Self-Stabilizing Population Protocols With Global Knowledge

Yuichi Sudo, *Member, IEEE*, Masahiro Shibata, Junya Nakamura, *Member, IEEE*, Yonghwan Kim, and Toshimitsu Masuzawa, *Member, IEEE*

Abstract—In the population protocol model, many problems cannot be solved in a self-stabilizing manner. However, global knowledge, such as the number of nodes in a network, sometimes enables the design of a self-stabilizing protocol for such problems. For example, it is known that we can solve the self-stabilizing leader election in complete graphs if and only if every node knows the exact number of nodes. In this article, we investigate the effect of global knowledge on the possibility of self-stabilizing population protocols in arbitrary graphs. Specifically, we clarify the solvability of the leader election problem, the ranking problem, the degree recognition problem, and the neighbor recognition problem by self-stabilizing population protocols with knowledge of the number of nodes and/or the number of edges in a network.

Index Terms—Population protocols, leader election, self-stabilization

1 Introduction

We consider the *population protocol* (PP) model [2] in this paper. A network called *population* consists of a large number of finite-state automata, called *agents*. Agents make *interactions* (i.e., pairwise communication) with each other to update their states. The interactions are opportunistic, i.e., they are unpredictable for the agents. Agents are strongly anonymous: they do not have identifiers, and they cannot distinguish their neighbors in the same states. One example represented by this model is a flock of birds where each bird is equipped with a sensing device with a small transmission range. Two devices can communicate (i.e., interact) with each other only when the corresponding birds come sufficiently close to each other. Therefore, an agent cannot predict when its next interaction will occur. A population is modeled by a graph $G=(V,E)$, where V represents the set of agents, and E indicates which pair of agents can interact. Each pair of agents $(u,v)\in E$ has interactions infinitely often, while each pair of agents $(u',v')\notin E$ never has an interaction. In the field of population protocols, many efforts have been devoted to devising protocols for a complete graph, i.e., a population where every pair of agents interacts infinitely often. In addition, several studies [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12] have investigated populations represented by a general graph.

Self-stabilization [13] is a fault-tolerant property whereby, even when any transient fault (e.g., memory crash) occurs, the network can autonomously recover from the fault. Formally, self-stabilization is defined as follows: (i) starting from an arbitrary configuration, a network eventually reaches a *safe configuration* (*convergence*), and (ii) once a network reaches a safe configuration, it maintains its specification forever (*closure*). Self-stabilization is of great importance in the PP model because self-stabilization tolerates any finite number of transient faults, and this is a necessary property in a network consisting of a large number of inexpensive and unreliable sensor nodes.

Consequently, many studies have been devoted to self-stabilizing population protocols [3], [4], [5], [6], [9], [10], [11], [14], [15], [16], [17], [18]. Angluin *et al.* [3] proposed self-stabilizing protocols for a variety of problems,i.e., leader election in rings whose sizes are not multiples of a given integer k (in particular, rings of odd size), token circulation in rings with a pre-selected leader, 2-hop coloring in degree-bounded graphs, consistent global orientation in undirected rings, and spanning-tree construction in regular graphs. The protocols for the first four problems use only a constant agent memory space, while the protocol for the last problem requires $O(\log D)$ bits of agent memory, where D is (a known upper bound[1] on) the diameter of the graph. Chen and Chen [6] gave a constant-space and self-stabilizing protocol for the leader election in rings with arbitrary size.

- *Yuichi Sudo is with Hosei University, Tokyo 102-8160, Japan. E-mail: sudo@hosei.ac.jp.*
- *Masahiro Shibata is with the Kyushu Institute of Technology, Kitakyushu 804-8550, Japan. E-mail: shibata@cse.kyutech.ac.jp.*
- *Junya Nakamura is with the Toyohashi University of Technology, Toyohashi 441-8580, Japan. E-mail: junya@imc.tut.ac.jp.*
- *Yonghwan Kim is with the Nagoya Institute of Technology, Nagoya 466-8555, Japan. E-mail: kim@nitech.ac.jp.*
- *Toshimitsu Masuzawa is with the Osaka University, Osaka 565-0871, Japan. E-mail: masuzawa@ist.osaka-u.ac.jp.*

Manuscript received 27 July 2020; revised 12 Apr. 2021; accepted 27 Apr. 2021. Date of publication 3 May 2021; date of current version 9 June 2021. (Corresponding author: Yuichi Sudo.) Recommended for acceptance by D. Yang. Digital Object Identifier no. 10.1109/TPDS.2021.3076769

1. In [3], D is defined as the diameter of the graph, not a known upper bound on it. However, since we must consider an arbitrary initial configuration, we require an upper bound on the diameter; Otherwise, the agents need memory of unbounded size. Fortunately, the knowledge of the upper bound is not a strong assumption in this case: any upper bound that is polynomial in the true diameter is acceptable since the space complexity is $O(\log D)$ bits.

On the negative side, Angluin *et al.* [3] proved that the self-stabilizing leader election (SS-LE) is impossible for arbitrary graphs. In particular, it immediately follows from their theorem that no protocol solves SS-LE in complete graphs with three different sizes, i.e., in all of K_i, K_j, and K_k for any distinct integers $i, j, k \geq 2$, where K_l is a complete graph with size l. Cai *et al.* [14] proved that no protocol solves SS-LE both in K_i and in K_{i+1} for any integer $i \geq 2$. In almost the same way, we can easily observe that no protocol solves SS-LE both in K_i and K_j for any distinct integers $i, j \geq 2$. (A more detailed explanation is provided in a previous study [18].) In other words, SS-LE is impossible in complete graphs unless the exact number of agents in the population is known to the agents. Note that Cai *et al.* [14] also gave a protocol that solves SS-LE in K_l for a given integer l. Thus, the knowledge of the exact number of agents is necessary and sufficient to solve SS-LE in a complete graph.

In addition to [3], [6], [14], many studies have focused on SS-LE because leader election is one of the most fundamental problems in the PP model. Several important protocols [2], [3], [19] require a pre-selected unique leader. In particular, as shown Angluin *et al.* [19], all semi-linear predicates can be solved very quickly if we have a unique leader. However, we have strong impossibility as mentioned above, i.e., SS-LE cannot be solved unless knowledge of the exact number of agents is given to the agents. In the literature, three approaches to overcome this impossibility are identified. One approach [14], [20] is to assume that every agent knows the exact number of agents. The protocol proposed by Cai *et al.* [14] uses $O(\log n)$ bits (n states) of memory space per agent and converges within $O(n^3)$ steps in expectation in the complete graph of n agents under *the uniformly random scheduler*, which selects a pair of interacting agents uniformly at random from all pairs at each step. Burman *et al.* [20] proposed three SS-LE protocols, also for the complete graph of n agents, performed faster than the protocol proposed by Cai *et al.* [14]. These self-stabilizing protocols [14], [20] solve not only the leader election problem but also *the ranking problem*, which requires ranking n agents by assigning them different integers from $0, 1, \ldots, n-1$. The results for the other two approaches to overcome the impossibility, i.e., SS-LE protocols with *oracles* [4], [5], [15] and *loosely-stabilizing protocols* [9], [10], [11], [16], [17], [18], are discussed in Section 1.2.

1.1 Our Contribution

As mentioned above, we can solve the self-stabilizing leader election in complete graphs if and only if we have knowledge of the exact number of agents. In this paper, we investigate how powerful global knowledge, such as the exact number of agents in the population, is for designing self-stabilizing population protocols for *arbitrary graphs*. Specifically, we consider two types of global knowledge, the number of agents and the number of edges (i.e., interactable pairs) in the population, and clarify the relationships between the knowledge and the solvability of the following four problems.

- leader election (**LE**): Elect exactly one leader,
- ranking (**RK**): Assign the agents in the population $G = (V_G, E_G)$ distinct integers (or *ranks*) from 0 to $|V_G| - 1$,
- degree recognition (**DR**): Let each agent recognize its degree in the graph,
- neighbor recognition (**NR**): Let each agent recognize the set of its neighbors in the graph. Since the population is anonymous, this problem also requires 2-hop coloring, i.e., all agents must be assigned integers (or *colors*) such that all neighbors of any agent have different colors.

In addition to the above specifications, we require that no agent changes its outputs (e.g., its *rank* in **RK**) after the population converges, i.e., it reaches a safe configuration.

We denote $A_1 \preceq A_2$ if problem A_1 is reducible to A_2. We have **LE** $\preceq$ **RK** and **DR** $\preceq$ **NR**. The first relationship holds because if the agents are labeled $0, 1, \ldots, |V_G| - 1$, **LE** is immediately solved by selecting the agent with label 0 as the unique leader. The second relationship is trivial.

To clarify our contributions, we formally define the global knowledge that we consider. Define $\mathcal{G}_{n,m}$ as the set of all the simple, undirected, and connected graphs with n nodes and m edges. Let ν and μ be any sets of positive integers such that $\nu \subseteq \mathbb{N}_{\geq 2} = \{n \in \mathbb{N} \mid n \geq 2\}$ and $\mu \subseteq \mathbb{N}_{\geq 1} = \{m \in \mathbb{N} \mid m \geq 1\}$. Then, we define $\mathcal{G}_{\nu,\mu} = \bigcup_{n \in \nu, m \in \mu} \mathcal{G}_{n,m}$. For simplicity, we define $\mathcal{G}_{\nu,*} = \mathcal{G}_{\nu,\mathbb{N}_{\geq 1}}$ and $\mathcal{G}_{*,\mu} = \mathcal{G}_{\mathbb{N}_{\geq 2},\mu}$ for any $\nu \subseteq \mathbb{N}_{\geq 2}$ and $\mu \subseteq \mathbb{N}_{\geq 1}$. We consider that ν and μ are global knowledge on the population. Here, ν is the set of the possible numbers of agents, and μ is the set of the possible numbers of interactable pairs. In other words, when we are given ν and μ, our protocol has to solve a problem only in the populations represented by the graphs in $\mathcal{G}_{\nu,\mu}$. We say that protocol P solves problem *Ain arbitrary graphs* given knowledge ν and μ if P solves A in all graphs in $\mathcal{G}_{\nu,\mu}$.

In this paper, we investigate the solvability of **LE**, **RK**, **DR**, and **NR** for arbitrary graphs with the knowledge ν and μ. Specifically, we prove the following propositions assuming that the agents are given knowledge ν and μ:

1) When the agents know nothing about the number of interactable pairs, i.e., $\mu = \mathbb{N}_{\geq 1}$, there exists a self-stabilizing protocol that solves **LE** and **RK** in arbitrary graphs if and only if the agents know the exact number of agents i.e., $\mathcal{G}_{\nu,\mu} = \mathcal{G}_{n,*}$ for some $n \in \mathbb{N}_{\geq 2}$.
2) There exists a self-stabilizing protocol that solves **NR** ($\succeq$ **DR**) in arbitrary graphs if the agents know the exact number of agents and the exact number of interactable pairs i.e., $\mathcal{G}_{\nu,\mu} = \mathcal{G}_{n,m}$ holds for some $n \in \mathbb{N}_{\geq 2}$ and $m \in \mathbb{N}_{\geq 1}$.
3) The knowledge of the exact number of agents is not sufficient to design a self-stabilizing protocol that solves **DR** ($\preceq$ **NR**) in arbitrary graphs if the agents do not know the number of interactable pairs *exactly*. Specifically, no self-stabilizing protocol solves **DR** in all graphs in $\mathcal{G}_{\nu,\mu}$ if $\mathcal{G}_{n,m_1} \cup \mathcal{G}_{n,m_2} \subseteq \mathcal{G}_{\nu,\mu}$ holds for some $n \in \mathbb{N}_{\geq 2}$ and some distinct $m_1, m_2 \in \mathbb{N}_{\geq 1}$ such that $\mathcal{G}_{n,m_1} \neq \emptyset$ and $\mathcal{G}_{n,m_2} \neq \emptyset$.

In the most of standard distributed computing models, each node always has its local knowledge, e.g., its degree and the set of its neighbors. In the PP model, the agents do not have the local knowledge *a priori*, and many impossibility results (e.g., the impossibility of SS-LE in complete graphs [3], [14]) come from the lack of the local knowledge. Interestingly, the third proposition yields that, for self-stabilizing population protocols, obtaining some local knowledge (degree recognition of each agent) is at least as difficult as obtaining the corresponding global knowledge (the number of interactable pairs). It is also worthwhile to mention that the PP model is greatly empowered if **LE** and **NR** are solved. After the agents recognize their neighbors correctly, the population can simulate one of the most standard distributed computing models, i.e., the message passing model, if each agent maintains a variable corresponding to a *message buffer* for each neighbor. Moreover, we have the unique leader in the population, by which we can easily break the symmetry of a graph and solve many important problems even in a self-stabilizing way. For example, we can construct a spanning tree rooted by the leader. This fact and the above propositions show how powerful this type of global knowledge is when designing self-stabilizing population protocols.

One may think that the assumption that all agents have global knowledge such as the number of agents and/or the number of edges is impractical. This may be true in some applications. However, this impracticality does not necessarily impair the importance of the contribution of this paper. Our results show that such global knowledge is necessary and sufficient to solve the above fundamental problems. Thus, to solve the problems, we must make some other assumptions on the original model of population protocols so that the agents can obtain the global knowledge. We leave open an interesting question what kind of (practical) assumptions (e.g., restricting the topology of graphs) enables the agents to compute the number of agents and/or the number of edges in a self-stabilizing fashion.

A preliminary extended abstract [1] of this article appeared in the proceedings of SIROCCO 2020.

1.2 Other Related Work

Several studies employ *oracles*, a kind of failure detectors, to solve SS-LE. Fischer and Jiang [15] introduced oracle $\Omega?$ that eventually tells all agents whether or not at least one leader exists and proposed two protocols that solve SS-LE for rings and complete graphs using $\Omega?$. Beauquier *et al.* [4] presented an SS-LE protocol for arbitrary graphs that uses two copies of $\Omega?$. One copy is used to detect the existence of a leader and the other is used to detect the existence of a special agent called *a token*. Canepa *et al.* [5] proposed two SS-LE protocols that use $\Omega?$ and require only a single bit of each agent. One is a deterministic protocol for trees and the other is a randomized protocol for arbitrary graphs although the position of the leader is not static and moves among the agents forever.

To solve SS-LE without oracles or the knowledge of the exact number of agents, Sudo *et al.* [17] introduced the concept of $loose-stabilization$, which relaxes the closure requirement of self-stabilization, but retains its advantage in practice. Specifically, loose-stabilization guarantees that (i) starting from any configuration, the population reaches a safe configuration within a relatively short time, and (ii) thereafter the problem specification, such as having a unique leader, must be sustained for a sufficiently long time, although not necessarily forever. In [17], a loosely-stabilizing leader election (LS-LE) protocol was given for the first time. Here, the proposed protocol assumes that the population is a complete graph and every agent knows a common *upper bound*N of n, where n is the number of agents in the population. This protocol is practically equivalent to an SS-LE protocol since it maintains the unique leader for an exponentially large number of steps in expectation (i.e., practically forever) after reaching a safe configuration within $O(nN\log N)$ steps in expectation. The assumption that an upper bound N of n can be used is practical because the protocol works correctly even if n is significantly overestimated, such as $N = 10n$. Izumi [16] proposed a method that reduces the number of steps for convergence to $O(nN)$. Later, Sudo *et al.* [18] presented a much faster loosely-stabilizing leader election protocol for complete graphs. Given parameter $\tau \geq 10$, it reaches a safe configuration within $O(\tau n\log^3 N)$ steps and thereafter it retains the unique leader for $\Omega(n^{10\tau})$ steps, both in expectation. Very recently, Sudo *et al.* [21] gave a time optimal protocol for complete graphs where the convergence time is $O(\tau n\log N)$ steps and the holding time (i.e., the number of steps where the unique leader is retained) is $\Omega(n^{\tau+1})$ steps in expectation. In addition, several studies have presented, LS-LE protocols for arbitrary graphs [9], [10], [11].

2 Preliminaries

A *population* is represented by a simple and connected graph $G = (V_G, E_G)$, where V_G is the set of the *agents* and $E_G \subseteq V_G \times V_G$ is the set of the *interactable pairs* of agents. If $(u, v) \in E_G$, two agents u and v can interact in the population G, where u serves as the *initiator* and v serves as the *responder* of the interaction. In this paper, we consider only *undirected* populations, i.e., we assume that, for any population G, $(u, v) \in E_G$ yields $(v, u) \in E_G$ for any $u, v \in V_G$. For formality, we sometimes use $\overline{G} = (V_G, E_{\overline{G}})$ to denote the undirected graph corresponding to population G, i.e., $E_{\overline{G}} = \{\{u, v\} \mid (u, v) \in E_G\}$. We define the set of the neighbors of agent v as $N_G(v) = \{u \in V_G \mid \{v, u\} \in E_{\overline{G}}\}$ and define the degree of v as $\delta_G(v) = |N_G(v)|$.

A *protocol* $P(Q, Y, T, \pi_{\mathrm{out}})$ consists of a finite set Q of states, a finite set Y of output symbols, a transition function $T : Q \times Q \to Q \times Q$, and an output function $\pi_{\mathrm{out}} : Q \to Y$. When two agents interact, T determines their next states according to their current states. The *output* of an agent is determined by π_{out}, i.e., the output of an agent in state q is $\pi_{\mathrm{out}}(q)$. As mentioned in Section 1, we assume that the agents can use knowledge ν and μ. Therefore, the four parameters of protocol P, i.e., Q, Y, T, and π_{out}, may depend on ν and μ. We sometimes write $P(\nu, \mu)$ explicitly to denote protocol P with knowledge ν and μ.

A *configuration* on population G is a mapping $C : V_G \to Q$ that specifies the states of all agents in G. The set of all configurations of protocol P on population G is denoted by $\mathcal{C}_{\mathrm{all}}(P, G)$. We say that a configuration C changes to C' by an interaction $e = (u, v)$, denoted by $C \overset{P,e}{\to} C'$, if $(C'(u), C'(v)) = T(C(u), C(v))$ and $C'(w) = C(w)$ for all $w \in V \backslash \{u, v\}$. We also denote $C \overset{P,G}{\to} C'$ if $C \overset{P,e}{\to} C'$ holds for some $e \in E_G$. We say that a configuration C' is reachable from C by P on population G if there is a sequence of configurations $C_0, C_1, \ldots, C_k$ such that $C = C_0$, $C' = C_k$, and $C_i \overset{P,G}{\to} C_{i+1}$ for $i = 0, 1, \ldots, k-1$. In addition, we say that a set $\mathcal{S}$ of configurations is *closed* if no configuration outside $\mathcal{S}$ is reachable from a configuration in $\mathcal{S}$.

An *execution* of protocol P on population G is an infinite sequence of configurations $\Xi = C_0, C_1, \ldots$ such that $C_i \overset{P,G}{\to} C_{i+1}$ for $i = 0, 1, \ldots$. We call C_0 the initial configuration of the execution Ξ. We must assume some kind of fairness of an execution; otherwise, for example, we cannot exclude an execution such that only one pair of agents have interactions in a row and all other pairs have no interaction forever. Unlike most distributed computing models in the literature, *global fairness* is usually assumed in the PP model. We say that an execution $\Xi = C_0, C_1, \ldots$ of P on population G satisfies global fairness (or Ξ is globally fair) if for any configuration C that appears infinitely often in Ξ, every configuration C' such that $C \overset{P,G}{\rightarrow} C'$ also appears infinitely often in Ξ.

A problem is P specified by a predicate on the outputs of the agents. We call this predicate the *specification* of P. We say that a configuration C satisfies the specification of P if the outputs of the agents satisfy it in C. We consider the following four problems in this paper.

Definition 1 (LE). *The specification of the leader election problem (*__LE__*) requires that exactly one agent outputs L and all other agents output F.*

Definition 2 (RK). *The specification of the ranking problem (*__RK__*) requires that in the population $G = (V_G, E_G)$, the set of the outputs of the agents in the population equals to $\{0, 1, \ldots, |V_G| - 1\}$.*

Definition 3 (DR). *The specification of the degree recognition problem (*__DR__*) requires that in the population $G = (V_G, E_G)$, every agent $v \in V_G$ outputs $|N_G(v)|$.*

Definition 4 (NR). *The specification of the neighbor recognition problem (*__NR__*) requires that in the population $G = (V_G, E_G)$, every agent $v \in V_G$ outputs a two-tuple $(c_v, S_v) \in \mathbb{Z} \times 2^{\mathbb{Z}}$ such that, for all $v \in V_G$, we have $S_v = \{c_u \mid u \in N_G(v)\}$ and $|S_v| = |N_G(v)|$.*

The definition of **NR** is complicated while those of the other three problems are simple, thus we will give the intuitive explanation about **NR**. The integer c_v represents the *color* of agent v. The first condition $S_v = \{c_u \mid u \in N_G(v)\}$ implies that v outputs the set of the colors of v's neighbors correctly. The second condition $|S_v| = |N_G(v)|$, together with the first condition, implies that no two neighbors of v have the same color. Thus, the colors are so-called two-hop coloring (or agents with distance 1 or 2 have different colors).

Now, we define self-stabilizing protocols in Definitions 5 and 6, where we use the definitions given in Section 1.1 for knowledge ν and μ and the set $\mathcal{G}_{\nu,\mu}$ of graphs. Note that Definition 5 is not sufficient if we consider *dynamic problems*, such as the token circulation, where the specifications must be defined as predicates on *executions* rather than configurations. However, in this paper, we only consider static problems; thus this definition is sufficient for our purpose.

Definition 5 (Safe configuration). *Given a protocol P and a population G, we say that a configuration $C \in \mathcal{C}_{\text{all}}(P(\nu, \mu), G)$ is* safe *for problem A if (i) C satisfies the specification of problem A, and (ii) no agent changes its output in any execution of P on G starting from C.*

Definition 6 (Self-stabilizing protocol). *For any ν and μ, we say that a protocol P is a self-stabilizing protocol that solves problem A in arbitrary graphs given knowledge ν and μ if every globally-fair execution of $P(\nu, \mu)$ on any population G, which starts from any configuration $C_0 \in \mathcal{C}_{\text{all}}(P(\nu, \mu), G)$, reaches a safe configuration for A.*

Finally, we define *the uniformly random scheduler*, which has been considered in most previous studies in the PP model [2], [9], [10], [11], [17], [18], [19], [22], [23], [24]. Under this scheduler, exactly one ordered pair $(u, v) \in E_G$ is selected to interact uniformly at random from all interactable pairs. This scheduler is required to evaluate the *time complexities* of protocols because global fairness only guarantees that an execution makes progress *eventually*. Formally, the uniformly random scheduler is defined as a sequence of interactions $\mathbf{\Gamma} = \Gamma_0, \Gamma_1, \ldots$, where each Γ_t is a random variable such that $\Pr(\Gamma_t = (u, v)) = 1/|E_G|$ for any $t \geq 0$ and any $(u, v) \in E_G$. Given a population G, a protocol $P(\nu, \mu)$, and an initial configuration $C_0 \in \mathcal{C}_{\text{all}}(P(\nu, \mu), G)$, the execution under the uniformly random scheduler is defined as $\Xi_{P(\nu,\mu)}(G, C_0, \mathbf{\Gamma}) = C_0, C_1, \ldots$ such that $C_t \overset{P(\nu,\mu),\Gamma_t}{\rightarrow} C_{t+1}$ for all $t \geq 0$. When we assume this scheduler, we can evaluate time complexities of a population protocol, e.g., the expected number of steps required to reach a safe configuration. We have the following observation.

Observation 1. *A protocol $P(\nu, \mu)$ is self-stabilizing for a problem A if and only if $\Xi_{P(\nu,\mu)}(G, C_0, \mathbf{\Gamma})$ reaches a safe configuration for A with probability 1 for any configuration $C_0 \in \mathcal{C}_{\text{all}}(P(\nu, \mu), G)$.*

Proof. Remember that we do not allow a protocol to have an infinite number of states. According to a previous study [2], we say that a set $\mathcal{C}$ of configurations is *final* if $\mathcal{C}$ is closed, and all configurations in $\mathcal{C}$ are reachable from each other. We also say that a configuration C is *final* if it belongs to a final set. Every protocol has at least one final configuration and every globally-fair execution eventually reaches a final configuration. Therefore, protocol P is self-stabilizing if and only if all final configurations are safe. Thus, it suffices to show that execution $\Xi = \Xi_{P(\nu,\mu)}(G, C_0, \mathbf{\Gamma})$ reaches a safe configuration for A with probability 1 for any $C_0 \in \mathcal{C}_{\text{all}}(P(\nu, \mu), G)$ if and only if all final configurations of $P(\nu, \mu)$ are safe for A. The sufficiency holds because Ξ reaches a final configuration with probability 1 regardless of C_0. We prove the necessary condition below. Suppose that there is a final configuration C that is not safe. By definition, C belongs to a final set $\mathcal{C}$. Since C is reachable from all configurations in $\mathcal{C}$, no configuration in $\mathcal{C}$ is safe. Since $\mathcal{C}$ is closed, Ξ will never reach a safe configuration if $C_0 = C$. □

3 Random Walk in Population Protocols

In this paper, we give two self-stabilizing protocols P_{rank} and P_{neigh}. Both protocols use n tokens that make the random walk, where n is the number of agents in the population. Specifically, all agents in the population always have exactly one token, and two agents swap their tokens whenever they have an interaction. In this section, we prove several lemmas about the movements of the tokens to analyze the expected number of steps until an execution of P_{rank} or P_{neigh} reaches a safe configuration. Very recently, Alistarh, Gelashvili, and Rybicki independently made a similar token-based analysis for population protocols and presented it in their pre-print [12].

3.1 Preliminaries

Fix a population $G = (V_G, E_G)$ and consider the execution $\Xi = C_0, C_1, \ldots$ of P_{rank} or P_{neigh}[2] under the uniformly random scheduler starting from an arbitrary configuration C_0. Let $\mathbf{\Gamma} = \Gamma_0, \Gamma_1, \ldots = (u_0, v_0), (u_1, v_1), \ldots$, i.e., we denote the ith interaction under the uniformly random scheduler $\mathbf{\Gamma}$ by (u_i, v_i). For each $w \in V_G$, we define token $t_w : \mathbb{N}_{\ge 0} \to V_G$ as follows:

- $t_w(0) = w$,
- $t_w(i) = \begin{cases} u_{i-1} & \text{if } t_w(i-1) = v_{i-1} \\ v_{i-1} & \text{if } t_w(i-1) = u_{i-1} \\ t_w(i-1) & \text{otherwise} \end{cases}$

for each $i > 0$.

We say that token t_v visits u in the ith step if $t_v(i) = u$. We also say that two tokens t_u and t_v meet in the ith step if $\Gamma_i = (t_u(i), t_v(i))$ or $\Gamma_i = (t_v(i), t_u(i))$ holds. In the rest of this section, we denote the number of agents and the number of interactable pairs by n and m, respectively, i.e., $n = |V_G|$ and $m = |E_{\overline{G}}|$. The diameter of $\overline{G}$ is denoted by d.

We introduce here two kinds of *hitting time*, $\mathbf{H}_G^{\mathcal{P}}$ and $\mathbf{H}_G^{\mathcal{S}}$. The former is the one about the random walks of the tokens defined above on population G, while the latter is the one about the (standard) simple random walk on an undirected graph $\overline{G}$. Define $\mathbf{H}_G^{\mathcal{P}}(u,v) = \mathbf{E}[\min\{t \ge 1 \mid t_u(t) = v\}]$ for any $u, v \in V_G$. Intuitively, $\mathbf{H}_G^{\mathcal{P}}(u,v)$ is the expected number of steps until token t_u reaches an agent v for the first time. To define $\mathbf{H}_G^{\mathcal{S}}$, we introduce a (discrete time) Markov Chain $\mathbf{S} = \{S(t) \in V_G \mid t = 0, 1, \ldots\}$, which corresponds to the simple random walk on graph $\overline{G}$. For any $t > 1$ and $u, v \in V_G$, the probability $\Pr(S(t) = v \mid S(t-1) = u)$ is independent of t and denoted by $P_{\mathbf{S}}(u,v)$. Probability $P_{\mathbf{S}}(u,v)$ is given as follows:

$$P_{\mathbf{S}}(u,v) = \begin{cases} 1/\delta_G(u) & \text{if } \{u,v\} \in E_{\overline{G}} \\ 0 & \text{otherwise.} \end{cases}$$

Define $\mathbf{H}_G^{\mathcal{S}}(u,v) = \mathbf{E}[\min\{t \ge 1 \mid S(t) = v\} \mid S(0) = u]$. Finally, we define $\mathbf{H}_G^{\mathcal{P}} = \max_{u,v \in V} \mathbf{H}_G^{\mathcal{P}}(u,v)$ and $\mathbf{H}_G^{\mathcal{S}} = \max_{u,v \in V} \mathbf{H}_G^{\mathcal{S}}(u,v)$.

We are also interested in how fast each token t_u moves on population G. To evaluate this, for any $k \ge 1$, we define $s_u(k)$ as the expected number of steps until t_u moves k times.

3.2 Lemmas

The main claims of this section are Lemmas 2 and 3, which is crucial to evaluate the time complexities of the proposed protocol in Sections 4 and 5. When t_v is located at vertex $u \in V_G$, it requires $m/\delta_G(u)$ steps in expectation to make one move (i.e., to move to one of the neighbors of u). Thus, the expected number of steps to make one move of t_v heavily depends on the degree of the agent at which the token is located, while the degree may range from $\Theta(1)$ to $\Theta(m)$. Hence, it is not trivial how $\mathbf{H}_G^{\mathcal{P}}$ and $\mathbf{H}_G^{\mathcal{S}}$ are related or how fast each token moves on the population. Lemmas 2 and 3 give good bounds for them. The former argues $\mathbf{H}_G^{\mathcal{P}} = O(n\mathbf{H}_G^{\mathcal{S}})$, while the latter argues that the expected number of steps to make k moves is well-bounded by $nk/2$ plus an additive overhead of $O(n\mathbf{H}_G^{\mathcal{S}})$ steps. Interestingly, the base value $nk/2$ corresponds to the average probability that an agent has an interaction at each step, $(1/n)\sum_{w \in V_G} \delta_G(w)/m = 2/n$. These two lemmas are very useful because we can apply a huge number of previously proven theorems about the random walk on a graph to the field of population protocols.

To prove Lemmas 2 and 3, we first introduce Lemma 1.

Lemma 1. *For any integer $k > 0$, there exists an agent $u \in V_G$ such that $s_u(k) \le nk/2$.*

Proof. If a token visits agent $w \in V_G$, then it requires $m/\delta_G(w)$ steps in expectation to leave w, i.e., move to another agent from w. Thus, we assign each agent $x \in V_G$*weight* $W(x) \overset{\mathrm{def}}{=} m/\delta_G(x)$. Then, with Markov chain $\mathbf{S} = \{S(t) \in V_G \mid t = 0, 1, \ldots\}$ defined in Section 3.1, we have $s_w(k) = \mathbf{E}\left[\sum_{t=0}^{k-1} W(S(t)) \middle| S(0) = w\right]$ for any $w \in V_G$. Note that $\pi_{\mathbf{S}} = (\pi_{\mathbf{S}}(w))_{w \in V_G}$ is a (unique) stationary distribution where $\pi_{\mathbf{S}}(w) = \delta_G(w)/2m$ for any $w \in V_G$, because we have $\pi_{\mathbf{S}} P_{\mathbf{S}} = \pi_{\mathbf{S}}$. Assume that the initial state $S(0)$ is now set according to this stationary distribution, i.e., $\Pr(S(0) = w) = \pi_{\mathbf{S}}(w) = \delta_G(w)/2m$ for any $w \in V_G$. Since $\pi_{\mathbf{S}}$ is a stationary distribution, we always have the same distribution thereafter, i.e., we have $\Pr(S(t) = w) = \pi_{\mathbf{Z}}(w)$ for any $t = 0, 1, \ldots$ and $w \in V_G$. Therefore, under this assumption, we have

$$\begin{aligned} \mathbf{E}\left[\sum_{t=0}^{k-1} W(S(t))\right] &= k \sum_{w \in V_G} \pi_{\mathbf{Z}}(w) W(w) \\ &= k \sum_{w \in V_G} \frac{\delta_G(w)}{2m} \cdot \frac{m}{\delta_G(w)} \\ &= \frac{nk}{2}. \end{aligned}$$

We also have

$$\begin{aligned} \mathbf{E}\left[\sum_{t=0}^{k-1} W(S(t))\right] &= \sum_{v \in V_G} \pi_{\mathbf{S}}(v) \cdot \mathbf{E}\left[\sum_{t=0}^{k-1} W(S(t)) \middle| S(0) = v\right] \\ &= \sum_{v \in V_G} \pi_{\mathbf{S}}(v) s_v(k). \end{aligned}$$

Thus, we have obtained $\sum_{w \in V_G} \pi_{\mathbf{S}}(w) s_w(k) = nk/2$. The lemma follows from $\sum_{w \in V_G} \pi_{\mathbf{S}}(w) = 1$. □

Lemma 2. $\mathbf{H}_G^{\mathcal{P}} = O(n\mathbf{H}_G^{\mathcal{S}})$.

Proof. By Lemma 1, there exists an agent $w \in V_G$ such that $s_w(6\mathbf{H}_G^{\mathcal{S}}) \le 3n\mathbf{H}_G^{\mathcal{S}}$. Let w be such an agent and u, v be any two agents in V_G. Let X denote the number of moves of t_w moves until it reaches u and thereafter v. Let Y denote the number of steps until t_w reaches u and thereafter v. By definition, it is immediate that $\mathbf{E}[X] = \mathbf{H}_G^{\mathcal{S}}(w,u) + \mathbf{H}_G^{\mathcal{S}}(u,v) \le 2\mathbf{H}_G^{\mathcal{S}}$. By Markov's inequality, we have $\Pr(X \ge 6\mathbf{H}_G^{\mathcal{S}}) \le 1/3$. Since $s_w(6\mathbf{H}_G^{\mathcal{S}}) \le 3n\mathbf{H}_G^{\mathcal{S}}$, the probability that t_w moves less than $6\mathbf{H}_G^{\mathcal{S}}$ times within $9n\mathbf{H}_G^{\mathcal{S}}$ steps is at most $1/3$ by Markov's inequality. Thus, by the union bound, we have observed that token t_w reaches u and

2. In this section, we do not care which protocol, P_{rank} or P_{neigh}, is executed because we only focus on the movement of tokens making the random walk.

thereafter v within $9n\mathbf{H}_G^{\mathcal{S}}$ steps with probability at least $1-(1/3+1/3)=1/3$, i.e., $\Pr(Y \le 9n\mathbf{H}_G^{\mathcal{S}}) \ge 1/3$. From the memoryless property of t_w's movement, this observation immediately yields that for any $u, v \in V_G$, token t_u reaches v within $9n\mathbf{H}_G^{\mathcal{S}}$ steps with probability at least $1/3$. Thus, for any $u, v \in V_G$ we have

$$\mathbf{H}_G^{\mathcal{P}}(u,v) \le 9n\mathbf{H}_G^{\mathcal{S}} + (1-1/3) \cdot \max_{u',v' \in V_G} \mathbf{H}_G^{\mathcal{P}}(u',v').$$

This yields that $\mathbf{H}_G^{\mathcal{P}} \le 9n\mathbf{H}_G^{\mathcal{S}} + (2/3) \cdot \mathbf{H}_G^{\mathcal{P}}$. Solving this inequality gives $\mathbf{H}_G^{\mathcal{P}} \le 27n\mathbf{H}_G^{\mathcal{S}} = O(n\mathbf{H}_G^{\mathcal{S}})$. □

By Lemma 2, we have the following two corollaries.

Corollary 1. *In execution Ξ, for any $v \in V_G$, all n tokens visit agent v within $O(n\mathbf{H}_G^{\mathcal{S}}\log n)$ steps in expectation.*

Proof. Let u be any agent in V_G. It immediately follows from Lemma 2 that token t_u visits agent v within $O(n\mathbf{H}_G^{\mathcal{S}})$ steps in expectation. Therefore, by Markov's inequality, t_u visits v within $O(n\mathbf{H}_G^{\mathcal{S}})$ steps with a sufficiently large hidden constant with probability at least $1/2$. Therefore, they meet within $2\log_2 n \cdot O(n\mathbf{H}_G^{\mathcal{S}}) = O(n\mathbf{H}_G^{\mathcal{S}}\log n)$ steps with probability at least $1-1/n^2$. By the union bound, all n tokens $(t_u)_{u \in V_G}$ visit v within $O(n\mathbf{H}_G^{\mathcal{S}}\log n)$ steps with probability $1-O(1/n)$, thus also in expectation. □

Corollary 2. *In execution Ξ, for any $v \in V_G$, token t_v visits all agents in V_G within $O(n\mathbf{H}_G^{\mathcal{S}}\log n)$ steps in expectation.*

Proof. Let u be an arbitrary agent in V_G. By Lemma 2 and Markov's inequality, t_v visits u with probability $1/2$ in every $O(n\mathbf{H}_G^{\mathcal{S}}\log n)$ steps with a sufficiently large hidden constant. Therefore, t_v visits u within $O(n\mathbf{H}_G^{\mathcal{S}}) \cdot (2\log_2 n) = O(n\mathbf{H}_G^{\mathcal{S}}\log n)$ steps with probability $1-(1/2)^{2\log_2 n} = 1-1/n^2$. The union bound for all $u \in V_G$ yields that t_v visits all agents within $O(n\mathbf{H}_G^{\mathcal{S}}\log n)$ steps with probability $1-O(1/n)$, thus also in expectation. □

Lemma 3. *For any integer $k \ge 1$ and $v \in V_G$, $s_v(k) = nk/2 + O(n\mathbf{H}_G^{\mathcal{S}})$ holds.*

Proof. By Lemma 1, there exists $w \in S_G$ such that $s_w(k) \le nk/2$. By Lemma 2, token t_v visits w within at most $O(n\mathbf{H}_G^{\mathcal{S}})$ steps in expectation. After visiting w, token t_v moves k times within at most $nk/2$ steps in expectation. In total, token t_v moves k times within at most $nk/2 + O(n\mathbf{H}_G^{\mathcal{S}})$ steps in expectation. Thus, we have $s_v(k) \le nk/2 + O(n\mathbf{H}_G^{\mathcal{S}})$.

Next, we analyze how many steps are required in expectation until all n tokens meet each other. (We say that two tokens t_u and t_v *meet* when an interaction happens such that t_u and t_v are swapped between two agents.) In our previous study [11], we proved that $O(mn^2 d\log n)$ steps are sufficient. In this paper, we improve this upper bound with the help of Lemma 3. Specifically, we obtain the following lemma here.[3] □

Lemma 4. *In execution Ξ, all the n tokens meet each other within $O(n\mathbf{H}_G^{\mathcal{S}}\log n)$ steps in expectation.*

3. Since it is well known that $\mathbf{H}_G^{\mathcal{S}} = O(md)$, Lemma 4 improves the upper bound $O(mn^2 d\log n)$ of [11] at least by a multiplicative factor of n.

Proof. Consider the following game:

Two particle x and y, initially located at different vertices in G, make the simple random walk on G. At each time round, x or y is selected by an adversary. The selected particle chooses one vertex uniformly at random from the neighbors of the current vertex and moves to that neighbor. The game ends when two particle meets, i.e., they are located at the same vertex.

Coppersmith *et al.* proved that this game ends within at most $2\mathbf{H}_G^{\mathcal{S}}$ rounds in expectation irrespective of the strategy of the adversary (Theorem 2 in [25]). Therefore, in an execution of P_{rank} and P_{neigh}, every two tokens t_u and t_v meet with probability at least $2/3$ each time they move $6\mathbf{H}_G^{\mathcal{S}}$ times in total, by Markov's inequality. Clearly, t_u and t_v moves at least $6\mathbf{H}_G^{\mathcal{S}}$ times in total when t_u moves $6\mathbf{H}_G^{\mathcal{S}}$ times, which occurs with probability $2/3$ in every $O(n\mathbf{H}_G^{\mathcal{S}})$ steps (with a sufficiently large hidden constant) by Lemma 3 and Markov's inequality. Thus, t_u and t_v meet within $(2\log_3 n) \cdot O(n\mathbf{H}_G^{\mathcal{S}}) = O(n\mathbf{H}_G^{\mathcal{S}}\log n)$ steps with probability at least $1-(1/3)^{2\log_3 n} = 1-1/n^2$. Therefore, by the union bound, all tokens meet each other within some $x = O(n\mathbf{H}_G^{\mathcal{S}}\log n)$ steps with a probability $1-\binom{n}{2}\cdot(1/n^2) \ge 1/2$.

Let T (resp. T') be the expected number of steps until all tokens meet each other by interactions $\mathbf{\Gamma} = \Gamma_0, \Gamma_1, \ldots$ (resp. $\mathbf{\Gamma}' = \Gamma_x, \Gamma_{x+1}, \ldots$). We have $T \le x + (1-1/2)T'$ from the above discussion. Since $T = T'$ holds, solving this inequality gives $T \le 2x = O(n\mathbf{H}_G^{\mathcal{S}}\log n)$. □

4 Leader Election and Ranking

The goal of this section is to provide a necessary and sufficient condition to solve **RK** and **LE** on knowledge ν, provided that μ gives no information, i.e., $\mu = \mathbb{N}_{\ge 1}$.

4.1 Necessary Knowledge

Lemma 5 ([3], [14], [18]). *Given knowledge ν and μ, there exists no self-stabilizing protocol that solves* **LE** *in arbitrary graphs if $\mathcal{G}_{n_1,*} \cup \mathcal{G}_{n_2,*} \subseteq \mathcal{G}_{\nu,\mu}$ for some two distinct $n_1, n_2 \in \mathbb{N}_{\ge 2}$.*

Proof. The lemma immediately follows from the fact that there exists no self-stabilizing protocol that solves **LE** in complete graphs of two different sizes, i.e., both in K_{n_1} and K_{n_2} for any two integers $n_1 > n_2 \ge 2$. As mentioned in Section 1, Sudo *et al.* [18] indicated how to prove this fact based on the proofs in two previous studies [3], [14]. □

4.2 Sufficient Knowledge

To give a sufficient condition, we provide a self-stabilizing protocol P_{rank} that solves the ranking problem (**RK**) in arbitrary graphs given the knowledge of the exact number of agents in a population. Specifically, this protocol assumes that the given knowledge ν satisfies $|\nu| = 1$. Protocol P_{rank} does not care about the number of interactable pairs, that is, $P_{\mathrm{rank}}(\nu, \mu)$ works even if μ does not give any knowledge (i.e., $\mu = \mathbb{N}_{\ge 1}$). Let n be the integer such that $\nu = \{n\}$. In the remainder of this section, we fix a population $G = (V_G, E_G) \in \mathcal{G}_{n,*}$, let $m = |E_{\overline{G}}|$, and let d be the diameter of G.

Algorithm 1. $P_{\mathrm{rank}}(\nu, \mu)$

```
    Assumption: $|\nu| = 1$. (Let $\nu = \{n\}$.)
    Variables:
      $\mathsf{id}_A, \mathsf{id}_T \in \{0, 1, \ldots, n-1\}$,
      $\mathsf{color}_A \in \{W, R, B\}$, $\mathsf{color}_T \in \{R, B\}$
    Output Function: $\mathsf{id}_A$
    Interaction between initiator $a_0$ and responder $a_1$:
 1: $(a_0.\mathsf{id}_T, a_0.\mathsf{color}_T) \leftrightarrow (a_1.\mathsf{id}_T, a_1.\mathsf{color}_T)$;
                                                    // Execute random walk
 2: if $a_0.\mathsf{id}_T = a_1.\mathsf{id}_T$ then
 3:   $a_1.\mathsf{id}_T \leftarrow a_1.\mathsf{id}_T + 1 \pmod n$)
 4: for all $i \in \{0, 1\}$ such that $a_i.\mathsf{id}_A = a_i.\mathsf{id}_T$ do
 5:   if $a_i.\mathsf{color}_A = W$ then
 6:     $a_i.\mathsf{color}_A \leftarrow a_i.\mathsf{color}_T$
 7:   else if $a_i.\mathsf{color}_A \neq a_i.\mathsf{color}_T$ then
 8:     $a_i.\mathsf{id}_A \leftarrow a_i.\mathsf{id}_A + 1 \pmod n$;
 9:     $a_i.\mathsf{color}_A \leftarrow W$;
10:   else
11:     $a_i.\mathsf{color}_A \leftarrow a_i.\mathsf{color}_T \leftarrow \begin{cases} B & \text{if } i = 0 \\ R & \text{if } i = 1 \end{cases}$;

      // Recolor $B$ or $R$ depending on the role, initiator or responder
```

If we focus only on complete graphs, the following simple algorithm [14] is sufficient to solve self-stabilizing ranking with the exact knowledge n of agents:

- Each agent v has only one variable $v.\mathsf{id} \in \{0, 1, \ldots, n-1\}$, and
- Each time two agents with the same id meet, one of them (the initiator) increases its id by one modulo n.

Since this algorithm assumes complete graphs, all pairs of agents in the population eventually have interactions. Therefore, as long as two agents have the same identifiers, they eventually meet and the collision of their identifiers is resolved. However, this algorithm does not work in arbitrary graphs, even if the exact number of agents is given. This is because some pair of agents may not be interactable in an arbitrary graph and they cannot resolve the conflicts of their identifiers by meeting each other.

It is worthwhile to mention that Angluin *et al.* [2] proves that for any population protocol P working on complete graphs, there exists a protocol that simulates P on arbitrary graphs. One may think that we can immediately obtain a self-stabilizing ranking protocol on arbitrary graphs using this transformer together with the above ranking protocol for complete graphs. However, we cannot apply the technique for our goal from the following two reasons. First, the technique cannot be applied to self-stabilizing protocols since it assumes that all the agents start with the common initial state, which violates the essential requirement of self-stabilization. Second, the technique targets the problems where all agents are required to output the same value, such as computing a function on the inputs of the agents. It cannot be applied to the problems that require the agents to output different values, such as **LE** and **RK**.

Protocol P_{rank} detects the conflicts between any (possibly non-interactable) two agents by traversing n tokens in a population where each agent always has exactly one token. This protocol is inspired by a self-stabilizing leader election protocol with *oracles* given by Beauquier *et al.* [4], where the agents traverse exactly one token in a population.

The pseudocode of P_{rank} is given in Algorithm 1. Our goal is to assign the agents the distinct labels $0, 1, \ldots, n-1$. Each agent v stores its label in a variable $v.\mathsf{id}_A \in \{0, 1, \ldots, n-1\}$ and outputs it as it is. To detect and resolve the conflicts of the labels in arbitrary graphs, each agent maintains four other variables $\mathsf{id}_T \in \{0, 1, \ldots, n-1\}$, $\mathsf{color}_A \in \{W, R, B\}$, and $\mathsf{color}_T \in \{R, B\}$. Each agent v has one color, white (W), red (R), or blue (B), while v's token has one color, red (R) or blue (B), maintained by variables $v.\mathsf{id}_A$ and $v.\mathsf{id}_T$, respectively.

The tokens always make *the random walk*: two agents swap their tokens whenever they interact (Line 1). If the two tokens have the same label, one of them increments its label by one modulo n (Lines 2–3). Since all tokens meet each other infinitely often by the random walk, they eventually have mutually distinct labels (id_T), after which they never change their labels. Thereafter, the conflicts of labels among the agents are resolved by using the tokens. Let x be any integer in $\{0, 1, \ldots, n-1\}$ and denote the token labeled x by T_x. Ideally, an agent labeled x always has the same color as that of T_x. Consider the case that an agent labeled x, say v, meets T_x, and v and T_x have different colors, blue and red. Then, v suspects that there is another agent labeled x, and v increases its label by one modulo n (Lines 7–8). The agent v, now labeled $x + 1 \pmod n$, changes its color to white (Line 9). The next time v meets $T_{x+1 \pmod n}$, it copies the color of the token to its color to synchronize a color with $T_{x+1 \pmod n}$ (Lines 5–6). Each time token T_x meets an agent labeled x with the same color, T_x and the agent change their common color randomly (Line 11). Specifically, if the agent is an initiator in this interaction, they choose B; Otherwise, they choose R. As a result, they keep their color with probability $1/2$ and flip their color with probability $1/2$. If there are two or more agents labeled x, this multiplicity is eventually detected because T_x makes a random walk forever changing its color randomly and eventually meets an agent labeled x with a different color. By repeating this procedure, the population eventually reaches a configuration where all the agents have distinct labels and the agent labeled x has the same color as that of T_x for all $x = 0, 1, \ldots, n-1$. No agent changes its label thereafter.

Note that this protocol works even if we do not use color W. We introduce this color to guarantee the fast stabilization time under the uniformly random scheduler. In the rest of this section, we prove the following theorem.

Theorem 1. *Given knowledge ν and μ, $P_{\mathrm{rank}}(\nu, \mu)$ is a self-stabilizing protocol that solves* **RK** *in arbitrary graphs if $\nu = \{n\}$ for some integer n, regardless of μ. Starting from any configuration C_0 on any population $G = (V_G, E_G) \in \mathcal{G}_{n,*}$, the execution of $P_{\mathrm{rank}}(\nu, \mu)$ under the uniformly random scheduler (i.e., $\Xi_{P_{\mathrm{rank}}(\nu,\mu)}(G, C_0, \mathbf{\Gamma})$) reaches a safe configuration within $O(n^2 \mathbf{H}_G^{\mathcal{S}} \log n)$ steps in expectation. Each agent uses $O(\log n)$ bits of memory space to execute $P_{\mathrm{rank}}(\nu, \mu)$.*

Recall that $\mathbf{H}_G^{\mathcal{S}}$ is the hitting time of the simple random walk on $\overline{G}$, which is formally defined in Section 3.1. We evaluate the convergence time of P_{rank} with $\mathbf{H}_G^{\mathcal{S}}$ in the above theorem in order to keep it a general result. It is well known that $\mathbf{H}_G^{\mathcal{S}} = O(md)$ holds for any simple, connected, and undirected graph $\overline{G}$, where d is the diameter of $\overline{G}$. Thus, the convergence time of P_{rank} is bounded by $O(mn^2 d \log n)$ steps. The hitting time is much smaller than the bound $O(md)$ for some specific class of the graphs, so the convergence time of P_{rank} is much smaller accordingly for such a class.

To prove Theorem 1, we define three sets $\mathcal{S}_{\mathrm{token}}$, $\mathcal{S}_{\mathrm{sync}}$, and $\mathcal{S}_{\mathrm{rank}}$ of configurations in $\mathcal{C}_{\mathrm{all}}(P_{\mathrm{rank}}(\nu,\mu), G)$ as follows.

- $\mathcal{S}_{\mathrm{token}}$: the set of all configurations in $\mathcal{C}_{\mathrm{all}}(P_{\mathrm{rank}}(\nu,\mu), G)$ where all tokens have distinct labels, i.e., $\forall u, v \in V_G : u.\mathsf{id}_T \neq v.\mathsf{id}_T$. In a configuration in $\mathcal{S}_{\mathrm{token}}$, there exists exactly one token labeled x in the population for each $x \in \{0,1,\dots,n-1\}$. We use the notation T_x to denote both the unique token labeled by x and the agent on which this token *currently* stays.
- $\mathcal{S}_{\mathrm{sync}}$: the set of all configurations in $\mathcal{S}_{\mathrm{token}}$ where proposition $Q_{\mathrm{token}}(x) \overset{\mathrm{def}}{\equiv} V_G(x) \neq \emptyset \Rightarrow (\exists u \in V_G(x) : u.\mathsf{color}_A = T_x.\mathsf{color}_T \vee u.\mathsf{color}_A = W)$ holds for any $x \in \{0,1,\dots,n-1\}$, where $V_G(x) \overset{\mathrm{def}}{=} \{v \in V \mid v.\mathsf{id}_A = x\}$.
- $\mathcal{S}_{\mathrm{rank}}$: the set of all the configurations in $\mathcal{S}_{\mathrm{sync}}$ where all the agents in V_G have distinct labels, i.e., $\forall u, v \in V_G : u.\mathsf{id}_A \neq v.\mathsf{id}_A$.

Lemma 6. *The set $\mathcal{S}_{\mathrm{token}}$ is closed for $P_{\mathrm{rank}}(\nu,\mu)$.*

Proof. A token changes its label only if it meets another token with the same label. Thus, no token changes its label in an execution starting from a configuration in $\mathcal{S}_{\mathrm{token}}$. □

Lemma 7. *Let $x \in \{0,1,\dots,n-1\}$. In an execution of $P_{\mathrm{rank}}(\nu,\mu)$ starting from a configuration in $\mathcal{S}_{\mathrm{token}}$, once $Q_{\mathrm{token}}(x)$ holds, it always holds thereafter.*

Proof. This lemma holds because (i) an agent must be white just after it changes its label from $x-1 \pmod n$ to x, (ii) a white agent labeled x changes its color only when token T_x visits it at an interaction, at which point this white agent gets the same color as that of T_x, (iii) an agent labeled x with the same color as that of T_x changes its color only when token T_x visits it at an interaction, at which this agent and T_x get the same new color. □

Lemma 8. *The set $\mathcal{S}_{\mathrm{sync}}$ is closed for $P_{\mathrm{rank}}(\nu,\mu)$.*

Proof. The lemma immediately follows from Lemma 7. □

Lemma 9. *Let $x \in \{0,1,\dots,n-1\}$. In an execution of $P_{\mathrm{rank}}(\nu,\mu)$ starting from a configuration in $\mathcal{S}_{\mathrm{sync}}$, once at least one agent is labeled x, the number of agents labeled x never becomes zero thereafter.*

Proof. The lemma holds in the same way as the proof of Lemma 7. □

Lemma 10. *The set $\mathcal{S}_{\mathrm{rank}}$ is closed for $P_{\mathrm{rank}}(\nu,\mu)$.*

Proof. The lemma immediately follows from Lemmas 8 and 9. □

The following lemma is useful to analyze the expected number of steps required to reach a configuration in $\mathcal{S}_{\mathrm{rank}}$ in an execution of $P_{\mathrm{rank}}(\nu,\mu)$.

Lemma 11. *Consider the following game with n players $p_0, p_1, \dots, p_{n-1}$. Each player always has one state in $\{0,1,\dots,n-1\}$. At each step, an arbitrary pair of players is selected and they check each other's states. If they have the same state, one player increases its state by one modulo n. Otherwise, their states do not change. Let ψ be any configuration (i.e., any combination of the states of all players) of this game. Then, there is at least one state $z \in \{0,1,\dots,n-1\}$ such that the transition from $z-1 \pmod n$ to z never occurs in any execution of this game starting from ψ.*

Proof. Let $\psi = (k_0, k_1, \dots, k_{n-1})$, where k_i represents the number of agents in state i in the configuration ψ. In this proof, we make every addition and subtraction in modulo n and omit the notation "$\pmod n$". For any $z \in \{0,1,\dots,n-1\}$, it is trivial that the transition from $z-1$ to z never occurs in any execution of this game starting from ψ if and only if z satisfies $\sum_{j=1}^{i} k_{z-j} \leq i$ for all $i \in \{1,2,\dots,n-1\}$. Burman *et al.* [20] proved that there is at least one integer z that satisfies the above condition from which the lemma immediately follows. For the completeness of this paper, we give here the proof of [20] (Lemma A.1.) with our notation. The goal is to show the existence of z that satisfies $\sum_{j=1}^{i} k_{z-j} \leq i$ for all $i \in \{1,2,\dots,n-1\}$. Let $S_i = \sum_{j=0}^{i}(k_j - 1)$. Note that $S_{n-1} = 0$ holds because $\sum_{j=0}^{n-1} k_j = n$. Choose $z \in \{1,2,\dots,n-1\}$ with the minimum S_{z-1} (i.e., $S_{z-1} \leq S_i$ for any i). Then, by the minimality of S_{z-1} and $S_{n-1} = 0$, we have

$$\forall i \leq z : \sum_{j=1}^{i} k_{z-j} = i + S_{z-1} - S_{z-i} \leq i,$$

$$\forall i > z : \sum_{j=1}^{i} k_{z-j} = i + S_{z-1} + S_{n-1} - S_{n-1-(i-z)} \leq i.$$

Thus, we have $\sum_{j=1}^{i} k_{z-j} \leq i$ for all $i \in \{1,2,\dots,n-1\}$. □

Lemma 12. *Starting from any configuration $C_0 \in \mathcal{C}_{\mathrm{all}}(P_{\mathrm{rank}}(\nu,\mu), G)$, an execution of $P_{\mathrm{rank}}(\nu,\mu)$ under the uniformly random scheduler (i.e., $\Xi_{P(\nu,\mu)}(G, C_0, \mathbf{\Gamma})$) reaches a configuration in $\mathcal{S}_{\mathrm{token}}$ within $O(n^2 \mathbf{H}_G^{\mathcal{S}} \log n)$ steps in expectation.*

Proof. By Lemma 11, there exists an integer $z \in \{0,1,\dots,n-1\}$ such that no *token* changes its label from $z-1 \pmod n$ to z. Then, the number of tokens labeled z becomes exactly one before or when all the tokens meet each other. Since the n tokens meet each other within $O(n\mathbf{H}_G^{\mathcal{S}} \log n)$ steps in expectation (Lemma 4), the number of tokens labeled z becomes exactly one within $O(n\mathbf{H}_G^{\mathcal{S}} \log n)$ steps in expectation. Thereafter, no token changes its label from z to $z+1 \pmod n$. Hence, the number of tokens labeled $z+1 \pmod n$ becomes one in the next $O(n\mathbf{H}_G^{\mathcal{S}} \log n)$ steps in the same way. Repeating this procedure, all the tokens have distinct labels within $O(n^2\mathbf{H}_G^{\mathcal{S}} \log n)$ steps in expectation. □

Lemma 13. *Starting from any configuration $C_0 \in \mathcal{S}_{\mathrm{token}}$, an execution of $P_{\mathrm{rank}}(\nu,\mu)$ under the uniformly random scheduler (i.e., $\Xi_{P(\nu,\mu)}(G, C_0, \mathbf{\Gamma})$) reaches a configuration in $\mathcal{S}_{\mathrm{sync}}$ within $O(n\mathbf{H}_G^{\mathcal{S}} \log n)$ steps in expectation.*

Proof. By Lemmas 6 and 7, it suffices to show that for each $x \in \{0,1,\dots,n-1\}$, $Q_{\mathrm{token}}(x)$ becomes true within $O(n H_G^{\mathcal{S}} \log n)$ steps in expectation in an execution of P_{rank}

(ν,μ) starting from C_0. Let S be the set of the agents labeled x in C_0. If $Q_{\mathrm{token}}(x) = \mathit{false}$ in C_0, we have $S \neq \emptyset$ and all agents in S have non-white color different from $T_x.\mathsf{color}_T$ in C_0. Then, $Q_{\mathrm{token}}(x)$ becomes true before or when T_x meets all of the agents in S. By Corollary 2, T_x visits (i.e., meets) all agents within $O(n\mathbf{H}_G^{\mathcal{S}}\log n)$ steps in expectation, from which the lemma follows. □

Lemma 14. *Let C_0 be a configuration in $\mathcal{S}_{\mathrm{sync}}$. Suppose that there are two distinct agents $u, v \in V_G$ such that they are labeled the same level $x \in \{0,1,\ldots,n-1\}$ and have non-white colors in C_0. Then, an execution of $P_{\mathrm{rank}}(\nu,\mu)$ starting from C_0 under the uniformly random scheduler (i.e., $\Xi_{P(\nu,\mu)}(G, C_0, \mathbf{\Gamma})$) reaches a configuration where u or v is labeled $x+1 \pmod n$ within $O(n\mathbf{H}_G^{\mathcal{S}}\log n)$ steps with probability $1-n^{-2}$.*

Proof. Let $\Xi_{P(\nu,\mu)}(G, C_0, \mathbf{\Gamma}) = C_0, C_1, C_2, \ldots$. Define two sequences of random variables $(\mathcal{T}_{u,i})_{i=1,2,\ldots}$ and $(\mathcal{T}_{v,i})_{i=1,2,\ldots}$ as follows:

- $\mathcal{T}_{u,1}$ is the minimum integer $t \geq 1$ such that $u = T_x$ holds in C_t,
- $\mathcal{T}_{v,i}$ is the minimum integer $t > \mathcal{T}_{u,i}$ such that $v = T_x$ holds in C_t for any $i = 1,2,\ldots$, and
- $\mathcal{T}_{u,i+1}$ is the minimum integer $t > \mathcal{T}_{v,i}$ such that $u = T_x$ holds in C_t for any $i = 1,2,\ldots$.

Note that $\mathcal{T}_{u,1} < \mathcal{T}_{v,1} < \mathcal{T}_{u,2} < \mathcal{T}_{v,2} < \ldots$ holds by definition. Let τ be the minimum integer such that u or v is labeled $x+1 \pmod n$ in C_τ. In the rest of this proof, we will show:

- $\tau \leq \mathcal{T}_{v,4\lceil \log_2 n\rceil}$ holds with probability $1-o(n^{-2})$, and
- $\mathcal{T}_{v,4\lceil \log_2 n\rceil} < 64\mathbf{H}_G^{\mathcal{P}}\lceil \log_2 n\rceil$ holds with probability $1-o(n^{-2})$,

from which the lemma immediately follows because $\mathbf{H}_G^{\mathcal{P}} = O(n\mathbf{H}_G^{\mathcal{S}})$ by Lemma 2.

Let i be any positive integer. At step $\mathcal{T}_{u,i}-1$, in which $C_{\mathcal{T}_{u,i}-1}$ changes to $C_{\mathcal{T}_{u,i}}$, token T_x visits agent u. At this time, unless $\tau \leq \mathcal{T}_{u,i}$ holds, T_x and u updates their color together to red with probability $1/2$ and to blue with probability $1/2$. Therefore, at step $\mathcal{T}_{v,i}-1$, T_x finds that v has a different color from T_x's color and increases v's color to $x+1 \pmod n$ with probability $1/2$. Note that T_x may visit an agent labeled x in the steps $\mathcal{T}_{u,i}, \mathcal{T}_{u,i}+1, \ldots, \mathcal{T}_{v,i}-2$, however, this does not impair the above discussion. Thus, we have $\Pr(\tau \leq \mathcal{T}_{v,i} \mid \tau > \mathcal{T}_{u,i}) \geq 1/2$ for any $i = 1,2,\ldots$. This yields that $\Pr(\tau \leq \mathcal{T}_{v,4\lceil \log_2 n\rceil}) \geq 1-(1/2)^{4\log_2 n} = 1-o(n^{-2})$.

By Markov's inequality, for each $i = 1,2,\ldots$, we have $\Pr(\mathcal{T}_{v,i} - \mathcal{T}_{u,i} \geq 2\mathbf{H}_G^{\mathcal{P}}) \leq 1/2$ and $\Pr(\mathcal{T}_{u,i+1} - \mathcal{T}_{v,i} \geq 2\mathbf{H}_G^{\mathcal{P}}) \leq 1/2$. Thus, by the memoryless property of T_x's movement, we have

$$\Pr(\mathcal{T}_{v,4\lceil \log_2 n\rceil} < 64\mathbf{H}_G^{\mathcal{P}}\lceil \log_2 n\rceil) \geq \Pr(X \geq 8\lceil \log_2 n\rceil),$$

where X is a binomial random variable with $32\lceil \log_2 n\rceil$ trials and success probability $1/2$. Chernoff bound gives

$$\Pr(X \geq 8\lceil \log_2 n\rceil) = \Pr(X \geq \mathbf{E}[X]/2) \geq 1-n^{-2.8}.$$

Thus, $\mathcal{T}_{v,4\lceil \log_2 n\rceil} < 64\mathbf{H}_G^{\mathcal{P}}\lceil \log_2 n\rceil$ holds with probability at least $1-n^{-2.8} = 1-o(n^{-2})$. □

Lemma 15. *Starting from any configuration $C_0 \in \mathcal{S}_{\mathrm{sync}}$, an execution of $P_{\mathrm{rank}}(\nu,\mu)$ under the uniformly random scheduler (i.e., $\Xi_{P(\nu,\mu)}(G, C_0, \mathbf{\Gamma})$) reaches a configuration in $\mathcal{S}_{\mathrm{rank}}$ within $O(n^2\mathbf{H}_G^{\mathcal{S}}\log n)$ steps in expectation.*

Proof. By Lemmas 9 and 11, there exists an integer $z \in \{0,1,\ldots,n-1\}$ such that no *agent* changes its label from $z-1 \pmod n$ to z. Therefore, at least one agent is labeled z in C_0. All such agents get non-white color, i.e., blue or red, or get a new label $z+1 \pmod n$ before or when T_z meets all agents, which requires only $O(n\mathbf{H}_G^{\mathcal{S}}\log n)$ steps in expectation (See Corollary 2). By Lemma 14 and the union bound, the number of agents labeled z becomes one within $O(n\mathbf{H}_G^{\mathcal{S}}\log n)$ steps with probability $1-\binom{n}{2}\cdot n^{-2} \geq 1/2$ and thus also in expectation. After that, no agent changes its label from z to $z+1 \pmod n$. Thus, the number of agents labeled $z+1 \pmod n$ becomes one in the next $O(n\mathbf{H}_G^{\mathcal{S}}\log n)$ steps in expectation for the same reason. Repeating this procedure, all agents get mutually distinct labels (i.e., id_A) within $O(n^2\mathbf{H}_G^{\mathcal{S}}\log n)$ steps in expectation. □

Proof of Theorem 1 By Lemmas 12, 13, and 15, $\Xi_{P_{\mathrm{rank}}(\nu,\mu)}(G, C_0, \mathbf{\Gamma})$ reaches a configuration in $\mathcal{S}_{\mathrm{rank}}$ within $O(n^2\mathbf{H}_G^{\mathcal{S}}\log n)$ steps in expectation. By Lemma 10, every configuration in $\mathcal{S}_{\mathrm{rank}}$ is a safe configuration for the ranking problem. □

Theorem 2. *Let ν be any subset of $\mathbb{N}_{\geq 2}$ and let $\mu = \mathbb{N}_{\geq 1}$. Given knowledge ν and μ $(= \mathbb{N}_{\geq 1})$, there exists a self-stabilizing protocol that solves* **LE** *and* **RK** *in arbitrary graphs if and only if the agents know the exact number of agents i.e., $\mathcal{G}_{\nu,\mu} = \mathcal{G}_{n,*}$ for some $n \in \mathbb{N}_{\geq 2}$.*

Proof. The theorem immediately follows from Lemma 5, Theorem 1, and the fact that **LE** $\preceq$ **RK**. □

5 Degree Recognition and Neighbor Recognition

Our goal is to prove the negative and positive propositions for **DR** and **NR** introduced in Section 1.

5.1 Necessary Knowledge

Lemma 16. *Let ν and μ be any sets such that $\nu \subseteq \mathbb{N}_{\geq 2}$ and $\mu \subseteq \mathbb{N}_{\geq 1}$. There exists no self-stabilizing protocol that solves* **DR** *in all graphs in $\mathcal{G}_{\nu,\mu}$ if $\mathcal{G}_{n,m_1} \cup \mathcal{G}_{n,m_2} \subseteq \mathcal{G}_{\nu,\mu}$ holds for some $n \in \mathbb{N}_{\geq 2}$ and some distinct $m_1, m_2 \in \mathbb{N}_{\geq 1}$ such that $\mathcal{G}_{n,m_1} \neq \emptyset$ and $\mathcal{G}_{n,m_2} \neq \emptyset$.*

Proof. Assume $m_1 < m_2$ without loss of generality. By definition, there must exist two graphs $G' = (V_{G'}, E_{G'}) \in \mathcal{G}_{n,m_1}$ and $G'' = (V_{G''}, E_{G''}) \in \mathcal{G}_{n,m_2}$ such that $V_{G'} = V_{G''}$ and $E_{G'} \subset E_{G''}$. Then, there exists at least one agent $v \in V_{G''}$ such that its degree differs in G' and G''. Let δ' and δ'' be the degrees of v in G' and G'', respectively. Assume for contradiction that there is a self-stabilizing protocol $P(\nu,\mu)$ that solves **DR** both in G' and G''. By definition, there must be at least one safe configuration S of protocol $P(\nu,\mu)$ on G'' for **DR**. In every execution of $P(\nu,\mu)$ starting from S on G'', agent v must always output δ'' as its degree. The configuration S can also be a configuration on G' because $V_{G'} = V_{G''}$. Since $P(\nu,\mu)$ is self-stabilizing in G', there must be a finite sequence of interactions $\gamma_0, \gamma_1, \ldots, \gamma_t$

of G' that put configuration S to a configuration where v outputs δ' as its degree. Since $E_{G'} \subset E_{G''}$, $\gamma_0, \gamma_1, \ldots, \gamma_t$ is also a sequence of interactions in G''. This implies that this sequence changes the output of v from δ'' to δ' starting from a *safe* configuration, a contradiction. □

5.2 Sufficient Knowledge

To prove the positive proposition for **DR** and **NR** described in Section 1, we give a self-stabilizing protocol P_{neigh}, which solves **NR** in arbitrary graphs given the knowledge of the exact number of agents and the exact number of interactable pairs, i.e., given knowledge ν and μ such that $|\nu| = |\mu| = 1$. In the rest of this section, let n and m be the integers such that $\nu = \{n\}$ and $\mu = \{m\}$, respectively. We fix a population $G = (V_G, E_G) \in \mathcal{G}_{n,m}$ and let d be the diameter of G.

The pseudocode of P_{neigh} is given in Algorithm 2. Our goal is to let the agents recognize the set of their neighbors. Each agent v stores its label in a variable $v.\mathsf{id}_A \in \{0, 1, \ldots, n-1\}$ and the set of the labels assigned to its neighbors in a variable $\mathsf{neighbors} \in 2^{\{0,1,\ldots,n-1\}}$. Each agent v outputs $(v.\mathsf{id}_A, v.\mathsf{neighbors})$.

Algorithm 2. $P_{\text{neigh}}(\nu, \mu)$

```
  Assumption:
    $|\nu| = 1$ and $|\mu| = 1$. (Let $\nu = \{n\}$ and $\mu = \{m\}$.)
  Variables:
    $\mathsf{id}_A, \mathsf{id}_T \in \{0, 1, \ldots, n-1\}$ // Updated by $P_{\text{rank}}$
    $\mathsf{degree}_T \in \{1, \ldots, n\}$, $\mathsf{sum} \in \{0, 1, \ldots, 2m+1\}$
    $\mathsf{reset} \in \{0, 1, \ldots, U_R\}$
    $\mathsf{neighbors}, \mathsf{counted} \in 2^{\{0,1,\ldots,n-1\}}$
  Output Function: $(\mathsf{id}_A, \mathsf{neighbors})$
  Interaction between initiator $a_0$ and responder $a_1$
1:  Execute $P_{\text{rank}}$;
2:  $(a_0.\mathsf{degree}_T, a_0.\mathsf{reset}) \leftrightarrow (a_1.\mathsf{degree}_T, a_1.\mathsf{reset})$;
          // tokens also carry $\mathsf{degree}_T$ and $\mathsf{reset}$.
3:  for all $i \in \{0, 1\}$ do
4:      $a_i.\mathsf{neighbors} \leftarrow a_i.\mathsf{neighbors} \cup \{a_{1-i}.\mathsf{id}_A\}$;
5:      if $a_i.\mathsf{id}_A = a_i.\mathsf{id}_T$ then
6:          $a_i.\mathsf{degree}_T \leftarrow |a_i.\mathsf{neighbors}|$
7:      if $a_i.\mathsf{id}_T \notin a_i.\mathsf{counted}$ then
8:          $a_i.\mathsf{sum} \leftarrow \min(2m+1, a_i.\mathsf{sum} + a_i.\mathsf{degree}_T)$;
9:          $a_i.\mathsf{counted} \leftarrow a_i.\mathsf{counted} \cup \{a_i.\mathsf{id}_T\}$;
10:     if $a_i.\mathsf{sum} = 2m+1$ then
11:         $a_i.\mathsf{reset} \leftarrow U_R$
12:     if $a_i.\mathsf{counted} = \{0, 1, \ldots, n-1\}$ then
13:         $(a_i.\mathsf{sum}, a_i.\mathsf{counted}) \leftarrow (0, \emptyset)$
14:     if $a_i.\mathsf{reset} > 0$ then
15:         $a_0.\mathsf{neighbors} \leftarrow a_1.\mathsf{neighbors} \leftarrow \emptyset$;
16:         $a_i.\mathsf{reset} \leftarrow a_i.\mathsf{reset} - 1$;
```

We use P_{rank} as a sub-algorithm to assign the agents the distinct labels $0, 1, \ldots, n-1$ and to let the n tokens make the random walk. Specifically, we first execute P_{rank} whenever two agents have an interaction (Line 1). Note that we do not update the variables used in P_{rank} in the other lines (Lines 2–16). Therefore, by Theorem 1, an execution of P_{neigh} starting from any configuration reaches a configuration in $\mathcal{S}_{\text{rank}}$ within $O(n^2 \mathbf{H}_G^{\mathcal{S}} \log n)$ steps in expectation. Thus, we only need to consider an execution after reaching a configuration in $\mathcal{S}_{\text{rank}}$. Then, we can assume that the population always has exactly one agent labeled x and exactly one token labeled x for each $x = \{0, 1, \ldots, n-1\}$. We denote this agent and token by A_x and T_x, respectively.

The agents compute their $\mathsf{neighbors}$ in a simple way: each time two agents u and v have an interaction, u adds $v.\mathsf{id}_A$ to $u.\mathsf{neighbors}$ and v adds $u.\mathsf{id}_A$ to $v.\mathsf{neighbors}$ (Line 4). However, this simple way to compute $\mathsf{neighbors}$ is not sufficient to design a self-stabilizing protocol because we consider an arbitrary initial configuration. Specifically, in an initial configuration, $v.\mathsf{neighbors}$ may include $u.\mathsf{id}_A$ for some $u \notin N_G(v)$. We call such $u.\mathsf{id}_A$ a *fake label*. To compute $v.\mathsf{neighbors}$ correctly, in addition to the above simple mechanism, it suffices to detect the existence of a fake label and reset the $\mathsf{neighbors}$ of all agents to the empty set if a fake label is detected.

Using the knowledge $\mu = \{m\}$, we can detect fake labels with the following strategy. Each token T_x carries $|A_x.\mathsf{neighbors}|$ in a variable $\mathsf{degree}_T \in \{1, \ldots, n\}$ (Line 2). Whenever T_x meets A_x, the value of $T_x.\mathsf{degree}_T$ is updated by the current value of $|A_x.\mathsf{neighbors}|$ (Line 6). Each agent always attempts to estimate $\sum_{v \in V_G} |v.\mathsf{neighbors}|$ using variables $\mathsf{sum} \in \{0, 1, \ldots, 2m+1\}$, and $\mathsf{counted} \in 2^{\{0,1,\ldots,n-1\}}$. Each time $A_x.\mathsf{counted}$ becomes *full*, i.e., $A_x.\mathsf{counted} = \{0, 1, \ldots, n-1\}$ holds, A_x resets its sum and $\mathsf{counted}$ to 0 and $\emptyset$, respectively (Lines 12–13). Whenever agent A_x meets T_y such that $y \notin A_x.\mathsf{counted}$, A_x executes $A_x.\mathsf{sum} \leftarrow \min(2m+1, A_x.\mathsf{sum} + T_y.\mathsf{degree}_T)$ and adds y to $A_x.\mathsf{counted}$ (Lines 7–9). We expect $A_x.\mathsf{sum} = \sum_{v \in V_G} |v.\mathsf{neighbors}|$ when A_x meets all of $T_0, T_1, \ldots, T_{n-1}$. If $A_x.\mathsf{sum}$ reaches $2m+1$, A_x concludes that at least one agent has a fake label, i.e., $u.\mathsf{neighbors} \not\subseteq \{w.\mathsf{id}_A \mid w \in N_G(u)\}$ for some $u \in V_G$.

When the existence of a fake label is detected, we reset the $\mathsf{neighbors}$s of all agents using a variable $\mathsf{reset} \in \{0, 1, \ldots, U_R\}$, where U_R is a design parameter of P_{neigh} such that $U_R \geq 4\mathbf{H}_G^{\mathcal{S}} \log_2 n$. We will explain how to assign U_R such a value in the end of this section. When $v.\mathsf{sum} = 2m+1$ holds, v emits an resetting signal by setting variable $v.\mathsf{reset}$ to U_R (Lines 10-11). Like variable degree_T, token T_x carries variable reset each time it moves from agent to agent (Line 2). Whenever $T_x.\mathsf{reset} > 0$ holds, T_x resets $T_x.\mathsf{neighbors}$ (and $u.\mathsf{neighbors}$, where u is the agent that T_x interacts with) to the empty set (Line 15). Each time T_x moves, $T_x.\mathsf{reset}$ decreases by one (Line 16), thus $T_x.\mathsf{reset}$ eventually becomes zero, at which the resetting signal becomes disabled.

By this mechanism, even if some agent has fake labels at the beginning of an execution, the population eventually reaches a configuration where no agent has fake labels. Thereafter, for any $x \in \{0, 1, \ldots, n-1\}$, T_x eventually meets A_x, after which $T_x.\mathsf{degree}_T \leq |N_G(A_x)|$ always holds. Therefore, by the periodical resets of variables sum and $\mathsf{counted}$ at Lines 12–13, the population eventually reaches a configuration from which no agent newly emits the error signal. Thereafter, the population will soon reach a configuration that satisfies $v.\mathsf{neighbors} = \{u.\mathsf{id}_A \mid u \in N_G(v)\}$ for all $v \in V_G$ by the above simple computation of $\mathsf{neighbors}$ (Line 4). Once it reaches such a configuration, no agent changes its $\mathsf{neighbors}$.

Define $\mathcal{S}_{\text{noFake}}$ as the set of all configurations in $\mathcal{S}_{\text{rank}}$ where no agent has a fake label in its variable $\mathsf{neighbors}$.

Lemma 17. *Starting from any configuration $C_0 \in \mathcal{C}_{\text{all}}(P_{\text{neigh}}(\nu, \mu), G)$, an execution of $P_{\text{neigh}}(\nu, \mu)$ under the uniformly* *random scheduler (i.e., $\Xi_{P(\nu,\mu)}(G, C_0, \mathbf{\Gamma})$) reaches a configuration in $\mathcal{S}_{\mathrm{noFake}}$ within $O((n^2\mathbf{H}_G^{\mathcal{S}} + nU_R)\log n)$ steps in expectation.*

Proof. By Theorem 1, $\Xi = \Xi_{P(\nu,\mu)}(G, C_0, \mathbf{\Gamma})$ reaches a configuration in $\mathcal{S}_{\mathrm{rank}}$ within $O(n^2\mathbf{H}_G^{\mathcal{S}}\log n)$ steps in expectation. Thus, we assume $C_0 \in \mathcal{S}_{\mathrm{rank}}$ without loss of generality. We also assume $C_0 \notin \mathcal{S}_{\mathrm{noFake}}$ because otherwise we need not discuss anything.

First, we will prove that some agent emits a new resetting signal, i.e., changes its reset to U_R at Line 11, within $O(n^2\mathbf{H}_G^{\mathcal{S}}\log n)$ steps in expectation. In this paragraph, we often omit the phrase "unless some agent emits a new resetting signal" for simplicity. Token T_x decreases T_x.reset by one each time it moves as long as $T_x.\mathsf{reset} > 0$. Hence, by Lemma 3, Markov's inequality, and the union bound, Ξ reaches a configuration where $T_x.\mathsf{reset} = 0$ for all $x = 0, 1, \ldots, n-1$ within $O((nU_R + n\mathbf{H}_G^{\mathcal{S}})\log n) = O(nU_R\log n)$ steps with a constant probability, thus also in expectation. All interactable pairs have at least one interaction within $O(m\log n)$ steps in expectation. Therefore, within the next $O(m\log n) = O(n^2\log n)$ steps in expectation, Ξ reaches a configuration where $\{u.\mathsf{id}_A \mid u \in N_G(v)\} \subseteq v.\mathsf{neighbors}$. In this configuration, $\sum_{v\in V} |v.\mathsf{neighbors}| > 2m$ holds since at least one agent has one or more fake labels in its neighbors. By Lemma 2, Markov's inequality, and the union bound, all tokens T_x meet A_x thus $T_x.\mathsf{degree}_T = |A_x.\mathsf{neighbors}|$ holds for all $x = 0, 1, \ldots, n-1$ within $O(\mathbf{H}_G^{\mathcal{P}}\log n) = O(n\mathbf{H}_G^{\mathcal{S}}\log n)$ steps with a constant probability, thus also in expectation. An agent v resets v.sum and v.counted before or when it meets all tokens, thereafter v adds $T_x.\mathsf{degree}_T$ to v.sum for all tokens T_x before or when it meets all tokens again. As a result, v.sum reaches $2m+1$ and emits a new resetting signal before or when it meets all tokens and thereafter meets all tokens again, which occurs within $O(n\mathbf{H}_G^{\mathcal{S}})$ steps in expectation by Corollary 1. To conclude, some agent emits a new resetting signal within $O((n^2\mathbf{H}_G^{\mathcal{S}} + nU_R)\log n)$ steps in expectation.

Suppose now that an agent v emits a new resetting signal and a token T_x is located at v at this time. By the definition of $\mathbf{H}_G^{\mathcal{S}}$, Markov's inequality, and the union bound, T_x visits all agents before or when it moves $U_R \geq 4\mathbf{H}_G^{\mathcal{S}} \log_2 n = (2\log_2 n)\cdot(2\mathbf{H}_G^{\mathcal{S}})$ times with a probability at least $1 - n\cdot(1/2)^{2\log n} = 1 - 1/n = 1 - o(1)$. This yields that T_x resets the variable neighbors of all agents to the empty set at least once before or when it visits all agents with a probability at least $1 - o(1)$. By Markov's inequality and Corollary 2, T_x visits all agents within $O(n\mathbf{H}_G^{\mathcal{S}})$ steps with a probability $p = \Omega(1)$, once an agent emits a new resetting signal.

From above, we have observed that all agents reset their neighbors at least once and thus Ξ reaches a configuration in $\mathcal{S}_{\mathrm{noFake}}$ within $(1/p)\cdot O((n^2\mathbf{H}_G^{\mathcal{S}} + nU_R)\log n) = O((n^2\mathbf{H}_G^{\mathcal{S}} + nU_R)\log n)$ steps in expectation. □

Lemma 18. *Starting from any configuration $C_0 \in \mathcal{S}_{\mathrm{noFake}}$, an execution of $P_{\mathrm{neigh}}(\nu,\mu)$ under the uniformly random scheduler (i.e., $\Xi_{P(\nu,\mu)}(G, C_0, \mathbf{\Gamma})$) reaches a safe configuration within $O((n^2\mathbf{H}_G^{\mathcal{S}} + nU_R)\log n)$ steps in expectation.*

Proof. $\Xi = \Xi_{P(\nu,\mu)}(G, C_0, \mathbf{\Gamma})$ reaches a configuration where $\sum_{x=0}^{n-1} T_x.\mathsf{degree}_T \leq 2m$ holds within $O(n\mathbf{H}_G^{\mathcal{S}}\log n)$ steps in expectation; because every T_x meets A_x within $O(n\mathbf{H}_G^{\mathcal{S}})$ steps in expectation for each $x \in \{0, 1, \ldots, n-1\}$ (Lemma 2). Similarly, all agents reset their sum and counted in the next $O(n\mathbf{H}_G^{\mathcal{S}}\log n)$ step in expectation. Thereafter, no agent sees $\mathsf{sum} = 2m+1$, thus no agent emits the resetting signal. Then, the resetting signal disappears from the population in the next $O((nU_R + n\mathbf{H}_G^{\mathcal{S}})\log n) = (nU_R\log n)$ steps in expectation by Lemma 2, Markov's inequality, and the union bound. The interactions between all interactable pairs occur in the next $O(m\log n)$ steps in expectation. Thus, $v.\mathsf{neighbors} = \{u.\mathsf{id}_A \mid u \in N_G(v)\}$ holds for all $v \in V_G$. After that, no agent v changes v.neighbors, which yields that Ξ has reached a safe configuration. □

Theorem 3. *Given knowledge ν and μ, $P_{\mathrm{neigh}}(\nu,\mu)$ is a self-stabilizing protocol that solves* **NR** *in arbitrary graphs if $\nu = \{n\}$ and $\mu = \{m\}$ for some integers n and m, respectively. Starting from any configuration C_0 on any population $G = (V_G, E_G) \in \mathcal{G}_{n,m}$, the execution of $P_{\mathrm{neigh}}(\nu,\mu)$ under the uniformly random scheduler (i.e., $\Xi_{P_{\mathrm{neigh}}(\nu,\mu)}(G, C_0, \mathbf{\Gamma})$) reaches a safe configuration within $O((n^2\mathbf{H}_G^{\mathcal{S}} + nU_R)\log n)$ steps in expectation. Each agent uses $O(n)$ bits of memory space to execute $P_{\mathrm{neigh}}(\nu,\mu)$.*

Proof. The correctness and the time complexities are immediate from Lemmas 17 and 18. Each agent uses only $O(n)$ bits: both variables neighbors and counted require n bits and all other variables used in P_{neigh} require $O(\log n)$ bits. □

Recall that we require $U_R \geq 4\mathbf{H}_G^{\mathcal{S}}\log_2 n$. It is well known that $\mathbf{H}_G^{\mathcal{S}} = O(md)$, where d is the diameter of $\overline{G}$. Since we know both n and m, we can substitute a sufficiently large $\Theta(mn\log n)$ value for U_R. Then, $P_{\mathrm{neigh}}(\nu,\mu)$ converges in $O(n^2\mathbf{H}_G^{\mathcal{S}}\log n + mn^2\log^2 n)$ steps in expectation. In addition, if an asymptotically tight upper bound H on $\mathbf{H}_G^{\mathcal{S}}$ is known to the agents, we can substitute $4H\log_2 n$ for U_R. Then, $P_{\mathrm{neigh}}(\nu,\mu)$ converges in $O(n^2\mathbf{H}_G^{\mathcal{S}}\log n)$ steps in expectation, which may be much smaller than the above bound for some class of graphs.

6 Discussion

While the vast majority of the studies on population protocols consider the population consisting of anonymous agents (i.e., the agents without identifiers), one may wonder why we do not consider agents with unique identifiers as in [26] and [27], which seems greatly strengthen the power of the model. This question is natural because our protocol uses a logarithmic or larger number of bits in the memory of each agent: P_{rank} uses $O(\log n)$ bits and P_{neigh} uses $O(n)$ bits, which is usually enough to store the identifier of each agent. However, unique identifiers do not help us to solve the four problems we consider in this paper: the impossibility results (i.e., Lemmas 5 and 16) still hold even if the agents have unique identifiers. (The proofs of both lemmas are still correct without any modification.)

It is worth mentioning the difference between the ranking **RK** and the assignment of the unique identifiers. The **RK** assigns the unique identifiers (or ranks) to the agents but has a much stronger requirement: the ranks must be the distinct integers from 0 to n-1, while the requirement of the unique identifiers allows assignment of integers larger than n-1. The difference makes a significance impact on solvability of the leader election **LE**. Actually, we cannot solve the self-stabilizing leader election even with the unique identifiers, whereas it is immediately solved once the self-stabilizing ranking is solved.

7 Conclusion

In this paper, we clarified the solvability of the leader election problem, the ranking problem, the degree recognition problem, and the neighbor recognition problem by self-stabilizing population protocols with knowledge of the number of nodes and/or the number of edges in a network. The protocols given in this paper require *exact* knowledge of the number of agents and/or the number of interactable pairs. It is interesting and still open whether *ambiguous* knowledge such as "the number of interactable pairs is at most M" and "the number of agents is not a prime number" is useful to design self-stabilizing population protocols.

Acknowledgments

This work was supported in part by JSPS KAKENHI under Grants 17K19977, 18K18000, 18K18029, 18K18031, 19H04085, 20H04140, and 20KK0232 and in part by JST SICORP under Grant JPMJSC1606.

References

[1] Y. Sudo, M. Shibata, J. Nakamura, Y. Kim, and T. Masuzawa, "The power of global knowledge on self-stabilizing population protocols," in *Proc. Int. Colloq. Struct. Inf. Commun. Complexity*, 2020, pp. 237–254.

[2] D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta, "Computation in networks of passively mobile finite-state sensors," *Distrib. Comput.*, vol. 18, no. 4, pp. 235–253, 2006.

[3] D. Angluin, J. Aspnes, M. J. Fischer, and H. Jiang, "Self-stabilizing population protocols," *ACM Trans. Auton. Adaptive Syst.*, vol. 3, no. 4, pp. 1–28, 2008.

[4] J. Beauquier, P. Blanchard, and J. Burman, "Self-stabilizing leader election in population protocols over arbitrary communication graphs," in *Proc. Int. Conf. Princ. Distrib. Syst.*, 2013, pp. 38–52.

[5] D. Canepa and M. G. Potop-Butucaru , "Stabilizing leader election in population protocols," 2007. [Online]. Available: http://hal.inria.fr/inria-00166632

[6] H.-P. Chen and H.-L. Chen, "Self-stabilizing leader election," in *Proc. ACM Symp. Princ. Distrib. Comput.*, 2019, pp. 53–59.

[7] G. Cordasco and L. Gargano, "Space-optimal proportion consensus with population protocols," in *Proc. Int. Symp. Stabilization, Saf., Secur. Distrib. Syst.*, 2017, pp. 384–398.

[8] G. B. Mertzios, S. E. Nikoletseas, C. L. Raptopoulos, and P. G. Spirakis, "Determining majority in networks with local interactions and very small local memory," in *Proc. Int. Colloq. Automata, Lang., Program.*, 2014, pp. 871–882.

[9] Y. Sudo, T. Masuzawa, A. K. Datta, and L. L. Larmore, "The same speed timer in population protocols," in *Proc. 36th IEEE Int. Conf. Distrib. Comput. Syst.*, 2016, pp. 252–261.

[10] Y. Sudo, F. Ooshita, H. Kakugawa, and T. Masuzawa, "Loosely-stabilizing leader election on arbitrary graphs in population protocols," in *Proc. Int. Conf. Princ. Distrib. Syst.*, 2014, pp. 339–354.

[11] Y. Sudo, F. Ooshita, H. Kakugawa, and T. Masuzawa, "Loosely stabilizing leader election on arbitrary graphs in population protocols without identifiers or random numbers," *IEICE Trans. Inf. Syst.*, vol. 103, no. 3, pp. 489–499, 2020.

[12] D. Alistarh, R. Gelashvili, and J. Rybicki, "Fast graphical population protocols," 2021, *arXiv:2102.08808*.

[13] E. Dijkstra, "Self-stabilizing systems in spite of distributed control," *Commun. ACM*, vol. 17, no. 11, pp. 643–644, 1974.

[14] S. Cai, T. Izumi, and K. Wada, "How to prove impossibility under global fairness: On space complexity of self-stabilizing leader election on a population protocol model," *Theory Comput. Syst.*, vol. 50, no. 3, pp. 433–445, 2012.

[15] M. J. Fischer and H. Jiang, "Self-stabilizing leader election in networks of finite-state anonymous agents," in *Proc. Int. Conf. Princ. Distrib. Syst.*, 2006, pp. 395–409.

[16] T. Izumi, "On space and time complexity of loosely-stabilizing leader election," in *Proc. Int. Colloq. Struct. Inf. Commun. Complexity*, 2015, pp. 299–312.

[17] Y. Sudo, J. Nakamura, Y. Yamauchi, F. Ooshita, H. Kakugawa, and T. Masuzawa, "Loosely-stabilizing leader election in a population protocol model," *Theor. Comput. Sci.*, vol. 444, pp. 100–112, 2012.

[18] Y. Sudo, F. Ooshita, H. Kakugawa, T. Masuzawa, A. K. Datta, and L. L. Larmore, "Loosely-stabilizing leader election with polylogarithmic convergence time," *Theor. Comput. Sci.*, vol. 806, pp. 617–631, 2020.

[19] D. Angluin, J. Aspnes, and D. Eisenstat, "Fast computation by population protocols with a leader," *Distrib. Comput.*, vol. 21, no. 3, pp. 183–199, 2008.

[20] J. Burman, D. Doty, T. Nowak, E. E. Severson, and C. Xu, "Efficient self-stabilizing leader election in population protocols," 2020, *arXiv: 1907.06068*.

[21] Y. Sudo, R. Eguchi, T. Izumi, and T. Masuzawa, "Time-optimal loosely-stabilizing leader election in population protocols," 2020, *arXiv: 2005.09944*.

[22] D. Alistarh and R. Gelashvili, "Polylogarithmic-time leader election in population protocols," in *Proc. 42nd Int. Colloq. Automata, Lang., Program.*, 2015, pp. 479–491.

[23] L. Gąsieniec, G. Stachowiak, and P. Uznanski, "Almost logarithmic-time space optimal leader election in population protocols," in *Proc. ACM Symp. Parallelism Algorithms Archit.*, 2019, pp. 93–102.

[24] Y. Sudo, F. Ooshita, T. Izumi, H. Kakugawa, and T. Masuzawa, "Time-optimal leader election in population protocols," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 11, pp. 2620–2632, Nov. 2020.

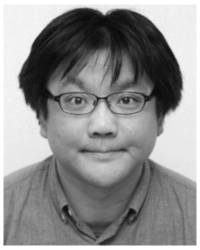
[25] D. Coppersmith, P. Tetali, and P. Winkler, "Collisions among random walks on a graph," *SIAM J. Discrete Math.*, vol. 6, no. 3, pp. 363–374, 1993.

[26] R. Guerraoui and E. Ruppert, "Even small birds are unique: Population protocols with identifiers," Dept. Comput. Sci. Eng., York Univ., York, ON, Canada, Tech. Rep. *TR-CSE-2007–04*, 2007.

[27] Y. Sudo, F. Ooshita, H. Kakugawa, T. Masuzawa, A. K. Datta, and L. L. Larmore, "Loosely-stabilizing leader election for arbitrary graphs in population protocol model," *IEEE Trans. Parallel. Distrib. Syst.*, vol. 30, no. 6, pp. 1359–1373, Jun. 2019.

Yuichi Sudo (Member, IEEE) received the BE, ME, and PhD degrees in information science and technology from Osaka University in 2009, 2011, and 2015, respectively. He was with Nippon Telegraph and Telephone Corporation and was engaged in research on network security during 2011–2017. He was an assistant professor with the Graduate School of Information Science and Technology, Osaka University, during 2017–2021. He has been an associate professor with the Faculty of Computer and Information Sciences, Hosei University, since April 2021. His research interests include distributed algorithms and graph theory. He is a member of the EATCS.

Masahiro Shibata received the BE, ME, and DE degrees in computer science from Osaka University in 2012, 2014, and 2017, respectively. Since 2017, he has been an assistant professor with the Kyushu Institute of Technology. His research interests include distributed algorithms and network management. He is a member of the IPSJ and IEICE.



Junya Nakamura (Member, IEEE) received the BE and ME degrees from the Toyohashi University of Technology, Japan, in 2006 and 2008, respectively, and the PhD degree in information science and technology from Osaka University in 2014. He is currently an associate professor with Information and Media Center, Toyohashi University of Technology. His research interests include theoretical and practical aspects of distributed algorithms and systems. He is a member of the IEEE Computer Society, IEICE, and IPSJ.

Yonghwan Kim received the BE double degree in electronic engineering and computing from Soongsil University, Seoul, South Korea, in 2009, the ME and PhD degrees in information science and technology from Osaka University in 2011 and 2015, respectively. He is currently an assistant professor with the Department of Computer Science, Graduate School of Engineering, Nagoya Institute of Technology, Japan. His research interests include distributed algorithms, fault-tolerance, autonomous mobile robots, and optimization problems. He is a regular member of the IPSJ and IEICE.

Toshimitsu Masuzawa (Member, IEEE) received the BE, ME, and DE degrees in computer science from Osaka University in 1982, 1984, and 1987. He was with Osaka University during 1987–1994, and was an associate professor with the Graduate School of Information Science, Nara Institute of Science and Technology during 1994–2000. He was also a visiting associate professor with the Department of Computer Science, Cornell University between 1993–1994. He is currently a professor with the Graduate School of Information Science and Technology, Osaka University. His research interests include distributed algorithms, parallel algorithms, and graph theory. He is a member of the ACM, IEICE, and IPSJ.