Constructing Self-Stabilizing Oscillators in Population Protocols

Colin Cooper[a], Anissa Lamani[b,*], Giovanni Viglietta[c], Masafumi Yamashita[b], Yukiko Yamauchi[b]

[a]*Department of Informatics, King's College, United Kingdom*
[b]*Department of Informatics, Graduate School of ISEE, Kyushu University, Japan*
[c]*School of Electrical Engineering and Computer Science, University of Ottawa, Canada*

Abstract

Population protocols (PPs) are used as a model for a collection of finite-state mobile agents which interact with each other to accomplish a common task. Unlike most of the previous works, which investigate their computational power, this paper throws light on an aspect of PPs as a model of chemical reactions. Motivated by the well-known *BZ reaction* that provides an autonomous chemical oscillator, we address the problem of autonomously generating an oscillatory execution from any initial configuration (i.e., in a self-stabilizing manner). For deterministic PPs under a deterministic scheduler, we show that the self-stabilizing leader election (SS-LE) problem and the self-stabilizing oscillation (SS-OS) problem are equivalent, in the sense that an SS-OS protocol is constructible from a given SS-LE protocol, and vice versa, which unfortunately implies that (1) resorting to a leader is inevitable (although we seek a decentralized solution), and (2) n states are necessary to create an oscillation of amplitude n, where n is the number of agents (although we seek a memory-efficient solution). Aiming at reducing the space complexity, we present and analyze some deterministic oscillatory PPs under a uniform random scheduler.

Keywords: autonomous systems, leader election, population protocols, self-organization, self-oscillation, self-stabilization

*Corresponding author
Email address: anissa.lamani@gmail.com (Anissa Lamani)

1. Introduction

We focus in this paper on self-oscillations which play fundamental roles in autonomous biological reactions, and investigate them as a phenomenon in distributed computing. Self-oscillations are often understood as a chemical oscillator provided by certain reactions, such as the Belousov–Zhabotinsky reaction. We use in our investigation the population protocol model.

The *population protocol* (PP) model introduced by Angluin et al. [1] is a model of passive agent systems. It is used as a theoretical model of a collection of finite-state mobile agents that interact with each other in order to solve a given problem in a cooperative fashion. In PPs, computations are done through pairwise interactions: When two agents interact, they exchange their information and update their respective states according to some common protocol. The interaction pattern is assumed to be unpredictable, that is, each agent has no control over which agent it interacts with. We thus assume the presence of an abstract mechanism called *scheduler* that chooses, at any time instant, a pair of agents for an interaction. The PP model can represent not only artificial distributed systems such as sensor networks and mobile agent systems, but also natural distributed systems such as animal populations and chemical reaction networks.

In the past decade, many problems have been investigated on PPs, including the problems of computing a function [2, 1, 3, 4, 5], electing a leader [6, 7, 8, 9], counting [10, 11, 12], coloring [4] and synchronizing [13]. Most of these problems consider the computational power of the population and hence are *static*; the agents are requested to eventually reach a configuration that represents the answer to the considered computation problem. The notion of termination is typically intended in the Noetherian sense (in the context of abstract rewriting systems); agents are not requested to eventually terminate, however, the execution is requested to repeat the target configuration that contains the answer of the problem that is considered, forever.

Unlike most of the past works in PPs, we throw light on an aspect of PPs as a model of chemical reactions. Specifically, we investigate a dynamic problem that consists of designing a PP that stabilizes to an oscillatory execution, no matter from which initial configuration it starts; that is, we explore a *self-stabilizing* PP that generates an oscillatory execution. The problem emerges in the project of designing molecular robots [14], and is directly motivated by the well-known Belousov–Zhabotinsky reaction, which is an example of

non-equilibrium thermodynamics providing a non-linear chemical oscillator. In biological systems, the oscillatory behavior is used as a natural clock to transmit signals and hence transfer information. In artificial distributed systems, aside from their theoretical interest, PPs that exhibit an oscillatory behavior could be used to distributely and autonomously implement a clock.

This paper shows that under a deterministic scheduler governed by an adversary, the self-stabilizing leader election (SS-LE) problem and the self-stabilizing oscillation (SS-OS) problem are equivalent, in the sense that an SS-OS protocol is constructible from a given SS-LE protocol, and vice versa, and hence costly in terms of the space complexity. Specifically, we show the followings: Let n be the size of the population. Cai et al. presented an SS-LE protocol $\mathcal{P}_{LE}$ whose space complexity (per agent) is exactly $\lceil \log n \rceil$ bits and showed that it is optimal; there is no SS-LE protocol whose space complexity is less than $\lceil \log n \rceil$ [6]. We first construct an SS-OS protocol $\mathcal{P}_{OS}$ from $\mathcal{P}_{LE}$ by using 2 more bits (per agent). Since any SS-LE protocol $\mathcal{P}'_{LE}$ requires at least $\lceil \log n \rceil$ bits and can be transformed into $\mathcal{P}_{LE}$, we can easily construct $\mathcal{P}_{OS}$ from $\mathcal{P}'_{LE}$ via $\mathcal{P}_{LE}$. We next show that an SS-LE protocol is constructible from any SS-OS protocol, again by using 2 more bits, which implies that the space complexity of any SS-OS protocol is at least $\lceil \log n \rceil - 2$ bits.

Although the space complexity of $\mathcal{P}_{OS}$ is $\lceil \log n \rceil + 2$ bits, it requires $4n - 2$ states. Aiming at the reduction of the number of states, under a uniform random scheduler, by modifying $\mathcal{P}_{OS}$, we propose three SS-OS protocols $\mathcal{PO}_I$ $(I = 1, 2, 3)$ which respectively require $2(n + \sqrt{n})$, $2(n + \log n)$ and $2(n + 2)$ states. For $\mathcal{PO}_1$ and $\mathcal{PO}_3$, the space complexity is reduced at the expense of either a lower average amplitude or a longer average period.

Apart from the difference of motivation, few works on *dynamic* problems are related to our work. Angluin et al. [4] provided a self-stabilizing token circulation protocol in a ring with a pre-selected leader. Beauquier and Burman investigated the self-stabilizing mutual exclusion problem, the self-stabilizing group mutual exclusion problem [15] and the self-stabilizing synchronization problem [13]. In the latter work [13], the authors have shown that the self-stabilizing synchronization problem in the PP model under a deterministic scheduler is impossible to solve without any additional assumptions and have hence proposed a solution, assuming the presence of an unlimited-resource agent called *Base Station*. Both the token circulation protocol proposed in [4] and the phase clock protocol presented in [13] can be used to implement a self-stabilizing oscillatory behavior. However, the first one works only for

ring shaped interacting graphs, while the second one uses the notion of cover time (the minimum number of interactions for an agent to have met with each other agent with certainty) and assumes an unlimited resource agent.

Roadmap. After introducing some concepts and notions in Section 2, we consider PPs under a deterministic scheduler governed by an adversary in Section 3. Under this scheduler, we show that the SS-LE problem and the SS-OS problem are equivalent; that is, an SS-OS protocol is constructible from a given SS-LE protocol, and vice versa. In Section 4, we consider PPs under a uniform random scheduler, i.e., the interacting agents are chosen uniformly at random. Under a uniform random scheduler, we present and analyze some oscillatory PPs, mainly aiming to reduce the space complexity. Section 5 is devoted to the conclusion and open problems.

2. Preliminaries

In this paper, we consider a population of n anonymous finite-state agents that update their states by interacting with other agents. The set of n agents in the population is denoted by $A = \{0, 1, \ldots, n-1\}$. We consider only pairwise interactions; each interaction involves exactly two agents, and they update their states according to a common protocol when they interact. Identities $i \in A$ are used for notation purposes only. The agents have no identity and cannot be distinguished from each other. In addition, all agents execute the same protocol. Any pair of agents i and j ($i \neq j$) in the population are susceptible to interact.

A *protocol* $\mathcal{P} = (Q, \delta)$ is a pair of a finite set of states Q and a transition function $\delta : Q \times Q \to Q \times Q$. When two agents interact with each other, δ determines the next states of both agents. Let p and q be the states of agents i and j, respectively. $\delta(p, q) = (p', q')$ indicates that the states of agents i and j, after interacting with each other, are p' and q', respectively. We distinguish the *initiator* and the *responder* in δ, so that $\delta(p, q) = (p', q')$ *may not* imply $\delta(q, p) = (q', p')$.

A *configuration* C is a mapping from A to Q that specifies the states of all the agents in the population. By $C(i)$ and $\mathcal{C}$, we refer to the state of a given agent i in a configuration C and the set of all possible configurations of the population, respectively. Given a configuration $C \in \mathcal{C}$ and an interaction $r = (i, j)$ between two agents i and j, we say that C' yields from C by

the interaction r, denoted by $C \xrightarrow{r} C'$, if $(C'(i), C'(j)) = \delta(C(i), C(j))$ and $C'(\ell) = C(\ell)$ for all $\ell \in A \setminus \{i, j\}$. An interaction r is said to be *active* in a configuration $C \in \mathcal{C}$, if it updates the state of at least one of the two interacting agents of r. Otherwise, r is called *inactive* and does not contribute to the computation.

An *execution* $\mathcal{E}$ of a protocol $\mathcal{P}$ is an infinite sequence of configurations and interactions $(C_0, r_0, C_1, r_1, \dots)$ such that r_t is an interaction (i.e., an ordered pair of agents) and $C_t \xrightarrow{r_t} C_{t+1}$ for all time instants $t = 0, 1, \dots$, assuming that each transition takes a unit of time. We say that C_t is reachable from C_s, denoted by $C_s \xrightarrow{*} C_t$, when $s \leq t$.

A *scheduler* chooses an ordered pair of agents to interact at each time $t \geq 0$. We consider two types of schedulers in this paper:

1. A deterministic *globally fair scheduler* which ensures that if there is a configuration that is reachable infinitely often, then the configuration is eventually reached.
2. A *uniform random scheduler* which chooses an interaction uniformly at random from the set of all the ordered pairs of distinct agents.

Whenever we investigate a PP under a scheduler, an execution means a one generated by the scheduler.

An execution $\mathcal{E}$ (under a scheduler) in general contains inactive interactions. Since these inactive interactions do not contribute to the computation, we assume without loss of generality that every transition in $\mathcal{E}$ is caused by an active interaction, or more clearly, given an execution $\mathcal{E}$ generated by the scheduler, we always investigate the execution constructed from $\mathcal{E}$ by ignoring (removing) all transitions caused by inactive interactions.

We define some notions that will be used throughout the paper. Let $\mathbb{N}$ be the set of natural numbers including 0. By $[a, b]$ we denote the set of natural numbers between a and b (including a and b), i.e., $\{a, a+1, \dots, b\} \subset \mathbb{N}$, where we assume $a \leq b$. Given an execution $\mathcal{E} = (C_0, r_0, C_1, r_1, \dots)$ of a population protocol $\mathcal{P} = (Q, \delta)$ on n agents and a subset S of states ($\subset Q$), by $f_S^{\mathcal{E}}$, we denote the function from $\mathbb{N}$ to $[0, n]$ that maps a time instant t into the number of agents whose state is in S in C_t. We frequently omit $\mathcal{E}$ of $f_S^{\mathcal{E}}$ to write f_S when it is obvious from the context.

Definition 1. ***(Oscillation)***
Let f be a function from $[a, b]$ to $\mathbb{N}$ for some natural numbers a and b. We say that f is an oscillation if there exists a number $c \in \mathbb{N}$ such that the following four conditions hold:

1. $a < c < b$,
2. $f(a) < f(c) > f(b)$,
3. $f(a) = f(b)$, *and*
4. f *is* weakly *increasing in* $[a, c]$ *and* weakly *decreasing in* $[c, b]$.

The value $f(c) - f(a)$ *is called the* amplitude *of the oscillation and is denoted by* ι_a*, whereas* $b - a$ *is called the* period *of the oscillation and is denoted by* ι_p*. The* increasing phase *(resp.,* decreasing phase*) of the oscillation is the interval in which* f *is weakly increasing (resp., weakly decreasing).*

Definition 2. ***(Oscillatory Behavior – Triangle Wave)***
Let $\mathcal{E}$ *and* S *be an execution of a population protocol* $\mathcal{P} = (Q, \delta)$ *and a subset of* Q*, respectively. We say that* $\mathcal{E}$ *exhibits an oscillatory behavior for the set of states* S *with amplitude* ι_a*, if there is a strictly increasing sequence* $\tau_0, \tau_1, \ldots$ *of time instants such that, for every* $i \geq 0$*, the restriction of* $f_S^{\mathcal{E}}$ *to* $[\tau_i, \tau_{i+1}]$ *is an oscillation of amplitude* ι_a*. That is,* $\mathcal{E}$ *exhibits an oscillatory behavior if* f_S *of* $\mathcal{E}$ *represents a triangle wave.*

Note that in the above definition, as assumed, $\mathcal{E}$ does not contain transitions caused by inactive interactions. Note also that oscillations in an oscillatory behavior may have different periods by the definition of oscillatory behavior.

3. Oscillations under Deterministic Globally Fair Scheduler

Provided that the initial configuration is arbitrary, we investigate the problem of generating oscillatory executions under a deterministic globally fair scheduler. We first define the self-stabilizing oscillator (SS-oscillator) under a deterministic globally fair scheduler.

Definition 3. ***(Oscillator)***
A population of agents executing a deterministic protocol $\mathcal{P}$*, under a deterministic globally fair scheduler, is a* (C_t, S, ι_a)*-oscillator if any execution* $\mathcal{E} = (C_t, r_t, C_{t+1}, r_{t+1}, \ldots)$ *of* $\mathcal{P}$*, exhibits an oscillatory behavior for the set of states* S*, with amplitude* ι_a*.*

Definition 4. ***(SS-Oscillator)***
A population of agents executing a deterministic protocol $\mathcal{P}$*, under a deterministic globally fair scheduler, is a self-stabilizing oscillator for the set of states* S *and amplitude* ι_a *if, starting from an arbitrary configuration* $C_0 \in \mathcal{C}$*,*

every execution $\mathcal{E} = (C_0, r_0, C_1, r_1, \dots)$ of $\mathcal{P}$, eventually reaches a configuration $C_t \in \mathcal{C}$ such that $\mathcal{P}$ is a (C_t, S, ι_a)-oscillator.

In this section, we show that a self-stabilizing oscillator (SS-OS) protocol of amplitude n is constructible from a given self-stabilizing leader election (SS-LE) protocol, and vice versa. We define the SS-LE in the following:

Definition 5. ***(SS-Leader Election)***
A PP $\mathcal{P} = (Q, \delta)$ solves the self-stabilizing leader election problem, under a deterministic globally fair scheduler, if there is a state $p_L \in Q$, and every execution $\mathcal{E} = (C_0, r_0, C_1, r_1, \dots)$ of $\mathcal{P}$ contains an agent $i \in A$ and a time instant $t_0 \in \mathbb{N}$ such that, for any time $t \geq t_0$ and for any agent $j \in A$, $C_t(j) = p_L$ if and only if $j = i$.

In [6], Cai et al. showed that the SS-LE problem is impossible to solve if the number of states per agent is less than n, where n is the size of the population. They also proposed a simple PP $\mathcal{P}_{LE}$ that solves the SS-LE problem, whose state set Q_{LE} is $\{0, 1, \dots, n-1\}$, where 0 is the leader state. Protocol $\mathcal{P}_{LE} = (Q_{LE}, \delta_{LE})$ ensures that eventually each agent has a unique state and hence one leader is eventually elected. The transition function δ_{LE} of $\mathcal{P}_{LE}$ is presented in Protocol 1. In Protocol 1, $(p, q) \to (p', q')$ means $(p', q') = \delta_{LE}(p, q)$, and for each unspecified pair of parameter values, e.g., for $\delta_{LE}(0, 1)$, δ_{LE} does not change the states of any of the two interacting agents, which causes an inactive interaction. We adopt these conventions to describe protocols in the rest of this paper.

Protocol 1 δ_{LE}

$(i, i) \to (i, (i+1) \mod n)$	$i \in [0, n-1]$

The next proposition holds:

Proposition 1. *[6] Protocol $\mathcal{P}_{LE}$ solves the SS-LE problem.*

3.1. Constructing SS-OS Protocol from $\mathcal{P}_{LE}$

In this subsection, we show that an SS-oscillator $\mathcal{P}_{OS} = (Q_{OS}, \delta_{OS})$ for $\iota_a = n$ is constructible from $\mathcal{P}_{LE}$. Here, $Q_{OS} = Q_L \cup Q_F$ is the union of the state set for the leader and the one for followers, and they are defined as $Q_L = \{(i, p, c) : i = 0,\ p \in [0, 1],\ c \in [1, n]\}$ and $Q_F = \{(i, p) : i \in [1, n-1],\ p \in [0, 1]\}$. The size of Q_{OS} is thus $2n + 2(n-1) = 4n - 2$.

Roughly, $\mathcal{P}_{OS}$ is regarded as a fair composition of $\mathcal{P}_{LE}$ and a protocol to govern the oscillations. However, its space complexity is reduced to $4n-2$ states instead of $\Omega(n^2)$ states using the fair composition described in [16].

Let us explain the idea behind $\mathcal{P}_{OS}$. It runs $\mathcal{P}_{LE}$, which regards the first component i of a state (i,p,c) or (i,p) as its state in Q_{LE} and solves the SS-LE problem. Then eventually a configuration C_t is reached such that every agent $j \in A$ has a different state in Q_{LE}; formally, for any $i \in [0,n-1]$, there is an agent $j \in A$ such that the first component of $C_t(j)$ is i. Since no rules in δ_{LE} are applicable to C_t, the first component i of any agent does not change after t. Observe that this is true only if the rules in δ_{LE} are the only ones that can update i, as it will be intended in designing δ_{OS}.

Once C_t is reached, all what $\mathcal{P}_{OS}$ needs to do to solve the SS-OS problem is to control the alternation of increasing and decreasing phases, which is realized by a transition function δ_{CNT}, using the second component p and the third component c of Q_L. Intuitively, $p=1$ (resp., $p=0$) indicates that the agent is involved in the increasing (resp., decreasing) phase of an oscillation, and c indicates the number of agents whose phase p is the same as that of the leader's. Transition function δ_{CNT} is applicable only to a pair of a leader state in Q_L and a follower state in Q_F, and is presented in Protocol 2. In Protocol 2, although we only describe the cases in which the initiator is the leader and the responder is a follower, i.e., the first parameter of δ_{CNT} is from Q_L and the second parameter is from Q_F, we assume that if $\delta_{CNT}(p,q)=(p',q')$ then $\delta_{CNT}(q,p)=(q',p')$. Observe that δ_{CNT} does not update the first component i of the agents' state.

Protocol 2 δ_{CNT}

1. $((0,p,i),(i,q)) \to ((0,p,i+1),(i,p))$	$i \in [1,n-1]$, $p,q \in [0,1]$
2. $((0,p,n),(i,q)) \to ((0,1-p,1),(i,q))$	$i \in [1,n-1]$, $p,q \in [0,1]$

Rule set δ_{CNT} works as follows: When the leader with a state $(0,p,c)$ interacts with another agent j whose state is (c,q), the leader increments its counter value c and j updates its phase q to the same phase p of the leader. When the leader reaches the maximum value of its counter (i.e., when $c=n$), it toggles its phase and re-initializes its counter to 1 at the next interaction. Then after the first re-initialization of the leader's counter value, when the leader reaches again the maximum value of its counter, the leader has interacted with every other agent in the population, and every agent has

updated its phase to the leader's phase. When the leader toggles its phase, the next phase of the oscillator is initiated. We prove the correctness of $\mathcal{P}_{OS}$ in the following.

Let $\mathcal{C}_{LE}$ be the set of configurations C such that the first component i of the state $C(j)$ of each agent $j \in A$ is distinct; that is, a leader is elected in C. Without loss of generality, we assume that agent 0 is the leader. Let $||C||$ be the number of agents (including the leader) that have the same phase p as the leader in a configuration C. By definition $||C|| \geq 1$.

Lemma 1. *Any execution $\mathcal{E} = (C_0, r_0, C_1, r_1, \dots)$ of δ_{CNT}[1] starting from a configuration $C_0 \in \mathcal{C}_{LE}$ eventually reaches a configuration $C_t \in \mathcal{C}_{LE}$ such that (1) $C_t(0) = (0, p, n)$ for some $p \in [0, 1]$ and (2) $n - c_0 + 1 \leq ||C_t|| \leq n$, after $n - c_0$ active interactions, where c_0 is the counter value of the leader in C_0 i.e., $C_0(0) = (0, p, c_0)$ for some $p \in [0, 1]$.*

Proof. If $c_0 = n$ then there is nothing to show. Otherwise, assume $1 \leq c_0 < n$. By definition, C_0 contains an agent $j \in A$ such that $C_0(j) = (c_0, q)$ for some $q \in [0, 1]$. Observe that the only rule applicable to C_0 is Rule 1 in an interaction between the leader and j. When the interaction occurs, the leader increments its counter to $c_0 + 1$ and j updates its state to (c_0, p) if $q = 1 - p$, that is $||C_1|| \geq ||C_0||$. Again, by definition, C_1 contains an agent $j' \in A$ such that $C_1(j') = (c_0 + 1, q)$ for some $q \in [0, 1]$. Like in the previous case, the only active interaction in C_1 is between the leader and j'. By interacting, Rule 1 is executed and hence the leader increments its counter and j' updates its state to $(c_0 + 1, p)$ if $q = 1 - p$. By repeating this argument, when the counter value becomes n, all agents in $[c_0, n - 1]$ have interacted with the leader and as a result have phase p of the leader. Thus, the lemma holds. ■

Lemma 2. *Let $C_0 \in \mathcal{C}_{LE}$ be any configuration such that $C_0(0) = (0, p, 1)$ for some $p \in [0, 1]$ and $||C_0|| = 1$. In any execution $\mathcal{E} = (C_0, r_0, C_1, r_1, \dots)$ of δ_{CNT} starting from C_0, the leader toggles its phase and re-initializes its counter value every n active interactions.*

Proof. In a configuration $C_t \in \mathcal{C}_{LE}$ in which $C_t(0) = (0, p, n)$ for some $p \in [0, 1]$ and $||C_0|| = n$. The only rule applicable to C_0 is Rule 2, and as the result of an interaction, a configuration $C \in \mathcal{C}_{LE}$ yields such that $C(0) = (0, 1 - p, 1)$ and $||C_0|| = 1$. By Lemma 1, the claim holds. ■

[1]More clearly, $\mathcal{E}$ is an execution of $\mathcal{P}_{CNT} = (Q_{OS}, \delta_{CNT})$.

We now define a set of states S_{OS} by $S_{OS} = \{(0,1,c) : c \in [1,n]\} \cup \{(i,1) : i \in [1,n-1]\}$, i.e., the set of all states with $p = 1$. the next lemma immediately holds.

Lemma 3. *Let $C_0 \in \mathcal{C}_{LE}$ be any configuration. Then any execution $\mathcal{E} = (C_0, r_0, C_1, r_1, \dots)$ of δ_{CNT} eventually reaches a configuration C_t such that δ_{CNT} is a (C_t, S_{OS}, n)-oscillator.*

Proof. By Lemma 1, starting from an arbitrary initial configuration C_0, any execution $\mathcal{E} = (C_0, r_0, C_1, r_1, \dots)$ of δ_{CNT} eventually reaches a configuration C_t satisfying $C_t(0) = (0, p, n)$. Since Rule 2 is the only rule applicable in C_t, a configuration C_{t+1} satisfying $C_{t+1}(0) = (0, 1-p, 1)$ is reached after one active interaction. By Lemma 1, a configuration $C_{t'}$ satisfying $C_{t'}(0) = (0, p, n)$ and $||C_{t'}|| = n$ is reached at some time instant $t' \geq t+1$. The only rule applicable is Rule 2, and hence the leader toggles its phase again and re-initializes its counter to 1. Thus, the claim holds by Lemma 2. ■

We now define δ_{OS} of $\mathcal{P}_{OS}$ as the union of δ_{CNT} and γ_{LE} described in Protocol 3, which is an implementation of δ_{LE} for Q_{OS}. That is, $\delta_{OS} = \gamma_{LE} \cup \delta_{CNT}$.

Protocol 3 γ_{LE}

0a. $((0,p,c),(0,p',c')) \to ((0,p,c),(1,p'))$	$p,p' \in [0,1]$, $c,c' \in [1,n]$
0b. $((n-1,p),(n-1,p')) \to ((n-1,p),(0,p',1))$	$p,p' \in [0,1]$
0c. $((i,p),(i,p')) \to ((i,p),(i+1,p'))$	$i \in [1,n-2]$, $p,p' \in [0,1]$

Theorem 1. *Given a population of n agents, $\mathcal{P}_{OS}$ is a self-stabilizing oscillator for the set of states S_{OS}, with amplitude $\iota_a = n$ and period $\iota_p = \Theta(n)$ active interactions.*

Proof. Let $\mathcal{E} = (C_0, r_0, C_1, r_1, \dots)$ be any execution of $\mathcal{P}_{OS}$ from any configuration C_0 of $\mathcal{P}_{OS}$, under a deterministic globally fair scheduler. Then a configuration $C_t \in \mathcal{C}_{LE}$ eventually yields by Proposition 1, and the theorem holds by Lemma 3. ■

Since the size of Q_{OS} is $4n-2$, the SS-OS problem is solvable by a PP with $4n-2$ states per agent. In order to generalize the result to any SS-LE protocol, recall that in [6], it has been shown that the exact information of the network size n is necessary and sufficient to solve the SS-LE problem. Since n

is known, any SS-LE protocol can be transformed into $\mathcal{P}_{LE}$ by using the same set of states and introducing new transition rules. Thus, our transformation can be applied.

As a note, $\mathcal{P}_{OS}$ can be generalized to solve the SS-OS problem of amplitude $\iota_a < n$. We modify δ_{CNT} in the following manner (we do not modify γ_{LE}): Recall that δ_{CNT} handles configurations in $\mathcal{C}_{LE}$. Without loss of generality, we assume that agent $i \in A \setminus \{0\}$ has a state (i, p) for some $p \in [0, 1]$ in C_0. The maximum value of the leader's counter c is set to ι_a.

Suppose that the leader interacts with an agent $i < \iota_a$. If $i = c$, which is the counter value of the leader, as in δ_{CNT}, the leader increments c, while agent $i(= c)$ updates its phase p to the leader's phase, as long as $c < \iota_a$ holds. When $c = \iota_a$, at the next interaction between the leader and any other agent, the leader re-initializes c to 1 and changes the phase.

Otherwise, when the leader interacts with an agent $i \geq \iota_a$, i updates its phase p to a default value, say 0. In addition, any interaction of i with the leader such that $i > \iota_a$ and $p = 0$ does not update the state of i. Since the scheduler is globally fair, every i with $i \geq \iota_a$ eventually interacts with the leader to fix its phase p to 0 forever.

Thus, exactly $\iota_a - 1$ agents toggle their phase with the leader. By defining S to be the set of states such that the phase of the agent is equal to 1, we obtain a self-stabilizing oscillator for the set of states S of amplitude ι_a and period $\iota_p = \Theta(\iota_a)$ active interactions.

3.2. Constructing SS-LE Protocol from any SS-OS Protocol

Let $\mathcal{P} = (Q, \delta)$ be any SS-oscillator on n agents for a set $S \subset Q$ and amplitude $\iota_a = n$. We assume $n \geq 3$. The case $n \leq 2$ is discussed separately. To construct an SS-LE protocol $\mathcal{P}^* = (Q^*, \delta^*)$ from $\mathcal{P}$, we first derive some properties of $\mathcal{P}$. Let $\mathcal{C}$ be the set of all configurations of $\mathcal{P}$. For any configuration $C \in \mathcal{C}$, by $|C|_S$, we denote the number of agents with a state in S in C. Since $\mathcal{P}$ is an SS-oscillator, every execution $\mathcal{E}$ eventually stabilizes and exhibits an oscillatory behavior of amplitude n.

We consider a sub-execution of $\mathcal{E}$ after it stabilizes, consisting of infinitely many increasing and decreasing phases. The set of configurations that constitutes an increasing phase may vary depending on the increasing phase under consideration, and the set of configurations that appear in some increasing phase may vary depending on the initial configuration of $\mathcal{E}$. For any $c \in [0, n]$, let $\mathcal{C}^{(c)}$ be the set of configurations $C \in \mathcal{C}$ that appears in some execution after stabilization and satisfies $|C|_S = c$. Note that $\mathcal{C}^{(c)}$ may not

contain all configurations such that $|C|_S = c$. Let $\mathcal{C}^+$ (resp., $\mathcal{C}^-$) be the set of all configurations that appear in an increasing (resp., decreasing) phase after stabilization, in an execution from some initial configuration. The above definitions introduce an ambiguity when $C \in \mathcal{C}^{(0)} \cup \mathcal{C}^{(n)}$, since C can belong both to $\mathcal{C}^+$ and to $\mathcal{C}^-$. To remove this ambiguity, we assume that C in $\mathcal{C}^{(0)} \cup \mathcal{C}^{(n)}$ is not a member of an increasing or a decreasing phase. That is, $1 \leq |C|_S \leq n-1$ for any $C \in \mathcal{C}^+ \cup \mathcal{C}^-$.

For any $C \in \mathcal{C}$, let R_C^- be the set of pairs (p, q) of states such that there is a pair of agents $r = (i, j) \in A^2$ satisfying that

1. $C(i) = p$ and $C(j) = q$, and
2. $|C'|_S < |C|_S$, where C' yields from C by the interaction r.

That is, R_C^- is the set of rules [2] applicable to C that decrease the number of agents in A with states in S. In the same manner, we define R_C^+ for any $C \in \mathcal{C}$, which is the set of rules applicable to C that increase the number of agents in A with states in S.

Lemma 4. 1. *For any $C \in \mathcal{C}^+$, R_C^- is empty.*
2. *For any $C \in \mathcal{C}^-$, R_C^+ is empty.*

Proof. We only show (1), since a proof of (2) is symmetrical. Assume that $r \in R_C^-$ for some $C \in \mathcal{C}^+$, to derive a contradiction. Then when the interaction r occurs, the increasing phase including C ends and the following decreasing phase starts before reaching a configuration in $\mathcal{C}^{(n)}$, which is a contradiction. ■

Let $R^- = \cup_{C \in \mathcal{C}^-} R_C^-$ and $R^+ = \cup_{C \in \mathcal{C}^+} R_C^+$. Consider any configuration $C_{-1} \in (\mathcal{C}^{(n-2)} \cup \mathcal{C}^{(n-1)})$ such that there is an interaction $r_{-1} = (i_{-1}, j_{-1}) \in A^2$ and a rule $(p_{-1}, q_{-1}) \in R^+$ satisfying that

1. $(p_{-1}, q_{-1}) = (C_{-1}(i_{-1}), C_{-1}(j_{-1}))$,
2. $\delta(p_{-1}, q_{-1}) \in S^2$, and
3. $C_0 \in \mathcal{C}^{(n)}$, where C_0 is the configuration generated from C_{-1} by interaction r_{-1}.

That is, $C_{-1} \in \mathcal{C}^+$ is a configuration immediately before achieving the amplitude n. Since C_{-1} is a configuration after stabilization, any execution

[2] Formally, a rule has a form "$(p', q') = \delta(p, q)$". We however abuse to call a pair (p, q) a rule when δ is obvious, since δ is a function and the rule can be identified by (p, q).

$\mathcal{E} = (C_0, r_0, C_1, r_1, \ldots)$ with initial configuration C_0 eventually starts the decreasing phase. Let C_{f+1} be the first configuration in $\mathcal{C}^-$. That is, $C_t \in \mathcal{C}^{(n)}$ for $t \in [0, f]$ ($f \geq 0$). Let $r_t = (i_t, j_t)$. Then obviously

1. $(p_t, q_t) \in S^2$ for any $t \in [0, f]$, where $(p_t, q_t) = (C(i_t), C(j_t))$
2. $\delta(p_t, q_t) \in S^2$ for any $t \in [0, f-1]$ if $f \geq 1$, and
3. $(p_f, q_f) \in R^-$, i.e., $\delta(p_f, q_f) \notin S^2$.

Given δ, since $|\mathcal{C}| < \infty$, we can construct a finite directed graph $DG_\delta = (\mathcal{C}, E_\delta)$ with edge labels. Here $((C, C'), r) \in E_\delta$ if and only if C' yields from C by the interaction r. Let $\Gamma_f = \{(C_f, r_f)\}$ be the set of the candidates (C_f, r_f).

3.2.1. Basic Idea

Let $G^{SC} = (V^{SC}, E^{SC})$ be the *component graph* of DG_δ. That is, V^{SC} is the set of strongly connected components of DG_δ, and $(\mathcal{C}_i, \mathcal{C}_j) \in E^{SC}$ if and only if there are configurations $C_i \in \mathcal{C}_i$ and $C_j \in \mathcal{C}_j$ such that $((C_i, C_j), r) \in DG_\delta$ for some r. Obviously G^{SC} is a DAG, and every execution of $\mathcal{P}$ eventually reaches a leaf $\mathcal{C}_\ell$ of G^{SC} and stays there forever, since the scheduler is globally fair. That is, every leaf of G^{SC} must represent oscillatory behavior. In $\mathcal{P}^*$, we use a set of markers to suppress some transitions, i.e., to remove some edges, to achieve our goal. Let us explain an outline of our idea.

When a configuration C_f where $(C_f, r_f) \in \Gamma_f$ is reached, we attach a marker L(eader) to agent i_f, if the corresponding transition from C_{-1} to C_0 has occurred. The effect of marker L is to suppress its participation to the execution. Hence transition $((C_f, C_{f+1}), r_f)$ cannot be executed and the state $C_f(i_f)$ of i_f, which is in S, is *freezed* as a result, i.e., it is not updated. If there are no other agents marked in C_f, the decreasing phase does not end because $C_f(i_f) \in S$ is freezed. Then the execution of $\mathcal{P}$ eventually elects a leader, in the sense that there is a time instant t_0 such that a unique agent i has a marker L in all configurations after t_0.

Formally, we memorize a marker L as a part of its state; if its current state is $p \in Q$, then the corresponding state is $(p, L) \in Q^*$. Generally, in $\mathcal{P}^*$, a state is a pair (p, m) of a state $p \in Q$ and a marker $m \in \Sigma$, where Σ is the set of markers we use in $\mathcal{P}^*$ and a marker F(ollower) $\in \Sigma$ denotes that no marker is attached.

Observe that the pair of interacting agents cannot recognize the current global configuration C from their current states. Hence, if we have the agents

attach a marker L to one of them whenever their state pair is $(C_f(i_f), C_f(j_f))$ in the execution, the execution would be led to a deadlock configuration containing more than one marked agents (since they freeze their states). One way to deal with this issue is to introduce a rule to discard L: When two marked agents interact, they discard both of their markers L. Then the execution can avoid deadlocks and may elect a leader by reaching a configuration with exactly one marker L. However, using this approach alone does not guarantee the convergence to a configuration with exactly one L, as markers L can be created and discarded infinitely often.

To solve this problem, we use the concept of non-deterministic choice, in which, whenever we need to make a decision, we guess a correct action and then verify its correctness. In $\mathcal{P}^*$, whenever interacting agents need to decide whether or not they should mark one of them, they non-deterministically decide. When they choose not to attach L, then the execution of $\mathcal{P}$ continues. When they decide to attach L, two cases should be considered: If $C = C_f$, then the execution eventually elects a leader. If $C \neq C_f$, then several agents might be marked. However, by using the previously introduced rule to delete the marks L, the execution can avoid deadlocks by removing all markers, if the number of marked agents is even. Otherwise, in the case where the number of marked agents is odd, the execution reaches a configuration C' with exactly one marked agent left. If the execution that starts from C' does not create a marker L, then the execution elects a leader. Otherwise, if it creates another marker L, then the execution of $\mathcal{P}$ continues.

This explanation of the basic idea shows that there is a chance for the execution to elect a leader, but does not explain the reason why the execution always elects a leader. We address this next. As mentioned previously, the execution $\mathcal{E}$ of $\mathcal{P}$ reaches a leaf $\mathcal{C}_\ell$ of G^{SC}, and stays in $\mathcal{C}_\ell$ forever, which implies that $\mathcal{E}$ passes every edge in the sub-graph of DG_δ induced by $\mathcal{C}_\ell$ infinitely many times, since the scheduler is globally fair. In particular, there is a pair $(C_f, r_f) \in \Gamma_f$ such that $C_f \in \mathcal{C}_\ell$, and $\mathcal{E}$ passes edge $((C_f, C_{f+1}), r_f)$ infinitely many times. When the agents i_f and j_f interact, they can choose either to mark or not to mark. Since $\mathcal{E}$ reaches C_f and select r_f as the interacting pair infinitely many times, the scheduler eventually chooses the branch leading to the leader election.

Let us explain in more details how we implement this non-deterministic execution.

(A) Implementing Non-deterministic Execution:

Consider i_f and j_f in C_f, whose states are $p = C_f(i_f)$ and $q = C_f(j_f)$, respectively. The rule $(p', q') = \delta(p, q)$ in R^- is applied, and their states are updated to p' and q' in C_{f+1} in $\mathcal{P}$. We design $\mathcal{P}^*$ so that i_f can non-deterministically choose one of two actions: to mark or not to mark. However, a direct implementation of $\delta^*((p, F), (q, F))$ must have two values, $((p, L), (q, F))$ to mark i_f and $((p', F), (q', F))$ not to mark, which is impossible as δ^* is a function. To overcome this issue, we introduce a new marker T(ransition) which declares that we do not choose to mark any of the two interacting agents. Now, for instance, we can define $\delta^*((p, F), (q, F)) = ((p, L), (q, F))$ and $\delta^*((p, T), (q, T)) = ((p', F), (q', F))$. We will explain later how to attach marker T to each of i_f and j_f, without changing the rest of C_f by the help of a third agent; thus, we assume that the number of agents n in the population is at least 3.

As a note, Angluin et al. [4] proposed a general method for eliminating non-deterministic transitions for a large family of problems called *elastic problems*. Their transformation uses a non-determinizer marker of two values ($\circ$ or $-$) and a choice counter $c \in [0, x-1]$, where $x = 2$ in our case. That is, in order to implement $\mathcal{P}^*$ by using the transformation in [4], we need six markers (including L and F), instead of three (L, T and F) in our case .

(B) Discarding Redundant Markers: We have explained a rough idea on how to create markers and how to use them to control the execution of $\mathcal{P}$. However, we need to discuss harmful effects caused by abuses of marker creations. As we have already explained, we can discard redundant Ls; two marked agents discard both of their markers L, when they interact.

A marker T is used to declare not to create a marker L and hence not to suppress a transition. Unlike L, T is not discarded until it participates in a transition to update its state; no T marker is redundant in this sense.

3.2.2. Protocol $\mathcal{P}^*$

In this subsection, we construct $\mathcal{P}^* = (Q^*, \delta^*)$ from an SS-OS protocol $\mathcal{P} = (Q, \delta)$, where $Q^* = Q \times \Sigma$ and $\Sigma = \{L, T, F\}$. In the rest of this subsection, we always assume that $(p', q') = \delta(p, q)$, where $(q', p') = \delta(q, p)$ may not hold. When we use "$\Rightarrow$" (instead of "$\rightarrow$"), $((p, \sigma), (q, \rho)) \Rightarrow ((p', \sigma'), (q', \rho'))$ means not only $((p', \sigma'), (q', \rho')) = \delta^*((p, \sigma), (q, \rho))$ but also $((p', \rho'), (q', \sigma')) = \delta^*((p, \rho), (q, \sigma))$. We say that $(p, \sigma) \in Q^*$ is *bad* if $p \notin S$ and $\sigma \in \{L, T\}$. Otherwise, we say that (p, σ) is *good*. Let $GOOD$ and BAD be the sets of

good and bad states, respectively. The next rule discards all bad states from a given configuration C.

Protocol 4 Discard bad states.

0. $((p,\sigma),(q,\rho)) \to ((p,\sigma'),(q,\rho'))$	$\sigma' = F$ if $(p,\sigma) \in BAD$ $\rho' = F$ if $(q,\rho) \in BAD$

Without loss of generality, we can assume that every execution does not contain a configuration with a bad state, since δ^* does not introduce a new bad state as one can easily observe from the following definition of δ^*. We only describe δ^* when both of the arguments are good.

The most important rules in δ^* are indeed inactive, and we do not need to describe them by convention. However, we explicitly state them because of their importance.

Protocol 5 Freeze interaction.

1a. $((p,L),(q,T)) \Rightarrow ((p,L),(q,T))$	$p,q \in S$
1b. $((p,L),(q,F)) \to ((p,L),(q,F))$	$p \in S$, $q \notin S$
1c. $((p,F),(q,L)) \to ((p,F),(q,L))$	$p \notin S$, $q \in S$

The leader can create a marker T.

Protocol 6 Assign T using L.

1d. $((p,L),(q,F)) \Rightarrow ((p,L),(q,T))$	$p,q \in S$

As mentioned previously, when two agents with a marker L interact, they both discard their markers.

Protocol 7 Discard L.

1e. $((p,L),(q,L)) \to ((p,F),(q,F))$	$p,q \in S$

Rule 1 enumerates all transitions in which L is involved. It is easy to observe the next property.

Property 1. *Once an agent is in state (p,L), its state does not change unless L is discarded by Rule 1e.*

Let $\mathcal{S}_L$ be the set of pairs $(p, q) \in S^2$ such that $(C_f(i_f), C_f(j_f))$ is (p, q) for some $(C_f, r_f) \in \Gamma_f$. Marker L is created by the following rule.

Protocol 8 Create L.

2. $((p, T), (q, F)) \Rightarrow ((p, L), (q, F))$	$(p, q) \in \mathcal{S}_L$

To create L, we need a marker T. We explain in the following how to create a marker T without using L, i.e., without applying Rule 1d.

Protocol 9 Create and transfer T.

3a. $((p, F), (q, F)) \to ((p, T), (q, F))$	$p \in S$
3b. $((p, F), (q, F)) \to ((p, F), (q, T))$	$p \notin S$, $q \in S$
3c. $((p, T), (q, F)) \Rightarrow ((p, T), (q, T))$	$p, q \in S$, $(p, q) \notin \mathcal{S}_L$

Property 2. *If $p \in S$, any agent with a state (p, F) can change its state to (p, T), possibly by changing only markers of the agents.*

Proof. Rule 3 guarantees that the lemma holds, if there is another agent with a state (q, F) or (q, T) with $(p, q) \notin \mathcal{S}_L$ for some state $q \in S$. Similarly, Rule 1d ensures that the lemma holds if there is another agent with a state (q, L) for some $q \in S$.

Recall that we have assumed $n \geq 3$. Let i, j and k be three agents, and assume that the state of i is (p, F). Let (q, σ) and (r, ρ) be the states of j and k, respectively. The remaining case worth considering is when $\sigma = T$ with $(p, q) \in \mathcal{S}_L$ and $\rho = T$ with $(p, r) \in \mathcal{S}_L$. Interacting with i, by Rule 2, k can change its marker to L and then j can also change its marker to L by the same rule through an interaction with i. Next, both j and k discard their markers by Rule 1e, which makes it possible for i to update its state (p, T) by Rule 3a.

■

We next define how to simulate a transition $(p', q') = \delta(p, q)$. Observe that if an agent has a marker T, the marker is discarded.

Protocol 10 Simulate δ.

4. $((p, \sigma), (q, \rho)) \to ((p', F), (q', F))$	$\sigma = T$ ($\rho = T$) if $p \in S$ ($q \in S$)

Lemma 5. *Let (p, σ) and (q, ρ) be the states of agents i and j, respectively. Suppose that $\sigma = L$ if and only if $\rho = L$. Then they can change their states respectively to (p', F) and (q', F), possibly by changing only markers of the agents, when $(p', q') = \delta(p, q)$.*

Proof. Since agents i and j can change their markers to F by Rule 1e if $\sigma = \rho = L$, we assume $\sigma, \rho \in \{T, F\}$ without loss of generality. It is sufficient to show that i (resp. j) can obtain a marker T if p (resp. q) is in S, to apply Rule 4. If $\{p, q\} \not\subseteq S$ or $(p, q) \notin \mathcal{S}_L$, by Rule 3, each of the agents can obtain a marker T when its (first element of) state is in S.

Let us thus concentrate on the case $(p, q) \in \mathcal{S}_L$, provided $(\sigma, \rho) \neq (T, T)$ (since otherwise Rule 4 is applicable). As assumed, there is a third agent k whose state is (r, ϱ). If $\varrho \in \{L, F\}$, then i and j can obtain a marker T by Rule 1d or 3a if necessary. If $\varrho = T$ and $(p, r) \notin \mathcal{S}_L$ (resp. $(q, r) \notin \mathcal{S}_L$), then i (resp. j) can obtain a marker by Rule 3c.

Now the case in which $\{(p, q), (p, r), (q, r)\} \subseteq \mathcal{S}_L$ holds remains. First, interacting with i, j changes its marker to L by Rule 2 (after changing ρ to T by Rule 3a when $\rho = F$). Next, interacting with i, k then updates its marker to L by Rule 2. Finally, j and k change their markers L to F by Rule 1e. Now we retrieve the case $\varrho = F$ we investigated above. ■

Property 3. *δ^* is a function.*

3.2.3. Correctness of $\mathcal{P}^*$

We show the correctness of Protocol $\mathcal{P}^*$. Let $\mathcal{C}^*$ be the set of all configurations of $\mathcal{P}^*$. That is, for any $C^* \in \mathcal{C}^*$ and $i \in A$, $C^*(i) = (p, \sigma)$, where $p \in Q$ and $\sigma \in \Sigma = \{L, T, F\}$. For C^*, there is the corresponding configuration $C \in \mathcal{C}$ of $\mathcal{P}$, where for any $i \in A$, $C(i) = p$ if $C^*(i) = (p, \sigma)$ for some $\sigma \in \Sigma$. That is, C is constructed from C^* by removing markers. By $C^* \simeq C$ we denote that C^* is constructed from C by assigning for each agent a marker in Σ.

Lemma 6. *Let C^* and D^* be two configurations of $\mathcal{P}^*$ and assume that $C^* \to D^*$. Then either $C = D$ or $C \to D$, where $C \simeq C^*$ and $D \simeq D^*$.*

Proof. Observe that only Rule 4 can change the (first component of) state of an agent by the definition of δ^*. If $C \neq D$, then Rule 4 applies to C^*, which implies $C \to D$. ■

Any execution $\mathcal{E}^* = (C_0^*, r_0, C_1^*, r_1, \ldots)$ of $\mathcal{P}^*$ thus "simulates" an execution $\mathcal{E} = (C_0, r_0, C_1, r_1, \ldots)$ of $\mathcal{P}$, if we regard the rule r_t as an inactive one,

whenenver it is not in Rule 4 in δ^*. We investigate how a given execution $\mathcal{E}$ can be simulated by an execution $\mathcal{E}^*$ in the following.

Let C and D be two configurations of $\mathcal{P}$ such that $C \to D$, and assume that $C^* \simeq C$ and $D^* \simeq D$, where no agent has a marker L in C^* and D^*. Then there exists an execution $C_0^*(= C^*) \to C_1^* \to \cdots \to C_k^*(= D^*)$ such that $C_i^* \simeq C$ for all $i \in [0, k-1]$ by Lemma 5. (Note that C_i^* may contain a marker L for some $i \in [1, k-1]$.) We have the following corollary of Lemma 5.

Corollary 1. *Let $\mathcal{E} = (C_0, r_0, C_1, r_1, \ldots)$ be any execution of $\mathcal{P}$ and let C_0^* be a configuration in $\mathcal{C}^*$ such that $C_0^* \simeq C_0$ and it does not contain an agent having a marker L. Then there is an execution $\mathcal{E}^* = (C_0^*, r_0^*, C_1^*, r_1^*, \ldots)$ that simulates $\mathcal{E}$ in the following sense: There is a strictly increasing sequence $t_0(= 0), t_1, \ldots$ such that, for all $k \in \mathbb{N}$,*

1. *$C_h^* \simeq C_k$ for all $h \in [t_k + 1, t_{k+1} - 1]$, in particular, $C_{t_k}^* \simeq C_k$,*
2. *$C_{t_k}^*$ does not contain an agent having a marker L, and*
3. *all agents in A_k have a marker F in $C_{t_k}^*$, where A_k is the set of agents that participate in an interaction r_h for some $h \in [0, k-1]$.*

Proof. The existence of an execution $\mathcal{E}^*$ satisfying Items (1) and (2) is obvious by Lemma 5. We concentrate on Item (3).

In order to apply Rule 4, an agent i whose state (p, σ) needs to change its marker to T, when $p \in S$ and $\sigma = F$. Recall the proofs of Property 2 and Lemma 7. To change the marker of i from F to T, another agent may needs to change its marker, but, fortuntely, from T or L to F. We have each agent change its marker to T, only immediately before Rule 4 applies. Then $\mathcal{E}^*$ satisfies Item (3). ■

Let $G^{SC} = (V^{SC}, E^{SC})$ be the component graph of DG_{δ^*} (not of DG_δ). Let $\mathcal{E}^* = (C_0^*, r_0, C_1^*, r_1, \ldots)$ be any execution of $\mathcal{P}^*$. Then $\mathcal{E}^*$ reaches a leaf $\mathcal{C}_\ell^*$ of G^{SC} and stays there forever. We say that a leaf $\mathcal{C}^*$ is *LE* if the following two conditions hold:

1. Every configuration of $\mathcal{C}^*$ contains exactly one agent with a marker L.
2. The same agent has a marker L in every configuration of $\mathcal{C}^*$.

If $\mathcal{C}_\ell^*$ is LE then the leader election has been solved. We show this fact in the sequel. We start with the following lemma.

Lemma 7. *There is a time instant t_0 such that sub-execution $\mathcal{E}_t^* = (C_t^*, r_t, C_{t+1}^*, r_{t+1}, \ldots)$ does not contain a bad state.*

Proof. Let t be the time instant that $\mathcal{E}^*$ reaches a leaf $\mathcal{C}_\ell^*$ of G^{SC}. Since (1) δ^* does not create a new bad state by definition, (2) Rule 0 is always applicable to reduce the number of bad states in a configuration when there is a bad state, (3) the sub-graph of DG_{δ^*} induced by $\mathcal{C}_\ell^*$ is strongly connected, and (4) the scheduler is globally fair, every configuration in $\mathcal{C}_\ell^*$ does not contain a bad state. Hence $\mathcal{E}_t^* = (C_t^*, r_t, C_{t+1}^*, r_{t+1}, \ldots)$ does not contain a bad state. ■

We thus assume that C_0^* does not contain a bad state, and hence $\mathcal{E}^*$ does not contain a configuration that contains a bad state, without loss of generality.

Lemma 8. *Leaf $\mathcal{C}_\ell^*$ is LE.*

Proof. To derive a contradiction, we assume that $\mathcal{C}_\ell^*$ is not LE. If every configuration of $\mathcal{C}_\ell^*$ contains exactly one agent with a marker L, then the same agent has a marker L in every configuration of $\mathcal{C}_\ell^*$, and $\mathcal{C}_\ell^*$ is LE. The number of agents with a marker L in a configuration of $\mathcal{C}_\ell^*$ is therefore not unique.

We first observe that there is a configuration D_0^* containing no agent with a marker L in $\mathcal{C}_\ell^*$. Let k_0 be the number of agents with a marker L in D_0^*, and assume that $k_0 > 0$. Without loss of generality, we can assume that k_0 is 1, since otherwise Rule 1e is applicable to reduce k_0. There is a configuration D_1^* with $k_1 (\neq k_0)$ agents having a marker L. Obviously $k_1 > k_0$. Since a marker L is created one by one by Rule 2, there is a configuration D_3^* containing two agents with a marker L, which implies that there is a configuration D_4^* containing no agent with a marker L by Rule 1e.

In the following, we show that there is a configuration F^* reachable from D_0^* such that every configuration reachable from F^* (including F^* itself) has exaclty one agent with a marker L. Then a contradiction is derived, since D_0^* is not reachable from F^*, and hence $\mathcal{C}_\ell^*$ is not a leaf. Let $D_0 \simeq D_0^*$. Since $\mathcal{P}$ is an SS-OS protocol, any execution $\mathcal{E}$ starting from D_0 eventually stabilizes and repeats oscillations (i.e., increasing and decreasing phases) infinitely many times. Suppose that $\mathcal{E}$ reaches a configuration C_f such that $(C_f, r_f) \in \Gamma_f$ for some r_f, when it is starting a decreasing phase (after at least one oscillation). Let $\mathcal{E}_D$ be a longest execution from C_f such that agent i_f is not a part of an interaction in it, where $r_f = (i_f, j_f)$. Since $C_f(i_f) \in S$, $\mathcal{E}_D$ forms a part of the decreasing phase and hence is finite. Let F be the last configuration in $\mathcal{E}_D$. Then all interactions in F are inactive unless i_f is involved.

Consider the following execution $\mathcal{E}^*$ from D_0^* corresponding to $\mathcal{E}$ with $\mathcal{E}_D$ as a subexecution. By Corollary 1, $\mathcal{E}^*$ reaches a configuration C_f^* such that $C_f^* \simeq C_f$ and all agents have a marker F (since $\mathcal{E}$ includes at least one oscillation before C_f). Since $(C_f(i_f), C_f(j_f)) \in \mathcal{S}_L$ is applicable to C_f, i_f can change its marker from F to T by Rule 3a and then to L by Rule 2. Although $C_f(i_f)$ is freezed, since i_f is not part of an interaction in $\mathcal{E}_D$, $\mathcal{P}^*$ can simulate $\mathcal{E}_D$ by Corollary 1 and $\mathcal{E}^*$ reaches a configuration F^* corresponding to F. Observe that $F^* \simeq F$ and i_f is the only agent with a marker L. By the definition of $\mathcal{E}_D$, all interactions in F^* include i_f and hence inactive, which implies that every configuration reachable from F^* contains exactly one agent with a marker L. ■

By Lemma 8, for any initial configuration $C_0^* \in \mathcal{C}^*$, every execution $\mathcal{E}^*$ with initial configuration C_0^* eventually reaches a leaf $\mathcal{C}_\ell^*$ of G^{SC}, which is LE, and hence elects a leader. We conclude the following theorem.

Theorem 2. *Given a population protocol of $n \geq 3$ agents, $\mathcal{P}^*$ solves the self-stabilizing leader election problem.*

We next present an SS-LE protocol $\mathcal{P}^* = (Q^*, \delta^*)$, given an SS-OS protocol $\mathcal{P} = (Q, \delta)$, when $n = 2$. We use two markers L and F. Then $Q^* = Q \times \Sigma$, where $\Sigma = \{L, F\}$. Transition function δ^* is described in the following.

Protocol 11 δ^*

a. $((p, F), (q, F)) \to ((p, L), (q, F))$	$p, q \in Q$
b. $((p, L), (q, L)) \to ((p, L), (q, F))$	$p, q \in Q$
c. $((p, L), (q, F)) \Rightarrow ((p, L), (q, F))$	$p, q \in Q$

Since every agent is part of an interaction at each time, it is obvious to observe the following property.

Property 4. *Protocol $\mathcal{P}^*$ solves the SS-LE problem, when $n = 2$.*

4. Oscillations under Uniform Random Scheduler

In Subsection 3.1, we construct an SS-OS protocol $\mathcal{P}_{OS}$ from an SS-LE protocol $\mathcal{P}_{LE}$, but it requires $4n - 2$ states per agent, where n is the size of the population. In this section, aiming at the reduction of the number of states, we investigate the SS-OS problem under a uniform random scheduler, which selects a pair of agents uniformly at random from the set of all the

pairs of distinct agents in the population. Let us first define a self-stabilizing oscillator under such a random scheduler.

Definition 6. ***(SS-Oscillator under a uniform random scheduler)*** *A (sufficiently large) population of agents executing a deterministic protocol $\mathcal{P}$ is a self-stabilizing oscillator under a uniform random scheduler, if starting from any configuration $C_0 \in \mathcal{C}$, any execution $\mathcal{E} = (C_0, r_0, C_1, r_1, \ldots)$ of $\mathcal{P}$ eventually reaches a configuration $C_t \in \mathcal{C}$, from which any execution $\mathcal{E}_t = (C_t, r_t, C_{t+1}, r_{t+1}, \ldots)$ exhibits an oscillatory behavior for the set of states S with an expected average amplitude ι_a and an expected average period ι_p.*

We present three deterministic protocols $\mathcal{PO}_I = (Q_k, \delta_I^k)$ $(I = 1, 2, 3)$ with a parameter $k \in \mathbb{N}$ to realize an SS-oscillator under a uniform random scheduler, each of which is a modification of $\mathcal{P}_{OS}$. Note that under a uniform random scheduler, $\mathcal{P}_{OS}$ itself correctly works to realize an SS-oscillator under a uniform random scheduler.

The three protocols $\mathcal{PO}_I$ commonly use the set of states Q_k defined as $Q_k = Q_L^k \cup Q_F$, where $Q_F = \{(i, p) : i \in [1, n-1],\ p \in [0, 1]\}$ is the set of states for followers as is defined in Q_{OS} and $Q_L^k = \{(i, p, c) : i = 0,\ p \in [0, 1],\ c \in [0, k-1]\}$ is the set of states of Q_k for the leader. Recall that the first component i of each state in Q_k indicates its name, where 0 is the leader, and the second component p indicates its phase. Since the number of states $|Q_k|$ is now $2(n + k - 1)$, a reduction of space complexity from $4n - 2$ to $2(n + k - 1)$ is achieved when $k < n$. A small difference here is that the counter value c starts from 0 (not 1), unlike $\mathcal{P}_{OS}$, for the convenience of protocol description.

The set of rules δ_I^k of each protocol $\mathcal{PO}_I$ commonly contains γ_{LE} as a subset (after changing the range of the counter from $[1, k]$ to $[0, k-1]$). That is, $\delta_I^k = \gamma_{LE} \cup \gamma_I^k$, where γ_I^k defines the result of an interaction between the leader and a follower, and like δ_{CNT}, (approximately) counts, in a random way, the number of followers with the same phase as the leader. We describe γ_I^k in the following to present $\mathcal{PO}_I$ and analyze its performance. As a result, we show that, for $\mathcal{PO}_1, \mathcal{PO}_2$ and $\mathcal{PO}_3$, $k = \sqrt{n}, \log n$ and 2 are respectively enough to realize an SS-oscillator under a uniform random scheduler, but for $\mathcal{PO}_1$ and $\mathcal{PO}_3$, at the expense of a lower average amplitude or a longer average period.

Finally, let $S = \{(0, 1, c) : c \in [0, k-1]\} \cup \{(i, 1) : i \in [1, n-1]\}$, be the set of all states with $p = 1$.

In the rest of this section, we analyze executions $\mathcal{E}$ generated by $\mathcal{PO}_I$ under a uniform random scheduler. We would like to make three remarks: First, since the analyses are based on probabilistic arguments, we always assume that the number n of agents in the population is sufficiently large. Second, since every $\mathcal{E}$ eventually reaches a configuration C in which the first component i of the state $C(j)$ of each agent $j \in A$ is distinct (and this correspondence between $j \in A$ and $i \in [0, n-1]$ is fixed forever) by Proposition 1, we always assume that all configurations in $\mathcal{E}$ satisfy this property. Finally, in the following analysis of performance, as assumed, an execution $\mathcal{E}$ consists only of active interactions.

4.1. Protocol $\mathcal{PO}_1$

Protocol $\mathcal{PO}_1$ presented in Protocol 12 works as follows: Suppose that the leader with state $(0, p, c)$ interacts with a follower with state (i, p). If $c < k-1$, the leader simply increments its counter. Otherwise, if $c = k-1$, it toggles its phase and re-initializes its counter c to 0; the follower does not update its state.

Suppose by contrast that the leader with state $(0, p, c)$ interacts with a follower with state $(i, 1-p)$. Then, regardless of c, the leader re-initializes its counter c to 0, while the follower flips its phase to the leader's phase p.

Protocol 12 γ_1^k

1. $((0,p,c),(i,p)) \to ((0,p,c+1),(i,p))$ $\quad i \in [1,n-1], p \in [0,1], c \in [0,k-2]$
2. $((0,p,k-1),(i,p)) \to ((0,1-p,0),(i,p))$ $\quad i \in [1,n-1], p \in [0,1]$
3. $((0,p,c),(i,1-p)) \to ((0,p,0),(i,p))$ $\quad i \in [1,n-1], p \in [0,1], c \in [0,k-1]$

We start analyzing the performance of $\mathcal{PO}_1$.

Lemma 9. *Suppose $k \gg \log n$.*

1. *When Rule 2 is applied to toggle the phase from p to $1-p$ in a configuration C, the number of followers with phase $1-p$ in C is $O((n/k)\log n)$ with high probability.*
2. *The period is at most $2k(n-1)$ active interactions.*

Proof. To prove Claim (1), suppose without loss of generality that the leader's phase is 0, and there are initially $n-1$ followers with phase 1 (which constitutes the worst case). Let X be a random variable to represent the number of followers with phase 1 at the end of this phase, i.e., when Rule 2 is applied for the first time. Let $P(x)$ be the probability that Rule 2 is applied when there are x followers with phase 1. Then

$$P(x) = \prod_{j=x+1}^{n-1} (1-(1-j/(n-1))^k)(1-x/(n-1))^k \leq (1-x/(n-1))^k,$$

since the probability that the counter re-initializes to 0 when there are j agents with phase 1 is $(1-j/(n-1))^k$.

Let $E = E(X)$ be the expected value of X, i.e.,

$$E = \sum_{x=0}^{n-1} xP(x).$$

Let $\omega = \omega(n)$ be some slowly growing function of n, and take $x \geq \omega(n-1)/k$. Then

$$P(x) \leq (1-x/(n-1))^k \leq e^{-kx/(n-1)} \leq e^{-\omega}.$$

Put $\omega = 2\log(n-1)$. Then the contribution to E from $\omega(n-1)/k \leq x \leq n-1$ is

$$\sum_{x=\omega(n-1)/k}^{(n-1)} xP(x) \leq (n-1)^2 e^{-2\log(n-1)} = 1,$$

which implies that

$$E \leq 1 + [(2\log(n-1))(n-1)/k] \sum_{x \leq (2\log(n-1))\ (n-1)/k} P(x)$$

$$= 1 + (2\log(n-1))(n-1)/k < (3\log(n-1))(n-1)/k.$$

For any small $\epsilon > 0$, put $a = \epsilon^{-1}(3\log(n-1))((n-1)/k)$. By the Markov inequality,

$$P(X \geq a) \leq E/a \leq \epsilon.$$

As for the proof of Claim (2), observe that re-initializations occur at most $n-1$ times in a phase, and that between two re-initializations, there are at most k active interactions with followers having the same phase. Thus, the length of a phase is at most $k(n-1)$ active interactions, and then (2) holds. ■

Theorem 3. *Protocol $\mathcal{PO}_1$ realizes an SS-oscillator under a uniform random scheduler for the set of states S, when parameter k is set to a value sufficiently larger than* $\log n$.

Proof. By Lemma 9, the theorem is now obvious. ■

We present in Figure 5 some simulation results of a population of 1000 agents executing $\mathcal{PO}_1$ for different values of k.

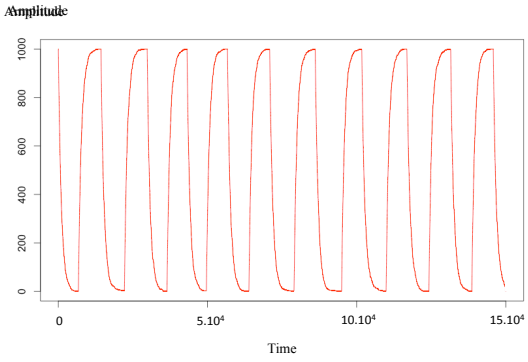

Figure 1: The case where k=n.

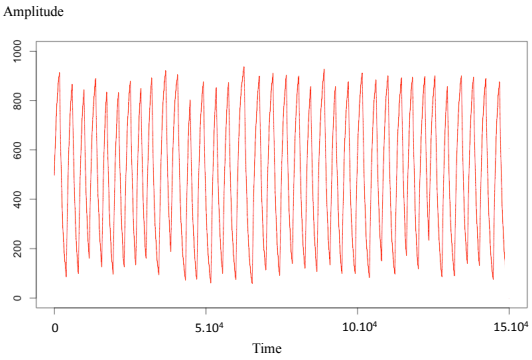

Figure 2: The case where $k = \sqrt{n}$.

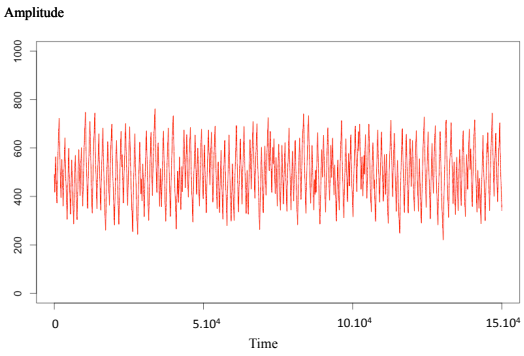

Figure 3: The case where $k = log(n)$.

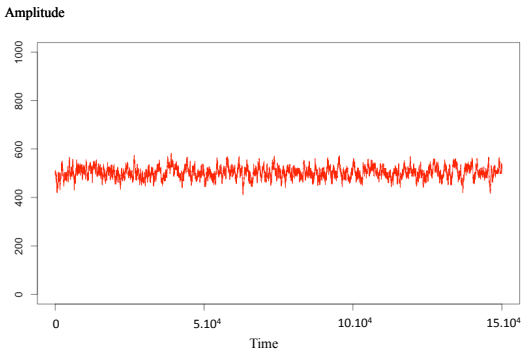

Figure 4: The case where $k = log(log\ n)$

Figure 5: Oscillatory behaviors by a population of 1000 agents executing $\mathcal{PO}_1$ for different values of k.

As we can observe from Figure 5, the population indeed exhibits an os-

cillatory behavior for the values of $k \gg log(n)$, i.e., for $k = n$ and $\sqrt{n}$, which validates the theoretical results obtained.

4.2. Protocol $\mathcal{PO}_2$

Aiming at reducing even more the space complexity, we propose in the sequel, two population protocols $\mathcal{PO}_2$ and $\mathcal{PO}_3$ that solve the SS-OS problem under a uniform random scheduler and that use, in addition to the leader, another agent that we call *marked agent*. This agent is the one with name 1.

Protocol $\mathcal{PO}_2$ presented in Protocol 13 works as follows: Suppose that the leader with state $(0, p, c)$ interacts with the marked agent with state $(1, q)$. If $c < k-1$ the leader increments its counter. Otherwise, if $c = k-1$, the leader toggles its phase and re-initializes its counter c to 0. Upon interacting with the leader, the marked agent also updates its phase to the one of the leader. Suppose now that the leader with state $(0, p, c)$ interacts with a follower with state $(i, 1-p)$. Then the follower updates its phase to the one of the leader.

In $\mathcal{PO}_2$, the leader simply counts the number of interactions with the marked agent. It toggles its phase after k interactions with the marked agent.

Protocol 13 γ_2^k

1. $((0, p, c), (1, q)) \to ((0, p, c+1), (1, p))$ $\qquad p, q \in [0, 1],\ c \in [0, k-2]$

2. $((0, p, k-1), (1, q)) \to ((0, 1-p, 0), (1, 1-p))$ $\qquad p, q \in [0, 1]$

3. $((0, p, c), (i, 1-p)) \to ((0, p, c), (i, p))$ $\qquad i \in [2, n-1],\ p \in [0, 1],\ c \in [0, k-1]$

We analyze the performance of $\mathcal{PO}_2$.

Theorem 4. *Protocol $\mathcal{PO}_2$ realizes an SS-oscillator under the uniform random scheduler for the set of states S, when parameter k is set to $\log n$. Its average amplitude and period are respectively n and $k(n-1)$.*

Proof. We first investigate the expected period. Let $X(i, i+1)$ be a random variable that represents the number of active interactions in order for c to be incremented from i to $i+1$. Recall that the leader's counter is only incremented by Rule 1, when the leader interacts with the marked agent, that is, $X(i, i+1)$ has a geometric distribution of parameter p:

$$P(X(i, i+1) = m) = (1-p)^{m-1}p,$$

where $p = 1/(n-1)$ is the probability that a given active interaction is a one between the leader and the marked agent. Let

$$X = \sum_{i=0}^{k-1} X(i, i+1)$$

be a random variable that represents the number of active interactions for the leader to toggle its phase to start the next phase of the oscillation. Then by the linearity of the expectations, we obtain

$$E[X] = E[\sum_{i=0}^{k-1} X(i, i+1)] = k/p = k(n-1),$$

since $E[X(i, i+1)] = 1/p$.

Assume without loss of generality that the leader's phase is equal to 0. We determine the expected amplitude ι_a. Let $B(t)$ denotes the number of followers with phase 1 at time t. The expected number of followers with phase 1 at time $t+1$, i.e., $B(t+1)$, is given by

$$B(t+1) = B(t) - \frac{B(t)}{n-1}.$$

That is, at time $t+1$, the number of followers with phase 1 either remains the same or decreases when there is an active interaction between the leader and such a follower. Approximately we have

$$\frac{B(t)}{dt} = -\frac{1}{n-1} B(t).$$

Hence

$$B(t) = B(0) e^{-(\frac{t}{n-1})}.$$

Recall that we know the expected number of interactions before reaching the amplitude. By replacing t by $E[X]$, we obtain the expected number A of followers with phase 0 when the amplitude is reached. That is,

$$A(E[X]) = n - B(0) e^{-k\frac{n-1}{n-1}} \simeq n(1 - e^{-k}).$$

For $k = log(n)$, $A(E[X]) = n - 1$, which implies that the average amplitude is n, since the leader is with phase 0. ■

We present in Figure 6 simulation results of a population of 1000 agents executing $\mathcal{PO}_2$ for $k = \log n$.

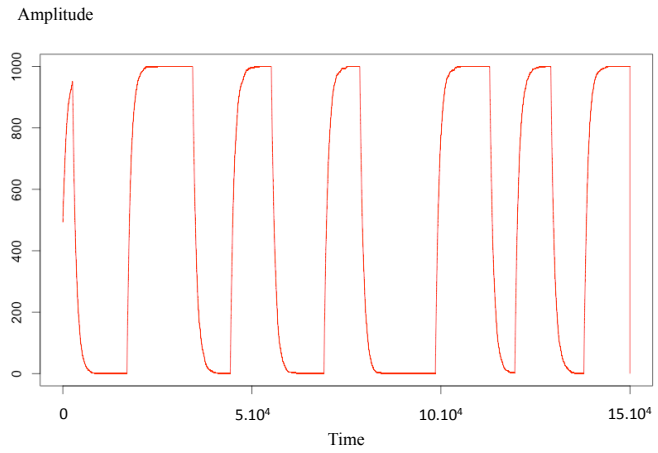

Figure 6: Oscillatory behavior by a population of 1000 agents executing $\mathcal{PO}_2$ for $k = \log n$.

Protocol $\mathcal{PO}_2$ uses less number of states than $\mathcal{PO}_1$ and achieves the average amplitude n with the same average period as $\mathcal{PO}_1$; $\mathcal{PO}_2$ is definitely better than $\mathcal{PO}_1$.

4.3. Protocol $\mathcal{PO}_3$

As the third protocol, we present $\mathcal{PO}_3$ and analyze its performance. Our aim is to solve the SS-OS problem under a uniform random scheduler with a counter of a constant size, i.e., the counter size k does not depend on the population size n. The idea of $\mathcal{PO}_3$ is similar to $\mathcal{PO}_3$ except that, whenever the leader interacts with a follower, it re-initializes its counter, regardless of the phase of the follower. Observe that the leader needs to interact k consecutive times with the marked agent in order to toggle its phase. The set of rules γ_3^k for $\mathcal{PO}_3$ is presented in Protocol 14.

The dynamics of the leader's counter value can be represented by a Markov chain shown in Figure 7, where state j represents the state in which the leader has the counter value $c = j$, $p = 1/(n-1)$ is the probability that a given active interaction is between the leader and the marked agent (and hence the counter is incremented by Rule 1), and $q = 1 - p$ is the probability

Protocol 14 γ_3^k

1. $((0,p,c),(1,q)) \to ((0,p,c+1),(1,p))$ $\qquad p,q \in [0,1],\ c \in [0,k-2]$

2. $((0,p,c),(i,q)) \to ((0,p,0),(i,p))$ $\qquad i \in [2,n-1],\ p,q \in [0,1]\ c \in [0,k-1]$

3. $((0,p,k-1),(1,q)) \to ((0,1-p,0),(1,1-p))$ $\qquad p,q \in [0,1]$

that a given active interaction is between the leader and a follower (and hence the counter is re-initialized to 0 by Rule 2). Finally, state k [3] denotes the state that is reached when the leader toggles its phase by Rule 3; it interacts with the marked agent when $c = k-1$.

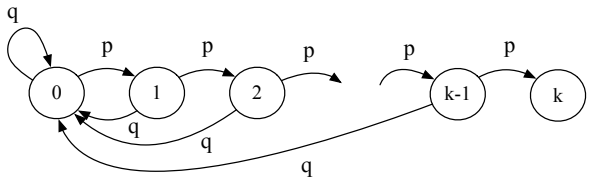

Figure 7: The Markov chain corresponding to $\mathcal{PO}_3$.

Theorem 5. *Protocol $\mathcal{PO}_3$ realizes an SS-oscillator under a uniform random scheduler for the set of states S, when parameter k is set to 2. Its average amplitude and period are respectively n and $(n-1)^2$.*

Proof. The average number of active interactions to reach node j (for the first time) in the Markov chain presented in Figure 7, starting from node 0, is denoted by H_i; in other words, H_j is the average hitting time of state j from state 0.

Obviously $H_0 = 0$, and it is easy to observe that

$$H_{j+1} = p^{-1}(H_j + 1)$$

for all $j \in [0, k-1]$. By a simple calculation, H_k, which is the average number

[3] State k is only used to show that one more interaction is needed for the leader to toggle its phase.

of active interactions necessary for the leader to toggle its phase, is

$$H_k = p^{-k}H_0 + \sum_{j=1}^{k} p^{-j} = \frac{p^{-k}-1}{q} \simeq (n-1)^k.$$

That is, the average period is $2H_k = 2(n-1)^k$.

As for the average amplitude, let $k \geq 2$ be a constant. Then $(n-1)\log n < (n-1)^k$ for all large n. By the argument in the proof of Theorem 4, the average amplitude is n. ■

We present in Figure 8 simulation results of a population of 1000 agents executing $\mathcal{PO}_3$ for $k = 2$. From the simulation results, we can observe that the periods of the oscillations are really variable, which probably reflects a large standard deviation.

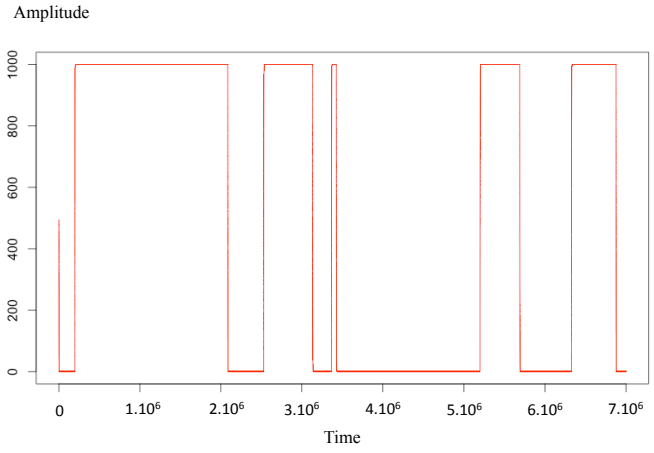

Figure 8: Oscillatory behavior by a population of 1000 agents executing $\mathcal{PO}_3$ for $k = 2$.

Protocol $\mathcal{PO}_3$ uses less number of states than $\mathcal{PO}_2$ and achieve the average amplitude n, but its average period $(n-1)^k$ is worse than $\mathcal{PO}_2$.

5. Conclusion

In this paper, we have addressed the problem of autonomously generating oscillatory executions in deterministic population protocols, and showed

that, under a deterministic globally fair scheduler, $\lceil \log n \rceil - 2$ bits per agent are necessary to solve the self-stabilizing oscillation (SS-OS) problem. This result emphasizes somehow the impact and the importance of randomization in biological systems and chemical reactions in creating self-oscillations. Recall that in order to prove the result, we have shown a relationship between the self-stabilizing leader election problem (SS-LE) and the SS-OS problem. That is, it is possible to solve the SS-OS problem if the SS-LE problem is solvable and vise versa. We have used in our solution a non-deterministic approach implemented by using markers. Solving the problem without resorting to non-determinism remains an open question.

Next, aiming at the reduction of the space complexity, we have proposed some protocols that solve the problem assuming a random scheduler. This is a preliminary work as several open questions arise about the SS-OS problem under the random scheduler:

1. All the proposed solutions in this paper assume a central control, that is, the agents first need to elect a leader in order to create the desired oscillatory behavior. This is really costly especially for these kinds of systems, since the number of agents is usually huge. Thus the problem of designing protocols that solve the SS-OS problem under the random scheduler in a decentralized way remains open. The main challenge is to achieve the SS-oscillatory behavior using a number of states that is independent from any global parameter of the system.
2. We have recently addressed the SS-OS problem under a uniform random scheduler in a slightly different setting, in which we assume that the population is synchronous i.e., each agent is part of an interaction at each instant t. We were able to implement a self-synchronized clock and use it to design primitive oscillators. The number of states used to solve the problem does not depend on the size of the population. However, it does depend on the defined period of the oscillator. Hence, it would be also interesting to investigate the impact of the degree of synchrony on the SS-OS problem.
3. It would be challenging to simulate, as for the *Fourier Transform*, in a self-stabilizing way, any periodic behavior of a given population using a finite number of deterministic oscillators. We were able to do so in a recent investigation assuming synchronous populations. Extending the investigation taking into account different levels of synchrony seems to be an interesting direction to investigate.

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to correct and improve the quality of the paper.

This work has been supported, in part, by a Grant-in-Aid for Scientific Research on Innovative Areas "Molecular Robotics" (No. 24104003) of the MEXT, Japan.

References

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

[2] D. Angluin, J. Aspnes, M. Chan, M. J. Fischer, H. Jiang, R. Peralta, Stably computable properties of network graphs, in: International Conference on Distributed Computing in Sensor Systems, Vol. 3560, 2005, pp. 63–74.

[3] D. Angluin, J. Aspnes, D. Eisenstat, Fast computation by population protocols with a leader, Distributed Computing 21 (3) (2008) 183–199.

[4] D. Angluin, J. Aspnes, M. J. Fischer, H. Jiang, Self-stabilizing population protocols, TAAS 3 (4).

[5] D. Angluin, J. Aspnes, D. Eisenstat, E. Ruppert, The computational power of population protocols, Distributed Computing 20 (4) (2007) 279–304.

[6] S. Cai, T. Izumi, K. Wada, How to prove impossibility under global fairness: On space complexity of self-stabilizing leader election on a population protocol model, Theory of Computing Systems 50 (3) (2012) 433–445.

[7] T. Izumi, On space and time complexity of loosely-stabilizing leader election, in: International Colloquium on Structural Information and Communication Complexity, Vol. 9439, 2015, pp. 299–312.

[8] J. Beauquier, P. Blanchard, J. Burman, Self-stabilizing leader election in population protocols over arbitrary communication graphs, in: International Conference on Principles of Distributed Systems, Vol. 8304, 2013, pp. 38–52.

[9] Y. Sudo, F. Ooshita, H. Kakugawa, T. Masuzawa, Loosely-stabilizing leader election on arbitrary graphs in population protocols without identifiers nor random numbers, in: International Conference on Principles of Distributed Systems, 2015.

[10] T. Izumi, K. Kinpara, T. Izumi, K. Wada, Space-efficient self-stabilizing counting population protocols on mobile sensor networks, Theory of Computing Systems 552 (2014) 99–108.

[11] Y. Mocquard, E. Anceaume, J. Aspnes, Y. Busnel, B. Sericola, Counting with population protocols, in: International Symposium on Network Computing and Applications, 2015, pp. 35–42.

[12] J. Beauquier, J. Burman, S. Clavière, D. Sohier, Space-optimal counting in population protocols, in: International Symposium on Distributed Computing, Vol. 9363, 2015, pp. 631–646.

[13] J. Beauquier, J. Burman, Self-stabilizing synchronization in mobile sensor networks with covering, in: International Conference on Distributed Computing in Sensor Systems, Vol. 6131, 2010, pp. 362–378.

[14] S. Murata, A. Konagaya, S. Kobayashi, H. Saito, M. Hagiya, Molecular robotics: A new paradigm for artifacts, New Generation Computing 31 (1) (2013) 27–45.

[15] J. Beauquier, J. Burman, Self-stabilizing mutual exclusion and group mutual exclusion for population protocols with covering, in: International Conference on Principles of Distributed Systems, Vol. 7109, 2011, pp. 235–250.

[16] S. Dolev, Self-Stabilization, MIT Press, 2000.