Information and Computation ••• (••••) •••–•••

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

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

ARTICLE INFO

Article history:
Received 31 October 2015
Available online xxxx

Keywords:
Autonomous systems
Leader election
Population protocols
Self-organization
Self-oscillation
Self-stabilization

ABSTRACT

We consider the population protocol (PP) model used to represent a collection of finite-state mobile agents that interact with each other to accomplish a common task. Motivated by the well-known *BZ reaction*, 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), (2) n states are necessary to create oscillations 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 some deterministic oscillatory PPs under a uniform random scheduler.

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.

* Corresponding author.
E-mail address: anissa.lamani@gmail.com (A. Lamani).

http://dx.doi.org/10.1016/j.ic.2016.12.002
0890-5401/© 2016 Published by Elsevier Inc.

In the past decade, many problems have been investigated on PPs, including the problems of computing a function [2,1,3–5], electing a leader [6–9], counting [10–12], 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 following: 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 $A = \{0, 1, \dots, n-1\}$. 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