# Space Complexity of Self-Stabilizing Leader Election in Passively-Mobile Anonymous Agents

Shukai Cai Taisuke Izumi Koichi Wada

Nagoya Institute of Technology

February 11, 2009

**Abstract**

A population protocol is one of distributed computing models for passively-mobile systems, where a number of agents change their states by pairwise interactions between two agents. In this paper, we investigate the solvability of the self-stabilizing leader election in population protocols without any kind of oracles. We identify the necessary and sufficient condition to solve the self-stabilizing leader election in population protocols from the aspects of local memory complexity and fairness assumptions. This paper shows that under the assumption of global fairness, no protocol using only $n-1$ states can solve the self-stabilizing leader election in complete interaction graphs, where $n$ is the number of agents in the system. To prove this impossibility, we introduce a novel proof technique, called closed-set argument. In addition, we propose a self-stabilizing leader election protocol using $n$ states that works even under the unfairness assumption. This protocol requires the exact knowledge about the number of agents in the system. We also show that such knowledge is necessary to construct any self-stabilizing leader election protocol.

## 1 Introduction

A *passively-mobile* system is a collection of agents that move in a certain region but have no control over how they move. Since the communication range of each agent is quite small compared to the size of the region, two agents can communicate only when they are sufficiently close to each other. Passive mobility appears in many real systems. A representative example is a network of smart sensors attached to cars or animals. In addition, a certain kind of natural computing, such as synthesis of chemical materials and complex biosystems, can be included in passively-mobile systems by regarding chemical interactions as communications. While these systems are different in the view of applications, all of them aim to a common goal, that is, how to organize and manipulate computing entities that are uncontrollable in the sense of mobility. Then, it is reasonable to think about some common principles underlying them. Revealing such principles from the aspect of theoretical computer science is an interesting and worthwhile challenge.

Recently, as a model for such passively-mobile systems, *population protocols* are introduced [1, 2, 7]. A population protocol consists of a number of agents, to which some program (protocol) is deployed. Following the deployed protocol, each agent changes its state by *pairwise interactions* to other agents (that is, two agents come closer to each other in the region and update their states by exchanging information). Typically, the capability of each agent is limited. It is often assumed that each agent has only constant-space memory and no identifier. A population protocol is a good abstraction that captures the feature of passively-mobile systems in spite of its mathematical simplicity. Therefore, in the last few years, it is receiving much attention among the community of the distributed computing [3, 4, 5, 6, 8, 10].

Population protocols are originated by Angluin et al. [1], which investigates a class of predicates that can be computed autonomously over population protocols. Its primary result is that any predicate in *semilinear* class (which includes the comparison, modulo and threshold predicates) can be computed on population protocols by proposing a protocol that stably computes any semilinear predicate. In the following paper [4], it is also shown that any computable predicate by population protocols belongs to semilinear, that is, semilinear is the necessary and sufficient class of the predicates that can be computed on population protocols of all-pairs interaction graphs (complete interaction graphs).

The protocols proposed in the above paper are assumed to start from a properly-formed system configuration. In this sense, it is not a *self-stabilizing* protocol: Self-stabilization is one of the desirable properties of distributed computations, which ensures that the system necessarily converges to the desired behavior regardless of its initial configuration. Self-stabilization on population protocols is considered in a number of previous papers [2, 9, 11], which have investigated the solvability of the *self-stabilizing leader election* (SS-LE) problems under some kinds of assumptions. The general model of population protocols introduces an *interaction graph*, which specifies the possibility of communication between two agents. The above papers show the solvability and unsolvability of SS-LE for specific classes of interaction graphs such as complete graphs, rings, rooted trees, directed acyclic graphs, and so on. Unfortunately, it is easily shown that SS-LE is almost impossible in general. Thus, the above papers also consider some additional (but reasonable) assumptions to make SS-LE solvable by introducing several notions extending the computational power of population protocols: *global fairness* and *leader detector* oracle $\Omega?$. Intuitively, global fairness guarantees the occurrence of any possible transition and thus it prevents livelock caused by some looped execution. The leader detector oracle is an abstracted virtual device that informs the existence and inexistence of a leader to all the agents in the system. Both of the assumptions give some additional computational power to population protocols, which is sufficient to solve SS-LE in some cases, but insufficient in some other cases. However, the complete characterization of system assumptions making SS-LE solvable is unknown. Currently, only a few results about the solvability of SS-LE on the complete interaction graphs are known:

1. Assuming global fairness and the oracle $\Omega?$, there exists an SS-LE protocol where each agent uses only one bit of memory [11].

2. Under the assumption of unfairness and no oracle, no uniform protocol can solve SS-LE, where "uniform protocol" means the one that works correctly on the system with arbitrary number of agents (that is, uniform protocols do not use any information about the total number of agents) [2].

3. Without $\Omega?$, any protocol using only one bit of memory cannot solve SS-LE even if we assume global fairness [9].

In this paper, we also investigate the solvability of SS-LE on population protocols. In particular, we are interested in self-stabilizing leader election protocols in complete interaction graphs without oracles. The primary contribution of our work is to identify the necessary and sufficient conditions such that SS-LE becomes solvable from the aspects of local memory space and fairness assumptions. More precisely, this paper shows the following three results:

1. Without oracles, there is no deterministic or probabilistic SS-LE protocol using only $n-1$ states of memory even if we assume global fairness, where $n$ is the number of agents in the system.

2. There exists an SS-LE protocol that uses $n$ states ($\lceil \log_2 n \rceil$ bits) of memory and correctly works under the unfairness assumption.

3. Even if we assume global fairness, without oracles, there is no uniform SS-LE protocol in the strong sense. That is, any SS-LE protocol working correctly on the population of $n$ agents does not work on the population of $n-1$ agents.

The third result implies that the upper bound for the number of agents is not sufficient knowledge to design SS-LE protocols, and thus it justifies the fact that the exact value of $n$ is necessary to construct the protocol shown in the second possibility result. It should be also noted that the first impossibility result is quite nontrivial and interesting. Global fairness is reasonable but sufficiently strong so that it can break essential ideas leading previous impossibility results. Actually, under the global fairness assumption, we cannot apply many of existing techniques to prove the impossibility. In this paper, we resolve such difficulty by introducing a novel proof technique based on *closed sets*. Our key idea is to identify the set of states that never creates the leader state. While this paper utilizes this technique to show the impossibility of SS-LE, we believe that it can be applied to more broader cases, including other problems and other graph classes, to prove the impossibility under the global fairness assumption. Moreover, we can show that the three results are all correct for both the traditional two-way protocol and the one-way protocol (the two-way protocol allows that two agents can change both of their states in the interaction, but the one-way one does not). That is, the impossibility result holds in the stronger two-way protocol and the possibility result even holds in the weaker one-way protocol.

### 1.1 Related Work

Leader election on population protocols are first introduced in [3]. In [2], a non-uniform population protocol is given to solve the self-stabilizing leader election problem in directed rings of odd size under the assumption of global fairness. The authors also show that there is no uniform self-stabilizing leader election protocol for any non-simple class of interaction graphs, where a class $C$ is non-simple if any graph in $C$ can be partitioned into two subgraphs belonging to $C$.

In [11], Fischer and Jiang introduce *eventual leader detector* $\Omega?$ to realize uniform self-stabilizing leader election protocols. They give a uniform SS-LE protocol for complete graphs under the weaker fairness assumption than global one (it is called *local fairness* and the most usual assumption of fairness) using only 1 bit of memory, and a uniform self-stabilizing leader election protocol for directed rings under the assumption of global fairness. It is also shown that there exists no uniform self-stabilizing leader election protocol for directed rings under the local fairness assumption. All the above results are obtained with the help of $\Omega?$.

Canepa and Gradinariu [9] investigates the feasibility of one-bit protocols: They give a uniform one-bit SS-LE protocol for rooted trees and acyclic graphs with only one sink-node. Also they give a probabilistic protocol for arbitrary graphs under local fairness. Moreover, they prove $\Omega?$ is necessary to realize uniform one-bit SS-LE protocols for any class of interaction graphs. All the results in the paper are under the assumption of using 1 bit of memory and with the help of $\Omega?$.

## 2 Model and Definitions

We introduce the formal definitions of population-protocol considered in this paper.

A population consists of $n$ agents, which can change their own states by interacting with each other. In the general model of population protocols, all pairs of agents do not necessarily have direct interactions. The possibility of direct interactions between two agents is specified by interaction graphs: An interaction graph G=(V, E) is a simple directed graph where each vertex, labeled by $v_1, v_2, v_3, \cdots$, corresponds to each agent. The edge from $v_i$ to $v_j$ implies that the agent corresponding to $v_i$ can interact to the agent for $v_j$, where $v_i$ is the *initiator* and $v_j$ is the *responder*. Throughout this paper, we assume that the interaction graph is complete. That is, any pair of agents is possible to interact with each other. For convenience, we use undirected complete graphs for the bidirectional completed graphs in what follows.

A *protocol* $P = (Q, \delta)$ is a pair of a finite set $Q$ of *states* and a *transition function* $\delta$ that maps each pair of states $Q \times Q$ to a nonempty subset of $Q \times Q$. The transition function, and the protocol, is *deterministic* if $\delta(p, q)$ always contains just one pair of states. Otherwise the protocol is called a *probabilistic* protocol. For convenience, in this paper, we only consider deterministic protocols, and thus we simplify the definition of a transition function to a mapping $\delta : Q \times Q \to Q \times Q$ (i.e., the states after each transition is uniquely determined). If the two agents involved in an interaction can learn the states of each other and change their states depending on the state of the other, we call the protocol a *two-way* one. By contrast, if the initiator has no chance to change its state and only the responder can change its state after an interaction, we call the protocol a *one-way* one. In the one-way protocol, for any transition $r : (p, q) \to (p', q')$, $p' = p$ for any $p \in Q$. Notice that a transition does not necessarily cause either of the nodes to change its state. That is, a transition $(p, q) \to (p, q)$ is possible. We define *silent* transitions as ones that do not change any state. The transition that is not silent is said to be *active*.

From the definition of the two-way and the one-way protocols, it is obviously that the one-way protocol is a special case of the two-way one. Thus the computational power of the one-way protocol is not stronger than the computational power of the two-way one. More precisely, the one-way protocol is also correct for the two-way one, and an unsolvable problem for the two-way protocol is still unsolvable for the one-way protocol.

Formally, a configuration $C$ is an $n$-tuple $(q_1, q_2, q_3, \cdots, q_n)$ of states where each entry $q_k$ corresponds to the state of the agent $v_k$. The state of an agent $v_k$ at the configuration $C$ is denoted by $C(v_k)$. Letting $C$ be a configuration, and $r$ be a transition that maps $(p, q)$ to $(p', q')$, we say that $r$ is *enabled* in $C$ if there exists an edge $(v_i, v_j)$ such that $C(v_i) = p$ and $C(v_j) = q$. Then, we say that $C$ can go to $C'$ via $r$, denoted by $C \xrightarrow{r} C'$, if $C'$ is the configuration that is obtained by changing the states of $v_i$ and $v_j$ to $p'$ and $q'$, respectively. We simply say that $C$ can go to $C'$, denoted $C \to C'$, if $C \xrightarrow{r} C'$ holds for some transition $r$. We define executions in population protocols as follows:

**Definition 1 (Execution)** Letting $P = (Q, \delta)$ be a protocol, an *execution* of $P$ is an infinite sequence of configurations and transitions $C_0, r_0, C_1, r_1, \cdots$ satisfying

1. for each $i$, $r_i$ is a transition of $\delta$ and $C_i \xrightarrow{r_i} C_{i+1}$, $i = 0, 1, \cdots$ holds , and
2. $r_i$ is active for infinitely many $i$ unless all the enabled transitions are silent.

Notice that the second condition ensures the progress of protocols (i.e., it excludes the meaningless executions such that only silent transitions appear).

## 2.1 Fairness Assumption

Fairness is an assumption that restricts the behavior of systems. Formally, it is defined as a constraint for executions. In this paper, we introduce the following fairness assumptions [11]:

**Definition 2 (Global fairness assumption $\mathbb{G}$)** An execution $E = C_0, r_0, C_1, r_1, \cdots$ is globally fair: for every $C$ and $C'$ such that $C \to C'$, if $C = C_i$ for infinitely many $i$, then $C_i = C$ and $C_{i+1} = C'$ for infinitely many $i$.

Intuitively, global fairness guarantees the possibility of the occurrence of any possible execution and thus it prevents the occurrence of livelock caused by some looped execution.

By contrast to global fairness, the assumption called local fairness is usually used. The local fairness only guarantees that each transition can be taken infinitely often is actually taken infinitely often.

In addition to the above, we also define *unfairness assumption* $\mathbb{U}$, which requires no assumption to executions. Given a protocol $P$ and a fairness assumption $X \in \{\mathbb{G}, \mathbb{U}\}$, we define $\mathcal{E}_X(P)$ be the set of all executions of $P$ satisfying the fairness assumption $X$.

### 2.2 Self-stabilization, Legitimate Configurations

Self-Stabilizing protocols guarantee the convergence to their desired behavior starting from any initial configuration. In this paper, we consider the self-stabilizing leader election over populations, which requires that the system eventually reach a *legitimate configuration*, where exactly one process keeps a special state, called *leader state*, and no other leader state is generated in any following execution. Formally, the self-stabilizing leader election problem is defined as follows:

**Definition 3 (Self-stabilizing leader election)** A protocol $P$ solves the self-stabilizing leader election under the fairness assumption $X$ if there is one special state $s$ and any execution $E$ in $\mathcal{E}_X(P)$ satisfies that there exist some $i$ and $v_k$ such that for any $j \geq i$ and $h \neq k$ , $C_j(v_k) = s$ and $C_j(v_h) \neq s$ hold.

## 3 Impossibility of Self-Stabilizing Leader Election Using $n-1$ States

In this section, we will show that without the help of Ω?, any self-stabilizing leader election two-way protocol is impossible in a complete network graph under global fairness using only distinct $n-1$ states.

### 3.1 Difficulty of Proving Impossibility under Global Fairness

In this subsection, we explain why it is a quite nontrivial and difficult task to prove impossibility under global fairness. We show that existing techniques used to prove the impossibility do not work under the global fairness assumption.

Roughly speaking, most of existing impossibility proofs for SS-LE are roughly divided into two types: One is the argument by *illegal loop*, and the other one is that by *partition.* We explain the details for both of them:

**Illegal loop argument:** The key idea of the illegal loop argument is to find a looped execution including a non-legitimate configuration. The infinite execution repeating the loop never converges to legitimate configurations, which contradicts the self-stabilization property. This kind of arguments is widely used in almost all areas of distributed computation. However, it cannot be applied to prove the impossibility under global fairness because the global fairness assumption does not allow the system to periodically repeat the same behavior: If the system does such looped behavior, any configuration in the loop appears infinitely often. Then, under global fairness, it is necessarily guaranteed that the system could escape from the looped execution if there exists a transition which can lead the system to exit from the looped execution.

**Partition argument:** Partition argument is the technique using the fact that it is difficult to break a certain kind of symmetry. The basic idea of the partition argument is to divide a given n-node interaction graph into two same subgraphs with size $n/2$ (in general, division to three or more subgraphs can be considered). By their symmetry, it is possible to show the existence of the execution that converges to the configuration where the two subgraphs independently and separately elect a leader respectively. Thus, it contradicts the uniqueness of leaders. However, this argument can be applied only to the case of uniform protocols because non-uniform protocols do not guarantee to elect one leader in the divided subgraph (that is, it is not guaranteed that the protocol works correctly on $n/2$ agents). Moreover, to make an execution where two subgraphs independently elect a leader respectively, we have to prohibit the interactions between the two subgraphs. However, if some interaction is enabled on an edge that joints two subgraphs infinitely often, it must occur necessarily under global fairness. We cannot eliminate the possibility that such interaction breaks the symmetry, and the system converges to the legitimate configuration.

To circumvent the problems which the above two arguments hold, in the following subsection, we newly introduce a proof technique based on closed sets. Intuitively, the closed set argument finds a set of states such that the interactions between any pair of two states in the set create no state out of the set. The key of our proof is to find a closed set excluding the leader state and obtain a contradiction.

## 3.2 Impossibility Using $n-1$ States

First, we introduce several notions necessary for the following proofs.

For convenience of the proof for the impossibility, we extend the definition of a configuration. A configuration $C$ is an n-tuple of states of agents or $\perp$, where $\perp$ is a special value that masks the state of the corresponding agent. For example, $C = (\perp, q_2, q_3, \cdots, q_n)$ is also a configuration. The size of a configuration $C$ is the number of non-$\perp$ values appearing in $C$, and it is denoted by $|C|$. A *subconfiguration* $C'$ of a configuration $C$ is an $n$-tuple obtained by replacing several entries in $C$ by $\perp$. For example, letting $C = (a, b, d, e)$ be a configuration, $C'_1 = (a, \perp, d, e)$, $C'_2 = (\perp, \perp, d, \perp)$, and $C'_3 = (\perp, b, d, \perp)$ are subconfigurations of $C$ whose sizes are 3, 1, and 2, respectively. In addition, $C'_2$ is also a subconfiguration of $C'_1$ and $C'_2$ itself, but $C'_3$ is not a subconfiguration of $C'_1$.

A *trace* is a sequence of transitions. We say a trace $T = r_1, r_2, \cdots, r_i$ is *applicable* to a configuration $C_0$ if there exists a sequence of configurations $C_0, C_1, \cdots, C_i$ such that $C_0 \xrightarrow{r_1} C_1 \xrightarrow{r_2} C_2 \xrightarrow{r_3} \cdots \xrightarrow{r_i} C_i$. We define the length of a trace $T$ as the number $i$ of transitions appearing in $T$. For a configuration $C$ and a trace $T$ applicable to $C$, we define $\sigma_T(C)$ as the configuration resulted by applying $T$ to $C$. If $C' = \sigma_T(C)$ holds, we often use the notation $C \xrightarrow{T} C'$.

A configuration $C'$ is *reachable* from a configuration $C$, denoted by $C \xrightarrow{*} C'$, if there exists a trace $T$ such that $C \xrightarrow{T} C'$. We say a configuration $C$ can *generate* state $p$, if there is a configuration $C'$ that is reachable from $C$ and contains $p$. See Figure 1. For a set $G$ of states,

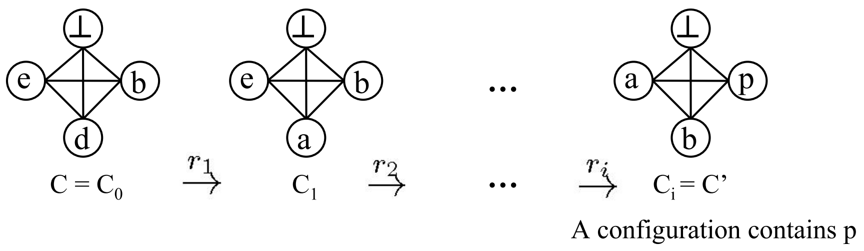

Figure 1: A configuration $C$ can generate state $p$

if a configuration $C$ cannot generate any state in $G$, we say $C$ cannot generate $G$. Letting $P = (Q, \delta)$ be a population protocol, a subset $G$ of $Q$ is called a *closed* set of $P$ if for any transition $r : (p, q) \to (p', q')$ in $\delta$, $p, q \in G$ implies $p', q' \in G$.

We first show three fundamental lemmas obtained from the above definitions. These proofs are omitted due to lack of space.

**Lemma 1** Let $C'$ be a subconfiguration of $C$. If a trace $T$ is applicable to $C'$, it is also applicable to $C$, and $\sigma_T(C')$ is a subconfiguration of $\sigma_T(C)$.

**Lemma 2** If a configuration $C$ cannot generate a set of states $G$, then for any configuration $C'$ such that $C \xrightarrow{*} C'$, $C'$ cannot generate $G$.

**Lemma 3** If a configuration $C$ cannot generate a set of states $G$, any subconfiguration of $C$ cannot generate $G$.

The following lemmas are the keys of our impossibility result.

**Lemma 4** Let $G$ ($|G| < n-1$) be a set of states, and $C$ ($|C| > 0$) be a configuration that cannot generate $G$. Then, either of the following conditions holds:

1: The complement of $G$ (denoted by $\bar{G}$) is closed.

2: There exist a configuration $C'$ and a superset $G'$ of $G$ such that $|C| - 1 \leq |C'|$ and $|G| + 1 = |G'|$ hold, and $C'$ cannot generate $G'$.

**Proof** We prove this lemma by showing that the condition 2 necessarily holds if the complement of $G$ is not closed. Assuming that $\bar{G}$ is not closed, there exists a transition $r : (p, q) \to (p', q')$ such that $p, q \notin G$ and at least one of $p'$ and $q' \in G$ (because if such a transition does not exist, any interaction of two states in $\bar{G}$ results in two states in $\bar{G}$, which implies that $\bar{G}$ is closed). Then we consider the following two cases:

1. One of $p$ and $q$ cannot be generated by $C$: Without loss of generality, we assume that $C$ cannot generate $p$. Then, $C$ cannot generate $\{p\} \cup G$. Therefore, we obtain $C' = C$ and $G' = G \cup \{p\}$ satisfying the condition 2.

2. Both of $p$ and $q$ can be generated by $C$: Since $C$ can generate $p$, there exists a configuration $D$ such that $C \overset{*}{\to} D$ and $p \in D$. We consider the subconfiguration $D'$ that is obtained by replacing the entry of $p$ in $D$ by $\perp$. Then, if we can show that $D'$ cannot generate $q$, the lemma is proved by letting $C' = D'$ and $G = G \cup \{q\}$. In the following, we show it actually holds: Suppose for contradiction that $D'$ can generate $q$. Then, there exists a trace $T$ that makes $D'$ reach a configuration with $q$. By Lemma 1, $T$ is also applicable to $D$, and $\sigma_T(D)$ includes both $p$ and $q$. This implies that $C$ can reach the configuration $\sigma_T(D)$ that includes both $p$ and $q$. Then, It is clear that $C$ can generate both $p'$ and $q'$ because the transition $r$ is enabled in the configuration $\sigma_T(D)$. However, either of $p'$ or $q'$ belongs to $G$ and thus it is contradict to that $C$ cannot generate $G$.

□

**Lemma 5** Any self-stabilizing leader election protocol $P$ has no closed set excluding its leader state.

**Proof** Suppose for contradiction that $P$ has a closed set $H$ which excludes its leader state in $P$. Consider an initial configuration $C$ whose states are all in $H$. Since $H$ is closed, so $C$ can only generate the states in $H$. Because the leader state is not in $H$, $C$ cannot generate a leader state. This implies that any execution starting from $C$ cannot reach a configuration with leader. It is contradiction. □

By using the above two lemmas, we can show the impossibility of self-stabilizing leader election using only $n-1$ states.

**Theorem 1** There is no self-stabilizing leader election protocol that uses only $n-1$ states.

**Proof** We assume for contradiction that a self-stabilizing leader election protocol $P$ which uses only distinct $n-1$ states. The $n-1$ states of the protocol $P$ are denoted by $Q = \{s_0, s_1, s_2, \cdots, s_{n-2}\}$, where $s_0$ is the leader state. The set of all transitions constituting $\mathcal{P}$ is denoted by $\delta_{\mathcal{P}}$.

Letting $C$ be a legitimate configuration, that is, exactly one leader exists in it and another leader is not newly created in any following execution. This implies that the subconfiguration $C'$

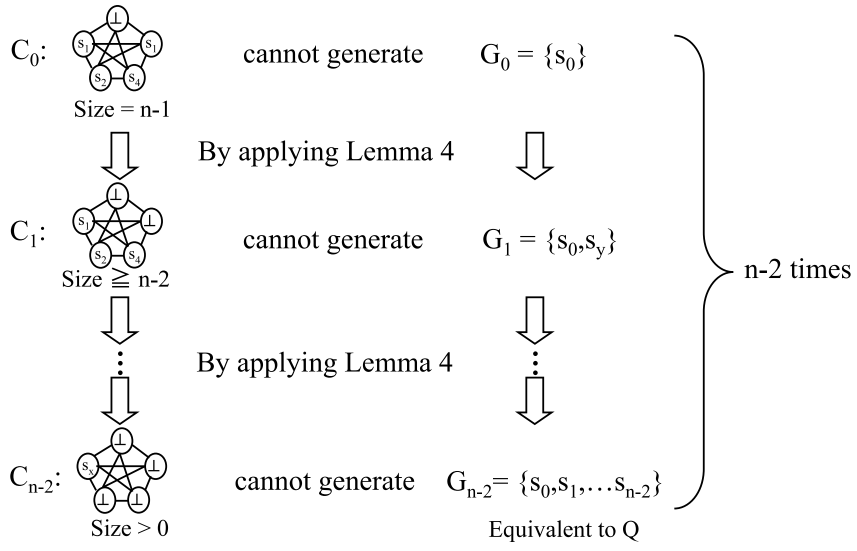

Figure 2: Prove the theorem by contradiction

which obtained by masking the leader state $s_0$ in $C$ cannot generate the leader state $s_0$. Thus, letting $C_0 = C'$ and $G_0 = \{s_0\}$, $C_0$ cannot generate $G_0$, and it holds that $|C_0| = n - 1$ and $|G_0| = 1$. By Lemma 5, there is no closed set excluding $s_0$ in $P$. Thus, the complement of $G_0$ is not closed. Then, by Lemma 4, we can obtain a configuration $C_1$ and a superset $G_1$ of $G_0$ satisfying that $|C_1| \geq |C_0| - 1 = n - 2$, $|G_1| = |G_0| + 1$, and $C_1$ cannot generate $G_1$. Similarly, we can also obtain $C_{i+1}$ and $G_{i+1}$ from $C_i$ and $G_i$ by applying Lemma 4 repeatedly. Finally, after applying the lemma $n - 2$ times, we have a configuration $C_{n-2}$ and a set $G_{n-2}$ satisfying $|C_{n-2}| \geq 1$, $|G_{n-2}| = n - 1$ and $C_{n-2}$ cannot generate $G_{n-2}$. See Figure 2. Then, $G_{n-2}$ is equivalent to $Q$, and thus $C_{n-2}$ cannot generate any state. However, $C_{n-2}$ is not empty, which implies a state $s_x$ $(0 \leq x \leq n - 2)$ in $C_{n-2}$ can be generated by $C_{n-2}$. This is contradiction. □

Since the one-way protocol is a special case of the two-way one, the impossibility result holds even for the one-way protocol. And any impossibility result for the two-way protocol also holds in the one-way one.

**Remarks 1** Noting that our proof does not request any constraint to the transition function. That means the impossibility result also holds for a probabilistic protocol. Thus, our impossibility result can be extended as follows: Without oracles, there is no deterministic or probabilistic SS-LE protocol using only $n - 1$ states of memory even if we assume global fairness.

## 4 Leader Election Protocol Using $n$ States

In this section, we will show a self-stabilizing leader election one-way protocol which uses distinct $n$ states. The $n$ states of the protocol are denoted by $Q = \{s_0, s_1, s_2, \cdots, s_{n-1}\}$, where $s_0$ is the leader state. The proposed protocol is quite simple: When two agents with the same state interact, the responder will increment the subscript of its state (modulo $n$). That is, when the state of the responder is $s_i$, it will changed to be $s_{i+1}$, $i = 0, 1, \cdots, n - 2$, exceptionally, $s_{n-1}$ will be changed to $s_0$.

**Protocol 1** $(s_i, s_i) \rightarrow (s_i, s_{(i+1)\ mod\ n})$, $(i = 0, 1, \cdots, n - 1)$

In what follows, we show that the above protocol correctly elects a unique leader. First, we introduce several notions necessary for the proofs. Throughout this section, we use another representation of each configuration $C = (m_0(C), m_1(C), \cdots, m_{n-1}(C))$, where $m_k(C)$ $(0 \leq k < n)$ is the number of agents having the state $s_k$ in $C$. We also define $\#_0(C)$ to be the number of $m_k(C)$ $(k = 0, 1, \cdots, n-1)$ such that $m_k(C) = 0$ holds.

**Lemma 6** A configuration $C$ with $\#_0(C) = 0$ is a legitimate configuration.

**Proof** From the definition of $\#_0(C)$, we know $\#_0(C) = 0$ means: for every $s_k$ $(k = 0, 1, \cdots, n-1)$, there exists an agent whose state is $s_k$ in $C$. Because the number of agents equals to the number of states, so the states of every agents are different. Therefore, in any following execution, there is only one leader state $s_0$ in the configuration and the state of each agent will not be changed. □

The correctness of the protocol is proved by the argument based on monotonically-decreasing function, which is a standard technique for proving the correctness of self-stabilizing protocols. We first define the *distance* between two states.

**Definition 4 (Distance Function)** For any configuration $C$, the distance $d_{k,j}(C)$ from the state $s_k$ to $s_j$ is defined as follows:

$$d_{k,j}(C) = \begin{cases} 0 & (m_j(C) \neq 0 \text{ or } k = j) \\ (j-k)(m_k(C) - 1) & (0 \leq k < j) \\ (j + n - k)(m_k(C) - 1) & (j < k \leq n) \end{cases}$$

The *total distance* $d_j(C)$ of state $s_j$ in $C$ is the sum of the distances from any state to $s_j$, that is, $d_j(C) = \Sigma_{k=0}^{n-1} d_{k,j}(C)$.

From the definition, a pair of different states $(s_k, s_j)$ can have a non-zero distance only if $m_j(C) = 0$ and $m_k(C) > 1$. That is, if the distance from $s_k$ to $s_j$ for a configuration $C$ is non-zero, no agent has the state $s_j$ and two or more agents necessarily have $s_k$. Then, the value $d_{k,j}(C)$ means how many interactions are necessary to create an agent with the state $s_j$ from an agent having $s_k$ in $C$. The distance $d_{k,j}(C)$ is obtained by multiplying the surplus number of agents having the state $s_k$ by such the necessary number of interactions.

The following lemmas show that if the total distance of some state becomes zero, it remains zero in any following execution.

**Lemma 7** If $m_k(C) > 0$ $(0 \leq k < n)$ holds in a configuration $C$, then $m_k(C') > 0$ in $C'$ holds for any configuration $C'$ such that $C \xrightarrow{*} C'$.

**Proof** Any active interaction of the Protocol 1 reduces the number of $m_k(C)$ by exactly one for some $k$. In addition, to enable the interaction that reducing $m_k(C)$, it is necessary that at least two agents have state $s_k$. So no interaction reduces $m_k(C)$ from 1 to 0. This implies that $m_k(C)$ never becomes zero after it becomes more than zero. □

**Lemma 8** Let $E$ be any unfair execution of Protocol 1, (i.e., $E \in \mathcal{E}_U(1)$). If $m_j(C) = 0$ holds for some $j$ in a configuration $C$ which appears in $E$, a configuration $C'$ such that $m_j(C') > 0$ is reachable from $C$ in $E$.

**Proof** Clearly, in $C'$, the total distance of the state $s_j$ is zero. In addition, for any configuration $C''$, if $m_j(C'') = 0$, two or more agents have the same state in $C''$, which implies that as long as no agent has the state $s_j$, some active interaction eventually occurs (notice that it holds even under the unfairness assumption). Thus, it is sufficient to show that any active interaction decreases the total distance of $s_j$ by one. Let two agents having state $s_k$ interact at a configuration

$C_1$, and $C_2$ be the resultant configuration of the interaction. Then, except for $i = k, k+1$, $d_{i,j}(C_1) = d_{i,j}(C_2)$ necessarily holds because $m_i(C_1) = m_i(C_2)$ holds for any $i$ other than $k$ and $k+1$. In addition, by the transition, the number of agents with $s_k$ decreases by one, and the number of agents with $s_{k+1}$ increases by one. Thus, by simple calculation, we can obtain $d_{k,j}(C_1) + d_{k+1,j}(C_1) = d_{k,j}(C_2) + d_{k+1,j}(C_2) + 1$. This implies that $d_j(C_1) = d_j(C_2) + 1$ holds, which means any active interaction decreases the total distance of $s_j$ by one and eventually $m_j(C)$ will increase from 0 to 1. □

By the above two lemmas, we know that if $m_j(C) = 0$, eventually there exists a configuration $C'$ which is reachable from $C$ such that $m_j(C') > 0$. And there is no execution can reduce $m_j(C) > 0$ to 0. Because the number of agents is the same as the number of distinct states, so we can show the following corollary.

**Corollary 1** For any execution $E = C_0, r_0, C_1, r_1, C_2, \cdots \in \mathcal{E}_U(1)$, there exists $i$ such that $\#_0(C_j) = 0$ holds for any $j \geq i$.

Corollary 1 and Lemma 6 directly imply the correctness of the protocol, and we can get the following theorem.

**Theorem 2** Protocol 1 is a self-stabilizing leader election one-way protocol working correctly under the unfairness assumption, and an arbitrary configuration converges to a legitimate configuration in $\Theta(n^2)$ active interactions.

## 5 No Single Protocol for Complete Graphs with Difference Sizes

In Section 4, we give a protocol using $n$ states to solve the self-stabilizing leader election in a complete graph of size $n$. In this section, we will show that there does not exist any single protocol to solve the self-stabilizing leader election in complete graphs with different sizes.

**Theorem 3** Letting $\mathcal{B}$ be a protocol which can solve the self-stabilizing leader election in complete graphs with size $n$, then $\mathcal{B}$ cannot work correctly in complete graphs with size $n-1$.

**Proof** Consider the legitimate configuration $C$ of a complete graph with size $n$. Since a new leader will not be created, so the subconfiguration $D$ which obtained by masking the leader state in $C$ where $|D| = n-1$, cannot generate the leader state. So consider an initial configuration $C' = D - \{\perp\}$ (a configuration of a complete graph with size $n-1$ and whose entries are the non-$\perp$ entries in $D$), from which the leader state will not be generated when using protocol $\mathcal{B}$. Noting that $C'$ is an initial configuration of a complete graph with size $n-1$, and from such the initial configuration, the legitimate configuration will never be reached. Hence, $\mathcal{B}$ cannot elect a leader correctly in complete graphs with size $n-1$. □

The above theorem also shows that even the upper bound for $n$ is not sufficient knowledge to realize any SS-LE protocol.

## 6 Conclusion

In this paper, we have shown the necessary and sufficient conditions to the solvability of the SS-LE in population protocols having no oracles and of complete interaction graphs. The conditions are characterized by local memory space and fairness assumptions. To prove the impossibility under global fairness, we introduce a new proof technique using closed sets.

## References

[1] Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, 18(4), pages 235-253, 2006.

[2] Dana Angluin, James Aspnes, Michael J. Fischer, Hong Jiang. Self-stabilizing population protocols. In *Proc. 9th International Conference on Principles of Distributed Systems(OPODIS)*, Vol. 3974 of *LNCS*, pages 103-117, 2005.

[3] Dana Angluin, James Aspnes, Melody Chan, Michael J. Fischer, Hong Jiang, René Peralta. Stably computable properties of network graphs. In *Proc. International Conference on Distributed Computing in Sensor Systems(DCOSS),* Vol. 3560 of *LNCS*, pages 63-74, June 2005.

[4] Dana Angluin, James Aspnes, David Eisenstat. Stably computable predicates are semilinear. In *Proc. 25th Annual ACM Symposium on Principles of Distributed Computing*, pages 292-299, 2006.

[5] Dana Angluin, James Aspnes, David Eisenstat. A simple protocol for fast robust approximate majority. In *Proc. 21st International Symposium on Distributed Computing(DISC)*, Vol. 4731 of *LNCS*, pages 20-32, 2007.

[6] Dana Angluin, James Aspnes, David Eisenstat, Eric Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4), pages 279-304, 2007.

[7] James Aspnes, Eric Ruppert. An introduction to population protocols. *Bulletin of the EATCS*, 93, pages 98-117, October 2007.

[8] Joffroy Beauquier, Julien Clement, Stephane Messika, Laurent Rosaz, Brigitte Rozoy. Self-stabilizing counting in mobil sensor networks. In *Proc. 21st International Symposium on Distributed Computing(DISC)*, Vol. 4731 of *LNCS*, pages 63-76, 2007.

[9] Davide Canepa, Maria Gradinariu Potop-Butucaru. Stabilizing leader election in population protocols. Unpublished, 2007.

[10] Carole Delporte-Gallet, Hugues Fauconnier, Rachid Guerraoui, Eric Ruppert. When birds die: Making population protocols fault-tolerant. In *Proc. 2nd IEEE International Conference on Distributed Computing in Sensor Systems(DCOSS)*, Vol. 4026 of *LNCS*, pages 51-66, 2006.

[11] Michael J. Fischer, Hong Jiang. Self-stabilizing leader election in networks of finite-state anonymous agent. In *Proc. 10th International Conference on Principle of Distributed Systems(OPODIS)*, Vol. 4305 of *LNCS*, pages 395-409, 2006.