Self-Stabilizing Oscillatory behavior in Population Protocols*

Colin Cooper[1] Anissa Lamani[2] Giovanni Viglietta[3]
Masafumi Yamashita[2] Yukiko Yamauchi[2]

[1] Department of Informatics, Kings College, United Kingdom
[2] Department of Informatics, Kyushu University, Japan
[3] University of Ottawa, Canada

Abstract

Motivated by the well-known *BZ reaction* that provides an autonomous chemical oscillator, we address in this paper, the problem of autonomously generating an oscillatory execution, assuming any initial configuration (i.e., in a self-stabilizing manner) and, considering the population protocol (PP) model designed to model a collection of finite-state mobile agents that interact with each other in order to accomplish a common task. While most of the works in the literature investigate the computational power of PPs, we throws light, in this paper, on an aspect of PPs as a model of chemical reactions and investigate the self-oscillatory behavior. For deterministic PPs, we show that the self-stabilizing leader election (SS-LE) and the self-stabilizing oscillation problem (SS-OSC) are equivalent, that is, an SS-OSC 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 oscillatory behavior of amplitude n, where n is the number of agents (although we seek a memory-efficient solution).

Keywords: Population protocol, Self-stabilization, Oscillatory behavior, Leader election, Distributed algorithm

1 Introduction

The *population protocol* (PP) model introduced by Angluin et al. [2] is a model of passive distributed systems. It is used as a theoretical model of a

*A detailed version of this work is published in [7]

collection of finite-state mobile agents that interact with each other in order to solve a given problem in a cooperative fashion. Computations are done through pairwise interactions, i.e., when two agents interact, they exchange their information and update their states accordingly. The interaction pattern, however, is unpredictable, that is, the agents have no control over which agent they interact with. We thus assume the presence of an abstract mechanism called *scheduler* that chooses at any time instant, the pair of agents that interact with each other. Observe that PPs can represent not only artificial distributed systems such as sensor networks and mobile agent systems, but also natural distributed systems such as chemical reactions and biological systems.

In the past few years, many problems have been investigated assuming PPs: computing a function, electing a leader, counting, coloring, synchronizing and naming [1, 2, 3, 6, 8, 4]. Most of the 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 given computation problem. The agents are not requested to eventually terminate, but the execution is requested to repeat the 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 the problem 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 [9], 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. We show that under a deterministic scheduler governed by an adversary, the self-stabilizing leader election problem and the self-stabilizing oscillation problem are equivalent, and hence costly in term of space complexity.

Apart from the difference of motivation, a few works on *dynamic* problems are related to our work. Angluin et al. [3] provided a self-stabilizing token circulation protocol in a ring with a pre-selected leader. Beauquier and Burman investigated the self-stabilizing mutual exclusion, group mutual exclusion problems [5]. Our problem also belongs to the class of dynamic problems.

In this paper, we use results from [6] that concern the SS-LE problem. In [6], it has been shown that the SS-LE is impossible to solve with less than n states where n is the size of the population. The paper also presents a PP that solves the SS-LE. The protocol ensures that eventually each agent has unique state.

2 Preliminaries

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

A *protocol* $\mathcal{P} = (\mathcal{Q}, \delta)$ 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 state 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 assume that if $\delta(p, q) = (p', q')$ then $\delta(q, p) = (q', p')$.

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

Let C_t be the configuration at time t and let r_t be the interaction on C_t at time t. An *execution* $\mathcal{E}$ of a protocol $\mathcal{P}$ is a sequence of configurations and transitions $(C_0, r_0, C_1, r_1, \ldots)$ such that $\forall\ i \geq 0$, r_i is a transition of δ and $C_i \xrightarrow{r_i} C_{i+1}$. When a configuration C' is reachable from C after a finite number of transitions we note $C \xrightarrow{*} C'$.

A *scheduler* chooses a pair of agents to interact at each time $t \geq 0$. In this paper, we consider a deterministic but globally fair scheduler that guarantees that if there is a configuration that is reachable infinitely often, then the configuration is eventually reached.

3 Self-stabilizing oscillators

We investigate in this section the problem of generating oscillatory executions under a global fair deterministic scheduler and starting from an arbitrary configuration. Let us first define some important notions that will be used in the sequel.

Definition 1 ***(Oscillation)*** *Let f: $[a, b] \subset \mathbb{N} \to \mathbb{R}$ be a function. We say that f is an oscillation if there exists $c \in \mathbb{N}$ such that:*
1) $a < c < b$ *2) $f(a) < f(c) > f(b)$,*

3) f is weakly increasing in $[a, c]$ and weakly decreasing in $[c, b]$.

The value $f(c) - (f(a) + f(b))/2$ 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 *(respectively,* decreasing phase*) of the oscillation is the interval in which f is weakly increasing (respectively, weakly decreasing).*

Definition 2 ***(Oscillatory behavior)*** *Given an execution $\mathcal{E}$ of a population protocol $\mathcal{P}$ and a set of states S, let $f_S : \mathbb{N} \to [0, n] \subset \mathbb{N}$ be the function mapping a time instant t into the number of agents whose state is in S at time t. Let $\{t_0, t_1, \dots\}$ be a strictly increasing sequence of time instants. We say that $\mathcal{E}$ exhibits an oscillatory behavior for the set of states S, if for every $i \geq 0$, the restriction of f_S to $[t_i, t_{i+1}]$ is an oscillation.*

Note that, according to the previous definitions, any execution exhibits an oscillatory behavior, unless the number of agents whose state is in S eventually stabilizes. However, we are also interested in evaluating the "quality" of the oscillations, in terms of their amplitude and period.

Definition 3 ***(Deterministic oscillator)*** *A population of agents executing a deterministic protocol $\mathcal{P}$, under a global fair scheduler, is a (C, S, ι_a, ι_p)-oscillator if starting from Configuration C, any execution $\mathcal{E}$ of $\mathcal{P}$ exhibits an oscillatory behavior for the set of states S, with amplitude ι_a and period ι_p.*

Definition 4 ***(Deterministic self-stabilizing oscillator)*** *A population of agents executing a deterministic protocol $\mathcal{P}$, under a global fair scheduler, is a self-stabilizing oscillator for the set of states S if, starting from an arbitrary configuration $C_0 \in \mathcal{C}$, every execution $\mathcal{E}$ of Protocol $\mathcal{P}$, reaches a configuration $C \in \mathcal{C}$ such that (C, S, ι_a, ι_p) is a deterministic oscillator.*

We consider in the following (C, S, n, ι_p)-oscillators and show that starting from an arbitrary initial configuration $C_0 \in \mathcal{C}$, the following two results hold:

1. If the SS-LE problem is solvable using M_{LE} states, then it is possible to solve the SS-OCS problem using $M_{LE} + O(n)$ states.

2. If the SS-OSC problem is solvable using M_{OSC} states, then it is possible to solve the SS-LE problem using $M_{OSC} + O(1)$ states.

3.1 SS-LE $\Rightarrow$ SS-OSC

We show that a deterministic population protocol $\mathcal{P}_{OSC}$ exists using $M_{LE} + O(n)$ states per agent (M_{LE} being the number of states necessary to solve the self-stabilizing leader election problem).

The idea of the solution is as follows: we build our SS-OSC protocol $\mathcal{P}_{OSC}$ on the SS-LE protocol $\mathcal{P}_{LE}$ proposed in [6] and that uses n distinct states per agent. When an interaction occurs between two agents, the two agents execute the enabled actions of both $\mathcal{P}_{OSC}$ and $\mathcal{P}_{LE}$. Protocol $\mathcal{P}_{LE}$ ensures that eventually one leader is elected and all the agents have a unique state [6]. Our solution takes advantage of this "identification" to create an oscillatory behavior. Indeed, using the identification created by Protocol $\mathcal{P}_{LE}$, the leader can somehow recognize the agents it has already interacted with.

The state of each agent i consists of a triplet of variables (id_i, p_i, Cnt_i). Variable id_i is used by Protocol $\mathcal{P}_{LE}$ ($id_i \in \{0, 1, 2, \ldots, n-1\}$), where 0 is the leader's state. According to [6], eventually each agent has a unique value of id_i. Variable $p_i \in \{0, 1\}$, indicates the phase of the oscillation Agent i is part of (increasing or decreasing phase). Variable Cnt_i is a counter variable such that $Cnt_i \in \{1, \ldots, M\}$ where $M = n$. The counter is used only by the leader to keep track of the agents it has already interacted with, hence, in the sequel, i is omitted when we refer to the counter. The state of a non-leader agent j is only represented by the pair (id_j, p_j).

The leader uses its counter value to recognize the next agent it has to interact with in order to increment its counter. More precisely, assume that $Cnt = i$ then, if a leader interacts with an agent i such that $id_i = Cnt$ then the leader increments its counter value and Agent i updates its phase to become in the same phase as the leader. When the leader's counter value reaches its maximum value, the leader toggles its phase and re-initializes its counter to 1. The formal description of the solution is given in Protocol 1. Character '?' indicates any state of a non leader agent. If '?' is used then, the corresponding non-leader agent does not update its state in the interaction.

Protocol 1 Self-stabilizing deterministic oscillator with central control

(C(Leader), C(¬ Leader)) → δ(C(Leader), C(¬ Leader))

1. $(0,0,i),(i,0) \rightarrow (0,0,i+1),(i,0)$ if $i \leq n-1$
2. $(0,0,i),(i,1) \rightarrow (0,0,i+1),(i,0)$ if $i \leq n-1$
3. $(0,0,n),? \rightarrow (0,1,1),$?
4. $(0,1,i),(i,0) \rightarrow (0,1,i+1),\ (i,1)$ if $i \leq n-1$
5. $(0,1,i),(i,1) \rightarrow (0,1,i+1),(i,1)$ if $i \leq n-1$
6. $(0,1,n),? \rightarrow (0,0,1),?$

We state the following result:

Theorem 1 *Under the global fair scheduler, if there exists a population protocol that solves the SS-LE problem using M_{LE} states then, there exists a population protocol that solves the SS-OSC problem using $M_{LE} + O(n)$ states.*

Remark 1. The number of states can be reduced to $M_{LE} + O(\iota_a)$ states where ι_a is the desired amplitude of the oscillator. This can be done by using the same strategy as in Protocol 1 and by setting the maximum value of the leader's counter $M = \iota_a$. In addition, when the leader interacts with a non leader agent j such that $id_j < M$, Agent j updates its phase p_j to the default value 0. Since we assume a global fair scheduler, $\forall\ j \in A$ such that $id_j \geq M$, j eventually interacts with the leader and hence $p_j = 0$. If we define the set S as the set of states such that $p = 1$ then only $(\iota_a - 1)$ agents toggle their phase with the leader and hence, we obtain an self-stabilizing oscillator of amplitude ι_a.

3.2 SS-OSC ⇒ SS-LE

We show that if the deterministic SS-OSC problem is solvable using M_{OSC} states, then it is also possible to solve the deterministic SS-LE problem using $M_{OSC} + O(1)$ states. To show this result, we build our self-stabilizing SS-LE protocol $\mathcal{P}'_{LE}$ on the top of the SS-OSC protocol $\mathcal{P}'_{OSC}$. By executing Protocol $\mathcal{P}'_{OSC}$, the system eventually exhibits an oscillatory behavior with respect to a given set of state S. Let us consider the population after the stabilization of $\mathcal{P}'_{OSC}$. We first show some important properties of a population that exhibits an oscillatory behavior. We assume that $\iota_a = n$. Given a configuration $C \in \mathcal{C}$, let $S(C)$ be the set of agents such that $\forall\ i \in A$, $i \in S(C)$ if $C(i) \in S$. The number of agents part of $S(C)$ is denoted by $\#_{S(C)}$. By C^+, we denote the set of configurations that can appear during the increasing phase of any oscillation before reaching the amplitude, that is, $\forall\ C \in C^+$, $\#_{S(C)} < n$. By C^*, we refer to the set of configurations such that $\forall\ C \in C^*$, $\#_{S(C)} = n$ (configurations in which all the agents have their states part of S, i.e., the amplitude is reached). The first step is to show that there is a non-empty subset of states that can only appear when the amplitude of the oscillation is reached. More precisely, in any configuration $C \in C^+$, the transition $\delta(C(i), C(j)) = (C'(i), C'(j))$ such that $\#_{S(C)} > \#_{S(C')}$ is never enabled when the system is stabilized. Let Q' be the set of states that enable such a transition then, $\forall\ C \in C^+$, $\forall\ i \in A$, $C'(i) \notin Q'$ and $\forall\ C' \in C^*$, $\exists\ i \in A$, $C'(i) \in Q'$ (States in Q' indicates that the next phase of the oscillation can be initiated). Next, we define a subset of special configurations that we denote by $C_{sp} \subset C^+$. A configuration $C \in C_{sp}$ satisfies the two following conditions: (1) $\exists!\ j \in A$ such that $C(j) \notin S$ and (2) $\forall\ i \in A$, $C(i) \notin Q'$. Observe that Condition (1) implies that $\forall\ i \in A \backslash \{j\}$, $C(i) \in S$. We show that a configuration $C \in C_{sp}$ is eventually reached and $\exists\ i, j \in A$ such that $\delta(C_{sp}(i), C_{sp}(a_j)) = (C'(i), C'(j))$ with $C(j) \notin S$ and $C'(j) \in S$ and either $(C'(i) \in Q')$ or $(C'(j) \in Q')$. That is, the amplitude is reached and at least one of the two interacting agents has a state part of Q'. We refer to such an interaction by r_{sp}. Finally, we prove that from a

configuration $C \in \mathcal{C}^*$, if $\exists\ i \in A$ such that $C(i) \notin Q'$ and $\exists j \in A$ such that $\delta(C(i), C(j)) = (C'(i), C'(j))$ with $C'(i) \in Q'$ then $C(j) \in Q'$, that is, when the amplitude is reached, a given agent can change its state to a state in Q' only if it interacts with an agent already in a state part of Q'. We take advantage of these properties and define now our protocol.

Protocol. In order to elect a leader starting from an arbitrary configuration $C_0 \in C$ using the SS-OSC population protocol $\mathcal{P}'_{OSC}$, we add to the state of each agent one bit of memory to indicate whether the agent is a leader or not ($l \in \{0, 1\}$). When r_{sp} is executed, if C' is the resulting configuration, then $\exists\ i \in A$ such that $C'(i) \in Q'$ (recall that r_{sp}:$\delta(C_{sp}(i), C_{sp}(j)) = (C'(i), C'(j))$ such that (i) $C(j) \notin S$. (ii) $C'(j) \in S$. (iii) $((C'(i) \in Q') \vee (C'(j)) \in Q'))$. Assume that after the execution of r_{sp}, $\exists!\ i \in A$ such that $C'(i) \in Q'$ (let us refer to this agent by a_{sp}). The idea of the protocol is as follows: when r_{sp} is executed, Agent a_{sp} becomes a leader. In addition, when a given agent i interacts with a leader then, neither Agent i nor the leader update their state (they both keep the same state). Observe that since we assume an arbitrary initial configuration, such a transition can be executed even if the population is not yet stabilized with respect to $\mathcal{P}'_{OSC}$. To be sure to create only one leader, if in a given configuration $C \in \mathcal{C}$, Agent i is a leader then Agent i becomes a non-leader in the next interaction if $C(i) \notin Q'$ or $C(i) \notin S$. In the same manner, Agent i becomes a non leader if it interacts either with another leader or with an agent j such that $C(j) \notin S$. Observe that if $\exists\ i \in A$ such that i is a leader, then i can only be enabled to become a non-leader, We show that:

Theorem 2 *Under the global fair scheduler, if there exists a population protocol $\mathcal{P}'_{OSC}$ that solves the SS-OSC problem with amplitude n using M_{OSC} states, then the SS-LE problem is also possible to solve using $M_{OSC} + O(1)$ states.*

Recall that it has been proved in [6] that the SS-LE problem is not solvable when $|Q| < n$ and hence impossible to solve in the case where n is arbitrary. Using Theorems 1 and 2 we deduce:

Corollary 1 *There exists no deterministic self-stabilizing oscillator if the number of states by agent is less than n, or if the size of the population is arbitrary.*

4 Conclusion

In this paper, we have considered the PPs model and have addressed the problem of autonomously generating oscillatory executions. We have considered the problem using deterministic protocols and have shown that, under the deterministic global fair scheduler, $\Omega(n)$ states are necessary to solve

the SS-OSC problem. This result emphasizes somehow the impact and the importance of randomization in biological systems and chemical reactions in creating self-oscillations that is, it would be interesting to consider the problem assuming a probabilistic scheduler i.e., the pair of agents chosen for the interactions are selected randomly.

References

[1] Dana Angluin, James Aspnes, Melody Chan, Michael J. Fischer, Hong Jiang, and René Peralta. Stably computable properties of network graphs. In *DCOSS*, volume 3560, pages 63–74, 2005.

[2] Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. In *PODC*, pages 290–299, 2004.

[3] Dana Angluin, James Aspnes, Michael J. Fischer, and Hong Jiang. Self-stabilizing population protocols. *TAAS*, 3(4), 2008.

[4] Joffroy Beauquier and Janna Burman. Self-stabilizing synchronization in mobile sensor networks with covering. In *Distributed Computing in Sensor Systems (DCOSS)*, volume 6131, pages 362–378, 2010.

[5] Joffroy Beauquier and Janna Burman. Self-stabilizing mutual exclusion and group mutual exclusion for population protocols with covering. In *OPODIS*, volume 7109, pages 235–250, 2011.

[6] Shukai Cai, Taisuke Izumi, and Koichi Wada. How to prove impossibility under global fairness: On space complexity of self-stabilizing leader election on a population protocol model. *Theory Comput. Syst.*, 50(3):433–445, 2012.

[7] Colin Cooper, Anissa Lamani, Giovanni Viglietta, Masafumi Yamashita, and Yukiko Yamauchi. Self-stabilizing synchronization in mobile sensor networks with covering. In *(SSS)*, 2015.

[8] Keigo Kinpara, Tomoko Izumi, Taisuke Izumi, and Koichi Wada. Improving space complexity of self-stabilizing counting on mobile sensor networks. In *OPODIS*, volume 6490, pages 504–515, 2010.

[9] Satoshi Murata, Akihiko Konagaya, Satoshi Kobayashi, Hirohide Saito, and Masami Hagiya. Molecular robotics: A new paradigm for artifacts. *New Generation Computing*, 31(1):27–45, 2013.