社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－AL－120 (4)
2008／11／7

Population protocolにおけるオラクルをもたない自己安定リーダー選挙問題の可解性に関して

Shukai Cai 泉 泰介 和田 幸一

名古屋工業大学

概要 本論文では, 任意のエージェント同士が干渉可能なpopulation protocolにおいて, 局所メモリ複雑度と公平性の面から, オラクルを許さない自己安定リーダー選挙問題を解決するための必要十分条件を与える. まず, n エージェントのpopulation protocolにおいて, システムの状態数が $n-1$ の場合自己安定リーダー選挙問題を解決するプロトコルは存在しないことを示す. この不可能性を証明するため, 我々がclosed-setと呼ばれる新たな証明技術を導入する.

また, 自己安定リーダー選挙問題を解く n 状態を使用するプロトコルが任意の公平性を仮定することなしに構成できることを示す. さらに, 自己安定リーダー選挙問題を解くプロトコルを実現するためは, システムのエージェント数を正確に把握することが必要であることを示す.

On Solvability of Self-Stabilizing Leader Election without Oracles in Population Protocols

Shukai Cai Taisuke Izumi Koichi Wada

Nagoya Institute of Technology

Abstract In this paper, we identify the necessary and sufficient conditions to solve the self-stabilizing leader election in population protocols without oracles 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 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 under the assumption of unfairness. This protocol requires the exact knowledge about the number of agents. It is also shown that such knowledge is necessary to construct the self-stabilizing leader election protocols.

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 region, two agents can communicate only when they are sufficiently close to each other in the region. 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 those 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 principle underlying them. Revealing such principle 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, 9]. 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 interaction to other agents (that is, interactions imply that an agent approaches to the close area of other agents in the region). Typically, the capability of each agent is limited. More precisely, it is often assumed that each agent has only constant-space memory and no identifier. The 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, the interest to it is rapidly growing among the community of the distributed computing [3, 4, 5, 6, 7, 8, 10].

Population protocols are originated by Angluin et al. [1], which investigate the class of predicates that can be computed autonomously over population protocols consisting of weak agents. The primary result of this paper is that any predicate in *semilinear* class can be computed on population protocols by proposing protocols that stably compute any predicate in the class. In the following paper [9], it is also shown that any computable predicate by population protocols belongs to semiliner, that is, semiliner is the necessary and sufficient class of the predicates that can be computed on population protocols.

The protocol computing semilinear predicate, proposed in the above paper, is 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. The primary benefit of self-stabilization is that self-stabilizing protocols require no global initialization. In addition, it also brings another benefit of resilience to any transient fault. Generally, it is not guaranteed that the system correctly comes back to its desired behavior after the recovery of transient faults because some effects caused by transient faults, such as memory corruption, still remain. However, by the nature of self-stabilization, self-stabilizing protocols necessarily recover to their correct behavior.

Self-stabilization on population protocols is considered in several previous papers [2, 13, 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 specifying the possibility 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, etc. Unfortunately, it is easily shown that SS-LE is almost impossible if we assume nothing. 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 $\Omega?$. A fairness assumption is a constraint for possible executions on population protocols. Intuitively, the global fairness prevents the occurrence of livelock caused by the loop of some illegal executions. The leader detector oracle is an abstracted virtual device that informs the existence and inexistence of the agents having the leader state. Both of the assumptions give some additional computational powers to population protocols, which are 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 solvablility of SS-LE on the complete interaction graphs are known:

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

2. Under the assumption of unfairness and no oracle, no uniform protocol can solve SS-LE, where a uniform protocol is 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).

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

In this paper, we also investigate the solvablity of SS-LE on population protocols. In particular, we are interested in self-stabilizing leader election protocols in complete interaction graphs that have no use of 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 SS-LE protocol using only $\log_2(n-1)$ bits of memory even if we assume the global fairness.

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

3. Even if we assume global fairness, without oracle, there is no uniform SS-LE protocol in strong sense. More precisely, we show that any SS-LE protocol working correctly on the population of n agents does not work on the population of $n+k$ agents ($k > 0$).

The third result implies that even the knowledge about the upper bound for the number of agents is insufficient to design SS-LE protocol, 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 also be noted that the first

impossibility result is quite nontrivial and interesting. The global fairness is reasonable but sufficiently strong so that it can break the essential ideas leading the previous impossibility results. Actually, under the global fairness assumption, many of the existing techniques to prove the impossibility cannot be adapted. In this paper, we resolve such difficulty by introducing a novel proof technique based on *closed sets*. The key idea of it is to identify the set of states that never create the leader state. While this paper utilizes this technique to show the impossibility of SS-LE, we believe that the proposed technique can be applied to more broader cases, including other problems and other graph classes, to prove the impossibility under the global fairness assumption.

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 its own state 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 defined 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 a node v_i to v_j implies that the agent corresponding to v_1 can interact to the agent for v_2, 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 are possible to interact with each other.

A *protocol* $P = (Q, \delta)$ is a pair of a finite set Q of *states* and a *transition function* δ 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. 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). For any transition $r : (p, q) \to (p', q')$ of δ, we call p and q the *prestates* of r, p' and q' are called the *poststates* of r. 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 called *active*.

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 *enable* 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 are 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 δ 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 enable 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 in it).

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 two fairness assumptions [13]:

Definition 2 (Global fairness assumption $\mathbb{G}$) An execution $E = C_0, r_0, C_1, r_1, \cdots$ is global fairness, if 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.

Definition 3 (Local fairness assumption $\mathbb{L}$) An execution $E = C_0, r_0, C_1, r_1, \cdots$ is local fairness, if for every transition r, if r is enable in C_i for infinitely many i, then $C_i \xrightarrow{r} C'_{i+1}$ for infinitely many i. Hence, the transition r is taken infinitely many times in E. We say the scheduler $\mathbb{S}$ is local fairness.

In addition to the above, we also define *unfairness assumption* $\mathbb{U}$, which requires no assumption to executions. Given a protocol P and an arbitrary fairness assumption $X \in \{\mathbb{G}, \mathbb{L}, \mathbb{U}\}$, we define $\mathcal{E}_X(P)$ to 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 an arbitrary 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 4 (Self-stabilizing leader election) A protocol P solves the self-stabilizing leader election under 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 Only $n-1$ States

In this section, we will show that without the help of $\Omega?$, the self-stabilizing leader election protocol is impossible in a complete network graphs under global fairness only use distinct $n-1$ states ($\log_2(n-1)$ bits of memory).

3.1 Difficulty of Proving Impossibility under the Global Fairness

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

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

Partition argument:

It is the technique using the fact that it is difficult to break a certain kind of symmetry. Its basic idea 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 two subgraphs independently and separately elect a leader respectively. However, it is contradict to the uniqueness of leader. First, this argument can be applied only to the case of uniform protocols because nonuniform protocol does 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). In addition, 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 the edge that joints two subgraphs infinitely often, it must occur necessarily under the global fairness. We cannot eliminate the possibility that such interaction breaks the symmetry, which can result in the union of two leaders.

To circumvent the problems 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 interaction between any pair of two states in the set creates no state out of the set. The key of our proof is to find a closed set excluding the leader state.

3.2 Impossibility Using $n-1$ States

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

Let C be a configuration. A *subconfiguration* C' of C is an n-tuple obtained by replacing several entries in C by $\perp$, where $\perp$ is the special value that masks the state of the corresponding agent. The above definition is simply extended in the case that C is a subconfiguration. That is, if a subconfiguration C' is obtained from another subcofiguration C'' by the replacement of entries, we say that C' is a subconfiguration of C''. The size of a subconfiguration C' is the number of non-$\perp$ values appearing in C', and it is denoted by $|C'|$. 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 $T = r_0, r_1, \cdots, r_i$ is *applicable* to a (sub)configuration C_0 if there exists a sequence of (sub)configurations $C_0, C_1, \cdots, C_{i+1}$ such that $C_0 \xrightarrow{r_0} C_1 \xrightarrow{r_1} C_2 \xrightarrow{r_2} \cdots \xrightarrow{r_i} C_{i+1}$. For a (sub)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 (sub)configuration C' is *reachable* from a (sub)configuration C, denoted by $C \xrightarrow{*} C'$, if there exists a trace T such that $C \xrightarrow{T} C'$. We say a (sub)configuration C can *generate* state p, if there is a (sub)configuration C' that is reachable from C and contains p. For a set G of states, if a (sub)configuration 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 δ, $p, q \in G$ implies $p', q' \in G$.

We first show three fundamental lemmas obtained from the above definitions.

The proof of Lemma 1 is omitted due to lack of space.

Lemma 1 Letting 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 (sub)configuration C cannot generate a set of states G, then for any (sub)configuration C' such that $C \xrightarrow{*} C'$, C' cannot generate G.

Proof Suppose for contradiction that C' can generate a state p in G. Then, there exists a (sub)configuration D such that $C' \xrightarrow{*} D$ and $p \in D$. Since $C \xrightarrow{*} C'$ holds, we obtain $C \xrightarrow{*} D$, which contradicts the fact that C cannot generate G.

Lemma 3 If a (sub)configuration C cannot generate a set of states G, any subconfiguration of C cannot generate G.

Proof Suppose for contradiction that a subconfiguration C' of C can generate a state p in G. Then, there exists a trace T such that $\sigma_T(C')$ includes the state p. By Lemma 1, T is also applicable to C and $\sigma_T(C')$ is a subconfiguration of $\sigma_T(C)$. This implies p belongs to $\sigma_T(C)$, which is contradiction.

The following lemmas are the key of our impossibility result.

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

1: The complement of G is closed.

2: There exists a subconfiguration C' and a set of states 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. Let $\bar{G}$ be the complement of G. Assuming that $\bar{G}$ is not closed, there exists a transition $r : (p, o) \to (p', q')$ 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 intraction of two states in $\bar{G}$ results in two states in $\bar{G}$, which implies $\bar{G}$ is closed). Then we consider the following 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 subconfiguration D such that $C \xrightarrow{*} D$ 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 subconfiguration 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 enable in the configuration $\sigma_T(D)$. However, either of p' or q' belongs to G and thus it is contradiction.

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 executions starting from C cannot reach configurations 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 implies the leader state. The set of all transitions constituting $\mathcal{A}$ is denoted by $\delta_{\mathcal{A}}$.

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' 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 they satisfy $|C_0| = n-1$ 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 subconfiguration C_1 and a set of states G_1 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 subconfiguration C_{n-2} and G_{n-2} satisfying $|C_{n-2}| \geq 1$, $|G_{n-2}| = n-1$ and C_{n-2} cannot generate G_{n-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 q in C_{n-2} can be generated by C_{n-2}. This is contradiction.

4 Leader Election Protocol Using n States

In this subsection, we will show a self-stabilizing leader election protocol $\mathcal{B}$ which uses distinct n states. The n states of the protocol $\mathcal{B}$ are denoted by $Q = \{s_0, s_1, s_2, \cdots, s_{n-1}\}$, where s_0 implies 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 $(s_i, s_i) \to (s_i, s_{(i+1) \bmod n})$, $(i = 0, 1, \cdots, n-1)$

In what follows, we show that the above protocol correctly elect a unique leader. First, we introduce several notions necessary for the proofs. Throughout this subsection, 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 the agents whose state is 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.

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

Definition 5 (Distance Function) For any configuration C, the distance $d_{k,j}(C)$ between the two states s_k and s_j is defined as follows:

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

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

From the definition, a pair of states s_k and s_j can have a non-zero distance only if $m_j(C) = 0$ and $m_k(C) > 1$. That is, if the distance between s_k and 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 $d_{k,j}(C)$ implies how many interactions are necessary to create the 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 to 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 executions.

The proof of the following lemmas are omitted due to lack of space.

Lemma 6 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 $C \xrightarrow{*} C'$.

Lemma 7 Let E be any unfair execution of Protocol 1, (i.e., $E \in \mathcal{E}_U(\mathcal{B})$). If $m_j(C) = 0$ holds for some j in a configuration C that appears in E, a configuration C' such that $m_j(C') > 0$ is reachable from C in E.

By the above two lemmas, we can show the following corollary.

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

The above corollary directly implies the correctness of the protocol.

Theorem 2 Protocol 1 is a self-stabilizing leader election protocol working correctly under the unfairness assumption and from an arbitrary initial configuration, the legitimate configuration C_j such that $\#_0(C_j) = 0$ will be reached in $\theta(n^2)$.

5 No Simple 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 complete graphs 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 $\mathcal{A}$ be a protocol which can solve the self-stabilizing leader election in complete graphs with size n, then $\mathcal{A}$ 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 $C' = D - \{\perp\}$, from which the leader state will not be generated. 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, $\mathcal{A}$ cannot elect a leader correctly in complete graphs with size $n-1$

6 Conclusion

In this paper, we showed the necessary and sufficient condition to the solvability of the SS-LE in population protocols having no oracles and complete interaction graphs. The condition is characterize by local memory space and fairness assumption. 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. *Twenty-Third ACM Symposium on Principles of Distributed Computing.* (2004) 290-299.

[2] Dana Angluin, James Aspnes, Michael J. Fischer, Hong Jiang. Self-stabilizing population protocols. In *Proc. Principles of Distributed Systems, 9th International Conference*, 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 DCOSS, pages 63-74, 2005.

[4] Dana Angluin, James Aspnes, David Eisenstat, Eric Ruppert. On the power of anonymous one-way communication. In *Proc. Principles of Distributed Systems, 9th International Conference*, pages 396-411, 2005.

[5] 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.

[6] Dana Angluin, James Aspnes, David Eisenstat. Fast computation by population protocols with a leader. In *Proc. Distributed Computing, 20th International Symposium*, pages 61-75, September 2006.

[7] Dana Angluin, James Aspnes, David Eisenstat. A simple protocol for fast robust approximate majority. In *Proc. Distributed Computing, 21th International Symposium*, pages 20-32, 2007.

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

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

[10] Joffroy Beauquier, Julien Clement, Stephane Messika, Laurent Rosaz, Brigitte Rozoy. Self-stabilizing counting in mobil sensor networks. Technical Report 1470, LRI, Université Pairs-Sud 11,2007.

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

[12] 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*, pages 51-66, 2006.

[13] Michael J. Fischer, Hong Jiang. Self-stabilizing leader election in networks of finite-state anonymous agent. In OPODIS, pages 395-409, 2006.