mathcal{A}$ *on initial configuration* c_0	$s(\mathcal{A}, c_0) = \max_{\alpha \in \Gamma(c_0)}\{s(\mathcal{A}, \alpha)\}$
N_{min}	Minimum multiplicity of a state in the initial configuration
$\mathcal{C}(N_{min})$	Set of all initial configurations c_0 such that $s(c_0) = N_{min}$
Symmetry $s(\mathcal{A}, N_{min})$ *of protocol* $\mathcal{A}$ *on* $\mathcal{C}(N_{min})$	$s(\mathcal{A}, N_{min}) = \min_{c_0 \in \mathcal{C}(N_{min})}\{s(\mathcal{A}, c_0)\}$

Table 1 summarizes the main notions defined in this section, to facilitate future reference by the reader.

3 Predicates of high symmetry

In this section, we try to identify predicates that can be stably computed with much symmetry. We first give an indicative example (partly serving as an illustration of the main definitions in action) and then we generalize to arrive at a partial characterization of the predicates that can be computed with maximum symmetry.

3.1 An example: count-to-x

The *Count-to-x* protocol (see Protocol 1, which is essentially a generalization of the *Count-to-5* of [3]) computes the predicate $N_1 \geq x$, which is true iff the number of 1s in the input is at least x. Initially, each node obtains an input from $\{0, 1\}$ and translates this to an initial state from $\{q_0, q_1\}$, respectively. The goal of the protocol is to aggregate a sum of at least x to some node if there are initially at least x 1s, and to never manage to do so if there are initially less than x 1s. This is implemented by having two interacting nodes in q_i, q_j update their states to q_{i+j}, q_0 (i.e., one keeps the sum of the indices and the other nothing), respectively, if the sum is less than x, and both to q_x if the sum is at least x. State q_x is an *alert* state, which is the only state that outputs 1 and which, once produced, floods the population via a 1-way epidemic (we call such states *disseminating* in Sect. 4.2).

Protocol 1 *Count-to-x*

$X = \{0, 1\}$, $Q = \{q_0, q_1, q_2, \ldots, q_x\}$
$I(\sigma) = q_\sigma$, for all $\sigma \in X$
$O(q_x) = 1$ and $O(q) = 0$, for all $q \in Q \backslash \{q_x\}$
δ:

$(q_i, q_j) \to (q_{i+j}, q_0)$, if $i + j < x$
$\to (q_x, q_x)$, otherwise

Proposition 1 *The symmetry of Protocol* Count-to-x *(Protocol 1), for any* $x = O(1)$, *is at least* $(2/3)\lfloor N_{min}/x \rfloor - (x - 1)/3$, *when* $x \geq 2$, *and* N_{min}, *when* $x = 1$; *i.e., it is* $\Theta(N_{min})$ *for any* $x = O(1)$.[4]

Proof Recall that $N_1(0)$ denotes the initial number of q_1s, and let us abbreviate this by N_1 throughout this proof. The scheduler[5] partitions the q_1s into $\lfloor N_1/x \rfloor$ groups of x q_1s each, possibly leaving an incomplete group of $r \leq x - 1$ q_1s residue. Then, in each complete group, it performs a sequential gathering of $x - 3$ other q_1s to one of the nodes, which will go through the states $q_1, q_2, \ldots, q_{x-1}$. The same gathering is performed in parallel to all groups, so every state that exists in one group will also exist in every other group, thus, its cardinality never drops below $\lfloor N_1/x \rfloor$. In the end, at step t, there are many q_0s, $N_{x-1}(t) = \lfloor N_1/x \rfloor$, and $N_1(t) = \lfloor N_1/x \rfloor + r$, where $0 \leq r \leq x - 1$ is the residue of q_1s. That is, in all configurations so far, the symmetry has not dropped below $\lfloor N_1/x \rfloor$.

Now, we cannot pick, as a symmetry maximizing choice of the scheduler, a perfect bipartite matching between the q_1s and the q_{x-1}s converting them all to the alarm state q_x, because this could possibly leave the symmetry-breaking residue of q_1s. What we can do instead, is to match in one step as many as we can so that, after the corresponding transitions, $N_x(t') \geq N_1(t')$ is satisfied. In particular, if we match y of the (q_1, q_{x-1}) pairs we will obtain $N_x(t') = 2y$, $N_{x-1}(t') = \lfloor N_1/x \rfloor - y$, and $N_1(t') = \lfloor N_1/x \rfloor - y + r$ and what we want is

[4] Throughout this paper, $x = O(1)$ is used to emphasize that $x \in \mathbb{N}$ is a constant which is independent of the size of the system (implying that it is also independent of N_{min}).

[5] Always meaning the *imaginary symmetry-maximizing scheduler* when lower-bounding the symmetry.

$$2y \geq \lfloor N_1/x \rfloor - y + r \Rightarrow 3y \geq \lfloor N_1/x \rfloor + r \Rightarrow$$
$$y \geq \frac{\lfloor N_1/x \rfloor + r}{3},$$

which means that if we match approximately $1/3$ of the (q_1, q_{x-1}) pairs then we will have as many q_x as we need in order to eliminate all q_1s in one step and all remaining q_{x-1}s in another step.

The minimum symmetry in the whole course of this schedule is

$$N_{x-1}(t') = \lfloor N_1/x \rfloor - y = \lfloor N_1/x \rfloor - \frac{\lfloor N_1/x \rfloor + r}{3}$$
$$= \frac{2}{3}\lfloor N_1/x \rfloor - \frac{r}{3} \geq \frac{2}{3}\lfloor N_1/x \rfloor - \frac{x-1}{3}.$$

So, we have shown that if there are no q_0s in the initial configuration, then the symmetry breaking of the protocol on the schedule defined above is at most $N_{min} - ((2/3)\lfloor N_1/x \rfloor - (x-1)/3) = N_{min} - ((2/3)\lfloor N_{min}/x \rfloor - (x-1)/3)$. Next, we consider the case in which there are some q_0s in the initial configuration. Observe that in this protocol the q_0s can only increase, so their minimum cardinality is precisely their initial cardinality N_0. Consequently, in case $N_0 \geq 1$ and $N_1 \geq 1$, and if $N_{min} = \min\{N_0, N_1\}$, the symmetry breaking of the schedule defined above is $N_{min} - \min\{N_0, N_{x-1}(t')\}$. If, for some initial configuration, $N_0 \geq N_{x-1}(t')$ then the symmetry breaking is $N_{min} - N_{x-1}(t') \leq N_{min} - ((2/3)\lfloor N_1/x \rfloor - (x-1)/3)$. This gives again $N_{min} - ((2/3)\lfloor N_{min}/x \rfloor - (x-1)/3)$, when $N_1 \leq N_0$, and less than $N_{min} - ((2/3)\lfloor N_{min}/x \rfloor - (x-1)/3)$, when $N_1 > N_0 = N_{min}$. If instead, $N_0 < N_{x-1}(t') < N_1$, then, in this case, the symmetry breaking is $N_{min} - \min\{N_0, N_{x-1}(t')\} = N_0 - N_0 = 0$. Finally, if $N_0 = n$, then the symmetry breaking is 0. We conclude that for every initial configuration, the symmetry breaking of the above schedule is at most $N_{min} - N_{x-1}(t') \leq N_{min} - ((2/3)\lfloor N_{min}/x \rfloor - (x-1)/3)$, for all $x \geq 2$, and 0, for $x = 1$. Therefore, the symmetry of the *Count-to-x* protocol is at least $(2/3)\lfloor N_{min}/x \rfloor + (x-1)/3 = \Theta(N_{min})$, for $x \geq 2$, and N_{min}, for $x = 1$. □

3.2 A general positive result

Theorem 1 *Any predicate of the form $\sum_{i\in[k]} a_i N_i \geq c$, for integer constants $k \geq 1$, $a_i \geq 1$, and $c \geq 0$, can be computed with symmetry more than $\lfloor N_{min}/(c/\sum_{j\in L} a_j + 2) \rfloor - 2 = \Theta(N_{min})$.*[6]

[6] The predicates captured by Theorem 1 form a large subset of what are know as *threshold predicates* (cf. [3]). In particular, their subset in which $a_i \geq 1$ and $c \geq 0$.

Protocol 2 *Positive-Linear-Combination*

$Q = \{q_0, q_1, q_2, \ldots, q_c\}$
$I(\sigma_i) = q_{a_i}$, for all $\sigma_i \in X$
$O(q_c) = 1$ and $O(q) = 0$, for all $q \in Q \backslash \{q_c\}$
δ:

$(q_i, q_j) \to (q_{i+j}, q_0)$, if $i + j < c$
$\to (q_c, q_c)$, otherwise

Proof We begin by giving a parameterized protocol (Protocol 2) that stably computes any such predicate, and then we shall prove that the symmetry of this protocol is the desired one.

Take now any initial configuration C_0 on n nodes and let $L \subseteq [k]$ be the set of indices of the initial states that are present in C_0. Let also q_{min} be the state with minimum cardinality, N_{min}, in C_0. Construct $\lfloor N_{min}/x \rfloor$ groups, by adding to each group $x = \lceil c/\sum_{j\in L} a_j \rceil$ copies of each initial state. There is always a way to construct these groups, because for any partitioning of nodes in q_{min} into groups there is an equivalent partitioning of a subset of the nodes in any initial state $q \neq q_{min}$ present in C_0, as $N_q(0) \geq N_{min}$. Observe that each group has total sum $\sum_{j\in L} a_j x = x \sum_{j\in L} a_j = \lceil c/\sum_{j\in L} a_j \rceil (\sum_{j\in L} a_j) \geq c$. Moreover, state q_{min} has a residue r_{min} of at most x and every other state q_i has a residue $r_i \geq r_{min}$. Finally, keep $y = \lceil (N_{min} + r_{min})/(x+1) \rceil - 1$ of those groups and drop the other $\lfloor N_{min}/x \rfloor - y$ groups making their nodes part of the residue, which results in new residue values $r'_j = x(\lfloor N_{min}/x \rfloor - y) + r_j$, for all $j \in L$. It is not hard to show that $y \leq r'_j$, for all $j \in L$.

We now present a schedule that achieves the desired symmetry. The schedule consists of two phases, the *gathering* phase and the *dissemination* phase. In the dissemination phase, the schedule picks a node of the same state from every group and starts aggregating to that node the sum of its group sequentially, performing the same in parallel in all groups. It does this until the alarm state q_c first appears. When this occurs, the dissemination phase begins. In the dissemination phase, the schedule picks one after the other all states that have not yet been converted to q_c. For each such state q_i, it picks a q_c which infects one after the other (sequentially) the q_is, until $N_c(t) \geq N_i(t)$ is satisfied for the first time. Then, in a single step that matches each q_i to a q_c, it converts all remaining q_is to q_c.

We now analyze the symmetry breaking of the protocol in this schedule. Clearly, the initial symmetry is N_{min}. As long as a state appears in the groups, its cardinality is at least y, because it must appear in each one of them. When a state q_i first becomes eliminated from the groups, its cardinality is equal to its residue r'_i. Thus, so far, the minimum cardinality of a state is

$$\min\{y, \min_{j\in L} r'_j\} = y = \left\lceil \frac{N_{min} + r_{min}}{x+1} \right\rceil - 1 > \left\lfloor \frac{N_{min}}{c/\sum_{j\in L} a_j + 2} \right\rfloor - 2.$$

It follows that the maximum symmetry breaking so far is less than $N_{min} - \left\lfloor \frac{N_{min}}{c/\sum_{j\in L} a_j + 2} \right\rfloor + 2$.

Finally, we must also take into account the dissemination phase. In this phase, the q_cs are $2y$ initially and can only increase, by infecting other states, until they become n and the cardinalities of all other states decrease until they all become 0. Take any state $q_i \neq q_c$ with cardinality $N_i(t)$ when the dissemination phase begins. What the schedule does is to decrement $N_i(t)$, until $N_c(t') \geq N_i(t')$ is first satisfied, and then to eliminate all occurrences of q_i in one step. Due to the fact that N_i is decremented by one in each step resulting in a corresponding increase by one of N_c, when $N_c(t') \geq N_i(t')$ is first satisfied, it holds that $N_i(t') \geq N_c(t') - 1 \geq N_c(t) - 1 \geq 2y - 1 \geq y$ for all $y \geq 1$, which implies that the lower bound of y on the minimum cardinality, established for the gathering phase, is not violated during the dissemination phase.

We conclude that the symmetry of the protocol in the above schedule is more than $\lfloor N_{min}/(c/\sum_{j\in L} a_j + 2)\rfloor - 2$. □

4 Harder predicates

In this section, we study the symmetry of predicates that, in contrast to single-signed linear combinations, do not allow for "output-honest" states (i.e., states that whenever they appear their output determines the output of the whole computation; see Sect. 4.2 for formal definitions). In particular, we focus on linear combinations containing mixed signs, like the *majority* predicate, and also on modulo predicates like the *parity* predicate. Recall that these predicates are not captured by the lower bound on symmetry of Theorem 1.

4.1 Bounds for mixed coefficients

We begin with a proposition stating that the majority predicate can be computed with symmetry that depends on the difference of the state-cardinalities in the initial configuration.

Proposition 2 *The majority predicate* $N_a - N_b > 0$ *can be computed with symmetry* $\min\{N_{min}, |N_a - N_b|\}$, *where* $N_{min} = \min\{N_a, N_b\}$.

Proof Initially, a node is in $(l, 1)$ if its input is a and in $(l, -1)$ if its input is b. The description of the protocol is given in Protocol 3.

Protocol 3 *Majority*

$Q = \{l, f\} \times \{-1, 1\}$
$I(a) = (l, 1)$ and $I(b) = (l, -1)$
$O(\cdot, -1) = 0$ and $O(\cdot, 1) = 1$
δ:

$(l, i), (l, j) \to (f, -1), (f, -1)$, if $i + j = 0$
$(l, i), (f, j) \to (l, i), (f, i)$
$(f, j), (l, i) \to (f, i), (l, i)$
$(f, -1), (f, 1) \to (f, -1), (f, -1)$
$(f, 1), (f, -1) \to (f, -1), (f, -1)$

We first argue about the correctness of the protocol. Initially all nodes are l-leaders and l-leaders can only decrease via an interaction between an $(l, 1)$ and an $(l, -1)$, in which case both become followers in state $(f, -1)$. The only things that followers do is to copy the data bit of the leaders (provided that at least one leader still exists) and to let the data bit -1 dominate a disagreement between two of them. Moreover, as long as there are at least two leaders with opposite data bits, due to fairness, an interaction between them will eventually occur. It follows that eventually, $\min\{N_a, N_b\}$ such eliminations will have occurred leaving $2 \cdot \min\{N_a, N_b\}$ followers and $n - 2 \cdot \min\{N_a, N_b\}$ leaders. All leaders will have data bit 1 in case as are the majority, data bit -1 in case bs are the majority, while there will be no leaders in case none of the two is a strict majority. In the first case, all followers will eventually copy 1, thus, all nodes will stabilize their output to 1. Observe that the 1 of a follower may change many times to -1 due to its interactions with other followers that have not yet set their data bit to 1, still fairness guarantees that eventually the unique (continuously reachable) stable configuration in which all followers have switched to 1 after interacting with the leaders will occur. In the second case, all followers will eventually copy -1, thus, all nodes will stabilize their output to 0 and in the third case there are only followers, so the data bit -1 eventually dominates due to the last two rules of the protocol and eventually all nodes will stabilize their output to 0. In summary, if the as form a strict majority all nodes stabilize to 1, otherwise all nodes stabilize to 0, thus, Protocol 3 stably computes the majority predicate.

For symmetry, consider first those initial configurations which satisfy $|N_a - N_b| \geq \min\{N_a, N_b\}$. Consider the schedule that matches $\min\{N_a, N_b\}$ leaders with opposite data bits in its first step, leaving $|N_a - N_b|$ leaders agreeing on the majority (i.e., in the same state) and $2 \cdot \min\{N_a, N_b\}$ followers in state $(f, -1)$. Up to this point, there is no symmetry breaking because the minimum cardinality that has appeared is still the initial minimum $\min\{N_a, N_b\}$. Next, the scheduler matches in one step $\min\{N_a, N_b\}$ followers to leaders and then in another step the rest $\min\{N_a, N_b\}$ followers to leaders, which leads to a stable configuration in which all fol-

lowers have their output agree with the data bit of the leaders. As the minimum cardinality never has fallen below the initial minimum $\min\{N_a, N_b\}$, the symmetry is in this case at least $\min\{N_a, N_b\}$.

Next, consider those initial configurations which satisfy $|N_a - N_b| < \min\{N_a, N_b\}$. Again, in the first step all opposite data bits are matched leaving $2 \cdot \min\{N_a, N_b\}$ followers and $|N_a - N_b| \geq 0$ leaders. Observe that if $|N_a - N_b| = 0$ then the configuration is already stable without any symmetry breaking. If $|N_a - N_b| \geq 1$, then the scheduler goes on by matching in one step $|N_a - N_b|$ followers to the leaders. Then it picks a leader and converts sequentially from the remaining followers, until precisely $|N_a - N_b|$ of them remain. Those are then converted in one step by being matched to the leaders. The minimum cardinality of a state in this schedule is $|N_a - N_b|$ and the initial minimum is $\min\{N_a, N_b\}$, so the symmetry breaking is $\min\{N_a, N_b\} - |N_a - N_b|$ and the symmetry is on those initial configurations $|N_a - N_b|$. □

Still, as we prove in the following theorem, it is possible to do better in the worst case, and achieve any desired constant symmetry.

Theorem 2 *For every constant $k \geq 1$, the majority predicate $N_a - N_b > 0$ can be computed with symmetry k.*

Proof The idea is to multiply both N_a and N_b by k so that their difference becomes $k|N_a - N_b|$. In this manner, the difference will become at least k (in absolute value) whenever there is a strict majority which can be exploited for computation with symmetry k. Fortunately, multiplying both N_a and N_b by $k \geq 1$ does not affect the value of the majority predicate, but only the winning difference.

To this end, we set the state of a node initially to (l, k) if its input is a and to $(l, -k)$ if its input is b. The definition of the protocol is given in Protocol 3.

For correctness, initially all nodes are l-leaders. Now, l-leaders, apart from decreasing as in Protocol 3, can also increase. This occurs whenever an (l, i), with $|i| \geq 2$, meets a follower, in which case the follower becomes a leader taking one unit of the other leader's count. Still, as in Protocol 3, as long as there are at least two leaders with opposite data bits, due to fairness, an interaction between two such leaders will eventually occur. Eventually, all leaders will have a positive data bit in case as are the majority, and a non-positive in case as are not the majority. From that point on, no leader can change its output and all followers will eventually copy this output.

For symmetry, in case $N_a = N_b$ the scheduler can pick a perfect bipartite matching between the (l, k)s and the $(l, -k)$s to convert them to $(f, 0)$ and, thus, stabilize to output 0 without any symmetry breaking. The case $N_b > N_a$ is simpler than the $N_a > N_b$ because the default output of the followers is 0, while in the $N_a > N_b$ case there is a small additional difficulty due to the fact that all $(f, 0)$s have to be converted to $(f, 1)$s. So, w.l.o.g. we focus on the $N_a > N_b$ case and we only give a proof for the special case in which $N_a = N_b + 1$ as the other cases are similar.

So, assume that $N_a = N_b + 1$. The construction requires that $n \geq 2k(k+1)$. The initial configuration consists of $N_a = N_b + 1$ nodes in state (l, k) and N_b nodes in $(l, -k)$. We present a schedule with symmetry k. The scheduler first picks a matching between (l, k)s and $(l, -k)$s of size $\lceil k/2 \rceil$. This introduces k copies of state $(f, 0)$ and leaves $N_b - \lceil k/2 \rceil$ nodes in state $(l, -k)$ and $N_b - \lceil k/2 \rceil + 1$ nodes in state (l, k). Now isolate $k+1$ nodes in state (l, k) and k nodes in state $(l, -k)$. The remaining nodes in states (l, k) and $(l, -k)$ (equal of each) are $n - k - (k+1) - k \geq 2k(k+1) - 3k - 1 = 2k^2 - k - 1$. These are converted to $(f, 0)$s in one step, so we now have the initial k $(f, 0)$s, the new $(f, 0)$s that are at least $2k^2 - k - 1$, $k+1$ (l, k)s, and k $(l, -k)$s. Together the (l, k)s and $(l, -k)$s hold a total count of $k(k+1) + k^2 = 2k^2 + k$ and together the new $(f, 0)$s, the (l, k)s, and the $(l, -k)$s are at least $2k^2 - k - 1 + (k+1) + k =$

Protocol 4 *k-Symmetry-Majority*

$Q = \{l, f\} \times \{-k, -(k-1), \ldots, 0, \ldots, k-1, k\}$
$I(a) = (l, k)$ and $I(b) = (l, -k)$
$O(\cdot, j) = 0$, for all $-k \leq j \leq 0$ and $O(\cdot, i) = 1$, for all $0 < i \leq k$
δ:

// leaders with opposite-signed values interact: one
// keeps the sum, the other becomes a follower with
// value 1 if the sum is positive and 0 otherwise
$(l, i), (l, j) \to (l, i + j), (f, 1)$, if i, j have opposite signs and $i + j > 0$
$\to (l, i + j), (f, 0)$, if i, j have opposite signs and $i + j < 0$
$\to (f, 0), (f, 0)$, if $i + j = 0$
// a leader with value ≥ 2 distributes one unit to a
// follower and turns that follower into a leader
$(l, i), (f, \cdot) \to (l, i - 1), (l, 1)$, if $i \geq 2$
$(f, \cdot), (l, i) \to (l, 1), (l, i - 1)$, if $i \geq 2$
// a leader with value ≤ -2 distributes one unit to a
// follower and turns that follower into a leader
$(l, j), (f, \cdot) \to (l, j + 1), (l, -1)$, if $j \leq -2$
$(f, \cdot), (l, j) \to (l, -1), (l, j + 1)$, if $j \leq -2$
// for the rest of the values (i.e., in $\{-1, 0, 1\}$)
// followers simply copy the value of a leader
$(l, i), (f, \cdot) \to (l, i), (f, i)$, if $i \in \{-1, 0, 1\}$
$(f, \cdot), (l, i) \to (f, i), (l, i)$, if $i \in \{-1, 0, 1\}$
// by default the 0 output dominates between followers
// but this affects neither correctness nor symmetry
$(f, 0), (f, 1) \to (f, 0), (f, 0)$
$(f, 1), (f, 0) \to (f, 0), (f, 0)$

$2k^2+k$ nodes. So, there are enough nodes (the initial $(f,0)$s excluded) to distribute on them the count as follows. First the scheduler picks the (l,k)s and matches them in one step to $k+1$ nodes in $(f,0)$. This leaves $k+1$ nodes in $(l,k-1)$ and $k+1$ nodes in $(l,1)$. Then it matches the $(l,k-1)$s to $(f,0)$s, introducing $k+1$ more $(l,1)$s and leaving $k+1$ nodes in $(l,k-2)$. It continues in the same way until no count is greater than 1, in this way distributing the counts of the $k+1$ nodes in (l,k) to $k(k+1)$ copies of $(l,1)$. Observe that during this process the initial (l,k)s are always in identical states going in parallel through the sequence of states $(l,k),(l,k-1),(l,k-2),\ldots,(l,1)$, so each state on them is the state of $k+1$ other nodes. Moreover, their first matching with $(f,0)$s introduces $k+1$ $(l,1)$s in one step and from that point on (during this particular process) $(l,1)$s only increase, so the cardinality of $(l,1)$s does not go below $k+1$. Next (or in parallel), it does the same with the k nodes in $(l,-k)$ leaving k^2 copies of $(l,-1)$. Observe that even though $(f,0)$s decrease during these processes, their cardinality never goes below k due to the initial set of k $(f,0)$s. So, at this point there are at least k $(f,0)$s, k^2+k $(l,1)$s, and k^2 $(l,-1)$, while the minimum multiplicity of a state has never dropped below k. The scheduler now matches in one step all the $(l,-1)$s to $(l,1)$s leaving in the population at least k^2+k $(f,0)$s and precisely k $(l,1)$s. Then, the scheduler matches all $(l,1)$s to $(f,0)$s, thus, introducing in one step k $(f,1)$s (and still having at least k^2 $(f,0)$s), and then picks an $(l,1)$ and starts converting sequentially $(f,0)$s to $(f,1)$s until precisely k $(f,0)$s have remained. Finally, it matches the remaining k $(f,0)$s to the k $(l,1)$s to convert all $(f,0)$s to $(f,1)$s in one step. At this point the protocol has stabilized and the multiplicity of no state has ever dropped below k. □

4.2 Output-honest states

In order to arrive at a strong impossibility result (presented in Sect. 4.3), we start by highlighting the role of *output-honest* states in symmetric computations. Informally, a state $q\in Q$ is called *output-honest* if its appearance in an execution guarantees that the output value $O(q)$ must be the output value of the execution. More formally, if q is output-honest and C is a configuration containing q, then the set of outputs of C' must contain $O(q)$, for all C' such that $C\rightsquigarrow C'$, where '$\rightsquigarrow$' means *reaches in one or more steps*. Moreover, if all executions under consideration stabilize to an agreement, meaning that eventually all nodes stabilize to the same output, then the above implies that if an execution ever reaches a configuration containing q, then the output of that execution is necessarily $O(q)$.

A state q is called *reachable* if there is an initial configuration C_0 and an execution on C_0 that can produce q. We can also define reachability just in terms of the protocol, under the assumption that if $Q_0\subseteq Q$ is the set of initial states, then any possible combination of cardinalities of states from Q_0 can be part of an initial configuration. A *production tree* for a state $q\in Q$, is a directed binary in-tree with its nodes labeled from Q, such that its root has label q, if a is the label of an internal node (the root inclusive) and b, c are the labels of its children, then the protocol has a rule of the form $\{b,c\}\to\{a,\cdot\}$ (that is, a rule producing a by an interaction between a b and a c in any direction),[7] and any leaf is labeled from Q_0. Observe now that if a path from a leaf to the root repeats a state a, then we can always replace the subtree of the highest appearance of a by the subtree of the lowest appearance of a on the path and still have a production tree for q. This implies that if q has a production tree, then q also has a production tree of depth at most $|Q|$, that is, a production tree having at most $2^{|Q|-1}$ leaves, which is a constant number, when compared to the population size n, that only depends on the protocol. Now, we can call a state q *reachable* (by a protocol $\mathcal{A}$) if there is a production tree for it. These are summarized in the following proposition.

Proposition 3 *Let $\mathcal{A}$ be a protocol, C_0 be any (sufficiently large) initial configuration of $\mathcal{A}$, and $q\in Q$ any state that is reachable from C_0. Then there is an initial configuration C_0' which is a sub-configuration of C_0 of size $n'\leq 2^{|Q|-1}$ such that q is reachable from C_0'.*

Proposition 3 is crucial for proving negative results, and will be invoked in Sect. 4.3.

Proposition 4 *Let p be a predicate. There is no protocol that stably computes p (all nodes eventually agreeing on the output in every fair execution), having both a reachable output-honest state with output 0 and a reachable output-honest state with output 1.*

Proof Let $\mathcal{A}$ be such a protocol and let q and q' be the two reachable output-honest states, such that $O(q)=0$ and $O(q')=1$. As both q and q' are reachable, there are initial configurations $\mathbf{c}_0$ and $\mathbf{c}_1$ (where $\mathbf{c}$ is just the vector notation for a configuration), such that $\mathbf{c}_0\xrightarrow{*}\mathbf{c}$ and $\mathbf{c}_1\xrightarrow{*}\mathbf{c}'$, where $\mathbf{c}$ contains q and $\mathbf{c}'$ contains q'. Observe now that $\mathbf{c}_0+\mathbf{c}_1$ is also a valid initial configuration and by additivity of reachability we get that $\mathbf{c}_0+\mathbf{c}_1\xrightarrow{*}\mathbf{c}+\mathbf{c}_1\xrightarrow{*}\mathbf{c}+\mathbf{c}'$. But the configuration $\mathbf{c}+\mathbf{c}'$ contains both q and q', which, by the definition of output-honest states, is a contradiction. □

An output-honest state q is called *disseminating* if $\{x,q\}\to(q,q)$, for all $x\in Q$.

Proposition 5 *Let $\mathcal{A}$ be a protocol with at least one reachable output-honest state, that stably computes a predicate p*

[7] Whenever we use an unordered pair in a rule, like $\{b,c\}$, we mean that the property under consideration concerns both (b,c) and (c,b).

and let $Q_s \subseteq Q$ be the set of reachable output-honest states of $\mathcal{A}$. Then there is a protocol $\mathcal{A}'$ with a reachable disseminating state that stably computes p.

Proof We first show how to construct $\mathcal{A}'$ from $\mathcal{A}$. Pick a single state $q \in Q_s$. Replace any occurrence of a $q' \in Q_s$ in the transition function δ by q, eliminate duplicate rules (as we may create many copies of the same rule by the previous replacements, it is sufficient to keep only one of those copies), and remove from Q all $q' \in Q_s \backslash \{q\}$. Finally, replace any rule $(x, y) \to (z, w)$, where $x = q$, $y = q$, $z = q$, or $w = q$ by the rule $(x, y) \to (q, q)$. This completes the construction of $\mathcal{A}'$. State q is a disseminating state of $\mathcal{A}'$ because, by the last step of the construction, it holds that $\{x, q\} \to (q, q)$ for all $x \in Q$. Moreover, q is reachable because every $q' \in Q_s$ is reachable in $\mathcal{A}$, q inclusive, and the above construction has only positively affected the reachability of q.

It remains to show that $\mathcal{A}'$ stably computes p. As $\mathcal{A}$ stably computes p, it suffices to show that when the two protocols are executed on the same schedule (including the choice of the initial configuration) their stable outputs are the same. Take any schedule in which $\mathcal{A}$ produces a $q' \in Q_s$ and consider the first time t that this happens. As q' is output-honest, the stable output of $\mathcal{A}$ on this schedule must be $O(q')$. Consider now $\mathcal{A}'$ on the same schedule. Before step t, the executions of $\mathcal{A}'$ and $\mathcal{A}$ must be equivalent, because the construction has only affected rules containing at least one output-honest state. At step t, $\mathcal{A}'$ produces its disseminating state q, thus, its output is $O(q) = O(q')$ (the fact that $q, q' \in Q_s \Rightarrow O(q) = O(q')$ follows from Proposition 4), so the outputs of the two protocols agree on schedules producing output-honest states. Finally, for any schedule in which $\mathcal{A}$ does not produce an output-honest state, the executions of $\mathcal{A}'$ and $\mathcal{A}$ are equivalent on this schedule, thus, again their outputs agree. □

Theorem 3 *Let $\mathcal{A}$ be a protocol with a reachable disseminating state q and let $\mathcal{C}_0^d$ be the subset of its initial configurations that may produce q. Then the symmetry of $\mathcal{A}$ on $\mathcal{C}_0^d$ is $\Theta(N_{min})$.*

Theorem 3 emphasizes the fact that disseminating states can be exploited for maximum symmetry. We have omitted its proof, because it is similar to the proofs of Proposition 1 and Theorem 1. This lower bound on symmetry immediately applies to single-signed linear combinations (where passing a threshold can safely result in the appearance of a disseminating state, because there are no opposite-signed numbers to inverse the process), thus, it can be used as an alternative way of arriving at Theorem 1. On the other hand, the next proposition shows that this lower bound does not apply to linear combinations containing mixed signs, because protocols for them cannot have output-honest states.

Proposition 6 *Let p be a predicate of the form $\sum_{i\in[k]} a_i N_i \geq c$, for integer constants $k \geq 1$, a_i, and $c \geq 0$ such that at least two a_is have opposite signs. Then there is no protocol, having a reachable output-honest state, that stably computes p.*

Proof Let $\mathcal{A}$ be such a protocol and let $q \in Q$ be a reachable output-honest state of $\mathcal{A}$. Take an initial configuration C_0 that can produce q. As q is output-honest, it must hold that the value of the predicate on C_0 is equal to $O(q)$. If $O(q) = 1$ then C_0 must satisfy $\sum_{i\in[k]} a_i N_i \geq c$. Construct now another initial configuration C_0' by adding to C_0 as many nodes in a negative-coefficient initial state as required to violate $\sum_{i\in[k]} a_i N_i \geq c$. The value of the predicate p on C_0' is equal to 0 but q can still be produced on the C_0 subconfiguration of C_0' implying that $\mathcal{A}$'s output on C_0' is 1. The latter violates the fact that $\mathcal{A}$ stably computes p. If, instead, $O(q) = 0$, then we can obtain a similar contradiction by adding a sufficient number of positive-coefficient nodes to C_0. □

Remark 2 Proposition 6 is also true for all "fluctuating" *modulo predicates*. In particular, a modulo predicate is a predicate of the form $\sum_{i\in[k]} a_i N_i \equiv c \pmod m$, for integer constants $k \geq 1$, a_i, c, and $m \geq 2$. We call a modulo predicate $\sum_{i\in[k]} a_i N_i \equiv c \pmod m$ *fluctuating* if (1) for all input assignments x such that $\sum_{i\in[k]} a_i x_i \equiv c \pmod m$, $\exists$ an input assignment y such that $\sum_{i\in[k]} a_i (x_i + y_i) \not\equiv c \pmod m$ and (2) for all input assignments x' such that $\sum_{i\in[k]} a_i x_i' \not\equiv c \pmod m$, $\exists$ an input assignment y' such that $\sum_{i\in[k]} a_i (x_i' + y_i') \equiv c \pmod m$. Intuitively, these are predicates in which no output value ever dominates as the input vector $\mathbf{x}$ (formed by the N_is) increases. Observe that the *parity* predicate, defined as $N_1 \equiv 1 \pmod 2$, is a fluctuating modulo predicate (as any time we add another 1 to the input we change the value of the predicate), therefore no protocol with a reachable output-honest state can compute it.

4.3 Predicates that cannot be computed with high symmetry

We now prove a strong impossibility result, establishing that there are predicates that cannot be stably computed with much symmetry. The result concerns the *parity* predicate, defined as $n \bmod 2 = 1$. In particular, all nodes obtain the same input, e.g., 1, and, thus, all begin from the same state, e.g., q_1. So, in this case, $N_{min} = n$ in every initial configuration, and we can here estimate symmetry as a function of n. The parity predicate is true iff the number of nodes is odd. So, whenever n is odd, we want all nodes to eventually stabilize their outputs to 1 and, whenever it is even, to 0. If symmetry is not a constraint, then there is a simple protocol, proposed in [2], that solves the problem. All nodes begin as leaders with data bit 1, i.e., in state $(l, 1)$. Whenever two leaders (l, i) and (l, j) interact, one of them remains a

leader in $(l, (i+j) \bmod 2)$ and the other becomes a follower in $(f, (i+j) \bmod 2)$. Followers can never become leaders again and only copy the data bit of the leaders they interact with. Observe that $n \bmod 2$ is always preserved as the sum of the leaders' data bits, that a unique leader is guaranteed to remain eventually with its data bit being equal to $n \bmod 2$, and that all followers will eventually copy the unique leader's data bit. In case this data bit is equal to 1, all output 1 and, in case it is equal to 0, all output 0. Unfortunately, not only this particular strategy, but any possible strategy for the problem, cannot achieve symmetry more than a constant that depends on the size of the protocol, as we shall now prove.

Theorem 4 *Let $\mathcal{A}$ be a protocol with set of states Q, that solves the parity predicate. Then the symmetry of $\mathcal{A}$ is less than $2^{|Q|-1}$.*

Proof For the sake of contradiction, assume $\mathcal{A}$ solves parity with symmetry $f(n) \geq 2^{|Q|-1}$. Take any initial configuration C_n for any sufficiently large odd n (e.g., $n \geq f(n)$ or $n \geq |Q| \cdot f(n)$, or even larger if required by the protocol). By definition of symmetry, there is an execution α on C_n that reaches stability without ever dropping the minimum cardinality of an existing state below $f(n)$. Call C_{stable} the first output-stable configuration of α. As n is odd, C_{stable} must satisfy that all nodes are in states giving output 1 and that no execution on C_{stable} can produce a state with output 0. Moreover, due to the facts that $\mathcal{A}$ has symmetry $f(n)$ and that α is an execution that achieves this symmetry, it must hold that every $q \in Q$ that appears in C_{stable} has multiplicity $C_{stable}[q] \geq f(n)$.

Consider now the initial configuration C_{2n}, i.e., the unique initial configuration on $2n$ nodes. Observe that now the number of nodes is even, thus, the parity predicate evaluates to false and any fair execution of $\mathcal{A}$ must stabilize to output 0. Partition C_{2n} into two equal parts, each of size n. Observe that each of the two parts is equal to C_n. Consider now the following possible finite prefix β of a fair execution on C_{2n}. The scheduler simulates in each of the two parts the previous execution α up to the point that it reaches the configuration C_{stable}. So, the prefix β takes C_{2n} to a configuration denoted by $2C_{stable}$ and consisting precisely of two copies of C_{stable}. Observe that $2C_{stable}$ and C_{stable} consist of the same states with the only difference being that their multiplicity in $2C_{stable}$ is twice their multiplicity in C_{stable}. A crucial difference between C_{stable} and $2C_{stable}$ is that the former is output-stable while the latter is not. In particular, any fair execution of $\mathcal{A}$ on $2C_{stable}$ must produce a state q_0 with output 0. But, by Proposition 3, q_0 must also be reachable from a sub-configuration C_{small} of $2C_{stable}$ of size at most $2^{|Q|-1}$. So, there is an execution γ restricted on C_{small} that produces q_0.

Observe now that C_{small} is also a sub-configuration of C_{stable}. The reason in that (i) every state in C_{small} is also a state that exists in $2C_{stable}$ and, thus, also a state that exists in C_{stable} and (ii) the multiplicity of every state in C_{small} is restricted by the size of C_{small}, which is at most $2^{|Q|-1}$, and every state in C_{stable} has multiplicity at least $f(n) \geq 2^{|Q|-1}$, that is, C_{stable} has sufficient capacity for every state in C_{small}. But this implies that if γ is executed on the sub-configuration of C_{stable} corresponding to C_{small}, then it must produce q_0, which contradicts the fact that C_{stable} is output-stable with output 1. Therefore, we conclude that $\mathcal{A}$ cannot have symmetry at least $f(n) \geq 2^{|Q|-1}$. □

Remark 3 Theorem 4 constrains the symmetry of any correct protocol for parity to be upper bounded by a constant that depends on the size of the protocol. Still, it does not exclude the possibility that parity is solvable with symmetry k, for any constant $k \geq 1$. The reason is that, for any constant $k \geq 1$, there might be a protocol with $|Q| > k$ (or even $|Q| \gg k$) that solves parity and achieves symmetry k, because $k < 2^{|Q|-1}$, which is the upper bound on symmetry proved by the theorem. On the other hand, the $2^{|Q|-1}$ upper bound of Theorem 4 excludes any protocol that would solve parity with symmetry depending on N_{min}.

We now strengthen the negative result of Theorem 4 by generalizing it to a subset of the predicates that are *not closed under doubling*. A predicate p is *closed under doubling* if for all $\mathbf{x} \in \mathbb{N}^k$, it holds that $p(\mathbf{x}) = p(2\mathbf{x})$ (where $\mathbf{x}$ is again the vector notation for an input assignment); see, e.g., [12]. We call a predicate p, *asymptotically* (w.r.t. symmetry) *not closed under doubling*, if for all $N_{min} \in \mathbb{N}$ there is a $N'_{min} \in \mathbb{N}$, $N'_{min} \geq N_{min}$, and an $\mathbf{x} \in \mathbb{N}^k$ with symmetry at least N'_{min}, such that $p(\mathbf{x}) \neq p(2\mathbf{x})$.

Corollary 1 *Let $\mathcal{A}$ be a protocol with set of states Q, that computes a predicate p that is asymptotically not closed under doubling. Then the symmetry of $\mathcal{A}$ is less than $2^{|Q|-1}$.*

5 Further research

In this work, we managed to obtain a first partial characterization of the predicates with symmetry $\Theta(N_{min})$, to exhibit a predicate (parity) that resists any non-constant symmetry, and to generalize the latter negative result to the asymptotically not closed under doubling predicates. The obvious next goal is to arrive at an exact characterization of the allowable symmetry of all semilinear predicates. An obvious first interesting open question to this end is: *"Is there a predicate closed under doubling (or even one that just does not satisfy the asymptotically not closed under doubling condition) for which we can prove that it cannot be computed with symmetry more than a constant?"*. It would be also interesting to look for predicates whose symmetry is upper bounded by 1, as these would mean that a leader *must* be

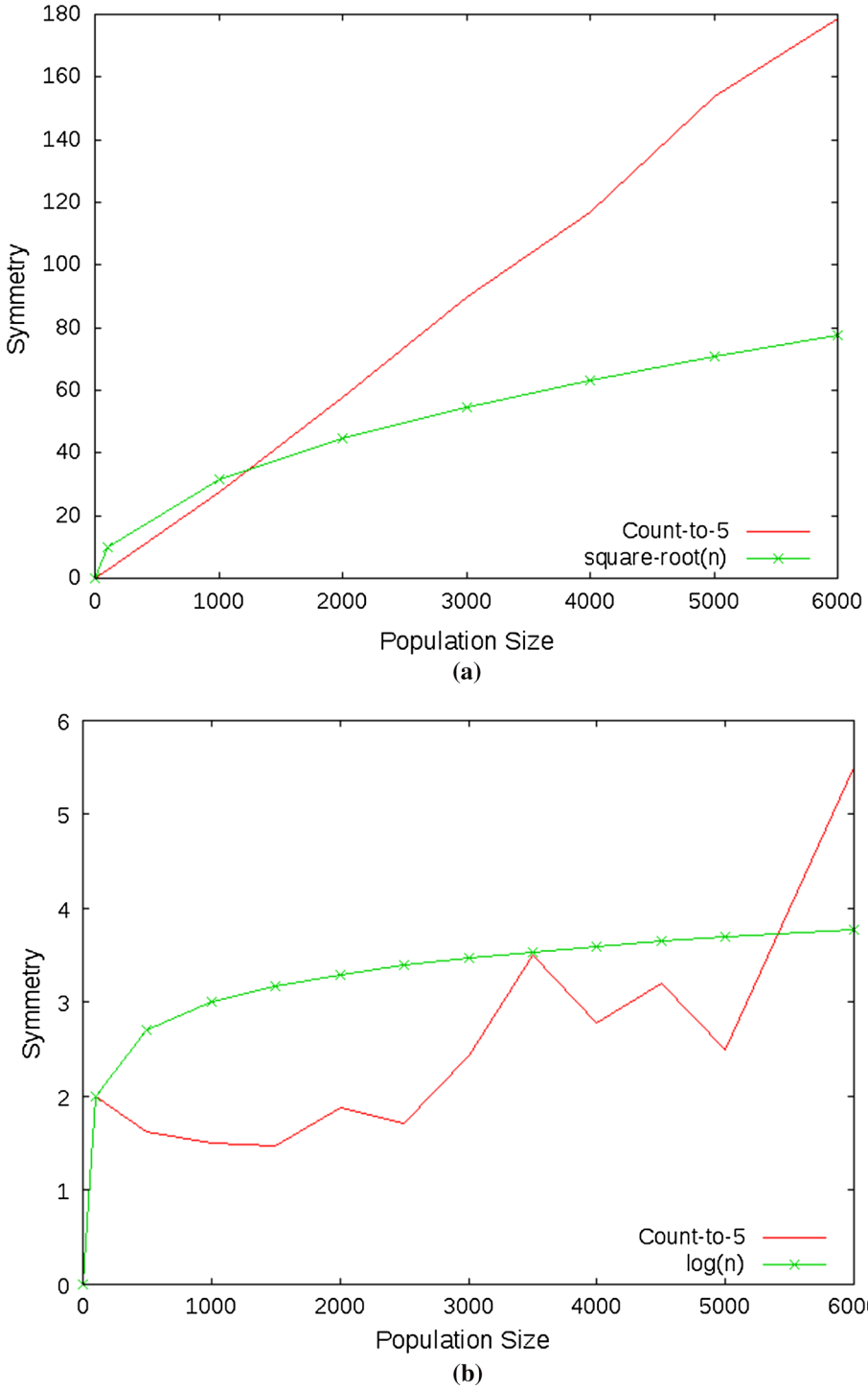

Fig. 1 The experiments were performed with the NETCS simulator [7]. The scheduler selects in every step a maximum cardinality matching, uniformly at random from all maximum matchings of the complete interaction graph. The implemented protocol is the *Count-to-x* protocol of Sect. 3.1, for $x = 5$. In a fraction of the populations of size up to 6000 nodes, several repetitions were performed and the average observed symmetry achieved by the protocol is plotted. The initial configuration is always the one resulting from assigning to all nodes the input value 1. (a) The observed symmetry of *Count-to-5* (red normal line) is calculated up to the point that the alert state q_5 first becomes an absolute majority in the population and seems to grow faster than $\sqrt{n}$ (green marked line). (b) The observed symmetry of *Count-to-5* (red normal line) is calculated up to stability and seems to grow as fast as $\log n$ (green marked line)

elected in order to compute the predicate. *"Is this, for example, true for all asymptotically not closed under doubling predicates or even any of them, e.g., parity?"*. Similarly, *"Is it possible that there is a protocol whose symmetry is exactly* $k = O(1), e.g.3$?". Moreover, it would be worth analyzing the symmetry of protocols that compute functions instead of predicates that we studied in this paper. Several examples of protocols that compute functions appear, e.g., in [17]. Another interesting next goal would be to extend our definitions of symmetry to stronger variants of population protocols, like Mediated Population Protocols [28], Network Constructors [34], Community Protocols [23], protocols with larger local memories [15], Absence Detectors [32], and Clocked Population Protocols [10]. Finally, it is possible that

there is a deeper connection between our approach and the approach followed by Chen et al. in [12]. Both papers are intuitively concerned with avoiding "low count states". In this paper we have mostly focused on avoiding the *existence* of low count states, whereas Chen et al. focused on avoiding *transitions between* "low count states". The main impossibility result of Chen et al. applies also to initial configurations with a leader, whereas our definition of symmetry makes such configurations immediately disqualifying for any "highly symmetric" protocol. There is also another result in [12] (Lemma 4.13) about protocols that initially have no leader. There could be a connection based on the following idea: if a protocol has no "low count state" initially, and also avoids "bottleneck" transitions, then any states that do end up with "low counts" can be pushed to count 0 (this is in fact how the proof of [19] goes). Perhaps one could use similar arguments to establish that the resulting executions (if some care is taken) are symmetric as well.

Another question concerns the parity predicate, but could possibly apply to other modulo or asymptotically not closed under doubling predicates as well. Some preliminary results of ours indicate that some amount of symmetry for parity (greater than 1 but, of course, in light of Theorem 4, still constant) can be achieved if the initial configuration has a sufficient number of *auxiliary nodes* in a distinct state q_0. It seems interesting to study how is symmetry affected by auxiliary nodes and whether they can be totally avoided.

Another very challenging direction for further research, concerns networked systems (either static or dynamic) in which the nodes have memory and possibly also unique IDs. Even though the IDs provide an a priori maximum symmetry breaking, still, solving a task and avoiding the process of "electing" one of the nodes may be highly non-trivial. But in this case, defining the role of a process as its complete local state is inadequate. As already discussed in Sect. 1, there are other plausible ways of defining the role of a process, but which one is best-tailored for such systems is still unclear and needs further investigation.

Finally, recall that in this work we focused on the *inherent* symmetry of a protocol as opposed to its *observed* symmetry. One way to study the observed symmetry would be to consider *random parallel schedulers*, like the one that selects in every step a maximum matching uniformly at random from all such matchings. Then we may ask *"What is the average symmetry achieved by a protocol under such a scheduler?"*. In some preliminary experimental results of ours, the expected observed symmetry of the *Count-to-5* protocol (i) if counted until the alert state q_5 becomes an absolute majority in the population, seems to grow faster than $\sqrt{n}$ and (ii) if counted up to stability, seems to grow as fast as $\log n$ (see Fig. 1).

Acknowledgements We would like to thank Dimitrios Amaxilatis for setting up and running, in the NETCS simulator of population protocols and network constructors [7], the experiments for the evaluation of the expected observed symmetry of the *Count-to-5* protocol. Finally, we would like to thank the anonymous reviewers of this article and of its preliminary version. Their thorough reading and comments have helped us to improve our work substantially.

References

1. Afek, Y., Attiya, H., Dolev, D., Gafni, E., Merritt, M., Shavit, N.: Atomic snapshots of shared memory. J. ACM (JACM) **40**(4), 873–890 (1993)
2. Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Computation in networks of passively mobile finite-state sensors. In: 23rd annual ACM Symposium on Principles of Distributed Computing (PODC), pp. 290–299. ACM (2004)
3. Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Computation in networks of passively mobile finite-state sensors. Distrib. Comput. **18**(4), 235–253 (2006)
4. Angluin, D., Aspnes, J., Eisenstat, D.: Fast computation by population protocols with a leader. Distrib. Comput. **21**(3), 183–199 (2008)
5. Angluin, D., Aspnes, J., Eisenstat, D., Ruppert, E.: The computational power of population protocols. Distrib. Comput. **20**(4), 279–304 (2007)
6. Alistarh, D., Gelashvili, R.: Polylogarithmic-time leader election in population protocols. In: 42nd International Colloquium on Automata, Languages and Programming (ICALP), Lecture Notes in Computer Science, vol. 9135, pp. 479–491. Springer (2015)
7. Amaxilatis, D., Logaras, M., Michail, O., Spirakis, P.G.: NETCS: A new simulator of population protocols and network constructors. arXiv preprint arXiv:1508.06731 (2015)
8. Angluin, D.: Local and global properties in networks of processors. In: Proceedings of the 12th Annual ACM Symposium on Theory of Computing (STOC), pp. 82–93. ACM (1980)
9. Aspnes, J., Ruppert, E.: An introduction to population protocols. In: Garbinato, B., Miranda, H., Rodrigues, L. (eds.) Middleware for Network Eccentric and Mobile Applications, pp. 97–120 (2009)
10. Aspnes, J.: Clocked population protocols. In: 36th ACM Symposium on Principles of Distributed Computing (PODC), pp. 431–440. ACM (2017)
11. Attiya, H., Welch, J.: Distributed Computing: Fundamentals, Simulations, and Advanced Topics, vol. 19. Wiley, Hoboken (2004)
12. Chen, H.-L., Cummings, R., Doty, D., Soloveichik, D.: Speed faults in computation by chemical reaction networks. In: Proceedings of the 28th International Symposium on Distributed Computing (DISC), Lecture Notes in Computer Science, vol. 8784, pp. 16–30 (2014) (**also in Distributed Computing**)
13. Chen, H.-L., Doty, D., Soloveichik, D.: Deterministic function computation with chemical reaction networks. Nat. Comput. **13**(4), 517–534 (2014)
14. Czyzowicz, J., Gasieniec, L., Kosowski, A., Kranakis, E., Spirakis, P.G., Uznański, P.: On convergence and threshold properties of discrete Lotka–Volterra population protocols. In: 42nd International Colloquium on Automata, Languages and Programming (ICALP), pp. 393–405. Springer (2015)

15. Chatzigiannakis, I., Michail, O., Nikolaou, S., Pavlogiannis, A., Spirakis, P.G.: Passively mobile communicating machines that use restricted space. Theor. Comput. Sci. **412**(46), 6469–6483 (2011)
16. Delporte-Gallet, C., Fauconnier, H., Guerraoui, R., Kermarrec, A.-M., Ruppert, E., Tran, H.: The Byzantine agreement with homonyms. In: Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing (PODC), pp. 21–30. ACM (2011)
17. Doty, D., Hajiaghayi, M.: Leaderless deterministic chemical reaction networks. Nat. Comput. **14**(2), 213–223 (2015)
18. Doty, D.: Timing in chemical reaction networks. In: Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 772–784 (2014)
19. Doty, D., Soloveichik, D.: Stable leader election in population protocols requires linear time. In: Proceedings of the 29th International Symposium on Distributed Computing (DISC), Lecture Notes in Computer Science, vol. 9363, pp. 602–616. Springer (2015)
20. Flocchini, P., Mans, B., Santoro, N.: Sense of direction: definitions, properties, and classes. Networks **32**(3), 165–180 (1998)
21. Förster, K.-T., Seidel, J., Wattenhofer, R.: Deterministic leader election in multi-hop beeping networks. In: Proceedings of the 28th International Symposium on Distributed Computing (DISC), Lecture Notes in Computer Science, vol. 8784, pp. 212–226. Springer (2014)
22. Ghaffari, M., Haeupler, B.: Near optimal leader election in multi-hop rast eddio networks. In: Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 748–766. SIAM (2013)
23. Guerraoui, R., Ruppert, E.: Names trump malice: tiny mobile agents can tolerate byzantine failures. In: 36th International Colloquium on Automata, Languages and Programming (ICALP), LNCS, vol. 5556, pp. 484–495. Springer (2009)
24. Johnson, R.E., Schneider, F.B.: Symmetry and similarity in distributed systems. In: Proceedings of the 4th annual ACM Symposium on Principles of Distributed Computing (PODC), pp. 13–22. ACM (1985)
25. Kuhn, F., Lynch, N., Oshman, R.: Distributed computation in dynamic networks. In: Proceedings of the 42nd ACM symposium on Theory of computing (STOC), pp. 513–522. ACM (2010)
26. Kranakis, E.: Symmetry and computability in anonymous networks: a brief survey. In: Proceedings of the 3rd International Conference on Structural Information and Communication Complexity, pp. 1–16 (1997)
27. Lynch, N.A.: Distributed Algorithms, 1st edn. Morgan Kaufmann, Burlington (1996)
28. Michail, O., Chatzigiannakis, I., Spirakis, P.G.: Mediated population protocols. Theor. Comput. Sci. **412**(22), 2434–2450 (2011)
29. Michail, O., Chatzigiannakis, I., Spirakis, P.G.: New models for population protocols. In: Lynch, N.A. (ed.) Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool, Williston (2011)
30. Michail, O.: Terminating distributed construction of shapes and patterns in a fair solution of automata. Distrib. Comput. 1–23 (2017). https://doi.org/10.1007/s00446-017-0309-z
31. Malkhi, D., Reiter, M.K., Wool, A., Wright, R.N.: Probabilistic quorum systems. Inf. Comput. **170**(2), 184–206 (2001)
32. Michail, O., Spirakis, P.G.: Terminating population protocols via some minimal global knowledge assumptions. J. Parallel Distrib. Comput. **81**, 1–10 (2015)
33. Michail, O., Spirakis, P.G.: How many cooks spoil the soup? In: Proceedings of the 23rd International Colloquium on Structural Information and Communication Complexity (SIROCCO), pp. 3–18. Springer (2016)
34. Michail, O., Spirakis, P.G.: Simple and efficient local codes for distributed stable network construction. Distrib. Comput. **29**(3), 207–237 (2016)
35. Michail, O., Spirakis, P.G.: Elements of the Theory of Dynamic Networks. Commun. ACM (2017). https://livrepository.liverpool.ac.uk/3006836/ (**to appear**)
36. Peleg, D.: Distributed Computing: A Locality-Sensitive Approach. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, Philadelphia (2000)
37. Peleg, D., Wool, A.: The availability of quorum systems. Inf. Comput. **123**(2), 210–223 (1995)
38. Soloveichik, D., Cook, M., Winfree, E., Bruck, J.: Computation with finite stochastic chemical reaction networks. Nat. Comput. **7**(4), 615–633 (2008)
39. Skeen, D.: A quorum-based commit protocol. Technical report, Cornell University (1982)
40. George, W., Sparks, J.: The Writings of George Washington. American Stationers' Company, Boston (1840)
41. Suomela, J.: Survey of local algorithms. ACM Comput. Surv. (CSUR) **45**(2), 24 (2013)
42. Yamashita, M., Kameda, T.: Computing on anonymous networks. I. Characterizing the solvable cases. IEEE Trans. Parallel Distrib. Syst. **7**(1), 69–89 (1996)