Population Protocols for Graph Class Identification Problems

Hiroto Yasumi
Nara Institute of Science and Technology, Japan

Fukuhito Ooshita
Nara Institute of Science and Technology, Japan

Michiko Inoue
Nara Institute of Science and Technology, Japan

Abstract

In this paper, we focus on graph class identification problems in the population protocol model. A graph class identification problem aims to decide whether a given communication graph is in the desired class (*e.g.* whether the given communication graph is a ring graph). Angluin et al. proposed graph class identification protocols with directed graphs and designated initial states under global fairness [Angluin et al., DCOSS2005]. We consider graph class identification problems for undirected graphs on various assumptions such as initial states of agents, fairness of the execution, and initial knowledge of agents. In particular, we focus on lines, rings, k-regular graphs, stars, trees, and bipartite graphs. With designated initial states, we propose graph class identification protocols for k-regular graphs and trees under global fairness, and propose a graph class identification protocol for stars under weak fairness. Moreover, we show that, even if agents know the number of agents n, there is no graph class identification protocol for lines, rings, k-regular graphs, trees, or bipartite graphs under weak fairness, and no graph class identification for lines, rings, k-regular graphs, stars, trees, or bipartite graphs with arbitrary initial states.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Theory of computation → Concurrent algorithms

Keywords and phrases population protocol, graph class identification, distributed protocol

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2021.13

Related Version *Full Version*: `https://arxiv.org/abs/2111.05111` [26]

Funding This work was supported by JSPS KAKENHI Grant Numbers 18K11167, 20H04140, 20J21849, and JST SICORP Grant Number JPMJSC1806.

1 Introduction

The population protocol model is an abstract model for low-performance devices, introduced by Angluin et al. [4]. In this model, a network, called population, consists of multiple devices called agents. Those agents are anonymous (i.e., they do not have identifiers), and move unpredictably (i.e., they cannot control their movements). When two agents approach, they are able to communicate and update their states (this communication is called an interaction). By a sequence of interactions, the system proceeds a computation. In this model, there are various applications such as sensor networks used to monitor wild birds and molecular robot networks [24].

In this paper, we study the computability of graph properties of communication graphs in the population protocol model. Concretely, we focus on graph class identification problems that aim to decide whether the communication graph is in the desired graph class. In most distributed systems, it is essential to understand properties of the communication graph in order to design efficient algorithms. Actually, in the population protocol model,

efficient protocols are proposed with limited communication graphs (*e.g.*, ring graphs and regular graphs) [2, 6, 16, 17]. In the population protocol model, the computability of the graph property was first considered in [3]. In [3], Angluin et al. proposed various graph class identification protocols with directed graphs and designated initial states under global fairness. Concretely, Angluin et al. proposed graph class identification protocols for directed lines, directed rings, directed stars, and directed trees. Moreover, they proposed graph class identification protocols for other graphs such as 1) graphs having degree bounded by a constant k, 2) graphs containing a fixed subgraph, 3) graphs containing a directed cycle, and 4) graphs containing a directed cycle of odd length. However, there are still some open questions such as "What is the computability for undirected graphs?" and "How do other assumptions (*e.g.*, initial states, fairness, etc.) affect the computability?" In this paper, we answer those questions. That is, we clarify the computability of graph class identification problems for undirected graphs under various assumptions such as initial states of agents, fairness of the execution, and an initial knowledge of agents. More concretely, in this paper, we consider the problems with designated or arbitrary initial states, under global or weak fairness, and with the number of agents n, the upper bound P of the number of agents, or no knowledge. The assumption of initial states bears on the requirement of initialization and the fault-tolerant property. To execute a protocol with designated initial states, it is necessary to initialize all agents. Alternatively, a protocol with arbitrary initial states does not need to initialize agents. This implies that, even if agents transition to incorrect states by transient faults, the protocol can recover to desired configurations. Fairness is an assumption of interaction patterns. Intuitively, global fairness guarantees that, if a reachable configuration can occur infinitely often, the reachable configuration actually occurs infinitely often. On the other hand, weak fairness only guarantees that interactions occur infinitely often between each pair of adjacent agents. The initial knowledge is given to agents for helping the agents solve the problem. The initial knowledge enables us to construct efficient protocols although it may be difficult to know the knowledge in some situations.

In the population protocol, researchers also considered other assumptions such as symmetry and randomness (deterministic or non-deterministic). In this paper, we consider only deterministic asymmetric protocols. Note that, with designated initial states under global fairness, there is a transformer that transforms an asymmetric protocol into a symmetric protocol by assuming additional states [13]. Although we deal only with asymmetric protocols, we can transform most of our asymmetric protocols to symmetric protocols by applying this transformer.

We remark that some protocols in [3] for directed graphs can be easily extended to undirected graphs with designated initial states under global fairness (see Table 1). Concretely, graph class identification protocols for directed lines, directed rings, and directed stars can be easily extended to protocols for undirected lines, undirected rings, and undirected stars, respectively. In addition, the graph class identification protocol for bipartite graphs can be deduced from the protocol that decides whether a given graph contains a directed cycle of odd length. This is because, if we replace each edge of an undirected non-bipartite graph with two opposite directed edges, the directed non-bipartite graph always contains a directed cycle of odd length. On the other hand, the graph class identification protocol for directed trees cannot work for undirected trees because the protocol uses a property of directed trees such that in-degree (resp., out-degree) of each agent is at most one on an out-directed tree (resp., an in-directed tree). Note that agents can identify trees if they understand the graph contains no cycle. However, the graph class identification protocol for graphs containing a directed cycle in directed graphs cannot be used to identify a (simple) cycle in undirected graphs. This is because, if we replace an undirected edge with two opposite directed edges, the two directed edges compose a directed cycle.

Table 1 The number of states to solve the graph class identification problems. n is the number of agents and P is an upper bound of the number of agents.

Model			Graph Properties					
Initial states	Fairness	Initial knowledge	*Line*	*Ring*	*Bipartite*	*Tree*	*k-regular*	*Star*
Designated	Global	n	$O(1)$†	$O(1)$†	$O(1)$†	$O(1)$*	$O(k \log n)$*	$O(1)$†
		P	$O(1)$†	$O(1)$†	$O(1)$†	$O(1)$*	$O(k \log P)$*	$O(1)$†
		None	$O(1)$†	$O(1)$†	$O(1)$†	$O(1)$*	–	$O(1)$†
	Weak	n	Unsolvable*					$O(n)$*
		P/None	Unsolvable*					
Arbitrary	Global/Weak	n/P/None	Unsolvable*					

* Contributions of this paper †Deduced from Angluin et al. [3]

Our Contributions

In this paper, we clarify the computability of graph class identification problems for undirected graphs under various assumptions. Recall that we consider only deterministic asymmetric protocols. A summary of our results is given in Table 1. Under global fairness, we propose two graph class identification protocols. One is a graph class identification protocol for trees with designated initial states. This protocol works with constant number of states even if no initial knowledge is given. The other is a graph class identification protocol for k-regular graphs with designated initial states and the initial knowledge of the upper bound P of the number of agents. On the other hand, under weak fairness, we show that there exists no graph class identification protocol for lines, rings, k-regular graphs, stars, trees, or bipartite graphs even if the upper bound P of the number of agents is given. In the case where the number of agents n is given, we propose a graph class identification protocol for stars and prove that there exists no graph class identification protocol for lines, rings, k-regular graphs, trees, or bipartite graphs. With arbitrary initial states, we prove that there is no protocol for lines, rings, k-regular graphs, stars, trees, or bipartite graphs. In this paper, because of space constraints, we omit the details of protocols and some proofs (see the full version in [26]).

Related Works

The population protocol model was proposed by Angluin et al. [4]. While they mainly studied the computability of the model in the paper, subsequent works studied various problems (*e.g.*, leader election [1, 11, 18, 21], counting [7, 8, 12, 22], majority [5, 10, 20], etc) under different assumptions (*e.g.*, fairness assumption [7, 8], initial states of agents [6, 9], and initial knowledge of agents [14, 25]).

Although those problems are usually considered with complete communication graphs (i.e., every pairwise interaction can occur), some researchers proposed efficient protocols with limited communication graphs (*e.g.*, ring graph, regular graph, etc.) [2, 6, 16, 17]. More concretely, Angluin et al. proposed a protocol that constructs a spanning tree with regular graphs [6]. Chen et al. proposed self-stabilizing leader election protocols with ring graphs [16] and regular graphs [17]. Alistarh et al. showed that protocols for complete graphs (including the leader election protocol, the majority protocol, etc.) can be simulated efficiently in regular graphs [2].

For graph class identification problems, after Angluin et al. studied some solvabilities [3], Chatzigiannakis et al. studied solvabilities for directed graphs with some properties on the mediated population protocol model [15], where the mediated population protocol model is an extension of the population protocol model. In this model, a communication link (on which agents interact) has a state. Agents can read and update the state of the communication

link when agents interact on the communication link. In [15], they proposed graph class identification protocol for some graphs such as 1) graphs having degree bounded by a constant k, 2) graphs in which the degree of each agent is at least k, 3) graphs containing an agent such that in-degree of the agent is greater than out-degree of the agent, 4) graphs containing a directed path of at least k edges, etc. Since Chatzigiannakis et al. proposed protocols for the mediated population protocol model, the protocols cannot work in the population protocol model. As impossibility results, they showed that there is no graph class identification protocol that decides whether the given directed graph has two edges (u, v) and (v, u) for two agents u and v, or whether the given directed graph is weakly connected.

As another perspective of communication graphs, Michail and Spirakis proposed a network constructors model that is an extension of the mediated population protocol [23]. The network constructors model aims to construct a desired graph on the complete communication graph by using communication links with two states. Each communication link only has *active* or *inactive* state. Initially, all communication links have inactive state. By activating/deactivating communication links, the protocol of this model constructs a desired communication graph that consists of agents and activated communication links. In [23], they proposed protocols that construct spanning lines, spanning rings, spanning stars, and regular graphs. Moreover, by relaxing the number of states, they proposed a protocol that constructs a large class of graphs.

2 Definitions

2.1 Population Protocol Model

A communication graph of a population is represented by a simple undirected connected graph $G = (V, E)$, where V represents a set of agents, and $E \subseteq V \times V$ is a set of edges (containing neither multi-edges nor self-loops) that represent the possibility of an interaction between two agents (i.e., only if $(a, b) \in E$ holds, two agents $a \in V$ and $b \in V$ can interact).

A protocol $\mathcal{P} = (Q, Y, \gamma, \delta)$ consists of a finite set Q of possible states of agents, a finite set of output symbols Y, an output function $\gamma : Q \to Y$, and a set of transitions δ from $Q \times Q$ to $Q \times Q$. Output symbols in Y represent outputs as the results according to the purpose of the protocol. Output function γ maps a state of an agent to an output symbol in Y. Each transition in δ is denoted by $(p, q) \to (p', q')$. This means that, when an agent a in state p interacts with an agent b in state q, their states become p' and q', respectively. We say that such a is an initiator and such b is a responder. When a and b interact as an initiator and a responder, respectively, we simply say that a interacts with b. Transition $(p, q) \to (p', q')$ is null if both $p = p'$ and $q = q'$ hold. We omit null transitions in the descriptions of protocols. Protocol $\mathcal{P} = (Q, Y, \gamma, \delta)$ is deterministic if, for any pair of states $(p, q) \in Q \times Q$, exactly one transition $(p, q) \to (p', q')$ exists in δ. Recall that we consider only deterministic protocols in this paper.

A configuration represents a global state of a population, defined as a vector of states of all agents. A state of agent a in configuration C is denoted by $s(a, C)$. Moreover, when C is clear from the context, we simply use $s(a)$ to denote the state of agent a. A transition from configuration C to configuration C' is denoted by $C \to C'$, and means that, by a single interaction between two agents, configuration C' is obtained from configuration C. For two configurations C and C', if there exists a sequence of configurations $C = C_0$, C_1, …, $C_m = C'$ such that $C_i \to C_{i+1}$ holds for every i $(0 \leq i < m)$, we say C' is reachable from C, denoted by $C \xrightarrow{*} C'$.

An execution of a protocol is an infinite sequence of configurations $\Xi = C_0, C_1, C_2, \ldots$ where $C_i \rightarrow C_{i+1}$ holds for every i ($i \geq 0$). An execution Ξ is weakly-fair if, for any adjacent agents a and a', a interacts with a' and a' interacts with a infinitely often[1]. An execution Ξ is globally-fair if, for each pair of configurations C and C' such that $C \rightarrow C'$, C' occurs infinitely often when C occurs infinitely often. Intuitively, global fairness guarantees that, if configuration C occurs infinitely often, then any possible interaction in C also occurs infinitely often. Then, if C occurs infinitely often, C' satisfying $C \rightarrow C'$ occurs infinitely often, and we can deduce that C'' satisfying $C' \rightarrow C''$ also occurs infinitely often. Overall, with global fairness, if a configuration C occurs infinitely often, then every configuration C^* reachable from C also occurs infinitely often.

In this paper, we consider three possibilities for an initial knowledge of agents: the number of agents n, the upper bound P of the number of agents, and no knowledge. Note that the protocol depends on this initial knowledge. When we explicitly state that an integer x is given as the number of agents, we write the protocol as $\mathcal{P}_{n=x}$. Similarly, when we explicitly state that an integer x is given as the upper bound of the number of agents, the protocol is denoted by $\mathcal{P}_{P=x}$.

2.2 Graph Properties and Graph Class Identification Problems

We define graph properties treated in this paper as follows:

- A graph G satisfies property *tree* if there is no cycle on graph G.
- A graph $G = (V, E)$ satisfies property *k-regular* if the degree of every agent in V is k.
- A graph G satisfies property *star* if G is a tree with one internal agent and $n - 1$ leaves.
- A graph $G = (V, E)$ satisfies property *bipartite* if V can be divided into two disjoint and independent sets U and W (i.e., $U \bigcap W = \emptyset$ holds and there is no edge connecting two agents in U or W).
- A graph $G = (V, E)$ satisfies property *line* if $E = \{(v_0, v_1), (v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n)\}$ for $V = \{v_1, v_2, \ldots v_n\}$.
- A graph $G = (V, E)$ satisfies property *ring* if the degree of every agent in V is 2.

Let *gp* be an arbitrary graph property. The *gp* identification problem aims to decide whether a given communication graph G satisfies property *gp*. In the *gp* identification problem, the output set is $Y = \{yes, no\}$. Recall that the output function γ maps a state of an agent to an output symbol in Y (i.e., *yes* or *no*). A configuration C is stable if C satisfies the following conditions: There exists $yn \in \{yes, no\}$ such that 1) $\forall a \in V : \gamma(s(a, C)) = yn$ holds, and 2) for every configuration C' such that $C \xrightarrow{*} C'$, $\forall a \in V : \gamma(s(a, C')) = yn$ holds.

An execution $\Xi = C_0, C_1, C_2, \ldots$ solves the *gp* identification problem if Ξ includes a stable configuration C_t that satisfies the following conditions.

1. If a given graph $G = (V, E)$ satisfies graph property *gp*, $\forall a \in V : \gamma(s(a, C_t)) = yes$ holds.
2. If a given graph $G = (V, E)$ does not satisfy graph property *gp*, $\forall a \in V : \gamma(s(a, C_t)) = no$ holds.

[1] We use this definition only for the lower bound under weak fairness. For the upper bound, we use a slightly weaker assumption. We show that our proposed protocol for weak fairness works if, for any adjacent agents a and a', a and a' interact infinitely often (i.e., it is possible that, for any interaction between some adjacent agents a and a', a becomes an initiator and a' never becomes an initiator). Note that, in the protocol, if a transition $(p, q) \rightarrow (p', q')$ exists for $p \neq q$, a transition $(q, p) \rightarrow (q', p')$ also exists.

A protocol $\mathcal{P}$ solves the gp identification problem under weak fairness (resp., under global fairness) if every weakly-fair execution (resp., every globally-fair execution) of protocol $\mathcal{P}$ solves the gp identification problem.

3 Graph Class Identification Protocols

3.1 Tree Identification with No Initial Knowledge under Global Fairness

In this section, we give a tree identification protocol with 18 states and designated initial states under global fairness (see the pseudocode in the appendix and the full version in [26]).

The basic strategy of the protocol is as follows. Initially agents think that the graph is a tree, and, if they detect a cycle, they confirm that the graph is not a tree. To detect a cycle, first, agents elect one leader token, one right token, and one left token. Initially, all agents have right tokens. When two agents with right tokens interact, agents make one of the right tokens transition to a left token. Similarly, when two agents with left tokens interact, agents make one of the left tokens transition to a leader token. When two agents with leader tokens interact, agents delete one of the leader tokens. Agents carry these tokens on a graph by interactions as if each token moves freely on the graph. Thus, by the above behaviors, eventually agents elect one leader token, one right token, and one left token.

Agents behave as if the leader token has an opinion (tree/non-tree), and agents follow the opinion (this opinion is hereinafter referred to as "decision"). Initially the leader token has the decision such that the graph is a tree (i.e., there is no cycle). Since the leader token moves freely on the graph and we assume global fairness, the leader token visits all agents infinitely often. Thus, eventually all agents will know the decision of the leader token.

Agents reset the decision of the leader token if agents notice that the token election is not yet over. Concretely, when two agents with leader tokens interact and delete one of the leader token, agents reset the decision of the remaining leader token. By this behavior, all agents virtually reset their decision because each agent follows the decision of the leader token. Hence, when agents complete the token election, all agents (and the leader token) virtually reset their decision. Now, we explain that, after the token election, agents correctly detect a cycle and the leader token make a correct decision.

After the election, agents repeatedly execute a trial to detect a cycle by using the tokens. The trial starts when an agent with the leader token places the right token and the left token to two adjacent agents x and y, respectively. During the trial, x and y hold the right token and the left token, respectively. To detect a cycle, agents use the right token and the left token as a single landmark. The right token and the left token correspond to a right side and a left side of the landmark, respectively. If agents can carry the leader token from the right side of the landmark to the left side of the landmark without passing through the landmark, the trial succeeds.

From now, we explain the behaviors of the trial in more details. An image of the trial is shown in Figure 1, where L_{se}, L_r, L_l, L_r^t, and L_l^t represent tokens (we will show the details later). To begin with, we explain the start of the trial (the first and second steps of Figure 1). To start the trial, agents place the left token and the right token next to each other. To distinguish between a moving token and a placed token, we use a trial mode. Agents regard right and left tokens in a trial mode as placed tokens. An L_r^t token (resp., an L_l^t token) represents the right token (resp., the left token) in the trial mode. An L_r token (resp., an L_l token) represents the right token (resp., the left token) in a non-trial mode. An L_{se} token represents the leader token.

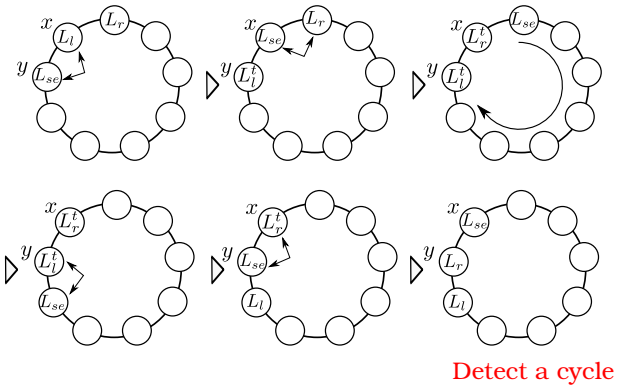

Figure 1 An image of the trial.

More concretely, agents start the trial as follows. When an agent x having the L_l token interacts with an agent y having the leader token, agents make the L_l token transition to an L_l^t token, and they exchange their tokens (the first step of Figure 1). Then, if agent x having the leader token interacts with an agent having the L_r token, agents make the L_r token transition to an L_r^t token, and they exchange their tokens (the second step of Figure 1). By above behaviors, agents place an L_r^t token next to the L_l^t token, and a trial of the cycle detection starts.

After that, agents try to carry the leader token to agent y without passing through agent x (the third step of Figure 1). To do this, agents try to carry the leader token to an agent having the L_l^t token. Note that the left and right tokens can move even while agents carry the leader token. However, if they move, they transition to the non-trial mode. Thus, if an agent having the leader token interacts with an agent having the L_l^t token, agents confirm that the L_l^t token is still placed at y (the fourth step of Figure 1). Then, to confirm that the L_r^t token is also still placed at x, agent y having the leader token tries to interact with an agent having the L_r^t token. Since the right token may move and the leader token may pass through agent x without meeting the right token, this confirmation is necessary. If agents succeed both confirmations, agents succeed the trial and decide that there is a cycle (the fifth step of Figure 1). Hence, in the case, the leader token makes a decision that the given graph is not a tree, and the decision is conveyed to all agents. Since each token moves freely on the graph and we assume global fairness, agents perform the trial on any place (i.e., agents place the left token and the right token on any adjacent agents). Thus, if there is a cycle, eventually agents decide that the given graph is not a tree. Recall that initially all agents think that the given graph is a tree. Hence, unless the trial succeeds, all agents continue to think that the given graph is a tree.

▶ **Theorem 1.** *There exists a protocol with constant states and designated initial states that solves the tree identification problem under global fairness.*

3.2 k-regular Identification with Knowledge of P under Global Fairness

In this subsection, we give a k-regular identification protocol with $O(k \log P)$ states and designated initial states under global fairness (see the pseudocode in the appendix and the full version in [26]).

The basic strategy of the protocol is as follows. First, agents elect a leader token. In this protocol, agents with leader tokens leave some information in agents. To keep only the information that is left after completion of the election, we introduce *level* of an agent. If an

agent at level i has the leader token, we say that the leader token is at level i. Agents with leader tokens leave the information with their levels. Before agents complete the election of leader tokens, agents keep increasing their levels, and agents discard the information with smaller levels when agents increase their levels. When agents complete the election of leader tokens, the agent with the leader token is the only agent that has the largest level. Then, all agents eventually converge to the level. We guarantee that the maximum level of agents is $\lfloor \log P \rfloor$. Since agents discard the information with smaller levels, agents virtually discard any information that was left before agents complete the election. From now on, we consider configurations after agents elect a leader token and discard any outdated information.

Now, we explain how the protocol solves the k-regular identification problem by using the leader token. First, an agent with the leader token examines whether its degree is at least k, and whether its degree is at least $k+1$. If the agent confirms that its degree is at least k but does not confirm that its degree is at least $k+1$, then the agent thinks that its degree is k. Since the leader token moves freely on the graph and we assume global fairness, eventually each agent confirms whether its degree is k. The agent with the leader token examines whether its degree is at least k as follows: An agent a with the leader token checks whether a can interact with k different agents. To check it, agent a with the leader token marks adjacent agents and counts how many times a has marked. Concretely, when agent a having the leader token interacts with an agent b, agent a marks agent b by making b change to a marked state. Agent a counts how many times a interacts with an agent having a non-marked state (hereinafter referred to as "a non-marked agent"). If agent a having the leader token interacts with k non-marked agents successively, a decides that a can interact with k different agents (i.e., its degree is at least k).

If an agent confirms that its degree is at least k, the agent stores this information locally. To do this, we introduce a variable loc_a at agent a: Variable $loc_a \in \{yes, no\}$, initialized to no, represents whether the degree of agent a is at least k. If $loc_a = yes$ holds, agent a thinks that its degree is at least k. If an agent a confirms that its degree is at least k, agent a makes loc_a transition from no to yes.

Next, we show how agents decide whether the graph is k-regular. In this protocol, first an agent with the leader token decides whether the graph is k-regular, and then the decision is conveyed to all agents by the leader token. We use variable reg_a at agent a for the decision: Variable $reg_a \in \{yes, no\}$, initialized to no, represents the decision of the k-regular graph. If $reg_a = yes$ holds for agent a, then $\gamma(s_a) = yes$ holds. If $reg_a = no$ holds, then $\gamma(s_a) = no$ holds. Whenever an agent a with the leader token makes loc_a transition to yes, agent a makes reg_a transition to yes. If an agent a with the leader token finds an agent b such that $loc_b = no$ or its degree is at least $k+1$, agents reset reg_a to no. Note that, since all agents follow the decision of the leader token, this behavior practically resets reg of each agent. If there is such agent b, agent a with the leader token eventually finds agent b since the leader token moves freely on the graph. Hence, if the graph is not k-regular, reg of the leader token (i.e., reg_a such that agent a has the leader token) transitions to no infinitely often. On the other hand, if the graph is k-regular, eventually loc_a of each agent a transitions from no to yes. Let us consider a configuration where loc of each agent other than an agent x is yes and loc_x is no. After the configuration, when agent x makes loc_x and reg_x transition to yes, agent x has the leader token (i.e., reg of the leader token transitions to yes). Hence, since there is no agent such that its loc is no or its degree is at least $k+1$, reg of the leader token never transitions to no afterwards and thus reg of the leader token converges to yes. Thus, since agents convey the decision of the leader token to all agents, eventually all agents make a correct decision.

▶ Remark. We have introduced the level of agents to reset all agents virtually. In the case of tree identification, only the leader token needs to be reset because, after the leader token is reset, the leader token is not affected by incorrect information (the decision of agent in this case). On the other hand, in the case of k-regular identification, all agents need to be reset because, even after the leader token is reset, the leader token is affected by incorrect information (the value of *loc* in this case). More concretely, the leader token has to refer to variable *loc* of agent in order to understand whether all agents have been examined. Hence, since agents may store an incorrect value in *loc*, the leader token may make a wrong decision unless the agents are reset.

To count the degree, agents require $O(k)$ states, and the maximum level of agents is $\lfloor \log P \rfloor$. Hence, the protocol works with $O(k \log P)$ states.

▶ **Theorem 2.** *If the upper bound P of the number of agents is given, there exists a protocol with $O(k \log P)$ states and designated initial states that solves the k-regular identification problem under global fairness.*

When the number of agents n is given, the protocol works even if the maximum level is $\lfloor \log n \rfloor$. Thus, we have the following theorem.

▶ **Theorem 3.** *If the number of agents n is given, there exists a protocol with $O(k \log n)$ states and designated initial states that solves the k-regular identification problem under global fairness.*

3.3 Star Identification with Knowledge of n under Weak Fairness

In this subsection, we give a star identification protocol with $O(n)$ states and designated initial states under weak fairness (see the pseudocode in the appendix and the full version in [26]). In this protocol, the number of agents n is given. Since a given graph is a star if $n \leq 2$ holds, we consider the case where n is at least 3.

The basic strategy of the protocol is as follows. Initially, each agent thinks that the given graph is not a star. First, agents elect an agent with degree two or more as a central agent (i.e., an agent that connects to all other agents in the star graph). Then, by counting the number of agents adjacent to the central agent, agents examine whether there is a star subgraph in the given graph such that the subgraph consists of n agents. Concretely, if the central agent confirms by counting that there are $n-1$ adjacent agents, agents confirm that there is the subgraph. In this case, agents think that the given graph is a star. Then, if two agents other than the central agent interact, agents decide that the graph is not a star. If such an interaction does not occur, agents continue to think that the given graph is a star.

To count $n-1$ agents adjacent to the central agents, agents require $O(n)$ states. Hence, the protocol works with $O(n)$ states.

▶ **Theorem 4.** *There exists a protocol with $O(n)$ states and designated initial states that solves the star identification problem under weak fairness if the number of agents n is given.*

4 Impossibility Results

4.1 A Common Property of Graph Class Identification Protocols for Impossibility Results

In this subsection, we present a common property that holds for protocols with designated initial states under weak fairness.

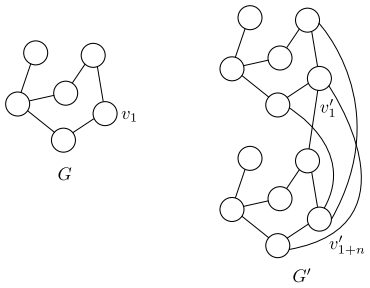

Figure 2 An example of graphs G and G'.

With designated initial states under weak fairness, we assume that a protocol $\mathcal{P}$ solves some of the graph class identification problems. From now, we show that, with $\mathcal{P}$, there exists a case where agents cannot distinguish between some different connected graphs. Note that $\mathcal{P}$ has no constraints for an initial knowledge (i.e., for some integer x, $\mathcal{P}$ is $\mathcal{P}_{n=x}$, $\mathcal{P}_{P=x}$, or a protocol with no initial knowledge). Because of space constraints, we omit the details of the proof (see the full version in the [26]).

▶ **Lemma 5.** *Let us consider a communication graph $G = (V, E)$, where $V = \{v_1, v_2, v_3, \ldots, v_n\}$. Let $G' = (V', E')$ be a communication graph that satisfies the following, where $V' = \{v'_1, v'_2, v'_3, \ldots, v'_{2n}\}$.*

- *$E' = \{(v'_x, v'_y), (v'_{x+n}, v'_{y+n}) \in V' \times V' \mid (v_x, v_y) \in E\} \cup \{(v'_1, v'_{z+n}), (v'_{1+n}, v'_z) \in V' \times V' \mid (v_1, v_z) \in E\}$ (Figure 2 shows an example of the graphs).*

Let Ξ be a weakly-fair execution of $\mathcal{P}$ with G. If there exists a configuration C of Ξ after which $\forall v \in V : \gamma(s(v)) = yn \in \{yes, no\}$ holds, there exists an execution Ξ' of $\mathcal{P}$ with G' such that there exists a configuration C' of Ξ' after which $\forall v' \in V' : \gamma(s(v')) = yn$ holds.

Proof. (Proof sketch) With G', we consider a particular weakly-fair execution Ξ'. In Ξ', agents repeat the following four sub-executions: 1) agents $v'_1, v'_2, \ldots, v'_n$ behave similarly to Ξ, 2) agents $v'_{1+n}, v'_{2+n}, \ldots, v'_{2n}$ behave similarly to Ξ, 3) agent v'_{1+n} and agents $v'_2, v'_3 \ldots, v'_n$ behave similarly to Ξ by joining v'_{1+n} instead of v'_1, and 4) agent v'_1 and agents $v'_{2+n}, v'_{3+n} \ldots, v'_{2n}$ behave similarly to Ξ by joining v'_1 instead of v'_{1+n}. Since v'_1 (resp., v'_{1+n}) can join interactions instead of v'_{1+n} (resp., v'_1) by the definition of G and G', this execution is possible. Clearly, in Ξ', the decision of each agent is the same as the decision of agent in Ξ. Therefore, the lemma holds. ◀

4.2 Impossibility with Knowledge of P under Weak Fairness

For the purpose of the contradiction, we assume that, for an integer x, there exists a protocol $\mathcal{P}_{P=x}$ that solves some of the graph class identification problems with designated initial states under weak fairness. We can apply Lemma 5 to $\mathcal{P}_{P=x}$ because we can apply the same protocol $\mathcal{P}_{P=x}$ to both G and G' in Lemma 5. Clearly, we can construct G and G' in Lemma 5 such that, for any of properties *line*, *ring*, *tree*, *k-regular*, and *star*, G is a graph that satisfies the property, and G' is a graph that does not satisfy the property. Therefore, we have the following theorem.

▶ **Theorem 6.** *Even if the upper bound of the number of agents is given, there exists no protocol that solves the line, ring, k-regular, star, or tree identification problem with the designated initial states under weak fairness.*

Note that, in Theorem 6, the bipartite identification problem is not included. However, we show later that there is no protocol that solves the bipartite identification problem even if the number of agents is given.

4.3 Impossibility with Knowledge of n under Weak Fairness

In this subsection, we show that, even if the number of agents n is given, there exists no protocol that solves the *line*, *ring*, *k-regular*, *tree*, or *bipartite* identification problem with designated initial states under weak fairness.

Case of Line, Ring, k-regular, and Tree

First, we show that there exists no protocol that solves the *line*, *ring*, *k-regular*, or *tree* identification problem. Concretely, we show that there is a case where a line graph and a ring graph are not distinguishable. To show this, first we give some notations. Let $G = (V, E)$ be a line graph with four agents, where $V = \{v_1, v_2, v_3, v_4\}$ and $E = \{(v_1, v_2), (v_2, v_3), (v_3, v_4)\}$. Let $G' = (V', E')$ be a ring graph with four agents, where $V' = \{v'_1, v'_2, v'_3, v'_4\}$ and $E' = \{(v'_1, v'_2), (v'_2, v'_3), (v'_3, v'_4), (v'_4, v'_1)\}$. Let s_0 be an initial state of agents. Let us consider a transition sequence $T = (s_0, s_0) \to (s_{a_1}, s_{b_1}), (s_{b_1}, s_{a_1}) \to (s_{b_2}, s_{a_2}), (s_{a_2}, s_{b_2}) \to (s_{a_3}, s_{b_3}), (s_{b_3}, s_{a_3}) \to (s_{b_4}, s_{a_4}), \ldots$. Since the number of states is finite, there are i and j such that $s_{a_i} = s_{a_j}$, $s_{b_i} = s_{b_j}$, and $i < j$ hold. Let sa and sb be states such that $sa = s_{a_i} = s_{a_j}$ and $sb = s_{b_i} = s_{b_j}$ hold.

Because of space constraints, we omit the details of the proof (see the full version in the [26]). The proof sketch is as follows: We construct a particular execution Ξ with G such that the decision of each agent converges to $yn \in \{yes, no\}$. In Ξ, agents repeat the following three sub-executions: 1) agents v_1 and v_2 interact repeatedly until v_1 and v_2 obtain sa and sb, respectively, 2) agents v_3 and v_4 interact repeatedly until v_3 and v_4 obtain sa and sb, respectively, and 3) v_3 and v_2 interact repeatedly until v_3 and v_2 obtain sa and sb again, respectively. Next, we construct a particular execution Ξ' with G'. In Ξ', agents repeat the following four sub-executions: 1) agents v'_1 and v'_2 interact repeatedly until v'_1 and v'_2 obtain sa and sb, respectively, 2) agents v'_3 and v'_4 interact repeatedly until v'_3 and v'_4 obtain sa and sb, respectively, 3) v'_3 and v'_2 interact repeatedly until v'_3 and v'_2 obtain sa and sb again, respectively, and 4) v'_1 and v'_4 interact repeatedly until v'_1 and v'_4 obtain sa and sb again, respectively. From the definition of sa and sb, we can construct those executions such that the executions satisfy weak fairness and agents converge to the decision yn.

▶ **Lemma 7.** *There exists a weakly-fair execution Ξ' of $\mathcal{P}$ with G' such that $\forall v' \in V'$: $\gamma(s(v')) = yn$ holds in a stable configuration of Ξ'.*

Note that, even if the number of agents is given, Lemma 7 holds because $|V| = |V'| = 4$ holds in the lemma. In Lemma 7, G is a line graph and a tree graph whereas G' is neither a line graph nor a tree graph. Furthermore, G' is a ring graph and a 2-regular graph whereas G is neither a ring graph nor a 2-regular graph. Hence, by Lemma 7, there is no protocol that solves the *line*, *ring*, *tree*, or *k-regular* identification problem, and thus we have the following theorem.

▶ **Theorem 8.** *Even if the number of agents n is given, there exists no protocol that solves the line, ring, k-regular, or tree identification problem with designated initial states under weak fairness.*

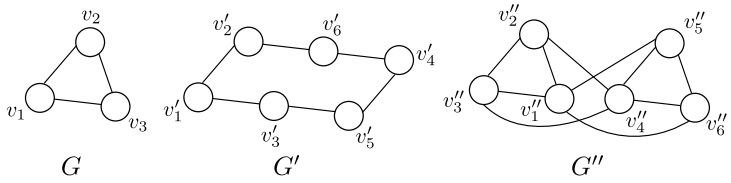

Figure 3 Graphs G, G', and G''.

Case of Bipartite

Next, we show that there exists no protocol that solves the bipartite identification problem. For the purpose of the contradiction, we assume that there exists a protocol $\mathcal{P}_{n=6}$ that solves the bipartite identification problem with designated initial states under weak fairness if the number of agents 6 is given.

We define a ring graph $G = (V, E)$ with three agents, a ring graph $G' = (V', E')$ with 6 agents, and a graph $G'' = (V'', E'')$ with 6 agents as follows:

- $V = \{v_1, v_2, v_3\}$ and $E = \{(v_1, v_2), (v_2, v_3), (v_3, v_1)\}$.
- $V' = \{v_1', v_2', v_3', v_4', v_5', v_6'\}$ and $E' = \{(v_1', v_2'), (v_2', v_6'), (v_6', v_4'), (v_4', v_5'), (v_5', v_3'), (v_3', v_1')\}$.
- $V'' = \{v_1'', v_2'', v_3'', v_4'', v_5'', v_6''\}$ and $E'' = \{(v_x'', v_y''), (v_{x+n}'', v_{y+n}'') \in V'' \times V'' \mid (v_x, v_y) \in E\} \cup \{(v_1'', v_5''), (v_1'', v_6''), (v_4'', v_2''), (v_4'', v_3'')\}$.

Figure 3 shows graphs G, G', and G''.

From now, we show that there exists an execution Ξ'' of $\mathcal{P}_{n=6}$ with G'' such that all agents converge to *yes* whereas G'' does not satisfy *bipartite*. To show this, we first show that, in any execution Ξ of $\mathcal{P}_{n=6}$ with G (i.e., the protocol for 6 agents is applied to a population consisting of 3 agents), all agents converge to *yes*. To prove this, we borrow the proof technique in [19]. In [19], Fischer and Jiang proved the impossibility of leader election for a ring graph.

▶ **Lemma 9.** *In any weakly-fair execution Ξ of $\mathcal{P}_{n=6}$ with G, all agents converge to yes. That is, in Ξ, there exists C_t such that $\forall v \in V : \gamma(s(v, C_i)) = yes$ holds for $i \geq t$.*

Proof sketch. For Ξ, we construct an execution Ξ' of $\mathcal{P}_{n=6}$ with G' such that v_1', v_2', and v_3' behave similarly to v_1, v_2, and v_3 in Ξ, respectively, and v_4', v_5', and v_6' also behave similarly to v_1, v_2, and v_3 in Ξ, respectively. Note that agents v_1', v_2', and v_3' and agents v_4', v_5', and v_6' operate in parallel. Since v_2' (resp., v_5') is not adjacent to v_3' (resp., v_6'), v_2' (resp., v_5') cannot interact with v_3' (resp., v_6'). When v_2' (resp., v_5') should interact with v_3' (resp., v_6'), v_2' (resp., v_5') interacts with v_6' instead of v_3' (resp., v_3' instead of v_6'). Since v_2' and v_5' (resp., v_3' and v_6') behave similarly to v_2 (resp., v_3), v_2' and v_5' (resp., v_3' and v_6') have the same state. Hence, even if v_2' (resp., v_5') interacts with v_6' instead of v_3' (resp., v_3' instead of v_6'), v_2' and v_3' (resp., v_5' and v_6') can transition similarly to v_2 and v_3.

In Ξ', since the number of agents is given correctly, a stable configuration exists. Hence, since G' is a bipartite graph, all agents converge to *yes* in Ξ'. This implies that all agents converge to *yes* even in Ξ. ◀

Now, we show that there exists execution Ξ'' of $\mathcal{P}_{n=6}$ with G'' such that all agents converge to *yes*. We show this by applying Lemma 5 to protocol $\mathcal{P}_{n=6}$ and graphs G and G''. Graphs G and G'' satisfy the condition of G and G' in Lemma 5, and the protocol $\mathcal{P}_{n=6}$ satisfies the condition of protocol $\mathcal{P}$ in Lemma 5. Thus, we can apply Lemma 5 to protocol

$\mathcal{P}_{n=6}$ and graphs G and G''. By applying Lemma 5, since all agents converge to *yes* in an execution of $\mathcal{P}_{n=6}$ with G by Lemma 9, there exists a weakly-fair execution Ξ'' of $\mathcal{P}_{n=6}$ with G'' in which all agents converge to *yes*.

▶ **Lemma 10.** *With the designated initial states, there exists a weakly-fair execution Ξ'' of $\mathcal{P}_{n=6}$ with G'' such that $\forall v'' \in V'' : \gamma(s(v'')) = yes$ in a stable configuration.*

Graph G'' does not satisfy *bipartite*. Thus, from Lemma 10, $\mathcal{P}_{n=6}$ is incorrect. Therefore, we have the following theorem.

▶ **Theorem 11.** *Even if the number of agents n is given, there exists no protocol that solves the bipartite identification problem with the designated initial states under weak fairness.*

4.4 Impossibility with Arbitrary Initial States

In this subsection, we show that, even if the number of agents n is given, there exists no protocol that solves the *line*, *ring*, *k-regular*, *star*, *tree*, or *bipartite* identification problem with arbitrary initial states under global fairness.

For the purpose of the contradiction, we assume that there exists a protocol $\mathcal{P}$ that solves some of the above graph class identification problems with arbitrary initial states under global fairness if the number of agents n is given. From now, we show that there are two executions Ξ and Ξ' of $\mathcal{P}$ such that the decision of all agents in the executions converges to the same value whereas Ξ and Ξ' are for graphs G and $G'(\neq G)$, respectively.

▶ **Lemma 12.** *Let $G = (V, E)$ and $G' = (V', E')$ be connected graphs that satisfy the following condition, where $V = \{v_1, v_2, v_3, \ldots, v_n\}$ and $V' = \{v'_1, v'_2, v'_3, \ldots, v'_n\}$.*

- *For some edge (v_α, v_β) in E, $E' = \{(v'_x, v'_y) \in V' \times V' \mid (v_x, v_y) \in E\} \backslash \{(v'_\alpha, v'_\beta)\}$.*

If there exists a globally-fair execution Ξ of $\mathcal{P}$ with G such that $\forall v \in V : \gamma(s(v)) = yn \in \{yes, no\}$ holds in a stable configuration of Ξ, there exists a globally-fair execution Ξ' of $\mathcal{P}$ with G' such that $\forall v' \in V' : \gamma(s(v')) = yn$ holds in a stable configuration of Ξ'.

Proof. Let $\Xi = C_0, C_1, C_2, \ldots$ be a globally-fair execution of $\mathcal{P}$ with G such that $\forall v \in V : \gamma(s(v)) = yn \in \{yes, no\}$ holds in a stable configuration C_t. For the purpose of the contradiction, we assume that there exists no execution of $\mathcal{P}$ with G' such that $\forall v' \in V' : \gamma(s(v')) = yn$ holds in a stable configuration.

Let us consider an execution $\Xi' = C'_0, C'_1, C'_2, \ldots, C'_{t'}, \ldots$ of $\mathcal{P}$ with G' such that, for $1 \leq i \leq n$, $s(v'_i, C'_0) = s(v_i, C_t)$ holds and $C'_{t'}$ is a stable configuration. By the assumption, $\exists v'_z \in V' : \gamma(s(v'_z, C'_{t'})) = yn'(\neq yn)$ holds.

Next, let us consider an execution $\Xi'' = C''_0, C''_1, C''_2, \ldots, C''_t, \ldots$ of $\mathcal{P}$ with G as follows:

- For $0 \leq i \leq t$, $C''_i = C_i$ holds (i.e., agents behave similarly to Ξ).
- For $t < i \leq t + t'$, when v'_x interacts with v'_y at $C'_{i-t-1} \to C'_{i-t}$, v_x interacts with v_y at $C''_{i-1} \to C''_i$. This is possible because $E' \subset E$ holds.

Since C_t is a stable configuration, C''_t is also a stable configuration and $\forall v \in V : \gamma(s(v, C''_t)) = yn$ holds. Since agents behave similarly to Ξ' after C''_t, $\gamma(s(v_z, C''_{t+t'})) = yn'$ holds. This contradicts the fact that C''_t is a stable configuration. ◀

We can construct a non-line graph, a non-ring graph, a non-star graph, a non-tree graph, and a non-bipartite graph by adding an edge to a line graph, a ring graph, a star graph, a tree graph, and a bipartite graph, respectively. Moreover, we can construct a k-regular graph by adding an edge to some non-k-regular graph. From Lemma 12, there is a case where the decision of all agents converges to the same value for each pair of graphs. Therefore, we have the following theorem.

▶ **Theorem 13.** *There exists no protocol that solves the line, ring, k-regular, star, tree, or bipartite identification problem with arbitrary initial states under global fairness.*

5 Concluding Remarks

In this paper, we consider the graph class identification problems on various assumptions. We have interesting open problems for future researches as follows:

- What is the space complexity of k-regular identification problem under global fairness with designated initial states and no initial knowledge.
- What is the time complexity of graph class identification problems?
- Are there some graph class identification protocols for other graph properties?

References

1 Dan Alistarh and Rati Gelashvili. Polylogarithmic-time leader election in population protocols. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming*, pages 479–491, 2015.

2 Dan Alistarh, Rati Gelashvili, and Joel Rybicki. Fast graphical population protocols. *arXiv preprint*, 2021. `arXiv:2102.08808`.

3 Dana Angluin, James Aspnes, Melody Chan, Michael J Fischer, Hong Jiang, and René Peralta. Stably computable properties of network graphs. In *Proceedings of International Conference on Distributed Computing in Sensor Systems*, pages 63–74, 2005.

4 Dana Angluin, James Aspnes, Zoë Diamadi, Michael J Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed computing*, 18(4):235–253, 2006.

5 Dana Angluin, James Aspnes, and David Eisenstat. A simple population protocol for fast robust approximate majority. *Distributed Computing*, 21(2):87–102, 2008.

6 Dana Angluin, James Aspnes, Michael J Fischer, and Hong Jiang. Self-stabilizing population protocols. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 3(4):13, 2008.

7 James Aspnes, Joffroy Beauquier, Janna Burman, and Devan Sohier. Time and space optimal counting in population protocols. In *Proceedings of International Conference on Principles of Distributed Systems*, pages 13:1–13:17, 2016.

8 Joffroy Beauquier, Janna Burman, Simon Claviere, and Devan Sohier. Space-optimal counting in population protocols. In *Proceedings of International Symposium on Distributed Computing*, pages 631–646, 2015.

9 Joffroy Beauquier, Julien Clement, Stephane Messika, Laurent Rosaz, and Brigitte Rozoy. Self-stabilizing counting in mobile sensor networks with a base station. In *Proceedings of International Symposium on Distributed Computing*, pages 63–76, 2007.

10 Petra Berenbrink, Robert Elsässer, Tom Friedetzky, Dominik Kaaser, Peter Kling, and Tomasz Radzik. Time-space trade-offs in population protocols for the majority problem. *Distributed Computing*, pages 1–21, 2020.

11 Petra Berenbrink, George Giakkoupis, and Peter Kling. Optimal time and space leader election in population protocols. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 119–129, 2020.

12 Petra Berenbrink, Dominik Kaaser, and Tomasz Radzik. On counting the population size. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 43–52, 2019.

13 Olivier Bournez, Jérémie Chalopin, Johanne Cohen, Xavier Koegler, and Mikael Rabie. Population protocols that correspond to symmetric games. *International Journal of Unconventional Computing*, 9, 2013.

14 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 of Computing Systems*, 50(3):433–445, 2012.

15 Ioannis Chatzigiannakis, Othon Michail, and Paul G. Spirakis. Stably decidable graph languages by mediated population protocols. In *Stabilization, Safety, and Security of Distributed Systems - 12th International Symposium, SSS*, volume 6366 of *Lecture Notes in Computer Science*, pages 252–266. Springer, 2010. doi:10.1007/978-3-642-16023-3_21.

16 Hsueh-Ping Chen and Ho-Lin Chen. Self-stabilizing leader election. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 53–59, 2019.

17 Hsueh-Ping Chen and Ho-Lin Chen. Self-stabilizing leader election in regular graphs. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, pages 210–217, 2020.

18 David Doty and David Soloveichik. Stable leader election in population protocols requires linear time. *Distributed Computing*, 31(4):257–271, 2018.

19 Michael Fischer and Hong Jiang. Self-stabilizing leader election in networks of finite-state anonymous agents. In *International Conference On Principles Of Distributed Systems*, pages 395–409. Springer, 2006.

20 Leszek Gąsieniec, David Hamilton, Russell Martin, Paul G Spirakis, and Grzegorz Stachowiak. Deterministic population protocols for exact majority and plurality. In *Proceedings of International Conference on Principles of Distributed Systems*, pages 14:1–14:14, 2016.

21 Leszek Gąsieniec, Grzegorz Stachowiak, and Przemyslaw Uznanski. Almost logarithmic-time space optimal leader election in population protocols. In *The 31st ACM on Symposium on Parallelism in Algorithms and Architectures*, pages 93–102. ACM, 2019.

22 Tomoko Izumi, Keigo Kinpara, Taisuke Izumi, and Koichi Wada. Space-efficient self-stabilizing counting population protocols on mobile sensor networks. *Theoretical Computer Science*, 552:99–108, 2014.

23 Othon Michail and Paul G Spirakis. Simple and efficient local codes for distributed stable network construction. *Distributed Computing*, 29(3):207–237, 2016.

24 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.

25 Yuichi Sudo, Masahiro Shibata, Junya Nakamura, Yonghwan Kim, and Toshimitsu Masuzawa. Self-stabilizing population protocols with global knowledge. *IEEE Transactions on Parallel and Distributed Systems*, 2021.

26 Hiroto Yasumi, Fukuhito Ooshita, and Michiko Inoue. Population protocols for graph class identification problems, 2021. arXiv:2111.05111.

A Pseudocode of Protocols

The descriptions of pseudocodes in this appendix are appeared in [26].

A.1 Tree Identification Protocol

Algorithm 1 shows the tree identification protocol in Subsection 3.1. The roles of the variables at agent a in Protocol 1 are as follows.

- $LF_a \in \{L_{se}, L_l, L_r, L^t_{se}, L^t_{se'}, L_{se'}, L^t_l, L^t_r, \phi\}$: Variable LF_a, initialized to L_r, represents a token held by agent a. If LF_a is not ϕ, agent a has LF_a token. There are three types of tokens: a leader token (L_{se}, L^t_{se}, $L^t_{se'}$, and $L_{se'}$), a left token (L_l and L^t_l), and a right token (L_r and L^t_r). L_l, and L_r tokens are the tokens in non-trial modes. L^t_l and L^t_r tokens represent the left token and the right token in the trial mode, respectively. L^t_{se}, $L^t_{se'}$, and $L_{se'}$ tokens represent a progress of a trial of the cycle detection. L^t_{se} token represents that the left token has been placed. $L^t_{se'}$ token represents that the right token has been placed. $L_{se'}$ token represents that the token confirms that the L^t_l token is still placed on the certain agent. ϕ represents no token.

- $tre_a \in \{yes, no\}$: Variable tre_a, initialized to yes, represents a decision of the tree. If $tre_a = yes$ holds for agent a, then $\gamma(s_a) = yes$ holds (i.e., a decides that the given graph is a tree). If $tre_a = no$ holds, then $\gamma(s_a) = no$ holds (i.e., a decides that the given graph is not a tree).

The protocol uses 18 states because the number of values taken by variable LF_a is 9 and the number of values taken by variable tre_a is 2.

Algorithm 1 A tree identification protocol (1/2).

Variables at an agent a:

$LF_a \in \{L_{se}, L_l, L_r, L^t_{se}, L^t_{se'}, L_{se'}, L^t_l, L^t_r, \phi\}$: Token held by the agent, initialized to L_r.

$tre_a \in \{yes, no\}$: Decision of the tree, initialized to yes.

1: **when** agent a interacts with agent b **do**

{ The election of tokens }

2: **if** $LF_a, LF_b \in \{L^t_r, L_r\}$ **then**

3: $LF_b \leftarrow L_l$

4: **else if** $LF_a, LF_b \in \{L^t_l, L_l\}$ **then**

5: $LF_b \leftarrow L_{se}$

6: **else if** $LF_a, LF_b \in \{L_{se}, L^t_{se}, L^t_{se'}, L_{se'}\}$ **then**

7: $LF_a \leftarrow L_{se}$, $LF_b \leftarrow \phi$

8: $tre_a \leftarrow yes$

{ Movement of tokens }

9: **else if** $LF_a \neq \phi \wedge LF_b = \phi$ **then**

10: **if** $LF_a \in \{L_{se}, L^t_{se}, L^t_{se'}, L_{se'}\}$ **then**

11: $tre_b \leftarrow tre_a$

12: **end if**

13: **if** $LF_a = L^t_\kappa$ for $\kappa \in \{l, r\}$ **then**

14: $LF_a \leftarrow L_\kappa$

15: **else if** $LF_a = L_{se'} \vee LF_a = L^t_{se'}$ **then**

16: $LF_a \leftarrow L_{se}$

17: **end if**

18: $LF_a \leftrightarrow LF_b$ *

{ Decision }

19: **else if** $LF_a = L_{se} \wedge LF_b = L_l$ **then**

20: $LF_a \leftarrow L^t_l$, $LF_b \leftarrow L_{se'}$

21: $tre_b \leftarrow tre_a$

22: **else if** $LF_a = L_{se'} \wedge LF_b = L_r$ **then**

23: $LF_a \leftarrow L^t_r$, $LF_b \leftarrow L^t_{se}$

24: $tre_b \leftarrow tre_a$

25: **else if** $LF_a = L^t_{se} \wedge LF_b = L^t_l$ **then**

26: $LF_a \leftarrow L_l$, $LF_b \leftarrow L^t_{se'}$

27: $tre_b \leftarrow tre_a$

28: **else if** $LF_a = L^t_{se'} \wedge LF_b = L^t_r$ **then**

29: $LF_a \leftarrow L_r$, $LF_b \leftarrow L_{se}$

30: $tre_b \leftarrow no$

* $p \leftrightarrow q$ means that p and q exchange values.

▷ Continued on the next page

Algorithm 1 A tree identification protocol (2/2).

```
31:     else if LF_a ≠ φ ∧ LF_b ≠ φ then
32:         if LF_ab ∈ {L_se, L^t_se, L^t_se', L_se'} for ab ∈ {a, b} then
33:             tre_a ← tre_ab, tre_b ← tre_ab
34:         end if
35:         if LF_ab = L^t_κ for ab ∈ {a, b} and κ ∈ {l, r} then
36:             LF_ab ← L_κ
37:         end if
38:         if LF_ab = L_se' ∨ LF_ab = L^t_se ∨ LF_ab = L^t_se' for ab ∈ {a, b} then
39:             LF_ab ← L_se
40:         end if
41:         LF_a ↔ LF_b
42:     end if
43: end
```

A.2 k-regular Identification Protocol

Algorithm 2 shows the k-regular identification protocol in Subsection 3.2.

The roles of the variables at agent a in Protocol 2 are as follows.

- $LF_a \in \{L_0, L_1, \ldots, L_k, \phi, \phi'\}$: Variable LF_a, initialized to L_0, represents states for a leader token and marked agents. If LF_a is neither ϕ nor ϕ', agent a has a leader token. In particular, if $LF_a = L_i (i \in \{0, 1, \ldots, k\})$ holds, agent a has an L_i token. Moreover, $LF_a = L_i$ represents that agent a has interacted with i different non-marked agents (i.e., agent a has at least i edges). If $LF_a = \phi$ holds, agent a has no leader token. If $LF_a = \phi'$ holds, agent a has no leader token and a is marked by other agents.
- $level_a \in \{0, 1, 2, \ldots, \lfloor \log P \rfloor\}$: Variable $level_a$, initialized to 0, represents the level of agent a.
- $reg_a \in \{yes, no\}$: Variable reg_a, initialized to no, represents the decision of the k-regular graph. If $reg_a = yes$ holds for agent a, then $\gamma(s_a) = yes$ holds. If $reg_a = no$ holds, then $\gamma(s_a) = no$ holds.

The protocol uses $O(k \log P)$ states because the number of values taken by variable LF_a is $k + 2$, the number of values taken by variable $level_a$ is $\lfloor \log P \rfloor + 1$, and the number of values taken by other variables (loc_a and reg_a) is constant.

A.3 Star Identification Protocol

Algorithm 3 shows the star identification protocol in Subsection 3.3.

The roles of the variables at agent a in Protocol 3 are as follows.

- $LF_a \in \{\phi, \phi', l', L_2, L_3, \ldots, L_{n-1}\}$: Variable LF_a, initialized to ϕ, represents a role of agent a. $LF_a = L_i$ means that a central agent a has marked i agents (i.e., agent a has at least i edges). $LF_a = l'$ means that a is a candidate of a central agent and is a marked agent. $LF_a = \phi$ means that agent a is a non-marked agent. $LF_a = \phi'$ means that agent a is a marked agent. When $LF_a = x$ holds, we refer to a as an x-agent.
- $star_a \in \{yes, no, never\}$: Variable $star_a$, initialized to no, represents a decision of a star. If $star_a = yes$ holds, $\gamma(s_a) = yes$ holds (i.e., a decides that a given graph is a star). If $star_a = no$ or $star_a = never$ holds, $\gamma(s_a) = no$ holds (i.e., a decides that a given graph is not a star). $star_a = never$ means the stronger decision of no. If agent a with $star_a = never$ interacts with agent b, $star_b$ transitions to $never$ regardless of the value of $star_b$.

Algorithm 2 A k-regular identification protocol.

Variables at an agent a:

$LF_a \in \{L_0, L_1, \ldots, L_k, \phi, \phi'\}$: States for a leader token and marked agents, initialized to L_0.

$level_a \in \{0, 1, 2, \ldots, \lfloor \log P \rfloor\}$: States for the level of agent a, initialized to 0.

$loc_a \in \{yes, no\}$: States representing whether the degree of agent a is at least k, initialized to no.

$reg_a \in \{yes, no\}$: Decision of the k-regular graph, initialized to no.

1: **when** agent a interacts with agent b **do**

⟨ The behavior when agents have the same level ⟩

2: **if** $level_a = level_b$ **then**

{ The election of leader tokens }

3: **if** $LF_a = L_x \wedge LF_b = L_y$ $(x, y \in \{0, 1, 2, \ldots, k\})$ **then**

4: $level_a \leftarrow level_a + 1$

5: $LF_a \leftarrow L_0$, $LF_b \leftarrow \phi$

6: $reg_a \leftarrow no$

7: $loc_a \leftarrow no$

{ Decision and movement of the token }

8: **else if** $LF_a = L_x \wedge LF_b = \phi$ $(x \in \{0, 1, 2, \ldots, k-2\})$ **then**

9: $LF_a \leftarrow L_{x+1}$, $LF_b \leftarrow \phi'$

10: **else if** $LF_a = L_x \wedge LF_b = \phi'$ $(x \in \{0, 1, 2, \ldots, k\})$ **then**

11: $LF_a \leftarrow \phi$, $LF_b \leftarrow L_0$

12: $reg_b \leftarrow reg_a$

13: **else if** $LF_a = L_{k-1} \wedge LF_b = \phi$ **then**

14: $LF_a \leftarrow L_k$, $LF_b \leftarrow \phi'$

15: **if** $loc_a = no$ **then**

16: $reg_a \leftarrow yes$

17: $loc_a \leftarrow yes$

18: **end if**

{ Reset of reg of the leader token (the degree of agent a is at least $k+1$) }

19: **else if** $LF_a = L_k \wedge LF_b = \phi$ **then**

20: $LF_a \leftarrow L_0$, $LF_b \leftarrow \phi'$

21: $reg_a \leftarrow no$

22: **end if**

{ Reset of reg of the leader token (loc_a or loc_b is no) }

23: **if** $loc_a = no \vee loc_b = no$ **then**

24: $reg_a \leftarrow no$, $reg_b \leftarrow no$

25: **end if**

⟨ The behavior when agents have different levels ⟩

26: **else if** $level_a > level_b$ **then**

27: $level_b \leftarrow level_a$

28: $loc_b \leftarrow no$

29: $LF_b \leftarrow \phi$

30: **end if**

31: **end**

The protocol is given in Algorithm 3. Algorithm 3 uses $3n + 3$ states because the number of values taken by variable LF_a is $n + 1$ and the number of values taken by variable $star_a$ is 3.

Algorithm 3 A star identification protocol.

A variable at an agent a:

$LF_a \in \{\phi, \phi', l', L_2, L_3, \ldots, L_{n-1}\}$: States that represent roles of agents, initialized to ϕ. L_i represents a central agent, l' represents a candidate of the central agent, ϕ' represents a marked agent, and ϕ represents a non-marked agent.

$star_a \in \{yes, no, never\}$: Decision of a star, initialized to no.

1: **when** agent a interacts with agent b **do**

⟨ The behavior when $star_a$ or $star_b$ is $never$ ⟩

2: **if** $star_a = never \lor star_b = never$ **then**

3: $star_a \leftarrow never$, $star_b \leftarrow never$

⟨ The behaviors when $star_a \neq never$ and $star_b \neq never$ holds ⟩

4: **else**

{ The election of a central agent }

5: **if** $LF_a = \phi \land LF_b = \phi$ **then**

6: $LF_a \leftarrow l'$, $LF_b \leftarrow l'$

7: **else if** $LF_a = l' \land LF_b = \phi$ **then**

8: $LF_a \leftarrow L_2$, $LF_b \leftarrow \phi'$

{ Counting the number of adjacent agents by the central agent }

9: **else if** $LF_a = L_i \land LF_b = \phi$ $(2 \leq i \leq n-2)$ **then**

10: $LF_a \leftarrow L_{i+1}$, $LF_b \leftarrow \phi'$

11: **end if**

12: **if** $LF_a = L_{n-1}$ **then**

13: $star_a \leftarrow yes$, $star_b \leftarrow yes$

14: **end if**

{ Decision of $never$ }

15: **if** $LF_a = \phi' \land LF_b = \phi'$ **then**

16: $star_a \leftarrow never$, $star_b \leftarrow never$

17: **else if** $LF_a = \phi' \land LF_b = l'$ **then**

18: $star_a \leftarrow never$, $star_b \leftarrow never$

19: **end if**

{ Conveyance of yes }

20: **if** $star_a = yes \lor star_b = yes$ **then**

21: $star_a \leftarrow yes$, $star_b \leftarrow yes$

22: **end if**

23: **end if**

24: **end**