Doctoral Dissertation

A Study on Uniform k-partition and Graph Class Identification in the Population Protocol Model

Hiroto Yasumi
Program of Information Science and Engineering
Graduate School of Science and Technology
Nara Institute of Science and Technology

Supervisor: Michiko Inoue
Dependable System Lab. (Division of Information Science)

Submitted on March 9, 2022

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of SCIENCE

Hiroto Yasumi

A Study on Uniform k-partition and Graph Class Identification in the Population Protocol Model[*]

Hiroto Yasumi

Abstract

In recent years, autonomous distributed systems consisting of low-performance devices have attracted attention. For example, there is growing interest in a technology called molecular robotics, in which molecules themselves are designed as robots. To construct algorithms specialized for such low-performance devices, we deal with a population protocol model; in this model, devices only have greatly limited resources. So far, researchers have studied various problems in this model. In this dissertation, we tackle with two important challenges of the population protocol model.

One is to handle multiple tasks. Since devices are low-performance, one device may not handle multiple tasks. Thus, it is necessary to develop a mechanism to handle multiple tasks. As an approach to achieve it, we address network partitioning on the population protocol model. By partitioning a network into some groups, devices can handle multiple tasks by assigning different tasks to each group. Concretely, we study the uniform k-partition problem on the population protocol model. This problem aims to divide a network into k groups of the same size, where k is a given positive integer. In this dissertation, to cope with various situations, we consider the problem on various assumptions. Moreover, since devices only have greatly limited resources, we mainly study space-complexity to solve the problem. As a result, on various assumptions, we clarify the solvability and space-complexity of this problem.

The other challenge is efficient task execution. To execute tasks efficiently, we address investigation of the structure of networks. As many studies have shown,

[*]Doctoral Dissertation, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD2021045, March 9, 2022.

the structure of networks has a high impact on the efficiency of the algorithm. Hence, by understanding the structure of networks, we can apply appropriate algorithms to execute tasks efficiently. In this dissertation, we study graph class identification problems that aim to understand the structure of networks (i.e., graph class of networks). In particular, as basic structures of networks, we focus on lines, rings, k-regular graphs, stars, trees, and bipartite graphs. As a result, we clarify the solvability and space-complexity of the problems for the graphs.

Keywords:

population protocol, uniform k-partition problem, graph class identification problem, distributed system, distributed algorithm

Contents

List of Figures

List of Tables

Part I

Introduction

1. The Background

In recent years, autonomous distributed systems consisting of low-performance devices have attracted much attention. For example, there is growing interest in a technology called molecular robotics [39], in which molecules themselves are designed as robots. In this system, a large number of robots cooperate in a human body to achieve an objective (e.g., carrying medicines). Another example is to construct a network to investigate the ecosystem by attaching sensors to a flock of wild small animals such as birds. In this system, since sensors are attached to small animals, sensors should be small and thus sensors should have small amount of memory.

The population protocol model is an abstract model for such low-performance devices, introduced by Angluin et al. [7]. In this model, a network (called population) consists of multiple devices (called agents). Those agents are greatly limited; they are anonymous (i.e., they do not have identifiers), move unpredictably, and have a small number of bits of memory. 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 dissertation, we tackle with two significant challenges of the population protocol model. One of the challenges is to handle multiple tasks. The other challenge is efficient task execution.

Challenge 1: Handling multiple tasks

Handling multiple tasks is a key issue especially on the population protocol model. This is because, since agents are low-performance, one agent may not handle multiple tasks and thus it is necessary to develop a mechanism to handle multiple tasks.

In order to achieve it, we focus on partitioning a population. By partitioning a population into some groups, agents can handle multiple tasks by assigning different tasks to each group. Concretely, we study the uniform k-partition problem on the population protocol model. This problem aims to divide a population into k groups of the same size, where k is a given positive integer. By solving this problem, since the population can be divided into an arbitrary number of groups, an arbitrary number of tasks can be assigned, and, since each group has the same number of agents, the tasks can proceed uniformly.

Challenge 2: Efficient task execution

To execute task efficiently, we focus on the structure of networks and address the investigation of the structure of networks. As many studies have shown, the structure of networks has a high impact on the efficiency of the algorithm. Actually, in the population protocol model, some efficient protocols are proposed with limited structure of networks [5, 10, 26, 27]. Hence, by understanding the structure of networks, we can apply appropriate algorithms to execute tasks efficiently. In this dissertation, we study graph class identification problems that aim to understand the structure of networks (i.e., graph class of networks). In particular, as basic graph classes, we deal with lines, rings, k-regular graphs, stars, trees, and bipartite graphs.

2. Overview of This Dissertation

In this dissertation, we study the space complexity of the problems. That is, we investigate the number of states per agent to solve the problems. In the population protocol model, the first goal is to execute tasks while the memory capacity of devices is severely limited. Hence, it is very important to study the space-complexity in this model.

2.1 Uniform k-partition Problem

The uniform k-partition problem aims to divide a population into k groups of equal size, where k is a given positive integer.

As a first step, to clarify the solvability of the problem in a simple case, we study the solvability of the uniform 2-partition on complete communication graphs. More concretely, we study the solvability of the problem under various assumptions; in order to be flexible for future applications, it is necessary to consider various situations. In this dissertation, we consider four types of assumptions: 1) with an initialized base station, a non-initialized base station, or no base station, 2) designated or arbitrary initial states of agents, 3) asymmetric or symmetric protocols, and 4) global or weak fairness. A base station is a special agent that is distinguishable from other agents and has powerful capability. An initialized base station means that the base station has a designated initial state in the initial configuration, whereas a non-initialized base station means that the base station has an arbitrary initial state in the initial configuration. The base station enables us to construct efficient protocols, though it is sometimes difficult to implement. The assumption of initial states bears on the requirement of initialization of agents (other than the base station) and the fault-tolerant property. If a protocol requires designated initial states, it is necessary to initialize agents to execute the protocol. Alternatively, if a protocol solves the problem with arbitrary initial states, we do not need to initialize agents. This implies that, when agents transition to arbitrary states by transient faults, the protocol can reach the desired configuration. Symmetry of protocols is related to the power of symmetry breaking in the population. Asymmetric protocols may include asymmetric transitions that make agents with the same states transition to different states. This needs a mechanism to break symmetry among agents and its implementation is not easy with heavily limited devices. Symmetric protocols do not include such asymmetric transitions. Fairness is an assumption of interaction patterns. Though weak fairness guarantees only that every pair of agents interact infinitely often, global fairness makes a stronger assumption on the order of interactions. As a result, we clarify the solvability of the uniform 2-partition problem for each combination of the assumptions. In addition, we clarify tight upper and lower bounds on the number of states to solve the uniform 2-partition problem for each combination of the assumptions.

Secondly, we extend the results to the general case of an arbitrary number of partitions (the uniform k-partition). Concretely, we clarify the solvability on complete graphs for each combination of the assumptions, and clarify tight upper and lower bounds on the number of states for most combinations of the assumptions.

Finally, we consider the uniform 2-partition problem even on *arbitrary* communication graphs. In the population protocol model, most existing works consider the complete communication graph model. However, in realistic networks, communication graphs may be unpredictable. Thus, we also need algorithms that can solve the problem on arbitrary communication graphs. On arbitrary communication graphs, we clarify the solvability of the uniform 2-partition problem with designated initial states.

2.2 Graph Class Identification Problems

Graph class identification problems aim to decide whether a given communication graph is in the desired class (e.g., whether the given communication graph is a ring graph). In the population protocol model, the computability of the graph property was first considered in [6]. In [6], 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 dissertation, we answer those questions. That is, we study the solvability 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. The initial knowledge is given to agents for helping the agents solve the problem. We consider three types of the initial knowledge: the number of agents n, the upper bound P of the number of agents, and no knowledge. The initial knowledge enables us to construct efficient protocols although it may be difficult to know the knowledge in some situations.

As a result, for most combinations of the assumptions, we clarify the solvability of graph class identification problems for various basic graphs such as *line* graphs, *ring* graphs, *k-regular* graphs, *star* graphs, *tree* graphs, and *bipartite* graphs.

3. Related Works

The population protocol model was introduced in [7, 9], where the class of computable predicates in this model was studied. Then, many fundamental tasks were studied, such as leader election, counting, majority, etc [4, 13]. These problems were studied under various assumptions, such as the existence of a base station, fairness of the execution, symmetry of protocols, initial states of agents, and an initial knowledge of agents.

3.1 Leader Election Problem

A population protocol solves a *leader election* problem if starting from an initially uniform population of agents, eventually a single agent outputs *leader*, while all others output *non-leader*. The leader election problem was studied from the perspective of time and space efficiency. Doty and Soloveichik [29] proved that $\Omega(n)$ expected parallel time is required to solve leader election with probability 1 if agents have a constant number of states. Relaxing the number of states to a polylogarithmic value, Alistarh and Gelashvili [3] proposed a leader election protocol in polylogarithmic expected stabilization time. Then, Gąsieniec et al. [32] designed a protocol with $O(\log\log n)$ states and $O(\log n \cdot \log\log n)$ expected time. Furthermore, the protocol of Gąsieniec et al. [32] is space-optimal for solving the problem in polylogarithmic time. Sudo et al. [43] presented a leader election protocol with $O(\log n)$ states and $O(\log n)$ expected time. This protocol is time-optimal for solving the problem. Finally, Berenbrink et al. [20] proposed a time and space optimal protocol that solves the leader election problem with $O(\log\log n)$ states and $O(\log n)$ expected time.

The designated initial states are assumed in above protocols whereas leader election protocols with arbitrary initial states have also been studied by many researchers [11, 14, 24, 26, 27, 30, 47]. Protocols with arbitrary initial are called self-stabilizing protocols. In [11], Angluin et al. proposed various self-stabilizing protocols such as leader election, spanning-tree construction, etc. Moreover, they proved that, in the case of arbitrary communication graphs, self-stabilizing leader election is impossible. After that, researchers studied self-stabilizing leader election protocols with restricted graphs [26, 27], with some oracles [14, 30], or with some initial knowledge [24, 47].

As a variant of the self-stabilizing protocol, loosely self-stabilizing protocols were also studied for the leader election (loose stabilization relates to the fact that correctness is only guaranteed for a very long expected amount of time) [42, 41, 44, 33, 46, 45, 40]. Loosely self-stabilizing leader election was introduced by Sudo et al. [42]. In [42], they proposed a loosely self-stabilizing leader election with complete graphs and initial knowledge of the upper bound P of the number of agents. The protocol obtains a safe configuration with $O(nN \log n)$ expected time, where the safe configuration is a configuration after which a unique agent outputs leader for a sufficiently long time. Then, with complete graphs, some researchers proposed time-efficient protocols [33, 46]. Recently, Sudo et al. proposed a time-optimal protocol with complete graphs [40]. On the other hand, with arbitrary graphs, researchers proposed protocols with identifier of agent, or non-deterministic protocols [44, 41]. Then, Sudo et al. proposed a deterministic protocol without identifier [45].

3.2 Counting Problem

The *counting* problem consists in counting how many agents participate to the protocol; As the agent's memory is typically constant, this number is output by a special agent that may maintain logarithmic size memory, the base station. The counting problem was introduced by Beauquier et al. [16] and popularized the concept of a base station. Space complexity was further reduced by follow-up works [15, 34], until Aspnes et al. [12] finally proposed a time and space optimal protocol. In the above works, they consider the problem with arbitrary initial states (however the base station has a designated initial state). On the other hand, by allowing the initialization of agents, the counting protocols without the base station were proposed for both exact counting [21] and approximate counting [1, 21]. Alistarh et al. [1] proposed a protocol that computes an integer k such that $\frac{1}{2}\log n < k < 9\log n$ in $O(\log n)$ time with high probability using $O(\log n)$ states. Then, Berenbrink et al. [21] designed a protocol that outputs either $\lfloor \log n \rfloor$

or $\lceil \log n \rceil$ in $O(\log^2 n)$ time with high probability using $O(\log n \cdot \log\log n)$ states. Moreover, they proposed the exact counting protocol that computes n in $O(\log n)$ time using $\tilde{O}(n)$ states with high probability.

3.3 Majority Problem

The *majority* problem aims to decide which, if any, initial state in a population is a majority. The majority problem was addressed under different assumptions (*e.g.*, with or without failures [8], deterministic [31, 35] or probabilistic [2, 17, 18, 35] solutions, with arbitrary communication graphs [37], etc.). Those works also consider minimizing the time and space complexity. Berenbrink et al. [19] show trade-offs between time and space for the problem.

3.4 Uniform k-partition Problem

To our best knowledge, only a few variants of the uniform k-partition have been considered so far. Lamani et al. [36] studied a group decomposition problem that aims to divide a population into groups of designated sizes. Delporte-Gallet et al. [28] proposed a k-partition protocol with relaxed uniformity constraints: the population is divided into k groups such that in any group, at least $n/(2k)$ agents exist, where n is the number of agents. In addition, they also constructed a uniform 2-partition protocol as a subroutine of the protocol.

3.5 Graph Class Identification Problems

For graph class identification problems, Chatzigiannakis et al. [25] studied solvabilities for directed graphs with some properties on the mediated population protocol model [38], 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 [25], 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.

4. Organization of This Dissertation

This dissertation consists of six parts. In Part II, we introduce a formal model definition of the population protocol model. Part III and IV focus on the uniform k-partition problem under various assumptions. In Part III, we clarify the solvability of the uniform k-partition problem on complete communication graphs. In Part IV, we clarify the solvability of the uniform 2-partition problem on arbitrary communication graphs. Part V focuses on graph class identification problems. Concretely, we clarify the solvability of graph class identification problems for *line*, *ring*, *tree*, *k-regular*, and *star*. In Part VI, we conclude this dissertation.

Part II

Model Definition

In this part, we give a formal definition of the 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 (called a population), 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 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 descriptions of algorithms. 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. We consider only deterministic protocols in this dissertation. Protocol $\mathcal{P} = (Q, Y, \gamma, \delta)$ is symmetric if, for every transition $(p, q) \to (p', q')$ in δ, $(q, p) \to (q', p')$ exists in δ. In particular, if a protocol $\mathcal{P} = (Q, Y, \gamma, \delta)$ is symmetric and transition $(p, p) \to (p', q')$ exists in δ, $p' = q'$ holds. In this dissertation, an arbitrary protocol is also called an asymmetric protocol. When we refer to a protocol as an asymmetric protocol, we simply emphasize that the protocol is not subject to the restriction of transitions (whereas symmetric protocols are subject to the restriction of transitions). Note that a symmetric protocol is also an asymmetric protocol (and an asymmetric protocol is not necessarily a symmetric protocol).

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, $\ldots$, $C_m = C'$ such that $C_i \to C_{i+1}$ holds for every i $(0 \le i < m)$, 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 $C_i \to C_{i+1}$ holds for every i $(i \ge 0)$. 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 $C \to C'$, 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 $C \to C'$ occurs infinitely often, and we can deduce that C'' satisfying $C' \to C''$ 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 dissertation, we consider three possibilities for the base station: initialized base station, non-initialized base station, and no base station. In the model with a base station, we assume that a single agent, called a base station, exists in V. Then, V can be partitioned into V_b, the singleton set containing the base station, and V_p, the set of agents except for the base station. The base station can be distinguished from other agents in V_p, although agents in V_p cannot be distinguished. Then, the state set Q can be partitioned into a state set Q_b for the base station, and a state set Q_p for agents in V_p. The base station has unlimited resources (with respect to the number of states), in contrast with other resource-limited agents (that are allowed only a limited number of states). So,

[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 $(p, q) \to (p', q')$ exists for $p \ne q$, a transition $(q, p) \to (q', p')$ also exists.

when we evaluate the space complexity of a protocol, we focus on the number of states $|Q_p|$ for agents in V_p and do not consider the number of states $|Q_b|$ that are allocated to the base station. In the sequel, we thus say a protocol uses x states if $|Q_p| = x$ holds. When we assume an initialized base station, the base station has a designated initial state. When we assume a non-initialized base station, the base station has an arbitrary initial state (in Q_b). When we assume no base station, there exists no base station and thus $V = V_p$ and $Q = Q_p$ hold. For simplicity, we use agents only to refer to agents in V_p in the following sections. To refer to the base station, we always use the term base station (not an agent).

In this dissertation, we consider three settings 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 $\mathcal{P}_{P=x}$. Unless otherwise mentioned, agents have no knowledge.

Part III

Uniform k-partition on Complete Graphs

1. Introduction

In this part, we aim to clarify the solvabilities of the uniform k-partition on complete graphs under various assumptions. As a first step, we consider a simple case of the uniform k-partition where the number of partitions is two (the uniform 2-partition). Then, we consider the general case (the uniform k-partition). We remark that some results of the uniform 2-partition is applied to the uniform k-partition. Note that, in this part, since we assume complete graphs, each pair of agents can interact.

1.1 Our Contributions

Results for the uniform 2-partition problem

Tables 1 and 2 show the solvability of the uniform 2-partition on complete graphs.

These tables show the number of states to solve the uniform 2-partition problem under various assumptions, where P is the known upper bound of the number of agents. In this dissertation, we provide tight upper and lower bounds of the number of states for each case. With a (initialized or non-initialized) base station and designated initial states, we show that three states are necessary and sufficient to solve the problem, regardless of fairness and symmetry. With an initialized base station and arbitrary initial states, we show that four states are necessary and sufficient to solve the problem under global fairness; under weak fairness, we show that P states (resp., $P+1$ states) are necessary and sufficient for asymmetric protocols (resp., symmetric protocols). With a non-initialized base station and arbitrary initial states, we prove that the uniform 2-partition problem is unsolvable, regardless of fairness and symmetry. With no base station and designated initial states, Delporte-Gallet et al. [28] constructed an asymmetric uniform 2-

Table 1. Minimum number of states to solve the uniform 2-partition problem under global fairness on complete graphs.

Base station	Initial states	Symmetry	Upper bound	Lower bound
Initialized base station	Designated	Asymmetric	3	3
		Symmetric	3	3
	Arbitrary	Asymmetric	4	4
		Symmetric	4	4
Non-initialized base station	Designated	Asymmetric	3	3
		Symmetric	3	3
	Arbitrary	Asymmetric	Unsolvable	
		Symmetric	Unsolvable	
No base station	Designated	Asymmetric	3 [28]	3
		Symmetric	4 [22]	4
	Arbitrary	Asymmetric	Unsolvable	
		Symmetric	Unsolvable	

Table 2. Minimum number of states to solve the uniform 2-partition problem under weak fairness on complete graphs.

Base station	Initial states	Symmetry	Upper bound	Lower bound
Initialized base station	Designated	Asymmetric	3	3
		Symmetric	3	3
	Arbitrary	Asymmetric	P	P
		Symmetric	$P+1$	$P+1$
Non-initialized base station	Designated	Asymmetric	3	3
		Symmetric	3	3
	Arbitrary	Asymmetric	Unsolvable	
		Symmetric	Unsolvable	
No base station	Designated	Asymmetric	3	3
		Symmetric	Unsolvable	
	Arbitrary	Asymmetric	Unsolvable	
		Symmetric	Unsolvable	

partition protocol that solves the problem with three states under global fairness. Moreover, with no base station and designated initial states, Bournez et al. [22] proposed a transformer that transforms an asymmetric protocol into a symmetric protocol under global fairness by assuming additional states. We use the same idea to construct a symmetric uniform 2-partition protocol under global fairness with no base station and designated initial states. As a result for lower bounds, we prove that three states (resp., four states) are necessary to solve the problem under global fairness for asymmetric protocols (resp., symmetric protocols). On the other hand, under weak fairness, we show that three states are necessary and sufficient for asymmetric protocols and there is no symmetric protocol, with no base station and designated initial states. With no base station and arbitrary initial states, we prove that the uniform 2-partition problem is unsolvable, regardless of fairness and symmetry.

Interestingly, in many impossibility results, we not only prove that there is no uniform 2-partition protocol, but also that there is no protocol that solves the 2-partition problem (which aims to divide a population into 2 groups, but the problem allows for a constant difference in the number of agents between the groups). Concretely, in all cases other than the case where we consider the problem with an initialized base station and arbitrary initial states under weak fairness, we show that the lower bounds of the 2-partition are the same as the lower bounds of the uniform 2-partition. Moreover, since uniform 2-partition protocols are also 2-partition protocols, possibility results of the uniform 2-partition can be applied directly to the 2-partition. Tables 3 and 4 show the results of the 2-partition on complete graphs. Overall, even in the case of the 2-partition, we clarify tight upper and lower bounds on the number of states in most settings.

For the case of an initialized base station, arbitrary initial states, and weak fairness, it is interesting to compare these results with those of naming protocols [23]. A naming protocol aims to assign different states to all agents, and hence it can be regarded as a uniform P-partition protocol (the size of each group is zero or one). Burman et al. [23] prove that, to construct naming protocols in the same setting, P states are necessary and sufficient for asymmetric protocols and $P + 1$ states are necessary and sufficient for symmetric protocols. That is, naming protocols have the same space complexity as uniform 2-partition protocols.

Table 3. Minimum number of states to solve the 2-partition problem under global fairness on complete graphs.

Base station	Initial states	Symmetry	Upper bound	Lower bound
Initialized base station	Designated	Asymmetric	3	3
		Symmetric	3	3
	Arbitrary	Asymmetric	4	4
		Symmetric	4	4
Non-initialized base station	Designated	Asymmetric	3	3
		Symmetric	3	3
	Arbitrary	Asymmetric	Unsolvable	
		Symmetric	Unsolvable	
No base station	Designated	Asymmetric	3 [28]	3
		Symmetric	4 [22]	4
	Arbitrary	Asymmetric	Unsolvable	
		Symmetric	Unsolvable	

Table 4. Minimum number of states to solve the 2-partition problem under weak fairness on complete graphs.

Base station	Initial states	Symmetry	Upper bound	Lower bound
Initialized base station	Designated	Asymmetric	3	3
		Symmetric	3	3
	Arbitrary	Asymmetric	P	-
		Symmetric	$P+1$	-
Non-initialized base station	Designated	Asymmetric	3	3
		Symmetric	3	3
	Arbitrary	Asymmetric	Unsolvable	
		Symmetric	Unsolvable	
No base station	Designated	Asymmetric	3	3
		Symmetric	Unsolvable	
	Arbitrary	Asymmetric	Unsolvable	
		Symmetric	Unsolvable	

Clearly naming protocols (or uniform P-partition protocols) require P states to assign different states to P agents. Interestingly uniform 2-partition protocols still require P states in this setting. On the other hand, the number of states is reduced to three or four when we assume designated initial states or global fairness.

Results for the uniform k-partition problem

Tables 5 and 6 show the solvability of the uniform k-partition on complete graphs. First of all, some results of the uniform 2-partition can be applied to the uniform k-partition. First, since the uniform 2-partition is a special case of the uniform k-partition, impossibility results of the uniform 2-partition can be applied directly to the uniform k-partition. In addition, we prove the impossibility of k states with the base station and designated initial states under global fairness by extending the impossibility of three states for the uniform 2-partition. Secondly, with an initialized base station under weak fairness, the P-state and $P+1$-state protocols for the uniform 2-partition also work even for the uniform k-partition. Moreover, with designated initial states and the (initialized or non-initialized) base station, the uniform 2-partition protocols are easily extended to the uniform k-partitions.

The main contribution for the uniform k-partition is to propose a protocol with designated initial states and no base station. Concretely, for the case with no base station and designated initial states under global fairness, we propose a symmetric protocol. This protocol requires $3k-2$ states for each agent. This space complexity is asymptotically optimal because the lower bound in this setting is $k+1$. Moreover, we evaluate the time complexity of the protocol by simulations. From the simulation results, we can observe that the time complexity increases exponentially with k but not exponentially with n, where n is the number of agents.

As in the case of the uniform 2-partition, most results of the uniform k-partition can also be applied to the k-partition (see Tables 7 and 8). Hence, even in the case of the k-partition, we clarify tight upper and lower bounds on the number of states in most settings.

Table 5. Minimum number of states to solve the uniform k-partition problem under global fairness on complete graphs.

Base station	Initial states	Symmetry	Upper bound	Lower bound
Initialized base station	Designated	Asymmetric	$k+1$	$k+1$
		Symmetric	$k+1$	$k+1$
	Arbitrary	Asymmetric	P	$k+1$
		Symmetric	$P+1$	$k+1$
Non-initialized base station	Designated	Asymmetric	$k+1$	$k+1$
		Symmetric	$k+1$	$k+1$
	Arbitrary	Asymmetric	Unsolvable	
		Symmetric	Unsolvable	
No base station	Designated	Asymmetric	$3k-2$	$k+1$
		Symmetric	$3k-2$	$k+1$
	Arbitrary	Asymmetric	Unsolvable	
		Symmetric	Unsolvable	

Table 6. Minimum number of states to solve the uniform k-partition problem under weak fairness on complete graphs.

Base station	Initial states	Symmetry	Upper bound	Lower bound
Initialized base station	Designated	Asymmetric	$k+1$	$k+1$
		Symmetric	$k+1$	$k+1$
	Arbitrary	Asymmetric	P	P
		Symmetric	$P+1$	$P+1$
Non-initialized base station	Designated	Asymmetric	$k+1$	$k+1$
		Symmetric	$k+1$	$k+1$
	Arbitrary	Asymmetric	Unsolvable	
		Symmetric	Unsolvable	
No base station	Designated	Asymmetric	-	$k+1$
		Symmetric	Unsolvable	
	Arbitrary	Asymmetric	Unsolvable	
		Symmetric	Unsolvable	

Table 7. Minimum number of states to solve the k-partition problem under global fairness on complete graphs.

Base station	Initial states	Symmetry	Upper bound	Lower bound
Initialized base station	Designated	Asymmetric	$k + 1$	$k + 1$
		Symmetric	$k + 1$	$k + 1$
	Arbitrary	Asymmetric	P	$k + 1$
		Symmetric	$P + 1$	$k + 1$
Non-initialized base station	Designated	Asymmetric	$k + 1$	$k + 1$
		Symmetric	$k + 1$	$k + 1$
	Arbitrary	Asymmetric	Unsolvable	
		Symmetric	Unsolvable	
No base station	Designated	Asymmetric	$3k - 2$	$k + 1$
		Symmetric	$3k - 2$	$k + 1$
	Arbitrary	Asymmetric	Unsolvable	
		Symmetric	Unsolvable	

Table 8. Minimum number of states to solve the k-partition problem under weak fairness on complete graphs.

Base station	Initial states	Symmetry	Upper bound	Lower bound
Initialized base station	Designated	Asymmetric	$k + 1$	$k + 1$
		Symmetric	$k + 1$	$k + 1$
	Arbitrary	Asymmetric	P	-
		Symmetric	$P + 1$	-
Non-initialized base station	Designated	Asymmetric	$k + 1$	$k + 1$
		Symmetric	$k + 1$	$k + 1$
	Arbitrary	Asymmetric	Unsolvable	
		Symmetric	Unsolvable	
No base station	Designated	Asymmetric	-	$k + 1$
		Symmetric	Unsolvable	
	Arbitrary	Asymmetric	Unsolvable	
		Symmetric	Unsolvable	

2. Problem Definition

Now, we give a formal definition of the uniform k-partition problem (and the k-partition problem).

Let V_p be a set of all agents except for the base station. Let $Y = \{color_1, color_2, \ldots, color_k\}$ be an output set of the problem. Let $\gamma : Q_p \to Y$ be a function that maps a state of an agent (except for the base station) to Y. We define a color of $v \in V_p$ as $\gamma(s(v))$. We say agent $v \in V_p$ belongs to the i-th group if $\gamma(s(v)) = color_i$ holds.

An execution $\Xi = C_0, C_1, C_2, \ldots$ solves the k-partition problem with size difference $x \geq 1$ if Ξ includes a stable configuration C_t that satisfies the following condition:

- There is a partition $\{H_1, H_2, \ldots, H_k\}$ of V_p such that, 1) $||H_i| - |H_j|| \leq x$ for any i and j, and, 2) for all C^* such that $C \xrightarrow{*} C^*$, each agent in H_i belongs to the i-th group at C^* (i.e., at C^*, any agent v in H_i satisfies $\gamma(s(v)) = color_i$).

In this dissertation, we assume that size difference x is a constant. If each execution Ξ of protocol $\mathcal{P}$ solves the k-partition problem with size difference $x \geq 1$, we say protocol $\mathcal{P}$ solves the k-partition problem with x. In particular, if each execution Ξ of protocol $\mathcal{P}$ solves the k-partition problem with size difference $x = 1$, we say protocol $\mathcal{P}$ solves the uniform k-partition problem.

3. Uniform 2-partition under Global Fairness on Complete Graphs

In this section, we show the upper and lower bounds of the uniform 2-partition on complete graphs under global fairness. Concretely, we consider three assumptions 1) a population with an initialized base station, non-initialized base station, or no base station, 2) symmetric or asymmetric protocols, and 3) designated or arbitrary initial states, and, for each combination of assumptions, we completely clarify solvability of the uniform 2-partition problem. Moreover, if solvable, we show the tight upper and lower bounds on the number of states.

In this section, since we consider the case of $k = 2$, function γ is defined as $\gamma : Q_p \rightarrow \{color_1, color_2\}$. In the case, we assign colors *red* and *blue* to $color_1$ and $color_2$, respectively, and we define γ as function $\gamma : Q_p \rightarrow \{red, blue\}$ that maps a state of a non base station agent to *red* or *blue*. We say agent $a \in V_p$ is *red* (resp., *blue*) if $\gamma(s(a)) = red$ (resp., $\gamma(s(a)) = blue$) holds.

3.1 Uniform 2-partition Protocols with Initialized Base Station

In this subsection, we consider the uniform 2-partition problem under the assumption of an initialized base station. Recall that the base station is distinguishable from other agents, and we do not care the number of states for the base station.

3.1.1 Upper Bound for Symmetric Protocols with Designated Initial States

In this protocol, the state set of agents is $Q_p = \{initial, red, blue\}$, and we set $\gamma(initial) = \gamma(red) = red$ and $\gamma(blue) = blue$. The designated initial state of all agents is *initial*. The idea of the protocol is to assign states *red* and *blue* to agents alternately when agents interact with the base station. To do this, the base station has a state set $Q_b = \{b_{red}, b_{blue}\}$, and its initial state is b_{red}. The protocol consists of the following two transitions.

1. $(b_{red}, initial) \rightarrow (b_{blue}, red)$
2. $(b_{blue}, initial) \rightarrow (b_{red}, blue)$

That is, when the base station in state b_{red} (resp., b_{blue}) and an agent in state *initial* interact, the base station changes the state of the agent to *red* (resp., *blue*) and the state of itself to b_{blue} (resp., b_{red}). When two agents (except for the base station) interact, no state transition occurs. Clearly, all agents evenly transition to state *red* or *blue*, and the difference in the numbers of *red* and *blue* agents is at most one. Note that the protocol contains no asymmetric transition and works correctly if every agent interacts with the base station. Therefore, we have the following theorem.

Theorem **1.** *In the model with an initialized base station, there exists a symmetric protocol with three states and designated initial states that solves the uniform 2-partition problem under global fairness.*

Remark. This protocol works even under weak fairness. Even under weak fairness, eventually all agents interact with the base station. Hence, eventually, all agents transition to either *red* or *blue*. Moreover, the protocol does not depend on the initial state of the base station; that is, the protocol can work with a non-initialized base station. From these facts, we can observe that the protocol solves the problem under weak fairness with a non-initialized base station.

Theorem **2.** *In the model with a non-initialized base station, there exists a symmetric protocol with three states and designated initial states that solves the uniform 2-partition problem under weak fairness.*

3.1.2 Lower Bound with Designated Initial States

Next, we show three states are necessary to construct an asymmetric protocol under global fairness. This implies that, in this case, three states are necessary for asymmetric or symmetric protocols under global fairness because a symmetric protocol is also asymmetric. That is, three states are necessary and sufficient in this case.

Theorem **3.** *In the model with an initialized base station, no asymmetric protocol with two states and designated initial states solves the 2-partition problem with size difference $x \geq 1$ under global fairness.*

Proof. For contradiction, we assume that such a protocol *Alg* exists. Without loss of generality, we assume $Q_p = \{s_1, s_2\}$, $f(s_1) = red$, $f(s_2) = blue$, and that the designated initial state of all agents is s_1. Let n be a number that is at least $x+4$. We consider the following three cases.

First, for population V of a single base station and n agents $a_1, a_2, \dots, a_n$, consider a globally fair execution $\Xi = C_0, C_1, \dots$ of *Alg* with size difference x. According to the definition, there exists a stable configuration C_t. That is, after C_t, the state of each agent does not change even if the base station and agents in states s_1 and s_2 interact in any order, and the difference in the numbers of *red* and *blue* agents is at most x. Note that, since n is at least $x+4$, the numbers of *red* and *blue* agents is at least 2 after C_t.

Next, for population V' of a single base station and $n+2x+1$ agents $a_1, a_2, \ldots, a_{n+2x+1}$, we define an execution $\Xi' = C'_0, C'_1, \ldots, C'_t, C'_{t+1}, \ldots$ of Alg as follows.

- From C'_0 to C'_t, the base station and n agents $a_1, a_2, \ldots, a_n$ interact in the same order as the execution Ξ.
- After C'_t, the base station and $n+2x+1$ agents interact so as to satisfy global fairness.

Since the base station and agents $a_1, \ldots, a_n$ change their states similarly to Ξ from C'_0 to C'_t, the number of *red* agents is $x+1$ or more than the number of *blue* agents at C'_t. Moreover, the state of the base station at C'_t is the same as the state of the base station at C_t. However, since the difference in the numbers of *red* and *blue* agents is at least $x+1$, C'_t is not a stable configuration. Consequently, after C'_t, some *red* or *blue* agent changes its state in execution Ξ'.

Lastly, we define execution $\Xi'' = C''_0, C''_1, \ldots$ for population V as follows. First, we make agents transition similarly to Ξ and reach stable configuration $C''_t (= C_t)$ in Ξ''. After that we apply interactions in Ξ' to execution Ξ''. That is, we make agents interact as follows after C''_t in Ξ'': 1) when the base station and an agent in state $s \in \{s_1, s_2\}$ interact at $C'_u \to C'_{u+1}$ $(u \geq t)$ in Ξ', the base station and an agent in state s interact at $C''_u \to C''_{u+1}$ in Ξ'', and 2) when two agents in states $s \in \{s_1, s_2\}$ and $s' \in \{s_1, s_2\}$ interact at $C'_u \to C'_{u+1}$ $(u \geq t)$ in Ξ', two agents in states s and s' interact at $C''_u \to C''_{u+1}$ in Ξ''. We can realize such interactions because, after stable configuration C''_t, at least two agents are in s_1 and at least two agents are in s_2. After C''_t, since interactions occur similarly to Ξ', some *red* or *blue* agent changes its state similarly to Ξ'. After such a state change occurs, we make agents interact so that Ξ'' satisfies global fairness. This implies that, in globally fair execution Ξ'', an agent changes its color after stable configuration C''_t. This is a contradiction. □

3.1.3 Upper Bound for Symmetric Protocols with Arbitrary Initial States

Here we show a symmetric protocol with four states under global fairness. In this protocol, each agent x has two variables $rb_x \in \{red, blue\}$ and $mark_x \in \{0, 1\}$. Variable rb_x represents the color of agent x. That is, for state s of agent x, $\gamma(s) = red$ holds if $rb_x = red$ and $\gamma(s) = blue$ holds if $rb_x = blue$. We define $\#red$ as the number of *red* agents and $\#blue$ as *blue* agents. We explain the role of variable $mark_x$ later.

The basic strategy of the protocol is that the base station counts *red* and *blue* agents by counting protocol *Count* [15] and changes colors of agents so that the numbers of *red* and *blue* agents become equal. Protocol *Count* is a symmetric protocol that counts the number of agents from arbitrary initial states under global fairness. Protocol *Count* uses only two states for each agent. We use variable $mark_x$ to maintain the state of protocol *Count*. In protocol *Count*, the base station has variable *Count.out* that eventually outputs the number of agents. More concretely, *Count.out* initially has value 0, gradually increases one by one, eventually equals to the number of agents, and stabilizes. The following lemma explains the characteristic of protocol *Count*.

Lemma **1** ([15]). *Let n be the number of agents. In the initial configuration, Count.out = 0 holds. When Count.out < n, Count.out eventually increases by one under global fairness. When Count.out = n, Count.out never changes and stabilizes.*

To count *red* and *blue* agents, the base station executes two instances of protocol *Count* in parallel to the main procedure of the uniform 2-partition protocol. We denote by $Count_{red}$ and $Count_{blue}$ instances of protocol *Count* to count *red* and *blue* agents, respectively. The base station executes $Count_{red}$ when it interacts with a *red* agent. That is, the base station updates variables of $Count_{red}$ at the base station and the *red* agent by applying a transition of protocol $Count_{red}$. By this behavior, the base station executes $Count_{red}$ as if the population contains only *red* agents. Therefore, after the base station initializes its own variables of $Count_{red}$, it can correctly count the number of *red* agents by $Count_{red}$ (i.e., $Count_{red}.out$ eventually stabilizes to $\#red$) as long as a set of *red* agents does not change. Similarly, the base station executes $Count_{blue}$ when it interacts with a *blue* agent, and counts the number of *blue* agents. The straightforward approach to use the counting protocols is to adjust colors of agents after $Count_{red}.out$ and $Count_{blue}.out$ stabilize. However, the base station cannot know whether the outputs have stabilized or not. For this reason, the base station maintains estimated numbers of *red* and *blue* agents, and it changes colors of agents when the difference in the estimated numbers of *red* and *blue* agents is two. Note that, since the counting protocols assume that a set of counted agents does not change, the base station must restart $Count_{red}$ and $Count_{blue}$ from the beginning when the base station changes colors of some agents.

We explain the details of this procedure. The base station records the estimated numbers of *red* and *blue* agents in variables $C^*_{rb}[red]$ and $C^*_{rb}[blue]$, respectively. In the beginning of execution, these variables are identical to outputs of $Count_{red}$ and $Count_{blue}$. If the difference between $C^*_{rb}[red]$ and $C^*_{rb}[blue]$ becomes two, the base station immediately changes colors of agents. At the same time, the base station updates $C^*_{rb}[red]$ and $C^*_{rb}[blue]$ to reflect the change of colors. After the base station changes colors of some agents, it restarts $Count_{red}$ and $Count_{blue}$ from the beginning by initializing its own variables of the counting protocols. Since the counting protocols allow arbitrary initial states of agents, the base station can correctly count *red* and *blue* agents after that. Note that the base station does not initialize $C^*_{rb}[red]$ and $C^*_{rb}[blue]$ because it knows such numbers of *red* and *blue* agents exist. If the output of $Count_{red}$ and $Count_{blue}$ exceeds $C^*_{rb}[red]$ and $C^*_{rb}[blue]$, the base station updates $C^*_{rb}[red]$ and $C^*_{rb}[blue]$, respectively. After that, if the difference between $C^*_{rb}[red]$ and $C^*_{rb}[blue]$ becomes two, the base station changes colors of agents. By repeating this behavior, the base station adjusts colors of agents.

The pseudocode of this protocol is given in Algorithm 1. We define $\overline{red} = blue$ and $\overline{blue} = red$. Recall that variable $mark_x$ is a two-state variable of counting protocols $Count_{red}$ and $Count_{blue}$. Since the base station restarts the counting protocols whenever it changes colors of agents, the base station keeps a set of *red* (resp., *blue*) agents unchanged until it restarts $Count_{red}$ (resp., $Count_{blue}$). In addition, each agent is involved in either $Count_{red}$ or $Count_{blue}$ at the same time. Hence it requires only a single variable $mark_x$ to execute $Count_{red}$ and $Count_{blue}$.

Algorithm 1 Uniform 2-partition protocol

```
Variables at the base station:
    $C^*_{rb}[c] (c \in \{red, blue\})$: the estimated number of $c$ agents, initialized to 0
    $Variables$: variables of $Count_c (c \in \{red, blue\})$
Variables at an agent $x$:
    $rb_x \in \{red, blue\}$: color of the agent, initialized arbitrarily
    $mark_x \in \{0, 1\}$: a variable of $Count_c (c \in \{red, blue\})$, initialized arbitrarily
 1: when an agent $x$ interacts with the base station do
 2:     Update $mark_x$ and variables of $Count_{rb_x}$ at the base station by applying
    a transition of $Count_{rb_x}$
 3:     if $C^*_{rb}[rb_x] < Count_{rb_x}.out$ then
 4:         $C^*_{rb}[rb_x] \leftarrow Count_{rb_x}.out$
 5:     end if
 6:     if $C^*_{rb}[rb_x] - C^*_{rb}[\overline{rb_x}] = 2$ then
 7:         $C^*_{rb}[rb_x] \leftarrow C^*_{rb}[rb_x] - 1$
 8:         $C^*_{rb}[\overline{rb_x}] \leftarrow C^*_{rb}[\overline{rb_x}] + 1$, $rb_x \leftarrow \overline{rb_x}$
 9:         reset variables of $Count_{red}$ and $Count_{blue}$ at base station
10:     end if
11: end
```

When two agents (except for the base station) interact, no state transition occurs in this protocol and counting protocols. When the base station and a *red* agent interact, they update $mark_x$ and variables of $Count_{red}$ at the base station by applying a transition of $Count_{red}$. This means that they execute $Count_{red}$ in parallel to the main procedure of the uniform 2-partition protocol. After that, if $Count_{red}.out$ is larger than $C^*_{rb}[red]$, $C^*_{rb}[red]$ is updated with $Count_{red}.out$. If the difference between $C^*_{rb}[red]$ and $C^*_{rb}[blue]$ becomes two, the *red* agent changes its color to *blue* and the base station updates $C^*_{rb}[red]$ and $C^*_{rb}[blue]$. After updating, the base station resets variables of $Count_{red}$ and $Count_{blue}$, and restarts counting. When the base station and a *blue* agent interact, they behave similarly.

Lemma **2.** *In any configuration,* $C^*_{rb}[red] \le \#red$*,* $C^*_{rb}[blue] \le \#blue$ *and* $|C^*_{rb}[red] - C^*_{rb}[blue]| \le 1$ *hold.*

Proof. We prove by induction on the index $k \ge 0$ of a configuration in an execution $C_0, C_1, C_2, \dots, C_k, \dots$. At the initial configuration C_0, the lemma holds. Let us assume that the lemma holds for configuration C_k and prove it for configuration C_{k+1}. From this assumption, $C^*_{rb}[red] \le \#red$, $C^*_{rb}[blue] \le \#blue$ and $|C^*_{rb}[red] - C^*_{rb}[blue]| \le 1$ hold at C_k. Assume that, when C_k transitions to C_{k+1}, the base station and agent x interact. If $Count_{rb_x}.out$ becomes larger than $C^*_{rb}[rb_x]$, the base station updates $C^*_{rb}[rb_x]$ by $C^*_{rb}[rb_x] \leftarrow Count_{rb_x}.out$ (line 3). Note that, in this case, $C^*_{rb}[rb_x]$ increases by one from Lemma 1. In addition, $C^*_{rb}[red] \le \#red$ and $C^*_{rb}[blue] \le \#blue$ still hold. Recall that $|C^*_{rb}[red] - C^*_{rb}[blue]| \le 1$ held before this update and $C^*_{rb}[rb_x]$ increases by one. Consequently, at this moment (before line 5), $|C^*_{rb}[rb_x] - C^*_{rb}[\overline{rb_x}]| \le 1$ or $C^*_{rb}[rb_x] - C^*_{rb}[\overline{rb_x}] = 2$ holds. Next, we consider lines 5 to 9. If $C^*_{rb}[rb_x] - C^*_{rb}[\overline{rb_x}] \le 1$ at line 5, lines 6 to 8 are not executed, and thus $C^*_{rb}[red] \le \#red$, $C^*_{rb}[blue] \le \#blue$ and $|C^*_{rb}[red] - C^*_{rb}[blue]| \le 1$ hold. If $C^*_{rb}[rb_x] - C^*_{rb}[\overline{rb_x}] = 2$ at line 5, agent x changes its color from rb_x to $\overline{rb_x}$, $C^*_{rb}[rb_x]$ decreases by one, and $C^*_{rb}[\overline{rb_x}]$ increases by one. This also preserves $C^*_{rb}[red] \le \#red$, $C^*_{rb}[blue] \le \#blue$ and $|C^*_{rb}[red] - C^*_{rb}[blue]| \le 1$. Therefore, the lemma holds. □

Theorem **4.** *Algorithm 1 solves the uniform 2-partition problem. That is, in the model with an initialized base station, there exists a symmetric protocol with* *four states and arbitrary initial states that solves the uniform 2-partition problem under global fairness.*

Proof. We define $phase = C^*_{rb}[red] + C^*_{rb}[blue]$. Initially, $phase = 0$ holds. We show that 1) $phase$ increases one by one if $phase < n$, and 2) Algorithm 1 solves the uniform 2-partition problem if $phase = n$.

First we consider the initial configuration. Since we assume global fairness, $Count_{red}.out$ or $Count_{blue}.out$ increases by one from Lemma 1 and at that time $phase$ increases by one.

Let us consider the transition $C \rightarrow C'$ such that $phase$ increases by one (i.e., line 4 is executed) and $phase < n$ holds at C'. We consider two cases.

- Case that lines 7 to 9 are not executed at $C \rightarrow C'$. In this case, since the base station does not change sets of red and $blue$ agents, it can correctly continue to execute $Count_{red}$ and $Count_{blue}$. Since $phase < n = \#red + \#blue$ holds, either $\#red > C^*_{rb}[red]$ or $\#blue > C^*_{rb}[blue]$ holds. Consequently, from Lemma 1, either $Count_{red}.out > C^*_{rb}[red]$ or $Count_{blue}.out > C^*_{rb}[blue]$ holds eventually because we assume global fairness. At that time, $C^*_{rb}[red]$ or $C^*_{rb}[blue]$ increases by one and hence $phase$ increases by one.

- Case that lines 7 to 9 are executed at $C \rightarrow C'$. In this case, the base station changes sets of red and $blue$ agents. At that time, the base station initializes its own variables of counting algorithms $Count_{red}$ and $Count_{blue}$. Since the counting algorithms work from arbitrary initial states of agents, the base station can correctly execute $Count_{red}$ and $Count_{blue}$ from the beginning under global fairness. Similarly to the first case, from Lemma 1, either $Count_{red}.out > C^*_{rb}[red]$ or $Count_{blue}.out > C^*_{rb}[blue]$ holds eventually. Then, $phase$ increases by one.

Lastly, we consider the transition $C \rightarrow C'$ such that $phase$ increases by one and $phase = n$ holds at C'. From $phase = n$, $C^*_{rb}[red] + C^*_{rb}[blue] = n = \#red + \#blue$ holds, and consequently $C^*_{rb}[red] = \#red$ and $C^*_{rb}[blue] = \#blue$ hold from Lemma 2. This implies that $Count_{red}.out$ and $Count_{blue}.out$ never exceed $C^*_{rb}[red]$ and $C^*_{rb}[blue]$ after that, respectively. Therefore, $C^*_{rb}[red]$ and $C^*_{rb}[blue]$ are never updated and consequently agents never change their colors any more. Since

$|\#red - \#blue| = |C^*_{rb}[red] - C^*_{rb}[blue]| \leq 1$ holds from Lemma 2, we have the theorem. □

3.1.4 Lower Bound with Arbitrary Initial States

Now, we show that four states are necessary to construct an asymmetric protocol with arbitrary initial states.

Theorem **5.** *In the model with an initialized base station, no asymmetric protocol with three states and arbitrary initial states solves the 2-partition problem with size difference $x \geq 1$ under global fairness.*

Proof. For contradiction, we assume that such a protocol Alg exists. Without loss of generality, we assume that the state set of agents is $Q_p = \{s_1, s_2, s_3\}$, $\gamma(s_1) = \gamma(s_2) = red$, and $\gamma(s_3) = blue$. We consider the following three cases.

First, we consider population $V = \{a_0, \dots, a_n\}$ of a single base station and n agents such that n is at least $x + 4$. Assume that a_0 is a base station. Since each agent has an arbitrary initial state, we consider an initial configuration C_0 such that $s(a_i) = s_3$ holds for any $i (1 \leq i \leq n)$. Note that the base station a_0 has a designated initial state at C_0. From the definition of Alg, for any globally fair execution $\Xi = C_0, C_1, \dots$, there exists a stable configuration C_t. Hence, both the number of *red* agents and the number of *blue* agents are at least 2 at C_t. After C_t, the color of agent a_i (i.e., $\gamma(s(a_i))$) never changes for any $a_i (1 \leq i \leq n)$ even if the base station and agents interact in any order.

Next, consider population $V' = \{a'_0, \dots, a'_{n+2x+1}\}$ of a single base station and $n + 2x + 1$ agents. Assume that agent a'_0 is a base station. We consider an initial configuration C'_0 such that $s(a'_i) = s_3$ holds for any i $(1 \leq i \leq n + 2x + 1)$. From this initial configuration, we define an execution $\Xi' = C'_0, C'_1, \dots, C'_t, \dots$ using the execution Ξ as follows.

- For $0 \leq u < t$, when a_i and a_j interact at $C_u \rightarrow C_{u+1}$, a'_i and a'_j interact at $C'_u \rightarrow C'_{u+1}$.
- For $t \leq u$, an interaction occurs at $C'_u \rightarrow C'_{u+1}$ so that Ξ' satisfies global fairness.

Since the base station and agents $a_1, \ldots, a_n$ change their states similarly to Ξ from C'_0 to C'_t, $s(a'_i) = s(a_i)$ holds for $1 \le i \le n$. Hence, the number of *blue* agents is $x + 1$ or more than the number of *red* agents at C'_t. Consequently C'_t is not a stable configuration. This implies that there exists a stable configuration $C'_{t'}$ for some $t' > t$. Clearly at least one *blue* agent becomes *red* from C'_t to $C'_{t'}$. That is, for some configuration $C'_{t^*}(t \le t^* < t')$, an agent in state s_3 transitions to state s_1 or s_2 at $C_{t^*} \to C_{t^*+1}$. Assume that t^* is the smallest value that satisfies the condition.

Finally, for V we define an execution $\Xi'' = C''_0, C''_1, \ldots$ using executions Ξ and Ξ' as follows.

- Let $C''_u = C_u$ for $0 \le u \le t$. That is, Ξ'' reaches stable configuration C''_t in similarly to Ξ.

- For $t \le u \le t^*$, we define an execution so that interaction at $C'_u \to C'_{u+1}$ also occurs at $C''_u \to C''_{u+1}$. Concretely, when a'_i and a'_j interact at $C'_u \to C'_{u+1}$, we define $a_{i'}$ and $a_{j'}$ as follows and they interact at $C''_u \to C''_{u+1}$. If $i \le n$, let $i' = i$. Otherwise, since $s(a'_i) = s_3$ holds at C'_u (because no agent in state s_3 changes its state from C'_t to C'_{t^*}), choose $i'(\le n)$ such that both $s(a_{i'}) = s_3$ and $i' \ne j$ hold. Similarly, if $j \le n$, let $j' = j$. Otherwise choose $j'(\le n)$ such that both $s(a_{j'}) = s_3$ and $j' \ne i'$ hold. Such i' and j' exist since at least two agents in state s_3 exist (because $n \ge x + 4$ holds and no agent in state s_3 changes its state from C'_t to C'_{t^*}).

- After $t^* < u$, an interaction occurs at $C''_u \to C''_{u+1}$ so that Ξ'' satisfies global fairness.

Clearly, for $t \le u \le t^*$ and $i \le n$, $s(a_i)$ at C''_u is equal to $s(a'_i)$ at C'_u. Additionally, at $C''_{t^*} \to C''_{t^*+1}$, an agent in state s_3 transitions to s_1 or s_2 as well as $C'_{t^*} \to C'_{t^*+1}$. This means that the agent changes its color at $C''_{t^*} \to C''_{t^*+1}$. That is, an agent changes its color after stable configuration C''_t in globally fair execution Ξ''. This is a contradiction. □

Remark. Note that, in the proof of Theorem 5, we consider a protocol with $Q_p = \{s_1, s_2, s_3\}$, $\gamma(s_1) = \gamma(s_2) = \mathit{red}$, and $\gamma(s_3) = \mathit{blue}$, and assume that every agent is in state s_3 at the initial configuration of Ξ, Ξ', and Ξ''. This means, even if we consider a protocol with three states and designated initial states, there exists no protocol such that the designated initial state does not have the same color as any other state. This fact holds even if the number of states is larger than three.

On the other hand, Section 3.1.1 gives a protocol with three states and designated initial states. In the protocol, the state set of agents is $Q_p = \{initial, red, blue\}$, we set $\gamma(initial) = \gamma(red) = red$ and $\gamma(blue) = blue$, and the designated initial state is *initial*. This implies that there exists a protocol if the designated initial state (i.e., *initial*) has the same color as one of other states (i.e., *red*).

3.1.5 Impossibility with Non-initialized Base Station and Arbitrary Initial States

In this section, we show that the problem cannot be solved with a non-initialized base station and arbitrary initial states. This result implies that, in the model with no base station, there is no protocol for uniform 2-partition problem with arbitrary initial states under global fairness.

Theorem **6.** *In the model with non-initialized base station, no protocol with arbitrary initial states solves the 2-partition problem with size difference $x \geq 1$ under global fairness.*

Proof. By way of contradiction, we assume that such a protocol Alg exists. Moreover, we assume that n is at least $x + 4$. We consider the following two cases.

First, we consider population $V = \{a_0, \dots, a_n\}$ of n agents and a non-initialized base station, where a_0 is the base station. For V, we consider an execution $\Xi = C_0, C_1, \cdots$ of Alg. From the definition of Alg, there exists a stable configuration C_t. Hence, both the number of *red* agents and the number of *blue* agents are at least 2 at C_t. By the definition of a stable configuration, the color of agent a_i (i.e., $\gamma(s(a_i))$) never changes for any a_i ($1 \leq i \leq n$) after C_t, even if agents interact in any order.

Next, we consider population $V' = \{a'_0\} \cup \{a'_i | \gamma(s(a_i, C_t)) = red\}$, where a'_0 is the base station. For V', we consider an execution $\Xi' = C'_0, C'_1, \cdots$ of Alg from the initial configuration C'_0 such that $s(a'_i, C'_0) = s(a_i, C_t)$ holds for any $a'_i \in V'$. Note that, because we assume a non-initialized base station, even the base station can have $s(a_0, C_t)$ as its initial state. Because all agents are *red* at C'_0, some agents must change their colors to reach a stable configuration. This implies that, after C_t in execution Ξ, agents change their colors if they interact similarly to Ξ'. This is a contradiction. $\square$

3.2 Uniform 2-partition Protocols with No Base Station

In this section, we consider the uniform 2-partition problem under the assumption of no base station. That is, all agents are identical.

3.2.1 Upper Bound for Asymmetric protocols with Designated Initial States

First, we consider asymmetric protocols in this case. Since three states are necessary in the model with a base station from Theorem 3, three states are also necessary in the model with no base station. In addition, Delporte-Gallet et al. [28] gives a protocol with three states. This implies that three states are necessary and sufficient in this case.

Here, we briefly explain the protocol proposed in [28]. In this protocol, the state set of agents is $Q_p = \{initial, red, blue\}$, and we set $\gamma(initial) = \gamma(red) = red$ and $\gamma(blue) = blue$. The designated initial state of all agents is *initial*. The protocol consists of a single asymmetric transition $(initial, initial) \to (red, blue)$. In this protocol, when two agents in state *initial* interact, one agent transitions to *red* and the other transitions to *blue*. This implies that the number of agents in state *red* is always the same as the number of agents in state *blue*. Eventually all agents (possibly except one agent) transition to state *red* or *blue*. From $\gamma(initial) = red$, the difference in the numbers of *red* and *blue* agents is at most one. Note that the protocol works correctly if every pair of agents interacts once.

Theorem **7** ([28])**.** *In the model with no base station, there exists an asymmetric protocol with three states and designated initial states that solves the uniform 2-partition problem under global fairness.*

Remark. This protocol works even under weak fairness. Even under weak fairness, if there are at least two agents in state *initial*, eventually an interaction between agents in state *initial* occurs. Hence, eventually, all agents (possibly except one agent) transition to either *red* or *blue*. From this fact, we can observe that the protocol solves the problem under weak fairness.

Theorem **8** ([28])**.** *In the model with no base station, there exists an asymmetric protocol with three states and designated initial states that solves the uniform 2-partition problem under weak fairness.*

3.2.2 Upper Bound for Symmetric Protocols with Designated Initial States

Next, we consider symmetric protocols in this case. For this setting, we show a protocol with four states and impossibility with three states. These results show that, in this case, four states are necessary and sufficient to construct a symmetric protocol under global fairness.

We can easily obtain a symmetric protocol with four states by a scheme proposed in [22]. The scheme transforms an asymmetric protocol with α states to a symmetric protocol with at most 2α states. By applying the scheme to an asymmetric protocol in Section 3.2.1 and deleting unnecessary states, we can obtain a symmetric protocol with four states.

For self-containment, we briefly explain the obtained protocol. Since no symmetric protocol solves the uniform 2-partition problem for a population of two agents, we assume that a population consists of at least three agents. In this protocol, the state set of agents is $Q_p = \{initial, initial', red, blue\}$, and we set $\gamma(initial) = \gamma(initial') = \gamma(red) = red$ and $\gamma(blue) = blue$. The designated initial state of all agents is *initial*. The protocol consists of the following seven transitions.

1. $(initial, initial) \rightarrow (initial', initial')$
2. $(initial', initial') \rightarrow (initial, initial)$
3. $(initial, initial') \rightarrow (red, blue)$
4. $(initial, red) \rightarrow (initial', red)$
5. $(initial, blue) \rightarrow (initial', blue)$

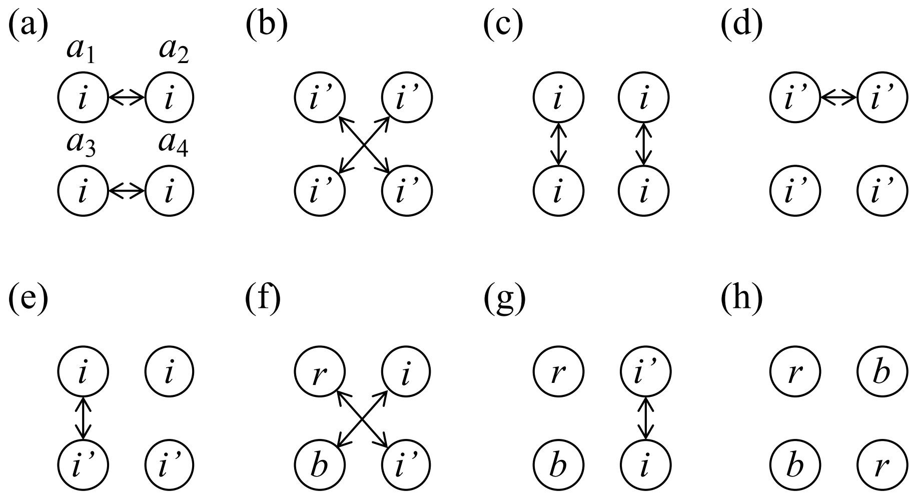


Figure 1. An example execution of the protocol. Symbols i, i', r, and b represent states *initial*, *initial*$'$, *red*, and *blue*, respectively. Arrows represent interactions of agents.

6. $(initial', red) \to (initial, red)$
7. $(initial', blue) \to (initial, blue)$

The main behavior of the protocol is similar to the previous asymmetric protocol with three states. However, since asymmetric transition $(initial, initial) \to (red, blue)$ is not allowed in symmetric protocols, the scheme in [22] introduces a new state *initial*$'$. Transition 3 implies that, when agents in states *initial* and *initial*$'$ interact, they become *red* and *blue*, respectively. In addition, agents in states *initial* and *initial*$'$ become *initial*$'$ and *initial* respectively when they interact with some agents (except for interaction between two agents in states *initial* and *initial*$'$). From global fairness, if at least two agents are in state *initial* or *initial*$'$, some two agents eventually enter states *initial* and *initial*$'$. After that, if the two agents interact, they enter states *red* and *blue*. Note that, since $\gamma(initial) = \gamma(initial') = red$ holds, the protocol solves the problem even if the number of agents is odd and an agent with state *initial* or *initial*$'$ remains forever.

Figure 1 shows an example execution of the protocol for a population of four agents. Initially all agents are in state *initial* (Fig. 1 (a)). After interactions (a_1, a_2) and (a_3, a_4), all agents enter state *initial′* (Fig. 1 (b)). Similarly, after interactions (a_1, a_4), (a_2, a_3), (a_1, a_3), and (a_2, a_4), all agents have the same state (Fig. 1 (c) and (d)). If these interactions happen infinite times, all agents keep the same state and never achieve the uniform 2-partition. However, under the global fairness, such interactions do not happen infinite times. This is because, if some configuration C occurs infinite times, every configuration reachable from C should occur. This implies that eventually interactions (a_1, a_2) and (a_1, a_3) happen in this order from a configuration in Fig. 1 (d). Then, a_1 and a_3 enter states *red* and *blue*, respectively (Fig. 1 (e) and (f)). After that, in a similar way, the remaining agents eventually enter *red* and *blue* like Fig. 1 (g) and (h).

Theorem 7 and correctness of the scheme in [22] derives the following theorem.

Theorem **9.** *In the model with no base station, when the number of agents is at least three, there exists a symmetric protocol with four states and designated initial states that solves the uniform 2-partition problem under global fairness.*

3.2.3 Lower Bound for Symmetric Protocols

Here we show that four states are necessary to construct a symmetric protocol with no base station.

Theorem **10.** *In the model with no base station, no symmetric protocol with three states and designated initial states solves the 2-partition problem with size difference $x \geq 1$ under global fairness.*

Proof. For contradiction, we assume that such a protocol *Alg* exists. Without loss of generality, we assume that the state set of agents is $Q_p = \{s_1, s_2, s_3\}$, $\gamma(s_1) = \gamma(s_2) = red$, and $\gamma(s_3) = blue$. Consider population $V = \{a_1, \ldots, a_n\}$ of n agents such that n at least $x + 6$. First, assume that the designated initial state of all agents is s_3. Clearly, *Alg* has transition $(s_3, s_3) \to (s_i, s_i)$ for some $i \neq 3$. However, since at least 3 agents in state s_3 exist at a stable configuration, some agents change their states from s_3 to s_i at the stable configuration. This implies that agents change their colors. Therefore, a designated initial state is s_1 or s_2.

Next, assume that the designated initial state of all agents is s_1 (Case of s_2 is the same). Since Alg is a symmetric protocol and all the initial states are s_1, Alg includes $(s_1, s_1) \to (s_i, s_i)$ for some $i \neq 1$. This implies that all agents can transition to state s_i from the initial configuration. Hence, Alg also includes $(s_i, s_i) \to (s_j, s_j)$ for some $j \neq i$. When $i = 3$, since at least 3 *blue* agents exist at a stable configuration and they are in state s_3, the *blue* agents become *red* by transition $(s_3, s_3) \to (s_j, s_j)$. Therefore, $i \neq 3$ holds.

The remaining case is $i = 2$. If $j = 3$, that is, Alg includes $(s_2, s_2) \to (s_3, s_3)$, *red* agents (i.e., agents in state s_1 or s_2) change their colors at a stable configuration because Alg includes $(s_1, s_1) \to (s_2, s_2)$ and $(s_2, s_2) \to (s_3, s_3)$. This implies $j = 1$. In this case, Alg includes $(s_2, s_2) \to (s_1, s_1)$. Since some agents should transition to state s_3, Alg includes $(s_1, s_2) \to (s_k, s_l)$ such that k or l is 3. At a stable configuration, there exist at least 3 agents with states s_1 or s_2. However, these agents can transition to state s_3 from transitions $(s_1, s_2) \to (s_k, s_l)$, $(s_2, s_2) \to (s_1, s_1)$, and $(s_1, s_1) \to (s_2, s_2)$. This is a contradiction. □

4. Uniform 2-partition under Weak Fairness on Complete Graphs

In this section, we show the upper and lower bounds of the uniform 2-partition on complete graphs under weak fairness. Concretely, for each combination of assumptions (base station, symmetry, and initial states), we completely clarify solvability of the uniform 2-partition problem. Moreover, if solvable, we show the tight upper and lower bounds on the number of states. More concretely, with arbitrary initial states and initialized base station, we prove that P states are necessary and sufficient to construct asymmetric protocols, and that $P + 1$ states are necessary and sufficient to construct symmetric protocols, where P is the known upper bound of the number of agents. Recall that proposed protocols in the setting are available for the uniform k-partition. Moreover, we show the impossibility of the uniform 2-partition for symmetric protocols with no base station and designated initial states. Note that, the impossibility results under global fairness remain valid even under weak fairness, and the 3-state protocols introduced in Section 3.1.1 and 3.2.1 work even under weak fairness.

As in Section 3, since we consider the uniform 2-partition in this section, function γ is defined as $\gamma : Q_p \to \{color_1, color_2\}$, and we assign colors *red* and *blue* to $color_1$ and $color_2$, respectively. Note that we define γ as function $\gamma : Q_p \to \{red, blue\}$ that maps a state of a non base station agent to *red* or *blue*. We say agent $a \in V_p$ is *red* (resp., *blue*) if $\gamma(s(a)) = red$ (resp., $\gamma(s(a)) = blue$) holds.

4.1 Lower Bounds for Initialized Base Station

In this section, we provide the lower bounds for asymmetric and symmetric protocols and the uniform 2-partition problem by proving impossibility. Clearly these lower bounds can be applied to the uniform k-partition problem for $k > 2$. Recall that, for an initialized base station, we assume weak fairness and arbitrary initial states.

4.1.1 Common Properties of Asymmetric and Symmetric Protocols

First, we present the basic properties that hold for both asymmetric and symmetric protocols. Let *Alg* be an arbitrary protocol that solves the uniform 2-partition. Recall that P is the known upper bound of the number of agents. Hence, *Alg* must solve the uniform 2-partition when the actual number of agents is at most P. In the remainder of this subsection, we consider the case where the actual number of agents is $P - 2$.

Lemma 3 shows that, in any execution for $P - 2$ agents, eventually all agents continue to maintain different states.

Lemma **3.** *In any weakly-fair execution $\Xi = C_0, C_1, C_2, \ldots$ of Alg with $P - 2$ agents and an initialized base station, there exists a configuration C_h such that 1) C_h is a stable configuration, and, 2) all agents have different states at $C_{h'}$ for any $h' \geq h$.*

Proof. First, we consider an arbitrary population $V = \{a_0, a_1, \ldots, a_{P-2}\}$ of $P - 2$ agents and a single base station, where a_0 is the base station. Let $\Xi = C_0, C_1, \ldots, C_t, \ldots$ be a weakly-fair execution of *Alg*, where C_t is a stable configuration. We assume, by way of contradiction, that two agents have the same state for infinitely many configurations after C_t. Since the number of agents is finite and the number of states is finite, there exist y, a_p, and $a_{p'}$ such that configurations that satisfy $y = s(a_p) = s(a_{p'})$ appear infinitely often after C_t.

Next, we consider population $V' = \{a'_0, \ldots, a'_P\}$ of P agents and a single base station, where a'_0 is a base station. We consider an initial configuration C'_0 such that the initial states of $a'_0, \ldots, a'_P$ are $s(a_0, C_0), \ldots, s(a_{P-2}, C_0)$, y, y, respectively. For V', we define an execution $\Xi' = C'_0, C'_1, \cdots, C'_t, \cdots$ using execution Ξ as follows:

- For $0 \le u \le t-1$, when a_i interacts with a_j at $C_u \to C_{u+1}$, a'_i interacts with a'_j at $C'_u \to C'_{u+1}$.

Clearly, $s(a'_i, C'_t) = s(a_i, C_t)$ holds for any i $(0 \le i \le P-2)$. Because $s(a'_P, C'_t) = s(a'_{P-1}, C'_t) = y$ holds, the difference between the numbers of *red* and *blue* agents remains two, and consequently, C'_t is not a stable configuration.

After C'_t, we define an execution as follows: This definition aims to make $P-2$ agents behave similarly to Ξ and two agents keep state y.

- Until $y = s(a'_p) = s(a'_{p'})$ holds, if a_i interacts with a_j at $C_u \to C_{u+1}$, a'_i interacts with a'_j at $C'_u \to C'_{u+1}$. Note that, because configurations that satisfy $y = s(a_p) = s(a_{p'})$ appear infinitely often after C_t, eventually, $y = s(a'_p) = s(a'_{p'})$ holds by behaving similarly to Ξ.

- To define the remainder of Ξ, we use procedure $Proc(q, q')$, which uses two indices q and q', and can be applied to a configuration that satisfies $y = s(a'_p) = s(a'_{p'}) = s(a'_{P-1}) = s(a'_P)$. Let $V(q, q') = (V' \backslash \{a'_p, a'_{p'}, a'_{P-1}, a'_P\}) \cup \{a'_q, a'_{q'}\}$. $Proc(q, q')$ creates a sub-execution similar to Ξ by making a'_q and $a'_{q'}$ join interactions instead of a'_p and $a'_{p'}$, and guarantees that, for any ordered pair (a^*_1, a^*_2) in $V(q, q') \times V(q, q')$, a^*_1 interacts with a^*_2 at least once, and the last configuration also satisfies $y = s(a'_p) = s(a'_{p'}) = s(a'_{P-1}) = s(a'_P)$. The formal definition of $Proc(q, q')$ is as follows:
  - When a_i interacts with a_j at $C_u \to C_{u+1}$, a'_i interacts with a'_j at $C'_u \to C'_{u+1}$ if $i, j \notin \{p, p'\}$.
  - If $i = p$ or $j = p$ holds, a'_q joins the interaction instead of a'_p.
  - If $i = p'$ or $j = p'$ holds, $a'_{q'}$ joins the interaction instead of $a'_{p'}$.

– Procedure $Proc(q, q')$ continues these behaviors until, for any ordered pair (a_1^*, a_2^*) in $V(q, q') \times V(q, q')$, a_1^* interacts with a_2^* at least once and satisfies $s(a'_q) = s(a'_{q'}) = y$.

Using $Proc(q, q')$, we define the remainder of Ξ' to satisfy weak fairness as follows: Repeat $Proc(p, p')$, $Proc(p, P-1)$, $Proc(p, P)$, $Proc(P-1, p')$, $Proc(P, p')$, and $Proc(P, P-1)$. Note that, because a'_q and $a'_{q'}$ cannot interact with agents in $\{a'_p, a'_{p'}, a'_{P-1}, a'_P\} \setminus \{a'_q, a'_{q'}\}$ in $Proc(q, q')$ (e.g., a'_P cannot interact with a'_{P-1} in $Proc(p, P)$), the above six $Proc$s are necessary to create interactions for each combination of a'_p, $a'_{p'}$, a'_{P-1}, and a'_P.

Clearly, Ξ' makes $P-2$ agents behave similarly to Ξ and two agents keep state y. Hence, Ξ' never converges to a stable configuration. Since Ξ' is weakly-fair, this is a contradiction. □

In the next lemma, we prove that there exists a configuration C such that, in any configuration reachable from C, all agents have different states. In addition, we also show that the system reaches C in some execution.

Definition **1.** *Configuration C is strongly-stable if 1) C is stable, and, 2) for any configuration C' with $C \xrightarrow{*} C'$, all agents have different states at C'.*

Lemma **4.** *Let P be an arbitrary even integer. For any P-state uniform 2-partition protocol Alg, when the number of agents other than the base station is $P-2$, there exists an execution of Alg that includes a strongly stable configuration.*

Proof. By way of contradiction, we assume that such an execution does not exist. First, we consider an arbitrary weakly-fair execution $\Xi = C_0, C_1, C_2, \ldots$ of an arbitrary P-state protocol Alg. By Lemma 3, after some configuration C_t, all agents have different states. From the assumption, C_t is not a strongly-stable configuration; that is, there exists a configuration reachable from C_t such that two agents have the same state. Hence, we can construct another weakly-fair execution $\Xi' = C'_0, C'_1, C'_2, \ldots, C'_t, \ldots, C'_u, \ldots$ of Alg as follows:

- For $0 \le i \le t$, C'_i is equal to C_i.

- After C'_t, the population moves to C'_u such that two agents have the same state.
- After C'_u, agents continue to interact so that Ξ' satisfies weak fairness.

By Lemma 3, since Ξ' is weakly-fair, all agents have different states after some configuration $C'_{t'}$. Hence, in a similar manner to Ξ', we can construct another weakly-fair execution $\Xi'' = C''_0, C''_1, \ldots, C''_{t'}, \ldots, C''_{u'}, \ldots$ that satisfies the following conditions:

- For $0 \leq i \leq t'$, C''_i is equal to C'_i.
- After $C''_{t'}$, the population moves to $C''_{u'}$ such that two agents have the same state.
- After C''_u, agents continue to interact so that Ξ'' satisfies weak fairness.

By repeating this construction, we can construct a weakly-fair execution such that two agents have the same state infinitely often. From Lemma 3, this is a contradiction. □

4.1.2 Lower Bound for Asymmetric Protocols

We show that the lower bound of the number of states for symmetric protocols is P. Recall that, when agents a and b interact as an initiator and responder, respectively, we say a interacts with b.

Theorem **11.** *In the model with an initialized base station, there is no asymmetric protocol that solves the uniform 2-partition problem with $P-1$ states from arbitrary initial states under weak fairness, if P is an even integer.*

To prove the theorem by contradiction, we assume that protocol Alg_{asym} exists. Let $Q_p = \{s_1, s_2, \ldots, s_{P-1}\}$ be a state set of agents other than the base station. Let $Q_{blue} = \{s \in Q_p \mid \gamma(s) = blue\}$ be a set of blue states and $Q_{red} = \{s \in Q_p \mid \gamma(s) = red\}$ be a set of red states. Without loss of generality, we assume that $|Q_{blue}| < |Q_{red}|$ holds. Recall that Lemmas 3 and 4 can be applied to both symmetric and asymmetric algorithms. Hence, the properties of the lemmas hold even in Alg_{asym}. In this proof, based on the properties, we construct an execution of P agents such that the base station does not recognize the difference between the execution of $P-2$ agents. We obtain a contradiction by proving that this execution does not achieve the uniform 2-partition.

By Lemma 3, clearly Alg_{asym} requires $P/2-1$ *blue* states and $P/2-1$ *red* states. Consequently, we have the following two corollaries.

Corollary **1.** *$|Q_{blue}| = P/2-1$ and $|Q_{red}| = P/2$ hold.*

Corollary **2.** *For any weakly-fair execution of Alg_{asym} with $P-2$ agents and an initialized base station, any strongly-stable configuration includes all states in Q_{blue}.*

To prove the main theorem, we focus on the following weakly-fair execution of Alg_{asym} with $P-2$ agents.

Definition **2.** *We consider a population $V = \{a_0, a_1, \ldots, a_{P-2}\}$ of $P-2$ agents and an initialized base station, where a_0 is the base station. We define $\Xi_\alpha = C_0, C_1, C_2, \ldots$ as a weakly-fair execution of Alg_{asym} for population V that satisfies the following conditions.*

- *Ξ_α includes a strongly-stable configuration C_t; and*
- *for any $u \geq 0$, when agent x interacts with agent y at $C_{t+2u} \rightarrow C_{t+2u+1}$, agent y interacts with agent x at $C_{t+2u+1} \rightarrow C_{t+2(u+1)}$ if $s(x, C_{t+2u+1}) = s(y, C_{t+2u})$ and $s(y, C_{t+2u+1}) = s(x, C_{t+2u})$ hold; otherwise, agent x also interacts with agent y at $C_{t+2u+1} \rightarrow C_{t+2(u+1)}$.*

Note that, in Ξ_α, the system reaches a strongly stable configuration C_t (this is possible from Lemma 4). Moreover, in Ξ_α, after C_t, any agent in Q_{blue} has the same state in C_{t+2}, C_{t+4}, C_{t+6}, ... We show why this is true later.

Definition **3.** *We define Q_t as a set of states that appear after C_t in Ξ_α.*

Note that, because C_t is strongly stable, Q_t includes at least $P-2$ states. This implies that Q_t includes all states in Q_p or does not include one state in Q_p. From Corollary 2, $Q_{blue} \subset Q_t$ holds. The following lemmas provide the key properties of Alg_{asym} required to prove Theorem 11. We present proofs of these lemmas later.

Lemma **5.** *For any distinct states p and q such that $p \in Q_{blue}$ and $q \in Q_t$ hold, transition rules $(p,q) \to (p',q')$ and $(q,p) \to (q'',p'')$ satisfy the following conditions:*

- *If $q \in Q_{red}$ or $q \in Q_b$ (i.e., q is a state of the base station) holds, $p'' = p' = p$ holds.*
- *If $q \in Q_{blue}$ holds, either $(p',q') = (p,q)$ or $(p',q') = (q,p)$ holds. Similarly, if $q \in Q_{blue}$ holds, either $(q'',p'') = (p,q)$ or $(q'',p'') = (q,p)$ holds.*

Lemma 5 indicates that, after a strongly stable configuration in Ξ_α, any agent with a state in Q_{blue} never changes its state as a result of two consecutive interactions with the same pair of agents. This is because, by the definition of a strongly stable configuration, any agent with state p can interact only with an agent with state $q \neq p$ after a strongly stable configuration.

Lemma **6.** *A non-empty state set $Q^* \subseteq Q_{blue}$ exists that satisfies the following conditions:*

- *For any state $p \in Q^*$, transition rule $(p,p) \to (p',q')$ satisfies $p' \in Q^*$ and $q' \in Q^*$.*
- *We assume that, in a configuration C, there exists a subset of agents V^* such that all agents in V^* have states in Q^*, and $|V^*| \geq |Q^*|+1$ holds. In this case, for any agent $a_r \in V^*$ and any state $q \in Q^*$, there exists an execution segment such that (1) the execution segment starts from C; (2) a_r has state q at the last configuration; (3) only agents in V^* join interactions; and (4) all agents in V^* have states in Q^* at the last configuration.*

Lemma 6 indicates that, if at least $|Q^*| + 1$ agents have states in Q^*, we can make an arbitrary agent with a state in Q^* transition to an arbitrary state in Q^*. Using these lemmas, we state the theorem by constructing a weakly fair execution of Alg_{asym} with P agents that cannot be distinguished from execution Ξ_α.

Proof of Theorem 11

We consider a population $V' = \{a'_0, \ldots, a'_P\}$ of P agents and an initialized base station, where a'_0 is the base station. Let C'_0 be an initial configuration such that the initial states of a'_0, …, a'_P are $s(a_0, C_0)$, …, $s(a_{P-2}, C_0)$, s^*, s^*, where s^* is a state in Q^*. For V', we construct an execution $\Xi_\beta = C'_0$, C'_1, …, C'_t, … using execution Ξ_α as follows:

- For $0 \le u \le t-1$, when a_i interacts with a_j at $C_u \to C_{u+1}$ in Ξ_α, a'_i interacts with a'_j at $C'_u \to C'_{u+1}$ in Ξ_β.

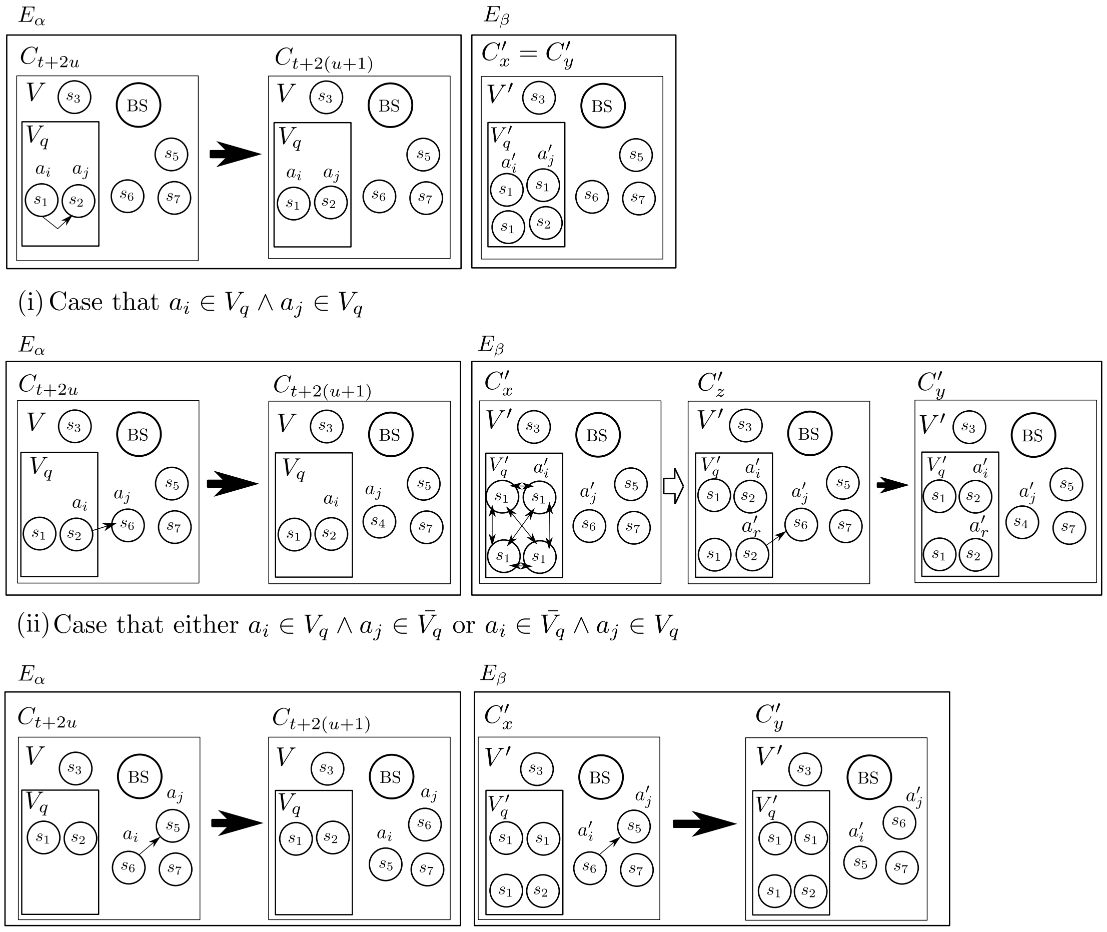
Clearly, $s(a'_i, C'_t) = s(a_i, C_t)$ holds for any i ($0 \le i \le P-2$). Since $s(a'_{P-1}, C'_t) = s(a'_P, C'_t) = s^*$ holds, the difference between the numbers of *red* and *blue* agents remains two and consequently C'_t is not a stable configuration.

To construct the remainder of Ξ_β, first we consider the characteristics of C'_t. Let $V_q \subseteq V$ be a set of agents that have states in Q^* at C_t, and let $\bar{V_q} = V - V_q$. Since all agents have different states and all states in Q_{blue} appear in C_t by Corollary 2, we have $|V_q| = |Q^*|$ from $Q^* \subseteq Q_{blue}$. Let $V'_q \subseteq V'$ be a set of agents that have states in Q^* at C'_t, and let $\bar{V}'_q = V' - V'_q$. Note that, for $i \le P-2$, $a_i \in V_q$ holds if and only if $a'_i \in V'_q$ holds. Since a'_{P-1} and a'_P are also in V'_q, we have $|V'_q| = |Q^*| + 2$. In the following, we construct the remainder of execution Ξ_β that includes infinitely many configurations similar to Ξ_α. We define similarity of configurations in Ξ_β and Ξ_α as follows.

Definition **4.** *We say configuration C'_u ($u \ge t$) in Ξ_β is similar to C_v ($v \ge t$) in Ξ_α if the following conditions hold:*

- *For any agent $a_i \in V_q$, $s(a_i, C_v) \in Q^*$ holds.*
- *For any agent $a'_i \in V'_q$, $s(a'_i, C'_u) \in Q^*$ holds.*
- *For any agent $a'_i \in \bar{V}'_q$ (i.e., $a_i \in \bar{V}_q$), $s(a'_i, C'_u) = s(a_i, C_v)$ holds.*

We focus on an execution segment $e = C_{t+2u}, C_{t+2u+1}, C_{t+2(u+1)}$ of Ξ_α for any $u \ge 0$. We consider a configuration C'_x of V' such that C'_x is similar to C_{t+2u}. Now, we explain how to construct an execution segment $e' = C'_x, \ldots, C'_y$ of Ξ_β that guarantees that C'_y is similar to $C_{t+2(u+1)}$. Since C'_t is similar to C_t, we can repeatedly apply this construction and construct an infinite execution Ξ_β. As a result, for any $u \ge 0$, Ξ_β includes a configuration C' that is similar to C_{t+2u}. Since C' includes $P-1$ *red* agents and $P+1$ *blue* agents, Ξ_β does not include a stable



(i) Case that $a_i \in V_q \wedge a_j \in V_q$

(ii) Case that either $a_i \in V_q \wedge a_j \in \bar{V}_q$ or $a_i \in \bar{V}_q \wedge a_j \in V_q$

(iii) Case that $a_i \in \bar{V}_q \wedge a_j \in \bar{V}_q$

Figure 2. Example of execution segment e' ($P = 8$, $Q^* = \{s_1, s_2\}$, $\gamma(s_1) = \gamma(s_2) = \gamma(s_3) = \mathit{blue}$, and $\gamma(s_4) = \gamma(s_5) = \gamma(s_6) = \gamma(s_7) = \mathit{red}$), where the thick black arrow represents two interactions between a_i and a_j (or a'_i and a'_j)

configuration. Note that Ξ_β is not necessarily weakly-fair, but later we explain how to construct a weakly-fair execution from Ξ_β.

We consider configuration C'_x that is similar to C_{t+2u}. We assume that, in Ξ_α, a_i interacts with a_j at $C_{t+2u} \to C_{t+2u+1}$. Recall that a_i and a_j also interact at $C_{t+2u+1} \to C_{t+2(u+1)}$. We construct execution segment e' as follows (Figure 2 shows an example of execution segment e' for each case):

- Case in which $a_i \in V_q \wedge a_j \in V_q$ holds: Because $s(a_i, C_{t+2u}) \in Q^* \subseteq Q_{blue}$ and $s(a_j, C_{t+2u}) \in Q^* \subseteq Q_{blue}$ hold, $s(a_i, C_{t+2(u+1)}) \in Q^*$ and $s(a_j, C_{t+2(u+1)}) \in Q^*$ also hold from Lemmas6 (the first condition) and 5. Because other agents do not change their states, C'_x is similar to $C_{t+2(u+1)}$. Hence, in this case, we consider that the constructed execution segment e' is empty.

- Case in which $a_i \in V_q \wedge a_j \in \bar{V}_q$ holds: In this case, $s(a'_i, C'_x) \in Q^*$ is not necessarily equal to $\sigma = s(a_i, C_{t+2u}) \in Q^*$. Hence, in the execution segment e', we first make some agent $a'_r \in V'_q$ enter state σ using interactions among agents in V'_q. By Lemma 6 (the second condition) and $|V'_q| = |Q^*| + 2$, such interactions exist and all agents in V'_q have states in Q^* after the interactions. Let C'_z be the resultant configuration. Clearly C'_z is similar to C_{t+2u} and $s(a'_r, C'_z) = s(a_i, C_{t+2u}) \wedge s(a'_j, C'_z) = s(a_j, C_{t+2u})$ holds. Then, a'_r and a'_j interact similarly to a_i and a_j; that is, first a'_r interacts with a'_j. Then, if a_i and a_j interact with a_j and a_i, respectively, in $C_{t+2u+1} \to C_{t+2(u+1)}$, a'_r and a'_j interact with a'_j and a'_r, respectively. We regard the resultant configuration as C'_y (i.e., the last configuration of the constructed execution segment e'). Clearly both $s(a'_r, C'_y) = s(a_i, C_{t+2(u+1)})$ and $s(a'_j, C'_y) = s(a_j, C_{t+2(u+1)})$ hold. Because C'_z is similar to C_{t+2u} and $s(a'_j, C'_y) = s(a_j, C_{t+2(u+1)})$, it is sufficient to prove $s(a'_r, C'_y) \in Q^*$ to guarantee that C'_y is similar to $C_{t+2(u+1)}$. From $s(a'_r, C'_z) \in Q^* \subseteq Q_{blue}$, $s(a'_r, C'_y) = s(a'_r, C'_z) \in Q^*$ holds by Lemma 5. Therefore, C'_y is similar to $C_{t+2(u+1)}$.

- Case in which $a_i \in \bar{V}_q \wedge a_j \in V_q$ holds: In this case, as in the previous case, clearly we can construct a configuration similar to $C_{t+2(u+1)}$.

- Case in which $a_i \in \bar{V}_q \wedge a_j \in \bar{V}_q$ holds: In this case, because $s(a'_i, C'_x) = s(a_i, C_{t+2u})$ and $s(a'_j, C'_x) = s(a_j, C_{t+2u})$ hold, a'_i and a'_j interact twice similarly to a_i and a_j. We regard the resultant configuration as C'_y (i.e., the last configuration of the constructed execution segment e'). Clearly, because a'_i and a'_j change their states similarly to a_i and a_j, C'_y is similar to $C_{t+2(u+1)}$.

We have constructed the infinite execution Ξ_β, but Ξ_β is not necessarily weakly fair. In the following, we construct a weakly fair execution Ξ_γ of population V' by slightly modifying Ξ_β. To guarantee that Ξ_γ is weakly fair, for any pair of agents (a'_i, a'_j), a'_i should interact with a'_j infinitely often in Ξ_γ. For a pair of agents (a'_i, a'_j) with $a'_i \in \bar{V}'_q$ and $a'_j \in \bar{V}'_q$, a'_i interacts with a'_j infinitely often in Ξ_β because Ξ_α is weakly fair and a'_i interacts with a'_j in Ξ_β when a_i interacts with a_j in Ξ_α. For a pair of agents (a'_i, a'_j) with $a'_i \in V'_q$ and $a'_j \in V'_q$, we can arbitrarily add interactions between them because, by Lemmas 6 (the first condition) and 5 (the second condition), they keep their states in Q^* and consequently do not influence the similarity of configurations. Hence, we consider the remaining pair (a'_i, a'_j); that is, either $a'_i \in V'_q \wedge a'_j \in \bar{V}'_q$ or $a'_i \in \bar{V}'_q \wedge a'_j \in V'_q$ holds. We show that any agent in V_q and $\bar{V}'_q$ can interact with any agent in $\bar{V}'_q$ and V_q, respectively, infinitely often without influencing the similarity of configurations. First, we consider the case in which $a'_i \in V'_q \wedge a'_j \in \bar{V}'_q$ holds. Because Ξ_α is weakly fair, an agent in V_q interacts with an agent in $\bar{V}'_q$ infinitely often in Ξ_α. Recall that these interactions correspond to interactions between a'_r and a'_j in Ξ_β, and a'_r can be arbitrarily selected from V'_q. For this reason, we can choose a'_r in a round-robin manner so that any agent in V'_q interacts with a'_j infinitely often. For example, when an agent in V_q and a_j first interact (after C_t), we choose an agent in V'_q as a'_r, and then in the next interaction of an agent in V_q and a_j, we choose another agent in V'_q as a'_r. Using this construction, any agent in V'_q can interact with a'_j infinitely often. Next, we consider the case in which $a'_i \in \bar{V}'_q \wedge a'_j \in V'_q$ holds. Clearly, as in the previous case, we can make any agent in $\bar{V}'_q$ interact with any agent in V'_q infinitely often. In this way, we can construct a weakly fair execution Ξ_γ similarly to Ξ_β. However, for any $u \geq 0$, Ξ_γ includes a configuration C'' that is similar to C_{t+2u}. Because C'' includes $P/2 - 1$ *red* agents and $P/2 + 1$ *blue* agents, Ξ_γ does not include a stable configuration. This is a contradiction, and thus we have the theorem. Moreover, Theorem 11 directly implies the following corollary.

Corollary **3.** *In the model with an initialized base station, there is no asymmetric protocol that solves the uniform partition problem with $P - 1$ states from arbitrary initial states under weak fairness if P is an even integer.*

Proofs of Lemmas 5 and 6

From now, we show the proofs of Lemmas 5 and 6. First, we prove Lemma 5.

Lemma **5.** *For any distinct states p and q such that $p \in Q_{blue}$ and $q \in Q_t$ hold,* *transition rules* $(p, q) \to (p', q')$ *and* $(q, p) \to (q'', p'')$ *satisfy the following conditions:*

- *If* $q \in Q_{red}$ *or* $q \in Q_b$ *(i.e.,* q *is a state of the base station) holds,* $p'' = p' = p$ *holds.*

- *If* $q \in Q_{blue}$ *holds, either* $(p', q') = (p, q)$ *or* $(p', q') = (q, p)$ *holds. Similarly, if* $q \in Q_{blue}$ *holds, either* $(q'', p'') = (p, q)$ *or* $(q'', p'') = (q, p)$ *holds.*

Proof. We consider execution Ξ_α defined in Definition 2. From the definition, each configuration C_u $(u \geq t)$ is strongly stable. From Corollary 2, each *blue* state exists in any strongly stable configuration. For this reason, p exists in any configuration C_u $(u \geq t)$. Additionally, since $q \in Q_t$ holds, there exists a configuration C_v $(v \geq t)$ such that state q exists at C_v. Let a_x and a_y be agents that have states p and q at C_v, respectively. We consider another execution Ξ'_α such that a_x and a_y interact at configuration C_v; that is, they change their states according to the transition rule $(p, q) \to (p', q')$ or $(q, p) \to (q'', p'')$. Let C'_v be the resultant configuration. Note that the transition does not change the colors of a_x and a_y because C_v is stable. Now, we consider the following two cases. First, we consider the case of $q \in Q_{red}$ or $q \in Q_b$. For contradiction, we assume that either $p' \neq p$ or $p'' \neq p$ holds. If $p' \neq p$ (resp., $p'' \neq p$) holds, we assume that $(p, q) \to (p', q')$ (resp., $(q, p) \to (q'', p'')$) occurs at $C_v \to C'_v$. Since $p', p'' \in Q_{blue}$ holds, p' and p'' exist in C_v by Corollary 2. Thus, since $p' \neq p$ (resp., $p'' \neq p$) holds, two agents have p' (resp., p'') at C'_v. This is a contradiction because C_v is strongly stable. Next, we consider the case of $q \in Q_{blue}$. For contradiction, we assume that, in C'_v, (1) a_x and a_y have p; (2) a_x and a_y have q; or (3) at least one of a_x and a_y has a state $r \notin \{p, q\}$. In the first two cases, two agents have the same state in C'_v. In the last case, since $r \in Q_{blue}$ holds, state r exists in C_v from Corollary 2, and consequently, two agents have r in C'_v. This is a contradiction because C_v is strongly stable. Therefore, the lemma holds. □

In the following, we prove Lemma 6. First, to prove the lemma, we provide some definitions.

Definition **5.** *For states* q *and* q'*, we say* $q \rightsquigarrow q'$ *if there exists a sequence of states* $q = q_0, q_1, \cdots, q_l = q'$ *such that, for any* $i (0 \leq i < l)$*, transition rule* $(q_i, q_i) \to (q_{i+1}, x_i)$ *or* $(q_i, q_i) \to (x_i, q_{i+1})$ *exists for some* x_i*.*

Definition **6.** *For states q and q', we say $q \overset{*}{\rightsquigarrow} q'$ if $x \rightsquigarrow q'$ holds for any x such that $q \rightsquigarrow x$ holds.*

Note that, in these definitions, we only consider interactions between agents with the same state. We say two agents are homonyms if they have the same state. Intuitively, $q \rightsquigarrow q'$ means that an agent with state q can transition to q' by only interacting with homonyms. Additionally, $q \overset{*}{\rightsquigarrow} q'$ means that, even if an agent with state q transitions to any state x by interactions with homonyms, it can still transition from x to q' by interactions with homonyms. By Corollary 1, without loss of generality, we assume that $Q_{blue} = \{s_1, s_2, \ldots, s_{P/2-1}\}$ and $Q_{red} = \{s_{P/2}, s_{P/2+1}, \ldots, /s_{P-1}\}$ hold.

Definition **7.** $Q_{bc} = \{p \in Q_{blue} \mid \exists q \in Q_{red} : p \rightsquigarrow q\}$.

Definition **8.** $Q_{nbc} = Q_{blue} \backslash Q_{bc}$.

Intuitively, $p \in Q_{bc}$ means that a *blue* agent with state p can become *red* by interactions with homonyms. Additionally, $p \in Q_{nbc}$ means that a *blue* agent with state p cannot become *red* by interactions with homonyms. The outline of the proof of Lemma 6 is as follows: First, we show that Q_{nbc} is not empty. This implies that, by the definition of Q_{nbc}, there exists a state p in Q_{nbc} such that $p \rightsquigarrow p$ holds. Then we show that there exists a state p^* in Q_{nbc} such that $p^* \overset{*}{\rightsquigarrow} p^*$ holds. If such a p^* does not exist, some state in Q_{nbc} can transition to a state not in Q_{nbc} by interactions with homonyms, which contradicts the definition of Q_{nbc}. The existence of p^* implies that there exists a state set $Q_{p*} \subseteq Q_{nbc}$ such that $Q_{p^*} = \{q \mid p^* \rightsquigarrow q\}$ holds. Finally, we show that this Q_{p*} satisfies the conditions of Lemma 6. Now we show that Q_{nbc} is not empty. To show this, we provide the following definition.

Definition **9.** *$DtR(s_i)$ is a function that satisfies the following property:*

- *If $s_i \in Q_{red}$ holds, $DtR(s_i) = 0$ holds.*
- *If $s_i \in Q_{bc}$ holds, $DtR(s_i) = \min\{DtR(s_j^1), DtR(s_j^2)\} + 1$ holds when transition rule $(s_i, s_i) \to (s_j^1, s_j^2)$ exists.*
- *If $s_i \in Q_{nbc}$ holds, $DtR(s_i) = \infty$ holds.*

Intuitively, $DtR(s_i)$ gives the minimum number of interactions to transition from s_i to a *red* state when allowing only interactions with homonyms.

Lemma **7.** $Q_{nbc} \neq \emptyset$.

Proof. The idea of the proof of the lemma is as follows: For contradiction, we assume $Q_{nbc} = \emptyset$. From the assumption, since $Q_{blue} = Q_{bc}$ holds, all states in Q_{blue} can transition to a state in Q_{red} by interactions with homonyms. Additionally, when homonyms with states in Q_{blue} interact, one of the agents transitions from s_i to s_j such that $DtR(s_i) > DtR(s_j)$ holds. When P is even, some *blue* agents have the same state in a stable configuration because the number of *blue* agents should be $P/2$ and $|Q_{blue}| = P/2 - 1$ holds. This implies that interactions with homonyms having states in Q_{blue} are always possible. Thus, by repeating the interactions, eventually, some agent transitions to s' such that $DtR(s') = 0$ holds (i.e., $s' \in Q_{red}$). This contradicts that the configuration is stable. Now we present the details of the proof. For contradiction, we assume $Q_{nbc} = \emptyset$. We consider a population $V' = \{a'_0, \dots, a'_P\}$ of P agents and an initialized base station, where a'_0 is the base station. Let $\Xi' = C'_0, C'_1, \dots, C'_t, \dots$ be a weakly fair execution of Alg_{asym}, where C'_t is a stable configuration. Without loss of generality, we assume $DtR(s_{P-1}) = \dots = DtR(s_{P/2}) = 0 < DtR(s_{P/2-1}) \leq \dots \leq DtR(s_1)$. Since $Q_{nbc} = \emptyset$ holds, $DtR(s_1) \neq \infty$ holds. From the definition of $DtR(s_i)$, if $DtR(s_i) > 0$ and $DtR(s_i) \neq \infty$ hold, an agent with state s_i transitions to s_j with $j > i$ by an interaction with homonyms. Using this property, we prove the following lemma.

Lemma **8.** *For $1 \leq i \leq P/2 - 1$, if $i + 1$ agents have states in $\{s_1, \dots, s_i\}$, one of these agents can transition to $s_h (h \geq i + 1)$ by interactions among these agents.*

Proof. We prove the lemma by induction on i. The base case is when two agents have state s_1. If they interact, one of them transitions to $s_h (h \geq 2)$. Thus, the lemma holds in this case. For the induction step, we assume that the lemma holds for $i = k - 1$ $(k \leq P/2 - 1)$, and prove that the lemma holds for $i = k$. To do this, we consider the scenario in which $k + 1$ agents have states in $\{s_1, \dots, s_k\}$. We consider three cases: (1) at least two agents have state s_k; (2) exactly one agent has state s_k; and (3) no agent has state s_k. First, we consider the case in which at least two agents have state s_k. In this case, if these two agents interact, one of them transitions to $s_h (h \geq k + 1)$. Hence, the lemma holds in this case.

Next, we consider the case in which exactly one agent has state s_k. In this case k agents have states in $\{s_1, \ldots, s_{k-1}\}$. Hence, from the inductive assumption, one of them can transition to $s_h (h \geq k)$ by interactions among the k agents. If $h > k$ holds, the lemma holds; otherwise, two agents have state s_k and hence the lemma holds similarly to the first case. Finally, we consider the case in which no agent has state s_k. This implies that $k+1$ agents have states in $\{s_1, \ldots, s_{k-1}\}$. Hence, from the inductive assumption, two of them can transition to $s_h (h \geq k)$ and $s_{h'} (h' \geq k)$. If $h > k$ or $h' > k$ holds, the lemma holds; otherwise, two agents have state s_k and hence the lemma holds similarly to the first case. □

Since C'_t is a stable configuration in Ξ', $P/2$ agents have states in $Q_{blue} = \{s_1, \ldots, s_{P/2-1}\}$. Hence, by Lemma 8, we can construct an execution segment that makes a *blue* agent transition to s_h for some $h \geq P/2$. This implies that the agent changes its color from *blue* to *red*. Since C'_t is a stable configuration, this is a contradiction. □

From now, we show that there exists a state $p \in Q_{nbc}$ such that $p \rightsquigarrow p$ holds. Furthermore, we also show that there exists a state $p \in Q_{nbc}$ such that $p \overset{*}{\rightsquigarrow} p$ holds.

Lemma **9.** *There exists a state $p \in Q_{nbc}$ such that $p \rightsquigarrow p$ holds.*

Proof. From Lemma 7, there exists a state p_0 in Q_{nbc}. From the definition, there exists a sequence $p_0 \rightsquigarrow p_1 \rightsquigarrow p_2 \rightsquigarrow p_3 \rightsquigarrow \cdots$ starting from p_0. Since the number of states is finite, there exist some p_i and p_j such that $j > i \geq 0$ and $p_i = p_j$ holds. This implies $p_i \rightsquigarrow p_i$. Clearly, $p_i \in Q_{nbc}$ holds because $p_0 \in Q_{nbc}$ holds. Therefore, the lemma holds. □

Lemma **10.** *There exists a state $p \in Q_{nbc}$ such that $p \overset{*}{\rightsquigarrow} p$ holds.*

Proof. For contradiction, we assume that such state p does not exist. By Lemma 9, there exists a state $p_0 \in Q_{nbc}$ such that $p_0 \rightsquigarrow p_0$ holds. Since $p_0 \overset{*}{\rightsquigarrow} p_0$ does not hold, there exists some state q such that $p_0 \rightsquigarrow q$ holds but $q \rightsquigarrow p_0$ does not hold. Since $p_0 \rightsquigarrow q$ holds, q belongs to Q_{nbc}. For this reason, there exists a state p'_0 such that $q \rightsquigarrow p'_0 \rightsquigarrow p'_0$ holds. Note that, since $q \rightsquigarrow p_0$ does not hold, p'_0 is not equal to p_0. Additionally, since $p'_0 \overset{*}{\rightsquigarrow} p'_0$ does not hold, there exists some state q' such that $p'_0 \rightsquigarrow q'$ holds but $q' \rightsquigarrow p'_0$ does not hold. This implies that there exists a state $p''_0 \in Q_{nbc}$ such that $q' \rightsquigarrow p''_0 \rightsquigarrow p''_0$, $p''_0 \neq p'_0$, and $p''_0 \neq p_0$ hold. By repeating similar arguments, since the number of states is finite, we can prove that there exists some state $p^*_0 \in Q_{nbc}$ such that $p^*_0 \overset{*}{\rightsquigarrow} p^*_0$ holds. This is a contradiction. □

In the following, we focus on a state $p^* \in Q_{nbc}$ such that $p^* \overset{*}{\rightsquigarrow} p^*$ holds. Let $Q_{p^*} = \{q \mid p^* \rightsquigarrow q\}$. Note that, if homonyms with a state in Q_{p*} interact, they transition to a state in Q_{p*}. This implies that Q_{p^*} satisfies the first condition of Q^* in Lemma 6. Moreover, we observe that, for any s, $s' \in Q_{p*}$, $s \overset{*}{\rightsquigarrow} s'$ holds because $p^* \overset{*}{\rightsquigarrow} s \overset{*}{\rightsquigarrow} p^*$ and $p^* \overset{*}{\rightsquigarrow} s' \overset{*}{\rightsquigarrow} p^*$ hold. To prove the second condition, we first show that, when $|Q_{p*}|$ agents have states in Q_{p*} initially, for any $s \in Q_{p*}$, there exists an execution such that only homonyms in the $|Q_{p*}|$ agents interact, and eventually, some agent transitions to state s. To show this, we define a potential function $\Phi(C, s)$ for configuration C and state $s \in Q_{p*}$. Intuitively, $\Phi(C, s)$ represents how far configuration C is from a configuration that includes an agent with state s. To define $\Phi(C, s)$, we define $DtQ(s_i, s)$ as follows:

Definition **10.** *$DtQ(s_i, s)$ is a function that satisfies the following properties .*

- *If $s_i = s$ holds, $DtQ(s_i, s) = 0$ holds.*
- *If $s_i \neq s$ and $s_i \in Q_{p^*}$ holds, $DtQ(s_i, s) = \min\{DtQ(s^1_j, s),\ DtQ(s^2_j, s)\} + 1$ holds when transition rule $(s_i, s_i) \to (s^1_j, s^2_j)$ exists.*
- *If $s_i \notin Q_{p^*}$ holds, $DtQ(s_i, s) = \infty$ holds.*

Intuitively, $DtQ(s_i, s)$ provides the minimum number of interactions required to transition from state s_i to state s when allowing only interactions with homonyms. Recall that, for any $s_i \in Q_{p*}$, $s_i \overset{*}{\rightsquigarrow} s$ holds because $p^* \overset{*}{\rightsquigarrow} s_i \overset{*}{\rightsquigarrow} p^*$ and $p^* \overset{*}{\rightsquigarrow} s \overset{*}{\rightsquigarrow} p^*$ hold.

Definition **11.** *We consider configuration C such that $z = |Q_{p^*}|$ agents $a_1, \ldots, a_z$ have states in Q_{p^*}. In this case, we define potential function $\Phi(C, s)$ as a multiset $\{DtQ(s(a_1, C), s),\ DtQ(s(a_2, C), s),\ DtQ(s(a_3, C), s),\ \ldots,\ DtQ(s(a_z, C), s)\}$.*

Definition **12.** *For distinct $\Phi(C_1, s)$ and $\Phi(C_2, s)$, we define their comparative operator as follows: Let i be the minimum integer such that the number of i-elements is different between $\Phi(C_1, s)$ and $\Phi(C_2, s)$. If the number of i-elements in $\Phi(C_1, s)$ is larger than that in $\Phi(C_2, s)$, we say $\Phi(C_1, s) < \Phi(C_2, s)$.*

Now we prove that, for any $s \in Q_{p*}$, there exists an execution such that only homonyms in the $|Q_{p*}|$ agents with states in Q_{p*} interact, and eventually, some agent transitions to state s.

Lemma **11.** *We consider a population $V_{p*} = \{a_1, a_2, \ldots, a_z\}$, where $z \geq |Q_{p^*}|$ holds. We consider an initial configuration C_0^q such that all agents in V_{p*} have states in Q_{p*}. For any $q \in Q_{p*}$, there exists an execution segment $e^q = C_0^q, C_1^q, C_2^q, \ldots, C_m^q$ such that (1) some agent has state q at the last configuration C_m^q; and (2) only homonyms interact in e^q.*

Proof. Let C be a configuration with V_{p*} such that all agents have states in Q_{p*}. If an agent with q in C, we can construct $e^q = C$. Hence, we consider that there does not exist an agent with q in C. Since at least $|Q_{p^*}|$ agents have states in Q_{p*} in C and there does not exist an agent with q in C, there exist homonyms in C. When homonyms with a state in Q_{p*} interact, they transition to a state in Q_{p*}. These imply that, when homonyms interact at $C \to C'$, either an agent with q or homonyms with a state in Q_{p*} exist in C'. Thus, for contradiction, we assume that there exists an infinite execution segment $e^q = C_0^q, C_1^q, C_2^q, \ldots$ with V_{p*} such that only homonyms interact and no agent ever has q in e^q. We consider the case in which a_x and a_y interact at $C_i^q \to C_{i+1}^q$ for $i \geq 0$. By the assumption, a_x and a_y have the same state $p \in Q_{p*}$. From the property of Q_{p*}, p satisfies $p \rightsquigarrow p^* \rightsquigarrow q$. Thus, $DtQ(s(a_x, C_i^q), q) > DtQ(s(a_x, C_{i+1}^q), q)$ or $DtQ(s(a_y, C_i^q), q) > DtQ(s(a_y, C_{i+1}^q), q)$ holds. Hence, from the property of Φ, $\Phi(C_i^q, q) > \Phi(C_{i+1}^q, q)$ holds. Since the number of possible values of $\Phi(C, q)$ is finite, $\Phi(C_j^q, q)$ includes 0 for some C_j^q and thus some agent has q in C_j^q. By the definition of e^q, this is a contradiction. $\square$

Now, we show Lemma 6 by using Lemma 11.

Lemma **6.** *A non-empty state set $Q^* \subseteq Q_{blue}$ exists that satisfies the following conditions:*

- *For any state $p \in Q^*$, transition rule $(p, p) \to (p', q')$ satisfies $p' \in Q^*$ and $q' \in Q^*$.*
- *We assume that, in a configuration C, there exists a subset of agents V^* such that all agents in V^* have states in Q^* and $|V^*| \geq |Q^*| + 1$ holds. In this*

case, for any agent $a_r \in V^*$ and any state $q \in Q^*$, there exists an execution segment such that (1) the execution segment starts from C; (2) a_r has state q at the last configuration; (3) only agents in V^* join interactions; and (4) all agents in V^* have states in Q^* at the last configuration.

Proof. We show that Q_{p^*} satisfies the condition of Q^*. Clearly, Q_{p^*} satisfies the first condition. Hence, we focus on the second condition. We consider a set of agents V^* with $|V^*| \geq |Q_{p*}|+1$, and consider an initial configuration $C_0^{p^*}$ such that all agents in V^* have states in Q_{p*}. Let a_r be an agent in V^* and let $s = s(a_r, C_0^{p^*})$. We consider a sequence of states $T = t_0, t_1, t_2, \dots, t_l$ such that $t_0 = s$, $t_l = q$, and for any i $(0 \leq i < l)$, transition rule $(t_i, t_i) \to (t_{i+1}, x_i)$ or $(t_i, t_i) \to (x_i, t_{i+1})$ exists for some x_i. From configuration $C_0^{p^*}$, we make a_r change its state according to T; that is, if a_r has state $t_i (0 \leq i < l)$, we make one of the remaining agents (i.e., $V^* - \{a_r\}$) transition to t_i by interactions with homonyms and then make the agent interact with a_r. Note that the number of remaining agents is at least $|Q_{p^*}|$. Such a procedure is possible because, by Lemma 11, one of the remaining agents can transition to any state in Q_{p^*} by interactions with homonyms. We observe that all agents in V^* keep states in Q_{p^*} when the procedure is applied. Hence, we can construct an execution segment under the second condition. Therefore, Q_{p^*} satisfies the conditions of Q^* and thus the lemma holds. □

4.1.3 Lower Bound for Symmetric Protocols

In this subsection, we show that the lower bound for symmetric protocols is $P+1$. To prove this, we use the ideas of the impossibility proof for the naming protocol [23]. This study showed that, in the model with an initialized base station, there is no symmetric naming protocol with P states from arbitrary initial states under weak fairness. We apply the proof in [23] to the uniform 2-partition but, because the problem considered in that study is different, we need to make a non-trivial modification.

Theorem **12.** *In the model with an initialized base station, there is no symmetric protocol that solves the uniform 2-partition problem with P states from arbitrary initial states under weak fairness, if P is an even integer.*

In the case of naming protocols [23], the impossibility proof proves that a unique state (called a sink state) always exists. However, in the case of uniform 2-partition protocols, sometimes no sink state exists. To consider this scenario, we additionally define a sink pair, which is a pair of two states that has similar properties to a sink state. We show that either a sink state or sink pair exists, and we prove that there is no symmetric protocol in both cases. Furthermore, Theorem 12 directly implies the following corollary.

Corollary **4.** *In the model with an initialized base station, there is no symmetric protocol that solves the uniform partition problem with P states from arbitrary initial states under weak fairness if P is an even integer.*

Proof of Theorem 12

We assume, by way of contradiction, that such an algorithm Alg_{sym} exists. Let $Q_p = \{s_1, \dots, s_P\}$ be a set of states of non-base station agents. Let $Q_{blue} = \{s \in Q_p \mid \gamma(s) = blue\}$ be a set of blue states and $Q_{red} = \{s \in Q_p \mid \gamma(s) = red\}$ be a set of red states. Without loss of generality, we assume that $|Q_{blue}| \le |Q_{red}|$ holds. First, we define a sink state similarly to [11].

Definition **13.** *For states q and q', we say $q \overset{sym}{\rightsquigarrow} q'$ if there exists a sequence of states $q = q_0, q_1, \cdots, q_l = q'$ such that, for any $i (0 \le i < l)$, transition rule $(q_i, q_i) \to (q_{i+1}, q_{i+1})$ exists.*

Definition **14.** *For state q, if $q \overset{sym}{\rightsquigarrow} q$ holds, q is called a loop state.*

Definition **15.** *State m is a sink state if $m \in Q_p$ satisfies the following conditions:*

1. *There exists a transition rule $(m, m) \to (m, m)$.*
2. *For any $s \in Q_p$, $s \overset{sym}{\rightsquigarrow} m$ holds.*
3. *If the number of agents is at most $P - 2$, m does not occur infinitely often for any execution.*

In the case of naming protocols [11], the impossibility proof proves that a sink state always exists. However, in the case of uniform 2-partition protocols, sometimes no sink state exists. To consider this scenario, we additionally define a sink pair, which is a pair of two states that has a similar property of a sink state. We prove that either a sink state or sink pair exists.

Definition **16.** *A pair of two states $m_1, m_2 \in Q_p$ is a sink pair if the following conditions hold:*

1. *There exist transition rules (1) $(m_1, m_1) \to (m_1, m_1)$ and $(m_2, m_2) \to (m_2, m_2)$ or (2) $(m_1, m_1) \to (m_2, m_2)$ and $(m_2, m_2) \to (m_1, m_1)$.*
2. *For any $s \in Q_p$, $s \overset{sym}{\rightsquigarrow} m_1$ or $s \overset{sym}{\rightsquigarrow} m_2$ holds.*
3. *If the number of agents is at most $P-2$, m_1 and m_2 do not occur infinitely often for any execution.*

The following lemma provides an important property to prove the existence of a sink state or sink pair.

Lemma **12.** *Let $\Xi = C_0, C_1, C_2, \ldots$ be a weakly fair execution of Alg_{sym} with $n \le P-2$ agents and an initialized base station. For any loop state $s_r \in Q_p$ (i.e., $s_r \overset{sym}{\rightsquigarrow} s_r$ holds), s_r does not occur infinitely often in Ξ.*

Proof. The idea of the proof is as follows: First, for contradiction, we consider an execution Ξ with $n \le P-2$ such that s_r occurs infinitely often in Ξ. Next, we consider an execution with P agents such that all additional agents have s_r as their initial states and other agents behave similarly to Ξ. In the execution, all additional agents do not join the interactions until some other agent has s_r. At that time, one of non-additional agents has state s_r and additional agents also have state s_r. We can prove that, from this configuration, non-additional agents cannot recognize the additional agents and hence they behave in the same manner as in Ξ. Additionally, the additional agents keep state s_r. Since the numbers of *red* and *blue* agents are balanced without the additional agents and the additional agents have the same state, the uniform 2-partition problem cannot be solved. Now we present the details of the proof. For contradiction, we assume that there exists a weakly fair execution Ξ such that s_r occurs infinitely often in Ξ. First, we consider a population $V = \{a_0, a_1, a_2, \ldots, a_n\}$ of $n \le P-2$ agents and an initialized base station, where a_0 is the base station. Since there exist a finite number of agents, there exists a particular agent a_x that has s_r infinitely often in Ξ. We can define infinite configurations $C_{u_0}, C_{u_1}, \ldots$ and infinite execution segments $e_0, e_1, e_2, \ldots$ of Ξ so that $\Xi = C_{u_0}$, e_0, C_{u_1}, e_1, C_{u_2}, e_2, C_{u_3}, $\ldots$ satisfies the following:

- For $w \geq 1$, agent a_x has state s_r in C_{u_w}.
- For $j \geq 0$, during execution segment $C_{u_j}, e_j, C_{u_{j+1}}$, any pair of agents in V interacts at least once (this is possible because Ξ is weakly fair).

Next, we consider a population $V' = \{a'_0, a'_1, \ldots, a'_P\}$ of P agents and an initialized base station, where a'_0 is the base station. We define execution $\Xi' = C'_{u_0}$, e'_0, C'_{v_0}, e^m_0, C'_{u_1}, e'_1, C'_{v_1}, e^m_1, C'_{u_2}, e'_2, C'_{v_2}, $\ldots$ as follows:

- In initial configuration C'_{u_0}, a'_i $(n \geq i \geq 0)$ has the same state as a_i in C_0 and $a'_{n+1}, a'_{n+2}, \ldots, a'_P$ have state s_r. Formally, $s(a'_i, C'_{u_0}) = s(a_i, C_{u_0})$ holds for any i $(n \geq i \geq 0)$, and $s(a'_{n+1}, C'_{u_0}) = s(a'_{n+2}, C'_{u_0}) = \cdots = s(a'_P, C'_{u_0}) = s_r$ holds.
- For $j \geq 0$, we construct execution segment $e'_j = \hat{C}^j_1$, $\hat{C}^j_2$, $\hat{C}^j_3$, $\ldots$, $\hat{C}^j_z$ and configuration C'_{v_j} using $e_j = C^j_1$, C^j_2, C^j_3, $\ldots$, C^j_z and $C_{u_{j+1}}$, where $z = |e_j|$ holds. Specifically, we construct e'_j as follows:
  - Case in which $j = (P - n + 1) \cdot y$ holds for some y $(y \geq 0)$. In this case, agents $a'_0, \ldots, a'_n$ interact in execution segment C'_{u_j}, e'_j, C'_{v_j} similarly to $a_0, \ldots, a_n$ in execution segment $C_{u_j}, e_j, C_{u_{j+1}}$. Formally, when a_g interacts with a_h at $C^j_f \to C^j_{f+1}$ for $z > f > 0$ (resp., $C_{u_j} \to C^j_1$ and $C^j_z \to C_{u_{j+1}}$), a'_g interacts with a'_h at $\hat{C}^j_f \to \hat{C}^j_{f+1}$ (resp., $C'_{u_j} \to \hat{C}^j_1$ and $\hat{C}^j_z \to C'_{v_j}$).
  - Case in which $j = (P - n + 1) \cdot y + l$ holds for some y $(y \geq 0)$ and l $(P - n \geq l \geq 1)$. In this case, a'_{n+l} joins interactions instead of a'_x. Note that, in C'_{u_j}, both a'_{n+l} and a'_x have state s_r. Formally we construct e'_j as follows: (1) when $a_g (g \neq x)$ interacts with $a_h (h \neq x)$ at $C^j_f \to C^j_{f+1}$ for $z > f > 0$ (resp., $C_{u_j} \to C^j_1$ and $C^j_z \to C_{u_{j+1}}$), a'_g interacts with a'_h at $\hat{C}^j_f \to \hat{C}^j_{f+1}$ (resp., $C'_{u_j} \to \hat{C}^j_1$ and $\hat{C}^j_z \to C'_{v_j}$), and (2) when a_x interacts with an agent $a_i (i \neq x)$ at $C^j_f \to C^j_{f+1}$ for $z > f > 0$ (resp., $C_{u_j} \to C^j_1$ and $C^j_z \to C_{u_{j+1}}$), a'_{n+l} interacts with a'_i at $\hat{C}^j_f \to \hat{C}^j_{f+1}$ (resp., $C'_{u_j} \to \hat{C}^j_1$ and $\hat{C}^j_z \to C'_{v_j}$). Similarly, when an agent $a_i (i \neq x)$ interacts with a_x at

$C^j_f \rightarrow C^j_{f+1}$ for $z > f > 0$ (resp., $C_{u_j} \rightarrow C^j_1$ and $C^j_z \rightarrow C_{u_{j+1}}$), a'_i interacts with a'_{n+l} at $\hat{C}^j_f \rightarrow \hat{C}^j_{f+1}$ (resp., $C'_{u_j} \rightarrow \hat{C}^j_1$ and $\hat{C}^j_z \rightarrow C'_{v_j}$).

- For $j \geq 0$, during execution segment $C'_{v_j}, e^m_j, C'_{u_{j+1}}$, agents a'_x, a'_{n+1}, a'_{n+2}, $\ldots$, a'_P interact so that, for any pair (a'_g, a'_h) of them, a'_g interacts with a'_h at least once, and eventually, they have state s_r. At the first configuration, agents a'_x, a'_{n+1}, a'_{n+2}, $\ldots$, a'_P have state s_r. Since $s_r \overset{sym}{\rightsquigarrow} s_r$ holds, each pair of them can go back to state s_r after some interactions. Thus, in $C'_{u_j}(j > 0)$, agents a'_x, a'_{n+1}, a'_{n+2}, $\ldots$, a'_P have state s_r.

In execution segment C'_{u_j}, e'_j, C'_{v_j} such that $j = (P - n + 1) \cdot y$ holds, for each pair of agents a and b in $V' - \{a'_{n+1}, a'_{n+2}, \ldots, a'_P\}$, a interacts with b. In execution segment C'_{u_j}, e'_j, C'_{v_j} such that $j = (P - n + 1) \cdot y + l$ holds, for every pair of agents a and b in $(V' - \{a'_x, a'_{n+1}, a'_{n+2}, \ldots, a'_P\}) \cup \{a'_{n+l}\}$, a interacts with b. Moreover, for every pair of agents a and b in $\{a'_x, a'_{n+1}, a'_{n+2}, \ldots, a'_P\}$, a interacts with b in e^m_j for $j > 0$. From these facts, execution Ξ' is weakly fair. In Ξ, let C_{u_t} be a stable configuration such that agent a_x has state s_r. Let R_{u_t} be a set of *red* agents in C_{u_t} and let B_{u_t} be a set of *blue* agents in C_{u_t}. Clearly, $||R_{u_t}| - |B_{u_t}|| \leq 1$ holds. Now, we consider two cases. One is the case in which n is even (i.e., $||R_{u_t}| - |B_{u_t}|| = 0$ holds). Another is the case in which n is odd (i.e., $||R_{u_t}| - |B_{u_t}|| = 1$ holds). Note that, in both cases, $s(a_i, C_{u_w}) = s(a'_i, C'_{u_w})$ holds for $n \geq i \geq 0$ and $w \geq 0$, and other $P - n$ agents have state s_r in C'_{u_w} for $w \geq 0$. Hence, for the number of $\gamma(s_r)$-agents, the difference between C_{u_w} and C'_{u_w} is $P - n$ for any $w \geq t$. First, we consider the case in which n is even. After C'_{u_t}, the number of $\gamma(s_r)$-agents is $P - n \geq 2$ more than the number of $\overline{\gamma(s_r)}$-agents. Thus, Ξ' never reaches a stable configuration. Next, we consider the case that n is odd. Since $n \leq P - 2$ holds and both $P - 1$ and n are odd, $n \leq P - 3$ holds and thus $P - n \geq 3$ holds. Hence, after C'_{u_t}, the number of $\gamma(s_r)$-agents is at least two more than the number of $\overline{\gamma(s_r)}$-agents. Thus, Ξ' never reaches a stable configuration. Since Ξ' is weakly fair, this is a contradiction. □

Using Lemma 12, Lemma 13 shows the existence of a sink state or sink pair.

Lemma **13.** *In any protocol Alg_{sym}, there exists either exactly one sink pair or exactly one sink state.*

Proof. For $q \in Q_p$, let $L_q = \{q' \mid q \overset{sym}{\leadsto} q' \text{ and } q' \overset{sym}{\leadsto} q'\}$; that is, L_q is a set of loop states such that an agent with state q can transition to the state by interactions with homonyms. For any $q_0 \in Q_p$, we consider a sequence of transition rules $(q_0, q_0) \to (q_1, q_1)$, $(q_1, q_1) \to (q_2, q_2)$, $(q_2, q_2) \to \cdots$. Since the number of states is finite, $q_i = q_j$ holds for some $j > i \geq 0$. Hence, $L_q \neq \emptyset$ holds. We define L as $L = L_{s_1} \cup L_{s_2} \cup L_{s_3} \cup \cdots \cup L_{s_P}$. Now we show that $|L| \leq 2$ holds by Lemmas 12 and 2. Recall that the properties of Lemma 2 hold even in a symmetric algorithm Alg_{sym}. By Lemma 12, a loop state does not occur infinitely often in any execution with $n \leq P - 2$ agents. Additionally, by Lemma 2, when the number of agents is $P-2$, there exists an execution such that at least $P-2$ states occur infinitely often. This implies that such $P - 2$ states are not loop states. Thus, the number of loop states is at most two; that is, $|L| \leq 2$ holds. If $|L| = 1$ holds, a loop state in L satisfies conditions 2 and 3 of a sink state in Definition 15. This is because an agent with any state can transition to the loop state in L by interactions with homonyms and the loop state does not occur infinitely often by Lemma 12. For a similar reason, if $|L| = 2$ holds, two loop states in L satisfy conditions 2 and 3 of a sink pair in Definition 16. Note that, in this case, the two loop states in L are not sink states because they cannot satisfy conditions 1 and 2 of a sink state in Definition 15 simultaneously. In the following, we show that a loop state and two loop states in L satisfy condition 1 of a sink state and sink pair, respectively. First, we consider the case of $|L| = 2$. Let m_1 and m_2 be states in L. For contradiction, we assume that there exists $(m_1, m_1) \to (s, s)$ such that $s \notin \{m_1, m_2\}$ holds. Since $m_1 \overset{sym}{\leadsto} m_1$ holds, $s \overset{sym}{\leadsto} s$ holds. However, by the assumption, such an s does not exist because only m_1 and m_2 are loop states. Hence, $(m_1, m_1) \to (s, s)$ does not exist, and hence either $(m_1, m_1) \to (m_1, m_1)$ or $(m_1, m_1) \to (m_2, m_2)$ exists. Similarly, $(m_2, m_2) \to (s, s)$ for $s \notin \{m_1, m_2\}$ does not exist, and hence either $(m_2, m_2) \to (m_1, m_1)$ or $(m_2, m_2) \to (m_2, m_2)$ exists. If both $(m_1, m_1) \to (m_1, m_1)$ and $(m_2, m_2) \to (m_1, m_1)$ exist, $m_2 \overset{sym}{\leadsto} m_2$ does not hold. Similarly, if both $(m_1, m_1) \to (m_2, m_2)$ and $(m_2, m_2) \to (m_2, m_2)$ exist, $m_1 \overset{sym}{\leadsto} m_1$ does not hold. Thus, m_1 and m_2 satisfy condition 1 of a sink pair. Next, we consider the case of $|L| = 1$. Let m be a state in L. For contradiction, we assume that there exists $(m, m) \to (s, s)$ such that $s \neq m$ holds. Since $m \overset{sym}{\leadsto} m$ holds, $s \overset{sym}{\leadsto} s$ holds. However, the loop state is only m. This is a contradiction, and m satisfies condition 1 of a sink state. Therefore, the lemma holds. □

We introduce a reduced execution that is also defined in [11].

Definition **17.** *In a reduced execution, if homonyms with a non-sink state (resp., neither of the sink pair) occur, they immediately enter a sink state (resp., one of the sink pair) by interactions with the homonyms. This procedure is called reducing .*

By condition 2 of a sink state and sink pair, such reducing is possible. We say configuration C is reduced if there are no homonyms except agents with a sink state or one of the sink pair. Note that there exists a reduced weakly fair execution of Alg_{sym} because any pair of agents can interact in a reduced configuration. We consider a reduced weakly fair execution Ξ of Alg_{sym} with $P-2$ agents. By Lemma 12, there exists a stable reduced configuration such that no agent has a sink state or states of the sink pair. Since no two agents have the same non-sink state, we have the following corollaries.

Corollary **5.** *When a sink state exists in Q_p, either*

- *the number of non-sink red states is $P/2$ and the number of non-sink blue states is $P/2-1$; or*
- *the number of non-sink blue states is $P/2$ and the number of non-sink red states is $P/2-1$.*

Corollary **6.** *When a sink pair exists in Q_p, the number of red (resp., blue) states not in the sink pair is $P/2-1$ (resp., $P/2-1$).*

Moreover, Corollary 5 can be extended to the following lemma.

Lemma **14.** *When a sink state m exists in Q_p, the number of non-sink $\gamma(m)$-states is $P/2-1$ and the number of non-sink $\overline{\gamma(m)}$-states is $P/2$. This implies that the number of $\gamma(m)$-states is $P/2$ and the number of $\overline{\gamma(m)}$-states is also $P/2$.*

Proof. By Corollary 5, the number of non-sink $\gamma(m)$-states is at least $P/2-1$ and at most $P/2$. Hence, for contradiction, we assume that the number of non-sink $\gamma(m)$-states is $P/2$. This implies that the number of $\gamma(m)$-states (including the sink state m) is $P/2+1$ and the number of $\overline{\gamma(m)}$-states is $P/2-1$. Now, we consider a reduced weakly fair execution Ξ of Alg_{sym} with P agents and an initialized base station. In Ξ, after a stable configuration, a reduced configuration occurs infinitely often. In a reduced configuration, each non-sink state is held by at most one agent. Thus, because all the $\overline{\gamma(m)}$-states are non-sink states and the number of them is $P/2-1$, in a stable reduced configuration the number of $\overline{\gamma(m)}$-agents is at most $P/2-1$ and the number of $\gamma(m)$-agents is at least $P/2+1$. This is a contradiction. □

Subsequently, we show that, when a sink state m exists and the number of agents is $P-1$, the number of $\gamma(m)$-agents is less than the number of $\overline{\gamma(m)}$-agents in a stable configuration.

Lemma **15.** *When a sink state m exists in Q_p, for any reduced weakly fair execution $\Xi = C_0, C_1, C_2, \ldots, C_t, \ldots$ of Alg_{sym} with $P-1$ agents and an initialized base station, the following is satisfied in a stable configuration C_t of Ξ.*

- *The number of $\gamma(m)$-agents is $P/2-1$ and the number of $\overline{\gamma(m)}$-agents is $P/2$.*

Proof. The idea of the proof is as follows: For contradiction, we consider a reduced execution Ξ with $P-1$ agents such that the number of $\gamma(m)$-agents is $P/2$ and the number of $\overline{\gamma(m)}$-agents is $P/2-1$ in a stable configuration of Ξ. Note that, by Lemma 14, the number of $\gamma(m)$-states is $P/2$. Thus, since the stable configuration is reduced, some agent has m in the stable configuration. Next, we consider an execution with P agents such that one additional agent has m as an initial state and other agents behave similarly to Ξ. In the execution, the additional agent does not join the interactions until $P-1$ agents converge to a stable configuration in Ξ. At that time, one of the $P-1$ agents has state m and the additional agent also has state m. We can prove that, from this configuration, $P-1$ agents cannot recognize the additional agent and hence they make the same behavior as in Ξ. Additionally, the additional agent can keep state m. Since the additional agent has $\gamma(m)$-state, the number of $\gamma(m)$-agents is $P/2+1$ and the number of $\overline{\gamma(m)}$-agents is $P/2-1$. This implies that the uniform 2-partition problem cannot be solved. Now we present the details of the proof. For contradiction, there exists a reduced weakly fair execution Ξ such that the number of $\gamma(m)$-agents is $P/2$ and the number of $\overline{\gamma(m)}$-agents is $P/2-1$ in a stable configuration of Ξ. First, we consider a population $V = \{a_0, a_1, a_2, \dots, a_{P-1}\}$ of $P-1$ agents and an initialized base station, where a_0 is the base station. We define a reduced weakly fair execution $\Xi = C_0,\ C_1,\ C_2,\ \dots,\ C_t,\ \dots,\ C_{t'_0}, e_1,\ C_{t'_1}, e_2,\ C_{t'_2}, e_3,\ \dots$ of Alg_{sym} with V as follows:

- C_t is a stable configuration.

- For any $u \geq 0$, $C_{t'_u}$ is a particular stable reduced configuration such that $C_{t'_0} = C_{t'_1} = C_{t'_2} = \cdots$ holds. Note that such a configuration (i.e., a stable reduced configuration that appears an infinite number of times) exists because the number of possible configurations is finite.

- For $j > 0$, e_j is an execution segment such that, during execution segment $C_{t'_{j-1}}, e_j, C_{t'_j}$, for any pair of agents a and b in V, a interacts with b at least once. This is possible because Ξ is weakly fair.

By Lemma 14, the number of $\gamma(m)$-states is $P/2$. Additionally, by the assumption, the number of $\gamma(m)$-agents is also $P/2$ in $C_{t'_u}$ for any u. From these facts, for any u, since $C_{t'_u}$ is a reduced configuration (i.e., there exist no homonyms except m), there exists a particular agent a_q that has state m in $C_{t'_u}$ for any u. Next, we consider a population $V' = \{a'_0, a'_1, \dots, a'_P\}$ of P agents and an initialized base station, where a'_0 is the base station. We define $\Xi' = C'_0,\ C'_1,\ C'_2,\ \dots,\ C'_t,\ \dots,\ C'_{t'_0}, e'_1,\ C'_{t'_1}, e'_2,\ C'_{t'_2}, e'_3,\ \dots$ using Ξ. First, we define the first part of Ξ'; that is, $C'_0,\ C'_1,\ C'_2,\ \dots,\ C'_t,\ \dots,\ C'_{t'_0}$ as follows:

- In initial configuration C'_0, $a'_0, \dots, a'_{P-1}$ have the same states as $a_0, \dots, a_{P-1}$ in C_0, and a'_P has state m. Formally, $s(a'_i, C'_0) = s(a_i, C_0)$ holds for i $(P-1 \geq i \geq 0)$, and $s(a'_P, C'_0) = m$ holds.

- From C'_0 to $C'_{t'_0}$, $a'_0, \dots, a'_{P-1}$ interact similarly to $a_0, \dots, a_{P-1}$ in Ξ. Formally, for any $u (t'_0 > u \geq 0)$, when a_g interacts with a_h at $C_u \to C_{u+1}$, a'_g interacts with a'_h at $C'_u \to C'_{u+1}$.

Clearly, $s(a'_i, C'_{t'_0}) = s(a_i, C_{t'_0})$ holds for i $(P-1 \geq i \geq 0)$, and $s(a'_P, C'_{t'_0}) = m$ holds. This implies that the number of $\gamma(m)$-agents is $P/2+1$ and the number of $\overline{\gamma(m)}$-agents is $P/2-1$ in $C'_{t'_0}$. We define the remaining part of Ξ'; that is, $C'_{t'_0}, e'_1,\ C'_{t'_1}, e'_2,\ C'_{t'_2}, e'_3,\ \dots$ as follows:

- For $j > 0$, we construct an execution segment $e'_j = \hat{C}^j_1, \hat{C}^j_2, \hat{C}^j_3, \ldots, \hat{C}^j_z, \hat{C}^j_{z+1}, \hat{C}^j_{z+2}$ using $e_j = C^j_1, C^j_2, C^j_3, \ldots, C^j_z$, where $z = |e_j|$ holds. Specifically, we construct $C'_{t'_{j-1}}, e'_j, C'_{t'_j}$ as follows:
  - Case in which j is even: In this case, agents $a'_0, \ldots, a'_{P-1}$ interact in execution segment $C'_{t'_{j-1}}, e'_j$ similarly to $a_0, \ldots, a_{P-1}$ in execution segment $C_{t'_{j-1}}, e_j, C_{t'_j}$. Formally, when a_g interacts with a_h at $C^j_f \to C^j_{f+1}$, for $z > f > 0$ (resp., $C_{t'_{j-1}} \to C^j_1$, and $C^j_z \to C_{t'_j}$), a'_g interacts with a'_h at $\hat{C}^j_f \to \hat{C}^j_{f+1}$, (resp., $C'_{t'_{j-1}} \to \hat{C}^j_1$, and $\hat{C}^j_z \to \hat{C}^j_{z+1}$.
  - Case in which j is odd: In this case, a'_P joins interactions instead of a'_q. Note that, in $C'_{t'_{j-1}}$, both a'_P and a'_q have state m. Formally, we construct e'_j as follows: (1) when $a_g (g \neq q)$ interacts with $a_h (h \neq q)$ at $C^j_f \to C^j_{f+1}$ for $z > f > 0$, (resp., $C_{t'_{j-1}} \to C^j_1$, and $C^j_z \to C_{t'_j}$), a'_g interacts with a'_h at $\hat{C}^j_f \to \hat{C}^j_{f+1}$, (resp., $C'_{t'_{j-1}} \to \hat{C}^j_1$, and $\hat{C}^j_z \to \hat{C}^j_{z+1}$; (2) when a_q interacts with an agent $a_i (i \neq q)$ at $C^j_f \to C^j_{f+1}$, (resp., $C_{t'_{j-1}} \to C^j_1$, and $C^j_z \to C_{t'_j}$), a'_P interacts with a'_i at $\hat{C}^j_f \to \hat{C}^j_{f+1}$ (resp., $C'_{t'_{j-1}} \to \hat{C}^j_1$, and $\hat{C}^j_z \to \hat{C}^j_{z+1}$. Similarly, when an agent $a_i (i \neq q)$ interacts with a_q at $C^j_f \to C^j_{f+1}$, (resp., $C_{t'_{j-1}} \to C^j_1$, and $C^j_z \to C_{t'_j}$, a'_i interacts with a'_P at $\hat{C}^j_f \to \hat{C}^j_{f+1}$ (resp., $C'_{t'_{j-1}} \to \hat{C}^j_1$, and $\hat{C}^j_z \to \hat{C}^j_{z+1}$).
  - a'_P interacts with a'_q at $\hat{C}^j_{z+1} \to \hat{C}^j_{z+2}$.
  - a'_q interacts with a'_P at $\hat{C}^j_{z+2} \to C'_{t'_j}$.

We can show inductively that, for any $x \geq 0$, $s(a'_i, C'_{t'_x}) = s(a_i, C_{t'_x})$ holds for any i $(P-1 \geq i \geq 0)$, and $s(a'_q, C'_{t'_x}) = s(a'_P, C'_{t'_x}) = s(a_q, C_{t'_x}) = m$ holds. Clearly, this holds for $x = 0$. We assume that this holds for $x = j$, and consider the case of $x = j+1$. When $j+1$ is even, during execution segment $C'_{t'_j}, e'_{j+1}$, agents in $V' - \{a'_P\}$ interact similarly to $C_{t'_j}, e_{j+1}, C_{t'_{j+1}}$. Hence, for any $j > 0$, $s(a'_i, \hat{C}^{j+1}_{z+1}) = s(a_i, C_{t'_{j+1}})$ holds for any i $(P-1 \geq i \geq 0)$ and $s(a'_q, \hat{C}^{j+1}_{z+1}) = s(a'_P, \hat{C}^{j+1}_{z+1}) = s(a_q, C_{t'_{j+1}}) = m$ holds. At $\hat{C}^{j+1}_{z+1} \to \hat{C}^{j+1}_{z+2}$ (resp., $\hat{C}^{j+1}_{z+2} \to C'_{t'_{j+1}}$), a'_P (resp. a'_q) interacts with a'_q (resp., a'_P), and hence, if they have state m, they do not change their states. Thus, $s(a'_i, C'_{t'_{j+1}}) = s(a_i, C_{t'_{j+1}})$ holds for any i $(P-1 \geq i \geq 0)$ and $s(a'_q, C'_{t'_{j+1}}) = s(a'_P, C'_{t'_{j+1}}) = s(a_q, C_{t'_{j+1}}) = m$ holds. When $j+1$ is odd, during execution segment $C'_{t'_j}, e'_{j+1}, a'_P$ joins interactions instead of a'_q, and agents in $V' - \{a'_q\}$ behave similarly to $C_{t'_{j-1}}$, e_j, $C_{t'_j}$. Hence, for any $j > 0$, $s(a'_i, \hat{C}^{j+1}_{z+1}) = s(a_i, C_{t'_{j+1}})$ holds for any i $(P-1 \geq i \geq 0)$ and $s(a'_q, \hat{C}^{j+1}_{z+1}) = s(a'_P, \hat{C}^{j+1}_{z+1}) = s(a_q, C_{t'_{j+1}}) = m$ holds. At $\hat{C}^{j+1}_{z+1} \to \hat{C}^{j+1}_{z+2}$ (resp., $\hat{C}^{j+1}_{z+2} \to C'_{t'_{j+1}}$), a'_P (resp., a'_q) interacts with a'_q (resp., a'_P), and hence, if they have state m, they do not change their states. Thus, $s(a'_i, C'_{t'_{j+1}}) = s(a_i, C_{t'_{j+1}})$ holds for any i $(P-1 \geq i \geq 0)$, and $s(a'_q, C'_{t'_{j+1}}) = s(a'_P, C'_{t'_{j+1}}) = s(a_q, C_{t'_{j+1}}) = m$ holds. For any j, the number of $\gamma(m)$-agents is $P/2$ and the number of $\overline{\gamma(m)}$-agents is $P/2-1$ in $C_{t'_j}$, and thus the number of $\gamma(m)$-agents is $P/2+1$ and the number of $\overline{\gamma(m)}$-agents is $P/2-1$ in $C'_{t'_j}$. Therefore, Ξ' cannot solve the uniform 2-partition. During $C'_{t'_{j-1}}$, e'_j, when j is even, for any pair of agents a and b in $V' - \{a'_P\}$, a interacts with b. Similarly, when j is odd, for any pair of agents a and b in $V' - \{a'_q\}$, a interacts with b. Moreover, for $j > 0$, at $\hat{C}^{j+1}_{z+1} \to \hat{C}^{j+1}_{z+2}$ (resp., $\hat{C}^{j+1}_{z+2} \to C'_{t'_{j+1}}$), a'_P (resp., a'_q) interacts with a'_q and a'_P. Thus, Ξ' is weakly fair. Since Ξ' cannot solve the uniform 2-partition, this is a contradiction. □

Using these lemmas, we show that, when a sink state exists in Q_p, Alg_{sym} does not work. We prove this in a similar way to the case of naming protocols in [11], but we need a non-trivial modification to apply the proof to uniform 2-partition protocols.

Definition **18.** *We assume that a sink state m exists in Q_p. We consider configurations C_0 and C_1 for a population V. We say that C_0 is far away from C_1 according to a non-sink state $s \neq m$ if there exists an agent a_x such that $s(a_y, C_0) = s(a_y, C_1)$ for any $a_y \in V \backslash \{a_x\}$, $s(a_x, C_0) = m$, and $s(a_x, C_1) = s \neq m$ hold. Then, C_0 is denoted by C_1^{-s} and C_1 is denoted by C_0^{+s}.*

We introduce the notion of equivalent configurations. We say that configurations C and C' are equivalent if a multiset of states in C is identical to that in C'.

Lemma **16.** *We assume that a sink state m exists in Q_p. We consider an execution segment $e = C_0, C_1, C_2, \ldots, C_k$ of Alg_{sym} with P agents and an initialized base station that satisfies the following conditions:*

- *e is a reduced execution segment.*
- *C_0 is reduced.*

- *There exists a non-sink state s such that, in any reduced configuration of e except the last configuration C_k, no agent has state s.*

Then, there exists the execution segment $e' = C'_0, C'_1, C'_2, \ldots, C'_k$ of Alg_{sym} with P agents and an initialized base station that satisfies the following conditions:

- *A particular agent a_x with m does not join interactions.*
- *$C_0 = C'_0$ holds.*
- *For any i $(0 < i \leq k)$, C'_i and C_i are equivalent.*

Proof. In a reduced configuration with P agents, if there exists a non-sink state that is held by no agent, there are at least two agents with a sink state. Hence, in any reduced configuration of e except the last configuration C_k, there exist at least two agents with a sink state. Using this property, we construct e' by induction on the index of configuration. First, since we can set the initial configuration C'_0 such that $C'_0 = C_0$ holds, the base case holds. For the induction step, we assume that there exists a configuration $C'_l (l \geq 0)$ such that C'_u and C_u are equivalent for any $u \leq l$ and a_x does not join interactions until C'_l (i.e., a_x has a sink state in C'_l). We consider two cases of interaction at $C_l \to C_{l+1}$. First, we consider the case in which an agent with a sink state does not join an interaction at $C_l \to C_{l+1}$. In this case, since C_l and C'_l are equivalent and a_x has a sink state in C'_l, a state transition that occurs at $C_l \to C_{l+1}$ can occur at $C'_l \to C'_{l+1}$. Thus, the lemma holds at C'_{l+1}. Next, we consider the case in which an agent with a sink state joins interaction at $C_l \to C_{l+1}$. In this case, C_l and C'_l are reduced. By the assumption, in C_l and C'_l, there are at least two agents with a sink state. Let $a_y \neq a_x$ be an agent that has a sink state in C'_l. Then, when agents a_i and a_j interact at $C_l \to C_{l+1}$, we consider the following two cases.

- Case in which a_i and a_j have a sink state: In this case, $C_l = C_{l+1}$ holds. Hence, we skip this interaction and regard C'_l as C'_{l+1}[2]. Clearly, C'_{l+1} and C_{l+1} are equivalent.

[2]Strictly speaking, this violates the definition of an execution because no interaction occurs at $C'_l \to C'_{l+1}$. However, by removing C'_{l+1} from e', we can construct execution e' that satisfies the definition of an execution. This modification does not affect the following proofs.

- Case in which either a_i or a_j has a sink state: Without loss of generality, we assume that a_i has a sink state. In this case, by making an interaction between a_y and a_j at $C'_l \to C'_{l+1}$, we can obtain C'_{l+1} such that C'_{l+1} and C_{l+1} are equivalent.

Then, we can obtain C'_{l+1} without making a_x join an interaction. Thus, the lemma holds. □

The following lemma is identical to the lemma in [11]. Although the lemma is proved for naming protocols in [11], we can use the lemma because the proof does not use the property of naming protocols. For completeness, we also provide the proof.

Lemma **17.** *We assume that a sink state m exists in Q_p. We consider two reduced configurations C_0 and C_0^{-s} with P agents. The difference between C_0 and C_0^{-s} is only whether an agent a_x has a non-sink state s or sink state m. We consider a reduced execution segment C_0^{-s}, e_0^{-s}, C_1 of Alg_{sym} with P agents and an initialized base station that satisfies the following conditions:*

- *During C_0^{-s}, e_0^{-s}, C_1, a_x does not join interactions.*
- *In any reduced configuration during C_0^{-s}, e_0^{-s}, there exists no agent with s.*
- *C_1 is a reduced configuration such that there exists an agent with s.*

If there exists such an execution segment, there also exists the reduced execution segment C_0, e_0, C_1^{-s} of Alg_{sym} with P agents and an initialized base station that satisfies the following conditions:

- *During C_0, e_0, C_1^{-s}, a_x does not join interactions except the last reducing.*
- *In any reduced configuration during C_0, e_0, there exists an agent with s.*
- *C_1^{-s} is a reduced configuration such that there does not exist an agent with s.*

Proof. By making an interaction similar to C_0^{-s}, e_0^{-s}, C_1, we can construct a reduced execution segment C_0, e'_0, C_1^{+s} starting from C_0. Because exactly two agents have state s in C_1^{+s}, C_1^{+s} can be reduced to C_1^{-s}. We denote this reducing procedure by C_1^{+s}, e^r, C_1^{-s}. Then, we can obtain an execution segment C_0, e_0', C_1^{+s}, e^r, C_1^{-s}. Note that, during C_0, e, C_1^{+s}, agent a_x does not join interactions. This implies that, during C_0, e_0', C_1^{+s}, e^r, C_1^{-s}, agent a_x does not join interactions except the last reducing. Hence, we can obtain the required execution segment C_0, e_0, C_1^{-s} such that $e_0 = e_0'$, C_1^{+s}, e^r holds. □

In the next lemma, we prove that a sink state does not exist.

Lemma **18.** *A sink state does not exist in Q_p.*

Proof. We use the idea of the impossibility proof for the naming protocol [11]. The idea of the proof is as follows: For contradiction, we assume that there exists a sink state $m \in Q_p$. First, we consider an execution segment e with P agents, such that (1) a particular agent a_x does not join interactions and other $P - 1$ agents interact until convergence; (2) a_x has m as an initial state; and (3) its final configuration C_h is a reduced configuration. Let s be a $\overline{\gamma(m)}$-state. When a_x has s as an initial state, by making other agents interact similarly to e, we can obtain C_h^{+s}. Moreover, by Lemma 15, since the number of $\overline{\gamma(m)}$-agents except for a_x is $P/2$ in C_h^{+s} and C_h is reduced, there exists an agent with s except for a_x in C_h^{+s}. Thus, C_h^{-s} can be obtained by reducing C_h^{+s}. Because every $\overline{\gamma(m)}$-state must be held by one agent in a stable reduced configuration, C_h^{-s} is not stable. Let C_{u_0}, $C_{u_0}^{-s}$, and $C_{u_0}^{+s}$ be C_h, C_h^{-s}, and C_h^{+s}, respectively. There is a reduced execution segment $C_{u_0}^{-s}$, e_0^{-s}, C_{u_1} such that any agent does not have s except C_{u_1}, and a single agent has s in C_{u_1}. This is because $C_{u_0}^{-s}$ is not stable, and state s must be held by one agent in a stable reduced configuration. Then, by Lemmas 16 and 17, we can construct C_{u_0}, e_0, $C_{u_1}^{-s}$. Since every $\overline{\gamma(m)}$-state must be held by one agent in a stable reduced configuration, $C_{u_1}^{-s}$ is not stable. By repeating similar arguments, for any $i \geq 0$, we can construct $C_{u_i}^{-s}$, e_i^{-s}, $C_{u_{i+1}}$ and C_{u_i}, e_i, $C_{u_{i+1}}^{-s}$. Because $C_{u_0}^{-s}$ can be reached from some initial configuration, by combining the above execution segments, we can construct a weakly fair execution of Alg_{sym} so that an agent with s disappears infinitely often. Since every $\overline{\gamma(m)}$-state must be held by one agent in a stable reduced configuration, such an execution cannot solve the uniform 2-partition and thus the lemma holds. Now we present the details of the proof. For contradiction, there exists a sink state $m \in Q_p$. We consider a population $V = \{a_0, a_1, a_2, \ldots, a_P\}$ of P agents and an initialized base station, where a_0 is the base station. First, we consider an execution segment $e = C_0, C_1, C_2, \ldots, C_h$ such that,

- a particular agent a_x has state m in C_0 and does not interact during e;
- $P - 1$ other agents (and the base station) interact until convergence, which implies that, by Lemma 15, the number of $\gamma(m)$-agents in $V - \{a_x\}$ is $P/2 - 1$ and the number of $\overline{\gamma(m)}$-agents in $V - \{a_x\}$ is $P/2$ after some configuration of e; and
- C_h is a reduced configuration.

In C_h, by Lemma 15, the number of $\gamma(m)$-agents is $P/2$ and the number of $\overline{\gamma(m)}$-agents is $P/2$ because additional agent a_x has state m. Note that all $\overline{\gamma(m)}$-states are non-sink states and, by Lemma 14, the number of $\overline{\gamma(m)}$-states is $P/2$. Since C_h is reduced, no two agents have the same non-sink state and thus every $\overline{\gamma(m)}$-state is held by exactly one agent in C_h. Let s be a $\overline{\gamma(m)}$-state. We consider three configurations: $C_{u_0} = C_h$, $C_{u_0}^{+s}$, and $C_{u_0}^{-s}$. Since a_x does not interact in e, $C_{u_0}^{+s}$ can be obtained by the same interactions in e if a_x has state s in the initial configuration (note that this execution may not be a reduced execution). In $C_{u_0}^{+s}$, since every $\overline{\gamma(m)}$-state is held by exactly one agent in $V - \{a_x\}$, there exists exactly one agent $a_s \neq a_x$ with s. Hence, we can obtain a reduced configuration $C_{u_0}^{-s}$ by reducing $C_{u_0}^{+s}$. Note that, since the number of $\overline{\gamma(m)}$-states is $P/2$, every $\overline{\gamma(m)}$-state must be held by one agent in any stable reduced configuration with P agents. Hence, since $C_{u_0}^{-s}$ is reduced and no agent has state s in $C_{u_0}^{-s}$, $C_{u_0}^{-s}$ is not stable. Hence, there exists an execution segment from $C_{u_0}^{-s}$ that leads to a stable configuration where some agent has state s. This implies that we can construct a reduced execution segment $\epsilon_1 = C_{u_0}^{-s}$, e_0^{-s}, C_{u_1} of Alg_{sym} starting from $C_{u_0}^{-s}$ as follows:

- C_{u_1} is reduced and exactly one agent a_y has state s in C_{u_1}.
- For any reduced configuration in e_0^{-s}, no agent has state s.

Moreover, by Lemma 16, since a_x has m in $C_{u_0}^{-s}$, we can construct ϵ_1 such that a_x does not join interactions. Hence, by Lemma 17, we can construct a reduced execution segment $\epsilon'_1 = C_{u_0}$, e_0, $C^{-s}_{u_1}$ of Alg_{sym}. Note that, in $C^{-s}_{u_1}$, a_y has state m. Similarly, we can construct a reduced execution segment $\epsilon_2 = C^{-s}_{u_1}$, e^{-s}_1, C_{u_2} such that

- C_{u_2} is reduced and exactly one agent a_z has state s in C_{u_2}; and
- for any reduced configuration in e^{-s}_1, no agent has state s.

By Lemma 16, we can construct ϵ_2 such that a_y does not join interactions. Hence, by Lemma 17, we can construct a reduced execution segment $\epsilon'_2 = C_{u_1}$, e_1, $C^{-s}_{u_2}$ of Alg_{sym}. By repeating similar arguments, we can construct an infinite execution segment $\epsilon^* = C^{-s}_{u_0}$, e^{-s}_0, C_{u_1}, e_1, $C^{-s}_{u_2}$, e^{-s}_2, C_{u_3}, ... Recall that, for $i \geq 0$, $C^{-s}_{u_i}$ is not stable because every $\overline{\gamma(m)}$-state must be held by one agent in a stable reduced configuration. Thus, ϵ^* cannot reach a stable configuration. As described above, there exists an execution segment e_{ini} that reaches $C^{-s}_{u_0}$ from some initial configuration. Hence, we can construct an execution $\Xi = e^{ini}$, $C^{-s}_{u_0}$, e^{-s}_0, C_{u_1}, e_1, $C^{-s}_{u_2}$, e^{-s}_2, C_{u_3}, ... that does not reach a stable configuration. The remaining step is to show that we can construct Ξ so that Ξ is weakly fair. Recall how to construct an execution segment $\epsilon_i = C^{-s}_{u_{i-1}}$, e^{-s}_{i-1}, C_{u_i}. Since $C^{-s}_{u_i}$ is reduced, any pair of agents can interact at the first interaction of ϵ_i. Consequently, we can construct $\epsilon_1, \epsilon_2, \ldots$ so that, for every pair of agents a and b in V, a interacts with b infinitely often in the first interactions of $\epsilon_1, \epsilon_2, \ldots$. Additionally, when a interacts with b in ϵ_i, a also interacts with b in ϵ'_i. Hence, we can construct Ξ so that, for every pair of agents a and b in V, a interacts with b infinitely often. This implies that Ξ is weakly fair, but Ξ cannot solve the problem. This is a contradiction. □

Finally, we prove that, even if a sink pair exists in Q_p, Alg_{sym} does not work.

Lemma **19.** *A sink pair does not exist in Q_p.*

Proof. For contradiction, we assume that there exists a sink pair in Q_p. Let m_1 and m_2 be a sink pair. We consider two cases: (1) $\gamma(m_1) = \gamma(m_2)$ holds; or (2) $\gamma(m_1) \neq \gamma(m_2)$ holds. First, we consider the case in which $\gamma(m_1) = \gamma(m_2)$ holds. By Corollary 6, the number of *red* (resp., *blue*) states not in the sink pair is $P/2 - 1$ (resp., $P/2 - 1$). Since we assume $|Q_{blue}| \leq |Q_{red}|$, $\gamma(m_1) = \gamma(m_2) = red$ holds. Hence, $|Q_{blue}| = P/2 - 1$ and $|Q_{red}| = P/2 + 1$ hold. We consider a reduced weakly fair execution Ξ^* of Alg_{sym} with P agents and an initialized base station. Since Ξ^* is a reduced execution, a stable reduced configuration occurs infinitely often in Ξ^*. Every state not in the sink pair is held by at most one agent in any reduced configuration, and thus, at most $P/2-1$ *blue* agents exist in any reduced configuration in Ξ^*. Hence, any reduced configuration in Ξ^* is not stable. This is a contradiction. Therefore, $\gamma(m_1) = \gamma(m_2)$ does not hold (i.e., $\gamma(m_1) \neq \gamma(m_2)$ holds). Next, we consider the case in which $\gamma(m_1) \neq \gamma(m_2)$ holds. The idea of the proof is as follows: We consider an execution Ξ with $P-1$ agents. Let m^* be a state in the sink pair such that the number of $\gamma(m^*)$-agents is $P/2$ in a stable reduced configuration of Ξ. This implies that there exists some agent with m^* in the configuration. Next, we consider an execution with P agents such that one additional agent has m^* as an initial state and other agents behave similarly to Ξ. In the execution, the additional agent does not join the interactions until $P-1$ agents converge to a stable reduced configuration in Ξ. At that time, one of the $P-1$ agents has state m^* and the additional agent also has state m^*. We can prove that, from this configuration, $P-1$ agents cannot recognize the additional agent and hence they make the same behavior as in Ξ. Moreover, the additional agent can keep state m^*. Since the additional agent has $\gamma(m^*)$-state, the number of $\gamma(m^*)$-agents is $P/2+1$ and the number of $\overline{\gamma(m^*)}$-agents is $P/2-1$. This implies that the uniform 2-partition problem cannot be solved. Now we present the details of the proof. Without loss of generality, we assume that $\gamma(m_1) = red$ and $\gamma(m_2) = blue$ hold. We consider a population $V = \{a_0, a_1, a_2, \ldots, a_{P-1}\}$ of $P-1$ agents and an initialized base station, where a_0 is the base station. We define a reduced weakly fair execution $\Xi = C_0, C_1, C_2, \ldots, C_t, \ldots, C_{t'_0}, e_1, C_{t'_1}, e_2, C_{t'_2}, e_3, \ldots$ of Alg_{sym} with V as follows:

- C_t is a stable configuration.
- For any $u \geq 0$, $C_{t'_u}$ is a particular stable reduced configuration such that $C_{t'_0} = C_{t'_1} = C_{t'_2} = \cdots$ holds. Note that such a configuration (i.e., a stable reduced configuration that appears an infinite number of times) exists because the number of possible configurations is finite.
- For $j > 0$, e_j is an execution segment such that, during execution segment $C_{t'_{j-1}}, e_j, C_{t'_j}$, for any pair of agents a and b in V, a interacts with b at least

once. This is possible because Ξ is weakly fair.

Let m^* be a state in the sink pair such that the number of $\gamma(m^*)$-agents is $P/2$ in $C_{t'_0}$. Note that m^* is uniquely chosen because the number of agents is $P-1$. Since every state not in the sink pair is held by at most one agent in a reduced configuration, there exists an agent a_q that has state m^* in $C_{t'_0}$. Subsequently, we consider a population $V' = \{a'_0, a'_1, a'_2, \dots, a'_P\}$ of P agents and an initialized base station, where a'_0 is the base station. We define a reduced weakly fair execution $\Xi' = C'_0, C'_1, C'_2, \dots, C'_t, \dots C'_{t'_0}, e'_1, C'_{t'_1}, e'_2, C'_{t'_2}, e'_3, \dots$ of Alg_{sym} with V'. First, we define the first part of Ξ', that is, $C'_0, C'_1, C'_2, \dots, C'_t, \dots C'_{t'_0}$ as follows:

- In initial configuration C'_0, $a'_0, \dots, a'_{P-1}$ have the same states as $a_0, \dots, a_{P-1}$ in C_0 and a'_P has state m^*. Formally, $s(a'_i, C'_0) = s(a_i, C_0)$ holds for i $(P-1 \geq i \geq 0)$, and $s(a'_P, C'_0) = m^*$ holds.

- From C'_0 to $C'_{t'_0}$, $a'_0, \dots, a'_{P-1}$ interact similarly to $a_0, \dots, a_{P-1}$ in Ξ. Formally, for any $u(t'_0 > u \geq 0)$, when a_g interacts with a_h at $C_u \to C_{u+1}$, a'_g interacts with a'_h at $C'_u \to C'_{u+1}$.

Clearly, $s(a'_i, C'_{t'_0}) = s(a_i, C_{t'_0})$ holds for i $(P-1 \geq i \geq 0)$ and $s(a'_P, C'_{t'_0}) = m^*$ hold. This implies that the number of $\gamma(m^*)$-agents is $P/2+1$ and the number of $\overline{\gamma(m^*)}$-agents is $P/2-1$ in $C'_{t'_0}$. Then, we construct the remaining part of Ξ'; that is, $C'_{t'_0}, e'_1, C'_{t'_1}, e'_2, C'_{t'_2}, e'_3, \dots$ as follows:

- For $j > 0$, we construct an execution segment $e'_j = \hat{C}^j_1, \hat{C}^j_2, \hat{C}^j_3, \dots, \hat{C}^j_z, \hat{C}^j_{z+1}, \hat{C}^j_{z+2}$ using $e_j = C^j_1, C^j_2, C^j_3, \dots, C^j_z$, where $z = |e_j|$ holds. Specifically, we construct $C'_{t'_{j-1}}, e'_j, C'_{t'_j}$ as follows:
  - Case in which j is even: In this case, agents $a'_0, \dots, a'_{P-1}$ interact in execution segment $C'_{t'_{j-1}}, \hat{C}^j_1, \hat{C}^j_2, \dots, \hat{C}^j_{z+1}$ similarly to $a_0, \dots, a_{P-1}$ in execution segment $C_{t'_{j-1}}, e_j, C_{t'_j}$. Formally, when a_g interacts with a_h at $C^j_f \to C^j_{f+1}$ for $z > f > 0$, (resp., $C_{t'_{j-1}} \to C^j_1$, and $C^j_z \to C_{t'_j}$), a'_g interacts with a'_h at $\hat{C}^j_f \to \hat{C}^j_{f+1}$ (resp., $C'_{t'_{j-1}} \to \hat{C}^j_1$ and $\hat{C}^j_z \to \hat{C}^j_{z+1}$).
  - Case in which j is odd: In this case, a'_P joins interactions instead of a'_q. Note that in $C'_{t'_{j-1}}$, both a'_P and a'_q have state m^*. Formally we construct e'_j as follows: (1) When $a_g (g \neq q)$ interacts with $a_h (h \neq q)$ at

$C_f^j \to C_{f+1}^j$ for $z > f > 0$ (resp., $C_{t'_{j-1}} \to C_1^j$, and $C_z^j \to C_{t'_j}$), a'_g interacts with a'_h at $\hat{C}_f^j \to \hat{C}_{f+1}^j$ (resp., $C'_{t'_{j-1}} \to \hat{C}_1^j$, and $\hat{C}_z^j \to \hat{C}_{z+1}^j$). (2) When a_q interacts with an agent $a_i (i \neq q)$ at $C_f^j \to C_{f+1}^j$ (resp., $C_{t'_{j-1}} \to C_1^j$ and $C_z^j \to C_{t'_j}$), a'_P interacts with a'_i at $\hat{C}_f^j \to \hat{C}_{f+1}^j$ (resp., $C'_{t'_{j-1}} \to \hat{C}_1^j$ and $\hat{C}_z^j \to \hat{C}_{z+1}^j$). Similarly, when an agent $a_i (i \neq q)$ interacts with a_q at $C_f^j \to C_{f+1}^j$ (resp., $C_{t'_{j-1}} \to C_1^j$ and $C_z^j \to C_{t'_j}$), a'_i interacts with a'_P at $\hat{C}_f^j \to \hat{C}_{f+1}^j$ (resp., $C'_{t'_{j-1}} \to \hat{C}_1^j$ and $\hat{C}_z^j \to \hat{C}_{z+1}^j$).

– a'_P interacts with a'_q at $\hat{C}_{z+1}^j \to \hat{C}_{z+2}^j$. Additionally, a'_q interacts with a'_P at $\hat{C}_{z+2}^j \to C'_{t'_j}$.

We can inductively show that, for any $x \geq 0$, $s(a'_i, C'_{t'_x}) = s(a_i, C_{t'_x})$ holds for any i $(P-1 \geq i \geq 0)$, and $s(a'_q, C'_{t'_x}) = s(a'_P, C'_{t'_x}) = s(a_q, C_{t'_x}) = m^*$ holds. Clearly, this holds for $x = 0$. We assume that this holds for $x = j$, and consider the case of $x = j+1$. When $j+1$ is even, during execution segment $C'_{t'_j}, \hat{C}_1^{j+1}, \hat{C}_2^{j+1}, \ldots, \hat{C}_{z+1}^{j+1}$, agents in $V' - \{a'_P\}$ interact similarly to $C_{t'_j}$, e_{j+1}, $C_{t'_{j+1}}$. Hence, $s(a'_i, \hat{C}_{z+1}^{j+1}) = s(a_i, C_{t'_{j+1}})$ holds for any i $(P-1 \geq i \geq 0)$ and $s(a'_q, \hat{C}_{z+1}^{j+1}) = s(a'_P, \hat{C}_{z+1}^{j+1}) = s(a_q, C_{t'_{j+1}}) = m^*$ holds. a'_P and a'_q interact at $\hat{C}_{z+1}^{j+1} \to \hat{C}_{z+2}^{j+1}$ and $\hat{C}_{z+2}^{j+1} \to C'_{t'_{j+1}}$. By the assumption of m^*, if two agents with m^* interact twice, they keep their state m^*. Since a'_P and a'_q have state m^* in $\hat{C}_{z+1}^{j+1}$, they also have state m^* in $C'_{t'_{j+1}}$, and thus, $\hat{C}_{z+1}^{j+1}$ is equal to $C'_{t'_{j+1}}$. Thus, $s(a'_i, C'_{t'_{j+1}}) = s(a_i, C_{t'_{j+1}})$ holds for any i $(P-1 \geq i \geq 0)$ and $s(a'_q, C'_{t'_{j+1}}) = s(a'_P, C'_{t'_{j+1}}) = s(a_q, C_{t'_{j+1}}) = m^*$ holds. When $j+1$ is odd, during execution segment $C'_{t'_j}, \hat{C}_1^{j+1}, \hat{C}_2^{j+1}, \ldots, \hat{C}_{z+1}^{j+1}$, a'_P interacts instead of a'_q, and agents in $V' - \{a'_q\}$ behave similarly to $C_{t'_{j-1}}$, e_j, $C_{t'_j}$. Hence, $s(a'_i, \hat{C}_{z+1}^{j+1}) = s(a_i, C_{t'_{j+1}})$ holds for any i $(P-1 \geq i \geq 0)$ and $s(a'_q, \hat{C}_{z+1}^{j+1}) = s(a'_P, \hat{C}_{z+1}^{j+1}) = s(a_q, C_{t'_{j+1}}) = m^*$ holds. Similarly, a'_P and a'_q interact at $\hat{C}_{z+1}^{j+1} \to \hat{C}_{z+2}^{j+1}$ and $\hat{C}_{z+2}^{j+1} \to C'_{t'_{j+1}}$, and they keep their state m^*. Thus, $s(a'_i, C'_{t'_{j+1}}) = s(a_i, C_{t'_{j+1}})$ holds for any i $(P-1 \geq i \geq 0)$ and $s(a'_q, C'_{t'_{j+1}}) = s(a'_P, C'_{t'_{j+1}}) = s(a_q, C_{t'_{j+1}}) = m^*$ holds. By the assumption, for any j, the number of $\gamma(m^*)$-agents is one more than the number of $\overline{\gamma(m^*)}$-agents in $C_{t'_j}$. Thus, for any j, the number of $\gamma(m^*)$-agents is two more than the number of $\overline{\gamma(m^*)}$-agents in $C'_{t'_j}$. Hence, Ξ' never converges to a stable configuration. During the execution segment $C'_{t'_{j-1}}$, e'_j, when j is even, for every pair of agents a and b in $V' - \{a'_P\}$, a interacts with b. Similarly, when j is odd, for every pair of agents a

and b in $V' - \{a'_q\}$, a interacts with b. Moreover, for $j > 0$, at $\hat{C}^j_{z+1} \to \hat{C}^j_{z+2}$ (resp., $\hat{C}^j_{z+2} \to C'_{t'_j}$), a'_P (resp., a'_q) interacts with a'_q (resp., a'_P). Thus, although Ξ' is weakly fair, Ξ' cannot solve the problem. This is a contradiction. Therefore, the lemma holds. $\square$

By Lemma 13, there exists either a sink pair or sink state. However, in both cases, Alg_{sym} does not work. Therefore, we have proved the theorem.

4.2 Upper Bounds for Initialized Base Station

In this section, we propose both asymmetric and symmetric protocols for the uniform k-partition problem. The asymmetric protocol requires P states and the symmetric protocol requires $P+1$ states. By the lower bounds, these protocols are space-optimal.

4.2.1 Upper Bound for Asymmetric Protocols

In this subsection, we describe a P-state asymmetric protocol for the uniform partition problem. The idea of the protocol is to assign states 0, 1, …, $n-1$ to n agents individually and then regard an agent with state s as a member of the (s mod k)-th group. One may think that, to implement this idea, we can directly use a naming protocol [23], where the naming protocol assigns different states to agents using P states if $n \le P$ holds. In fact, if $n = P$ holds, the naming protocol assigns states 0, 1, …, $P-1$ to P agents individually and hence it achieves the uniform partition. However, if $n < P$ holds, the naming protocol does not always achieve the uniform partition. For example, in the case of $(n-1)k < P$, the naming protocol may assign states 0, k, $2k$, …, $(n-1)k$ to n agents individually, which implies that all agents are in the 0-th group.

Algorithm 2 shows a P-state asymmetric protocol for the uniform k-partition problem. Note that this protocol does not take into account whether an agent is an initiator or responder if two agents with different states interact. If two agents with the same state interact (line 8), one of them may change its state. In the protocol, the base station assigns states 0, 1, …, $n-1$ to n agents individually. To do this, the base station maintains variable M, which represents the state the base station will assign next. The base station sets $M = 0$ initially, and

Algorithm 2 Asymmetric uniform k-partition protocol

```
A variable at the base station
    $M$: The state that the base station assigns next, initialized to 0
A variable at an agent $a$:
    $S_a \in \{0, 1, 2, \ldots, P-1\}$: The agent state, initialized arbitrarily. Agent $a$
    belongs to the $(S_a \mod k)$-th group.
1: when an agent $a$ and the base station interact do
2:     if $M \leq S_a$ then
3:         $S_a = M$
4:         $M = M + 1$
5:     end if
6: end
7: when two agents $a$ and $b$ interact do
8:     if $S_a = S_b$ and $S_a < P - 1$ then
9:         $S_a = S_a + 1$
10:    end if
11: end
```

increments M whenever it assigns M to an agent. We consider an interaction between the base station and an agent with state x. If x is smaller than M, the base station identifies that it has already assigned a state to the agent, and hence it does not update the state. If x is M or larger, the base station assigns state M to the agent and increments M. When the base station assigns state x to an agent, there may exist another agent with state x because of arbitrary initial states. To consider this case, when two agents with the same state x interact, one enters state $x+1$ and the other keeps its state x. By repeating such interactions, eventually, exactly one agent has state x. Using this behavior, the base station eventually assigns states 0, 1, ..., $n-1$ to n agents individually, and hence the protocol achieves the uniform k-partition.

As a result, we obtain the following theorem.

Theorem **13.** *Algorithm 2 solves the uniform k-partition problem. This means that, in the model with an initialized base station, there exists an asymmetric protocol with P states and arbitrary initial states that solves the uniform k-partition problem under weak fairness, where P is the known upper bound of the number of agents.*

To prove the theorem, we show the following two lemmas.

Lemma **20.** *Let $\Xi = C_0, C_1, \ldots$ be a weakly-fair execution of Algorithm 2. In configuration C_i ($i \geq 0$), for any s with $0 \leq s \leq M-1$, at least one agent with state s exists.*

Proof. We prove the lemma by induction on the index of a configuration in execution Ξ. The base case is vacuously true because M is initialized to 0 in the initial configuration C_0. For the induction step, we assume that the lemma holds at $C_k(0 \leq k)$; that is, at C_k, at least one agent with state s exists for any s with $0 \leq s \leq M-1$. We consider two cases of interaction at transition $C_k \rightarrow C_{k+1}$. First, we consider the case in which the base station joins the interaction. If the base station interacts with an agent with state less than M, the base station and the agent do not change their states. If the base station interacts with an agent with state M or more, the base station assigns M to the agent and then increases M. Thus, the lemma holds at C_{k+1}. Next, we consider the case in which the base station does not join the interaction. When two agents interact, at least one agent keeps its state. Because M is not changed, the lemma holds at C_{k+1}. □

Lemma **21.** *Let $\Xi = C_0, C_1, \ldots$ be a weakly-fair execution of Algorithm 2. There exists a configuration C_i such that $M = n$ holds.*

Proof. By way of contradiction, we assume that M does not become n in Ξ. Because M is monotonically increasing, the base station eventually stops updating M. Let $l < n$ be the last value of M, and C_j be the first configuration with $M = l$. First, we can say that all agents have states less than l after C_j; otherwise; some agent a' with state l' ($l' \geq l$) exists after C_j. From the algorithm, a' never decreases its state unless a' and the base station interact. Because Ξ is weakly fair, a' and the base station eventually interact. At that time, a' has state at least $l' \geq l = M$, and consequently, the base station increases M. This contradicts the assumption. From Lemma 20, for each s with $0 \leq s \leq l-1$, at least one agent with state s exists at configuration C_j. Additionally, because each agent has one of states 0 to $l-1$ with $l < n$, at least two agents have state q for some $q < l$. From the algorithm, an agent with state less than l $(= M)$ changes its state only by an interaction with the homonyms. Hence, two agents with state q eventually interact, and then one of them enters state $q+1$. If $q+1 \leq l-1$ holds, at least two agents with state $q+1$ exist and similarly, one of them enters state $q+2$. Hence, eventually, some agent enters state l. This is a contradiction. □

From Lemma 21, the base station eventually sets $M = n$. From Lemma 20, when $M = n$ holds, for each s with $0 \leq s \leq n - 1$, exactly one agent has state s. This implies that, after $M = n$ holds, line 2 of the pseudocode is never executed, and thus the base station never updates M. Clearly, the configuration achieves the uniform partition and no agent subsequently updates its state. Therefore, Theorem 13 holds.

Remark. Interestingly, when P is odd, Algorithm 2 solves the uniform 2-partition, even if the number of agent states is $P - 1$. Specifically, let $S_a \in \{1, 2, 3, \dots, P - 1\}$ be a set of agent states, and initialize variable M to 1. Then, Algorithm 2 converges to a configuration such that there exist two agents with state $P - 1$ (and other states are held by exactly one agent). This is because, in the algorithm, the base station assigns $P - 1$ agents to $P - 1$ states individually, and because the algorithm works under weak fairness, the remaining agent shifts its state until state $P - 1$. In the configuration, the difference in the numbers of *red* and *blue* agents is one. Moreover, each agent does not change its state after the configuration. Hence, the uniform 2-partition is solved.

4.2.2 Upper Bound for Symmetric Protocols

In this subsection, we propose a $(P+1)$-state symmetric protocol for the uniform partition problem. We can easily obtain the protocol using a scheme proposed in [15]. In [15], a P-state symmetric protocol for the counting problem was proposed. The counting protocol assigns different states in $\{1, \dots, n\}$ to n agents and keeps the configuration if $n < P$ holds. Hence, by regarding $P + 1$ as the upper bound of the number of agents and allowing $P + 1$ states, the protocol assigns different states in $\{1, ..., n\}$ to n agents for any $n \leq P$. This implies that, as in the previous subsection, the protocol can achieve the uniform partition by regarding an agent with x as a member of the $(x \mod k)$-th group.

Theorem **14.** *In the model with an initialized base station, there exists a symmetric protocol with $P + 1$ states and arbitrary initial states that solves the uniform k-partition problem under weak fairness, where P is the known upper bound of the number of agents.*

4.3 Impossibility with No Base Station and Designated Initial States for Symmetric Protocols

In the following, we show that the problem cannot be solved with no base station and designated initial states for symmetric protocols.

Theorem **15.** *In the model with no base station, no symmetric protocol with designated initial states solves the 2-partition problem with size difference $x \geq 1$ under weak fairness.*

Proof. By way of contradiction, we assume such a protocol Alg exists. We assume the state set of agents is $Q_p = \{s_1, s_2, \ldots\}$. We consider population $V = \{a_1, \ldots, a_n\}$ of n agents, where n is even and at least $x+1$. Let s_{i_1} be the designated initial state of all agents; that is, $s(a_i) = s_{i_1}$ holds for any i $(1 \leq i \leq n)$ at the initial configuration. Clearly, the symmetric protocol Alg has transition $(s_{i_1}, s_{i_1}) \rightarrow (s_{i_2}, s_{i_2})$ for some s_{i_2}. This implies that if all pairs of agents in state s_{i_1} interact, all agents transition to s_{i_2}. Similarly, if all pairs of agents in state s_{i_2} interact, all agents transition to the same state (say s_{i_3}).

When the above execution is repeated, configurations such that all agents have the same state appear infinitely often. By changing pairs of agents, we can perform the above execution under weak fairness. If all agents are in the same state, such a configuration is not stable because n is at least $x + 1$. This is a contradiction. □

5. Uniform k-partition under Global Fairness on Complete Graphs

In this section, we consider the solvability of the uniform k-partition problem. More concretely, we prove that, with initialized base station and designated initial states on complete graphs, there is no asymmetric protocols with k states. Moreover, we propose a symmetric uniform k-partition protocol with designated initial states under global fairness on complete graphs.

5.1 Lower Bound for Initialized Base Station with Designated Initial States

Here we show $k+1$ states are necessary to construct an asymmetric protocol with initialized base station and designated initial states. We prove the impossibility similarly to Theorem 3.

Theorem **16.** *In the model with an initialized base station, no asymmetric protocol with k states and designated initial states solves the k-partition problem with size difference* $x \geq 1$ *under global fairness on complete graphs.*

Proof. For contradiction, we assume such a protocol Alg exists. Without loss of generality, we assume $Q_p = \{s_1, s_2, \ldots, s_k\}$, and the designated initial state of all agents is s_1. Let n be a number such that $n = ik$ holds for some $i \geq x+1$. We consider the following three cases.

First, for population V of a single base station and n (non base station) agents $a_1, a_2, \ldots, a_n$, consider an execution $\Xi = C_0, C_1, \ldots$ of Alg. According to the definition, there exists a stable configuration C_t. That is, after C_t, the state of each agent does not change even if the base station and agents in states s_1, s_2, $\ldots$, s_k interact in any order.

Next, for population V' of a single base station and $n+2x+1$ agents a_1, a_2, $\ldots$, a_{n+2x+1}, we define an execution $\Xi' = C'_0, C'_1, \ldots, C'_t, C'_{t+1}, \ldots$ of Alg as follows.

- From C'_0 to C'_t, the base station and n agents a_1, a_2, $\ldots$, a_n interact in the same order as the execution Ξ.
- After C'_t, the base station and $n+2x+1$ agents interact so as to satisfy global fairness.

Since the base station and agents $a_1, \ldots, a_n$ change their states similarly to Ξ from C'_0 to C'_t, the number of agents in state s_1 is $x+1$ or more than the number of agents in state s_y for $2 \leq y \leq k$ at C'_t. Moreover, the state of the base station at C'_t is the same as the state of the base station at C_t. However, since the difference in the numbers of agents with s_1 and other states is at least $x+1$, C'_t is not a stable configuration. Consequently, after C'_t, some agent changes its state in execution Ξ'.

Lastly, we consider execution Ξ for population V again. Here, we consider interactions after stable configuration C_t, and apply interactions in Ξ' to execution Ξ. That is, we consider the following execution after C_t: 1) when the base station and an agent in state $s \in \{s_1, s_2, \ldots, s_k\}$ interact at $C'_u \to C'_{u+1}$ $(u \geq t)$ in Ξ', the base station and an agent in state s interact at $C_u \to C_{u+1}$ in Ξ, and 2) when two agents in states $s \in \{s_1, s_2, \ldots, s_k\}$ and $s' \in \{s_1, s_2, \ldots, s_k\}$ interact at $C'_u \to C'_{u+1}$ $(u \geq t)$ in Ξ', two agents in states s and s' interact at $C_u \to C_{u+1}$ in Ξ. We can construct such an execution because, after stable configuration C_t, at least two agents have each s_1, s_2, $\ldots$, s_k. In this execution Ξ, since interactions occur similarly to Ξ', some agent changes its state similarly to Ξ' after C_t. This is a contradiction because C_t is a stable configuration. $\square$

5.2 Upper Bound for Uniform k-partition Protocol

Here we show a symmetric uniform k-partition protocol with designated initial states under global fairness on complete graphs. The summary of the protocol is given in Algorithm 3.

In this protocol, a set of agent states is divided into four subsets, i.e., $Q = I \cup G \cup M \cup D$, where $I = \{initial, initial'\}$, $G = \{g_1, g_2, \ldots, g_k\}$, $M = \{m_2, m_3, \ldots, m_{k-1}\}$, and $D = \{d_1, d_2, \ldots, d_{k-2}\}$. The designated initial state of agents is *initial*, that is, the state of every agent is *initial* in the initial configuration. State g_i in G indicates that the agent belongs to the i-th group, that is, $\gamma(g_i) = i$ holds for any $g_i \in G$. For other state s, we define $\gamma(s)$ as follows:

- $\gamma(ini) = 1$ holds for any $ini \in I$.
- $\gamma(d_i) = 1$ holds for any $d_i \in D$.
- $\gamma(m_i) = i$ holds for any $m_i \in M$.

We say an agent is free if its state is in I. We define $\overline{initial} = initial'$ and $\overline{initial'} = initial$.

We will describe the details of the protocol in Sections 5.2.1 and 5.2.2. In the basic strategy (Section 5.2.1), the protocol makes k agents enter states $g_1, g_2, \ldots, g_k$ by using states in M as intermediate states. However, this strategy may increase

```
Algorithm 3 Uniform $k$-partition protocol
A state set
    $Q = I \cup G \cup M \cup D$ where
      $I = \{initial, initial'\}$,
      $G = \{g_1, g_2, \ldots, g_k\}$,
      $M = \{m_2, m_3, \ldots, m_{k-1}\}$, and
      $D = \{d_1, d_2, \ldots, d_{k-2}\}$.
A mapping function to groups
    $\gamma(ini) = 1$ holds for any $ini \in I$.
    $\gamma(g_i) = i$ holds for any $g_i \in G$.
    $\gamma(m_i) = i$ holds for any $m_i \in M$.
    $\gamma(d_i) = 1$ holds for any $d_i \in D$.
Transition rules
     1. $(initial, initial) \to (initial', initial')$
     2. $(initial', initial') \to (initial, initial)$
     3. $(d_i, ini) \to (d_i, \overline{ini})$ ($d_i \in D$ and $ini \in I$)
     4. $(g_i, ini) \to (g_i, \overline{ini})$ ($g_i \in G$ and $ini \in I$)
     5. $(initial, initial') \to (g_1, m_2)$
     6. $(ini, m_i) \to (g_i, m_{i+1})$ ($ini \in I$ and $2 \le i \le k-2$)
     7. $(ini, m_{k-1}) \to (g_{k-1}, g_k)$ ($ini \in I$)
     8. $(m_i, m_j) \to (d_{i-1}, d_{j-1})(2 \le i, j \le k-1)$
     9. $(d_i, g_i) \to (d_{i-1}, initial)(2 \le i \le k-2)$
    10. $(d_1, g_1) \to (initial, initial)$
```

the number of agents in some groups beyond n/k. In Section 5.2.2, we overcome such a situation by using states in D.

5.2.1 Basic strategy

The basic strategy of the protocol is as follows: First two free agents transition to states g_1 and m_2. After that, for each i ($2 \le i \le k-2$), when an agent in state m_i and a free agent interact, they transition to states m_{i+1} and g_i, respectively. Lastly, when an agent in state m_{k-1} and a free agent interact, they transition to states g_k and g_{k-1}. By this behavior, k free agents can change their states to $g_1, g_2, \ldots, g_k$. That is, the size of each group is increased by one. To achieve this, the protocol includes the following transitions.

1. $(initial, initial) \to (initial', initial')$
2. $(initial', initial') \to (initial, initial)$
3. $(d_i, ini) \to (d_i, \overline{ini})$ ($d_i \in D$ and $ini \in I$)
4. $(g_i, ini) \to (g_i, \overline{ini})$ ($g_i \in G$ and $ini \in I$)
5. $(initial, initial') \to (g_1, m_2)$
6. $(ini, m_i) \to (g_i, m_{i+1})$ ($ini \in I$ and $2 \le i \le k-2$)
7. $(ini, m_{k-1}) \to (g_{k-1}, g_k)$ ($ini \in I$)

First we explain transitions 1 to 5, which make two free agents transition to states g_1 and m_2. Recall that all agents are in state *initial* in the initial configuration. Since we consider symmetric protocols, two agents in state *initial* cannot transition to states g_1 and m_2 at one interaction. This is the reason why we introduce state *initial*$'$. Each agent in state *initial* (resp., *initial*$'$) transitions to *initial*$'$ (resp., *initial*) when it interacts with an agent in a state in $I \cup D \cup G$ (except for interaction between one in state *initial* and one in state *initial*$'$). Transition 5 implies that, when agents in states *initial* and *initial*$'$ interact, they become g_1 and m_2, respectively. From global fairness, if at least two free agents and no agents in a state in M exist, two free agents eventually enter states *initial* and *initial*$'$, respectively, and then enter states g_1 and m_2 by an interaction. Transition 6 implies that, when a free agent and an agent in state m_i interact, they become g_i and m_{i+1}, respectively. By these transitions, free agents transition to states $g_1, \dots, g_{k-2}$ one by one. After that, from transition 7, when a free agent and an agent in state m_{k-1} interact, they become g_{k-1} and g_k, respectively. From this behavior, the size of each group is increased by one.

Figure 3 is an example execution of the protocol for a population of six agents. Initially all agents are in state *initial* (Fig. 3 (a)). After interactions (a_1, a_2), (a_3, a_4), and (a_5, a_6), all agents enter state *initial*$'$ (Fig. 3 (b)). After interactions (a_1, a_6), (a_2, a_3), and (a_4, a_5), all agents enter state *initial* (Fig. 3 (c)). If such interactions happen infinitely, the protocol never solves the uniform k-partition problem. However, under the global fairness, such interactions do not occur infinitely. This is because, if some configuration C occurs infinitely often, every

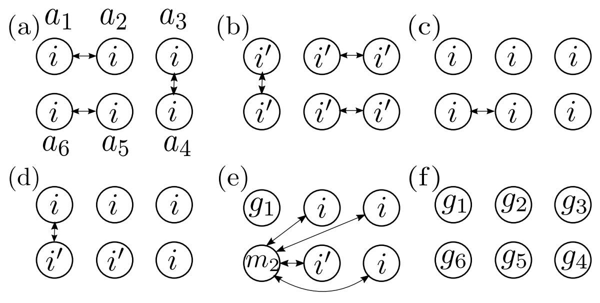


Figure 3. An example of k-partition

configuration reachable from C should occur. That is, eventually interactions (a_5, a_6) and (a_1, a_6) happen in this order from such a configuration (Fig. 3 (d) and (e)). Then, a_1 and a_6 enter states g_1 and m_2, respectively (Fig. 3 (e)). After that, if interactions (a_6, a_2), (a_6, a_3), (a_6, a_4), and (a_6, a_5) occur in this order, agent a_6 changes its state from m_2 to m_3, m_4, m_5, and g_6, and agents a_2, a_3, a_4, and a_5 enter g_2, g_3, g_4, and g_5, respectively (Fig. 3 (f)).

5.2.2 A problem of the basic strategy and its solution

However, in the protocol of the basic strategy, $\lceil n/k \rceil$ or more agents in state m_1 can appear. In this case, the above transitions do not achieve a uniform k-partition. For example, in the case of $n = 12$ and $k = 4$, if four agents enter state m_1, agents can transition to states g_1, g_2, m_3, g_1, g_2, m_3, g_1 ,g_2, m_3, g_1, g_2, m_3. To solve this problem, we introduce states in D and add the following transitions.

8. $(m_i, m_j) \to (d_{i-1}, d_{j-1}) (2 \le i, j \le k-1)$
9. $(d_i, g_i) \to (d_{i-1}, \mathit{initial}) (2 \le i \le k-2)$
10. $(d_1, g_1) \to (\mathit{initial}, \mathit{initial})$

By transition 8, when two agents in states in m_i and m_j interact, they transition to states in d_{i-1} and d_{j-1}, respectively. Intuitively, an agent in state d_i makes agents in $g_1, g_2, \ldots, g_i$ go back to state *initial*. Recall that an agent in state m_{i+1} can enter state d_i and an agent in state m_{i+1} has made agents in states g_1, g_2, $\ldots$, g_i. This means an agent in state d_i initializes agents that it makes enter states g_1, g_2, $\ldots$, g_i. More concretely, an agent in a state in D works as follows:

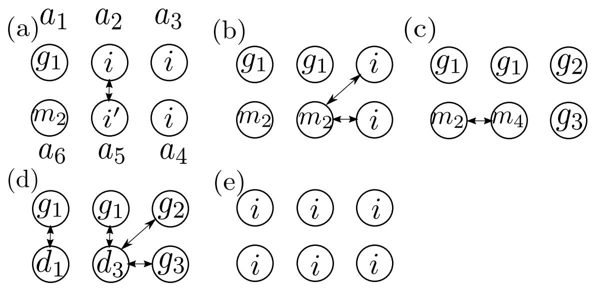


Figure 4. Another example of k-partition

- For $2 \le i \le k-2$, when agents in states d_i and g_i interact, they become d_{i-1} and *initial* by transition 9, respectively.
- After that, from transition 10, when agents in states d_1 and g_1 interact, they become *initial*.

Figure 4 is an example that shows the impact of states in D. Similarly to Fig. 3, agents can transition to a configuration in Fig. 4 (a). If interactions (a_2, a_5), (a_3, a_5), and (a_4, a_5) occur in this order from Fig. 4 (a), agents transition to a configuration in Fig. 4 (c). In this configuration, transitions of the basic strategy (transitions 1 to 7) are not applied. However, transition 8 can be applied, that is, interaction (a_5, a_6) eventually occurs. By the interaction, a_5 and a_6 enter states d_3 and d_1, respectively (Fig. 4 (d)). After that, interactions (a_1, a_6), (a_4, a_5), (a_3, a_5) and (a_2, a_5) happen, and then all agents enter state *initial* (Fig. 4 (e)).

Clearly, agents can repeatedly enter state g_i and go back to *initial* many times. However, after an agent enters state g_k, one set of agents in states $g_1, \ldots, g_k$ never goes back to *initial*. Thus, if there are h agents in state g_k, the number of agents in state g_i is at least h for each i. In addition, when there are h agents in state g_k and $n - kh \ge k$ holds, there is an execution that makes some agent enter state g_k. This implies that, from the global fairness, some agent eventually enters state g_k. When $n - kh = r < k$ holds, there is an execution that makes the remaining agents transition to $g_1, g_2, \ldots, m_r$. From the global fairness, the remaining agents eventually enter these states. In this configuration, agents achieve a uniform k-partition and after that all agents never change their states.

5.2.3 Correctness

In this section, we prove the correctness of the proposed protocol. If $k = 2$, the protocol is exactly the same as a uniform 2-partition protocol in Section 3.2.2. Thus, the protocol solves the uniform k-partition problem for $k = 2$. In the rest of this section, we assume that $k \geq 3$ holds.

First, we define the notations to consider the number of states at a configuration. We denote by $\#ini$ the number of free agents (i.e., agents in states *initial* or *initial′*). We denote by $\#g_x$, $\#m_p$, and $\#d_q$ the numbers of agents in state g_x, m_p, and d_q, respectively ($1 \leq x \leq k$, $2 \leq p \leq k-1$, $1 \leq q \leq k-2$).

The first lemma gives invariants that hold for any configuration reachable from the initial configuration C_0. In the following, when configuration C is reachable from C_0, we simply say C is reachable.

Lemma **22.** *For any reachable configuration C, $\#g_x = \sum_{p=x+1}^{k-1} \#m_p + \sum_{q=x}^{k-2} \#d_q + \#g_k$ holds for any x ($1 \leq x \leq k$) at C.*

Proof. First we intuitively explain the invariants. Let us fix x. An agent in state m_p ($2 \leq p \leq k-1$) has made $p-1$ agents enter $g_1, g_2, \ldots, g_{p-1}$. Hence, for each agent in state m_p with $p > x$, there exists an agent in state g_x that corresponds to the agent. Consequently, there exist $\sum_{p=x+1}^{k-1} \#m_p$ agents in state g_x that correspond to agents in states in M. Since an agent in state d_q has changed its state from m_{q+1} to d_q, it has made q agents enter $g_1, g_2, \ldots, g_q$. Hence, for each agent in state d_q with $q \geq x$, there exists an agent in state g_x that corresponds to the agent. Consequently, there exist $\sum_{q=x}^{k-2} \#d_q$ agents in state g_x that correspond to agents in states in D. An agent in state g_k has made $k-1$ agents enter $g_1, g_2, \ldots, g_{k-1}$. Hence, there exist $\#g_k$ agents in state g_x that correspond to agents in state g_k. Therefore, we have the above invariants.

We prove the lemma formally by induction. First let us consider the initial configuration. Since $\#g_x = 0$, $\#m_p = 0$, and $\#d_q = 0$ hold for any x, p, and q ($1 \leq x \leq k$, $2 \leq p \leq k-1$, $1 \leq q \leq k-2$), the lemma holds.

Next, assume that the lemma holds at some configuration C. We show that, for any C' satisfying $C \rightarrow C'$, the lemma holds at C'. Clearly, if transition 1, 2, 3, or 4 occurs in $C \rightarrow C'$, the lemma holds at C' because $\#g_x$, $\#m_p$, and $\#d_q$ do not change for any x, p, and q ($1 \leq x \leq k$, $2 \leq p \leq k-1$, $1 \leq q \leq k-2$). Hence, we consider the remaining six transitions.

First, we consider the case of transition 5. This transition increases $\#g_1$ and $\#m_2$ by one, and consequently it affects the formula of $x = 1$. Since the left and right sides of the formula increase by one, the lemma holds in this case.

Let us consider the case of transition 6. This transition increases $\#g_i$ and $\#m_{i+1}$ by one, and decreases $\#m_i$ by one. Consequently, it affects the formula of $x \le i$. For $x < i$, since $\sum_{p=x+1}^{k-1} \#m_p$ and $\#g_x$ do not change, the left and right sides of the formula do not change. For $x = i$, both $\#g_i$ and $\sum_{p=i+1}^{k-1} \#m_p$ increase by one, the left and right sides of the formula increase by one. Hence, the lemma holds in this case.

Let us consider the case of transition 7. This transition increases $\#g_{k-1}$ and $\#g_k$ by one, and decreases $\#m_{k-1}$ by one. Consequently, it affects the formula of $x \le k$. For $x < k-1$, since $\sum_{p=x+1}^{k-1} \#m_p$ decreases and $\#g_k$ increases by one, the left and right sides of the formula do not change. For $x = k-1$, both $\#g_k$ and $\#g_{k-1}$ increase by one and $\#m_{k-1}$ is not included in $\sum_{p=x+1}^{k-1} \#m_p$, the left and right sides of the formula increase by one. For $x = k$, the formula always holds. Hence, the lemma holds in this case.

Let us consider the case of transition 8. This transition increases $\#d_{i-1}$ and $\#d_{j-1}$ by one, and decreases $\#m_i$ and $\#m_j$ by one. Consequently, it affects the formula of $x \le \max\{i,j\}-1$. Since this transition increases $\sum_{q=x}^{k-2} \#d_q$ and decreases $\sum_{p=x+1}^{k-1} \#m_p$ by the same number for any $x \le \max\{i,j\}-1$, the lemma holds in this case.

Let us consider the case of transition 9. This transition increases $\#d_{i-1}$ by one, and decreases $\#d_i$ and $\#g_i$ by one. Consequently, it affects the formula of $x \le i$. For $x < i-1$, since $\sum_{q=x}^{k-2} \#d_q$ and $\#g_x$ do not change, the left and right sides of the formula do not change. For $x = i$, both $\#g_i$ and $\sum_{q=x}^{k-2} \#d_q$ decrease by one, the left and right sides of the formula decrease by one. Hence, the lemma holds in this case.

Finally, consider the case of transition 10. Since this transition decreases $\#d_1$ and $\#g_1$ by one, it affects only formula of $x = 1$. Clearly, the left and right sides of the formula decrease by one. Hence, the lemma holds in this case. $\square$

The invariants in Lemma 22 explain some properties of the proposed protocol. For example, $\#g_x \ge \#g_k$ holds for any x $(1 \le x \le k)$. This means the number of agents in each group is at least $\#g_k$. Since $\#g_k$ is never decreased from the protocol, the number of agents in each group is never decreased below $\#g_k$ after that. By Lemmas 23 to 25, we prove that $\#g_k$ eventually becomes $\lfloor n/k \rfloor$. That is, the number of agents in each group eventually becomes $\lfloor n/k \rfloor$.

Lemma **23.** *Let $\mathcal{C}_1$ be a set of all reachable configurations such that $\#ini \geq k$ holds. For any configuration C in $\mathcal{C}_1$, there exists C' such that $C \xrightarrow{*} C'$ holds and $\#g_k$ at C' is increased by one from C.*

Proof. If there exist no agents in state *initial* at C, there exist at least three agents in state *initial'* exist (because of $k \geq 3$). Consequently two of them enter state *initial* by interacting each other (transition 2). Similarly, if there exist no agents in state *initial'* at C, some agents can enter state *initial'* (transition 1). Hence, there exists a reachable configuration from C where at least one agent in state *initial'* and at least one agent in state *initial*. Let a_1 and a_2 be agents in state *initial'* and *initial*, respectively. After a_1 and a_2 interact, they become m_2 and g_1, respectively (transition 5). At this moment, there exist at least $k-2$ agents in state *initial* or *initial'*. After that, these $k-2$ agents can interact with a_2 one by one. As a result, these $k-2$ agents enter $g_2, g_3, \ldots, g_{k-1}$, and a_2 enters g_k (transitions 6 and 7). Therefore, $\#g_k$ is increased by one from C. □

Lemma **24.** *Let $\mathcal{C}_2$ be a set of all reachable configurations such that $\#ini < k$ and $n - k \cdot \#g_k \geq k$ hold. For any configuration C in $\mathcal{C}_2$, there exists C' such that $C \xrightarrow{*} C'$ holds and $\#g_k$ at C' is increased by one from C.*

Proof. We prove that, from C, there exists a transition such that 1) $\#g_k$ is increased by one or 2) $\#ini$ is increased. In the former case, the lemma directly holds. In the latter case, since $\#g_k$ is not increased, $n - k \cdot \#g_k \geq k$ still holds. Consequently, we can repeatedly apply this claim, and eventually $\#ini$ exceeds k or $\#g_k$ is increased by one. If $\#ini$ exceeds k, $\#g_k$ is eventually increased from Lemma 23. Therefore, the lemma holds.

To prove the above claim, we divide $\mathcal{C}_2$ into the following four sets of configurations $\mathcal{C}_d$, $\mathcal{C}_{m2}$, $\mathcal{C}_{m1}$, and $\mathcal{C}_{m0}$.

- $\mathcal{C}_d$ is a set of configurations (in $\mathcal{C}_2$) such that $\#d_q > 0$ holds for some q $(1 \leq q \leq k-2)$.

- $\mathcal{C}_{m2}$ is a set of configurations (in $\mathcal{C}_2$) such that $d_q = 0$ holds for any q $(1 \le q \le k-2)$ and $\sum_{p=2}^{k-1} \#m_p \ge 2$ holds.
- $\mathcal{C}_{m1}$ is a set of configurations (in $\mathcal{C}_2$) such that $d_q = 0$ holds for any q $(1 \le q \le k-2)$ and $\sum_{p=2}^{k-1} \#m_p = 1$ holds.
- $\mathcal{C}_{m0}$ is a set of configurations (in $\mathcal{C}_2$) such that $d_q = 0$ holds for any q $(1 \le q \le k-2)$ and $\sum_{p=2}^{k-1} \#m_p = 0$ holds.

First we consider a configuration $C \in \mathcal{C}_d$. Let q be an integer such that $d_q > 0$ holds in C. From Lemma 22, $\#g_q > 0$ holds. Consequently, when agents in states d_q and g_q interact, at least one of them enters *initial* by transition 9 or 10. Thus, $\#ini$ is increased.

Next consider a configuration $C \in \mathcal{C}_{m2}$. From the definition of $\mathcal{C}_{m2}$, there exist two distinct agents a_i and a_j whose states are m_i and m_j, respectively. When a_i and a_j interact, they enter states d_{i-1} and d_{j-1} by transition 8, respectively. This configuration belongs to $\mathcal{C}_d$, and thus $\#ini$ is eventually increased.

Let us consider a configuration $C \in \mathcal{C}_{m1}$. Let i be an integer such that $\#m_i = 1$ holds. From Lemma 22, $\#g_x = 1 + \#g_k$ holds for $x \le i-1$ and $\#g_x = \#g_k$ holds for $x \ge i$. Since a population consists of one agent in state m_i and agents in states $g_x (1 \le x \le k)$, *initial*, and *initial′*, we have $\#ini = n - 1 - \sum_{x=1}^{k} \#g_x = n - k \cdot \#g_k - i \ge k - i$. Let a be the agent in state m_i and $a_i, a_{i+1}, \ldots, a_{k-1}$ be agents in state *initial* or *initial′*. If a interacts with $a_i, a_{i+1}, \ldots, a_{k-1}$ in this order, $a_i, a_{i+1}, \ldots, a_{k-1}$ transition to $g_i, g_{i+1}, \ldots, g_{k-1}$, respectively and a transitions to g_k. Thus, $\#g_k$ is increased by one.

Finally, we consider a configuration $C \in \mathcal{C}_{m0}$. In this case, $\sum_{q=1}^{k} \#g_q + \#ini = n$ holds. From Lemma 22, $\#g_x = \#g_k$ holds for any x $(1 \le x \le k)$. That is, $\sum_{x=1}^{k} \#g_x + \#ini = k \cdot \#g_k + \#ini = n$ holds. Hence, $\#ini = n - k \cdot \#g_k \ge k$ holds. This means no configuration is in $\mathcal{C}_{m0}$.

Therefore, the lemma holds. □

Lemma **25.** *For any execution $\Xi = C_0, C_1, \ldots$, there exists C_t such that $n - k \cdot \#g_k < k$ holds.*

Proof. First, we show that, when $n - k \cdot \#g_k \ge k$ holds at a configuration C_i, $\#g_k$ is increased by one at C_j for some j $(j > i)$. For contradiction, we assume that such C_j does not exist. Since $\#g_k$ is never decreased from the protocol, $\#g_k$ is never changed and $n - k \cdot \#g_k \geq k$ continuously holds after C_i. Since the number of such configurations is finite, some configuration C'_i occurs infinitely often after C_i in Ξ. From Lemmas 23 and 24, there exists C'_j such that $C'_i \xrightarrow{*} C'_j$ and $\#g_k$ in C'_j is increased by one from C'_i. That is, there exists a sequence of configurations $C'_1, C'_2, \dots, C'_l$ such that $C'_i = C'_1 \to C'_2 \to \cdots \to C'_l = C'_j$ holds. From global fairness, since $C'_i = C'_1$ occurs infinitely often, C'_2 occurs infinitely often. Similarly, $C'_3, \dots, C'_l = C'_j$ occur infinitely often. That is, $\#g_k$ at C'_j is increased by one from C_i. This is a contradiction. Thus, if $n - k \cdot \#g_k \geq k$ holds, $\#g_k$ is eventually increased by one. Therefore, the lemma holds. □

Note that, since $n \geq \sum_{x=1}^{k} \#g_x \geq k \cdot \#g_k$ holds from Lemma 22, $n - k \cdot \#g_k < k$ derives $\#g_k = \lfloor n/k \rfloor$. Hence, Lemma 25 implies that $\#g_k = \lfloor n/k \rfloor$ eventually holds. This implies that the number of agents in each group eventually becomes $\lfloor n/k \rfloor$ or $\lfloor n/k \rfloor + 1$ from Lemma 22. Let $r = n - k \cdot \lfloor n/k \rfloor$. If $r = 0$ holds, the uniform k-partition has been solved. If $r \geq 1$ holds, there exist r remaining agents. Lemma 26 shows resultant states of the remaining agents. If $r = 1$ holds, the one remaining agent is in state *initial* or *initial'*. If $r \geq 2$ holds, r agents enter states $g_1, g_2, \dots, g_{r-1}$ and m_r.

Lemma **26.** *Assume that $r = n - k \cdot \lfloor n/k \rfloor > 0$ holds. Let $\mathcal{C}_3$ be a set of reachable configurations such that $n - k \cdot \#g_k < k$ holds (i.e., $\#g_k = \lfloor n/k \rfloor$). For any configuration $C \in \mathcal{C}_3$, there exists C' such that 1) $C \xrightarrow{*} C'$ holds, 2) $\#g_x = \lfloor n/k \rfloor + 1$ holds for any x ($1 \leq x \leq r-1$), 3) $\#g_x = \lfloor n/k \rfloor$ holds for any x ($r \leq x \leq k$), and 4) $\#ini = 1$ holds if $r = 1$ and $\#m_r = 1$ holds if $r \geq 2$.*

Proof. From Lemma 22, $\#g_x \geq \#g_k = \lfloor n/k \rfloor$ holds for any x ($1 \leq x \leq k$) at C. Let $V' \subset V$ be a set of agents that include $\lfloor n/k \rfloor$ agents in state g_x at C for any x ($1 \leq x \leq k$), and let $V_r = V - V'$.

Let us consider the case of $r = 1$. In this case V_r does not contain an agent in state m_p for any p because otherwise V_r also contains agents in state g_r ($r \leq p-1$) from Lemma 22. Similarly, V_r does not contain an agent in state d_q for any q. Hence, V_r contains one agent in state *initial* or *initial'*. Thus, if $r = 1$, the lemma holds.

In the following, we assume $r \geq 2$. Similarly to Lemma 24, we can prove that r agents in V_r transition to $g_1, g_2, \dots, g_{r-1}$ and m_r. That is, we can easily observe the following facts. If all agents in V_r are in *initial* or *initial'*, they can transition to $g_1, g_2, \dots, g_{r-1}$ and m_r by interacting one by one. If an agent in state d_q exists in V_r for some q, it eventually transitions to *initial*. If two agents in states m_i and m_j exist in V_r for some i and j, they can transition to d_{i-1} and d_{j-1}. If V_r contains exactly one agent in state m_p for some p, V_r contains $p-1$ agents in states $g_1, g_2, \dots, g_{p-1}$ and $r-p$ agents in states *initial* and *initial'*. In this case, agents in state m_p, *initial*, and *initial'* can transition to $g_p, g_{p+1}, \dots, g_{r-1}$ and m_r.

Since V' contains $\lfloor n/k \rfloor$ agents in state g_x for every x and V_r contains r agents in states $g_1, g_2, \dots, g_{r-1}$ and m_r, the lemma holds. □

Lemma 26 proved that a configuration specified in the lemma is reachable from a configuration specified in Lemma 25. Thus, similarly to Lemma 25, we can obtain the following lemma.

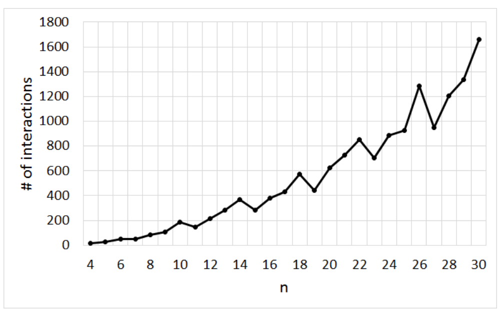
Lemma **27.** *Assume that $r = n - k \cdot \lfloor n/k \rfloor > 0$ holds. For any execution $\Xi = C_0, C_1, \dots,$ there exists C_t such that 1) $\#g_x = \lfloor n/k \rfloor + 1$ holds for any x ($1 \leq x \leq r-1$), 2) $\#g_x = \lfloor n/k \rfloor$ holds for any x ($r \leq x \leq k$), and 3) $\#ini = 1$ holds if $r = 1$ and $\#m_r = 1$ holds if $r \geq 2$.*

Let $r = n - k \cdot \lfloor n/k \rfloor$. From Lemmas 25 and 27, a population eventually reaches a configuration C^* such that 1) $\#g_x = \lfloor n/k \rfloor + 1$ holds for any x ($1 \leq x \leq r-1$), 2) $\#g_x = \lfloor n/k \rfloor$ holds for any $x \geq r$, and 3) $\#ini = 1$ holds if $r = 1$ and $\#m_r = 1$ holds if $r \geq 2$. Since $\gamma(g_x) = x$ holds for x ($1 \leq x \leq k$), $\gamma(m_p) = p$ holds for p ($2 \leq p \leq k-1$), and $\gamma(ini) = 1$ holds for $ini \in \{initial, initial'\}$, the number of agents in each group is $\lfloor n/k \rfloor$ or $\lfloor n/k \rfloor + 1$. In addition, no transition can happen at C^*. This implies that C^* is stable. Therefore, we have the following theorem.

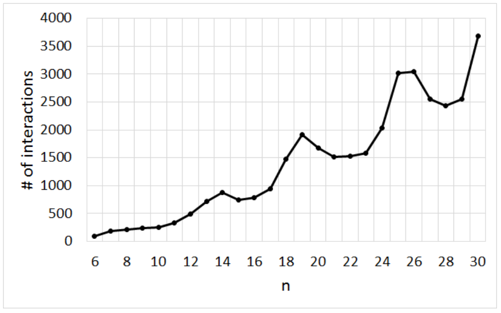
Theorem **17.** *The proposed protocol solves the uniform k-partition problem. That is, there exists a symmetric protocol with $3k-2$ states and designated initial states that solves the uniform k-partition problem under global fairness.*

5.2.4 Simulation Results

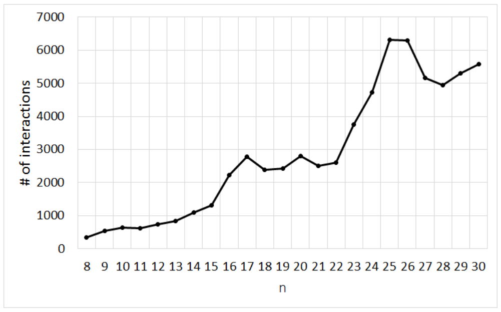
In this section, we discuss the time complexity of the proposed protocol by simulations. We evaluate the time complexity by the total number of interactions until



(a) $k = 4$



(b) $k = 6$



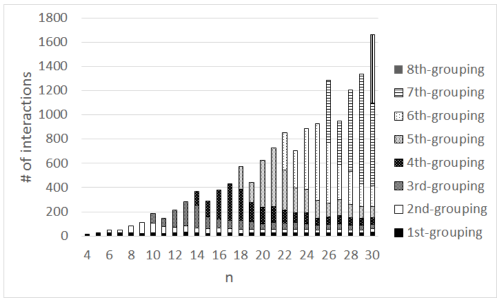
(c) $k = 8$

Figure 5. The number of interactions for $k \in \{4, 6, 8\}$ with changing the population size n

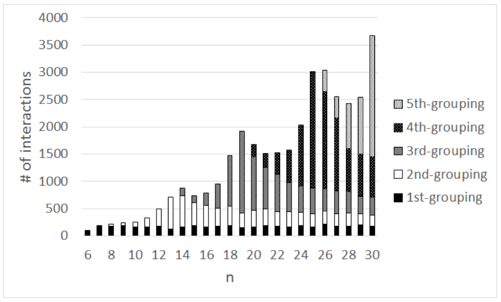
a population reaches a stable configuration. In the simulations, we construct an execution by selecting two agents uniformly at random in each configuration and making them interact. Note that, if we construct an infinite execution by this way, the execution satisfies global fairness with probability 1. For all simulation settings, we conduct a simulation 100 times and show the average values as the results.

Varying the population size n

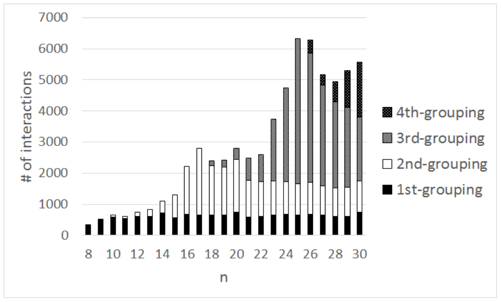
Figure 5 shows the number of interactions for $k \in \{4, 6, 8\}$ with changing the population size (i.e., the number of agents) n. As n increases, the number of interactions tends to increase. However, the number of interactions sometimes decreases when n increases. We can observe that such a phenomenon is repeated with a period of a length of k. That is, $n \bmod k$ influences the number of interactions.



(a) $k = 4$



(b) $k = 6$



(c) $k = 8$

Figure 6. The number of interactions to achieve the i-th grouping

To observe the details of executions, we focus on the number of interactions required to construct one set of agents in states $g_1, g_2, \ldots, g_k$. We refer to this construction by grouping. Recall that, once an agent enters state g_k, the set of agents never goes back to *initial*. Let NI_i be the number of interactions required to construct the i-th set of agents in states $g_1, g_2, \ldots, g_k$. We define $NI_0 = 0$. We count $NI'_i = NI_i - NI_{i-1}$, i.e., the number of interactions to achieve the i-th grouping. We show the results in Figure 6. In this figure, we show NI'_1 at the bottom of the figure (denoted by 1st-grouping), NI'_2 at the second to the bottom (denoted by 2nd-grouping), and so on. Figure 6 shows that $NI'_1 < NI'_2 < \cdots$ holds except for the last part (i.e., transitions of the remaining $n \bmod k$ agents). This is because, as the execution proceeds, the number of agents not in a group decreases and consequently agents require more interactions to achieve the grouping. In addition, we can observe that, for any positive integer c, when $n = c{\cdot}k+2, c{\cdot}k+3, \ldots, c{\cdot}k+(k+1)$ holds, the number of interactions to achieve the $(c+1)$-th grouping (shown in the top of each graph) increases steeply with n. In

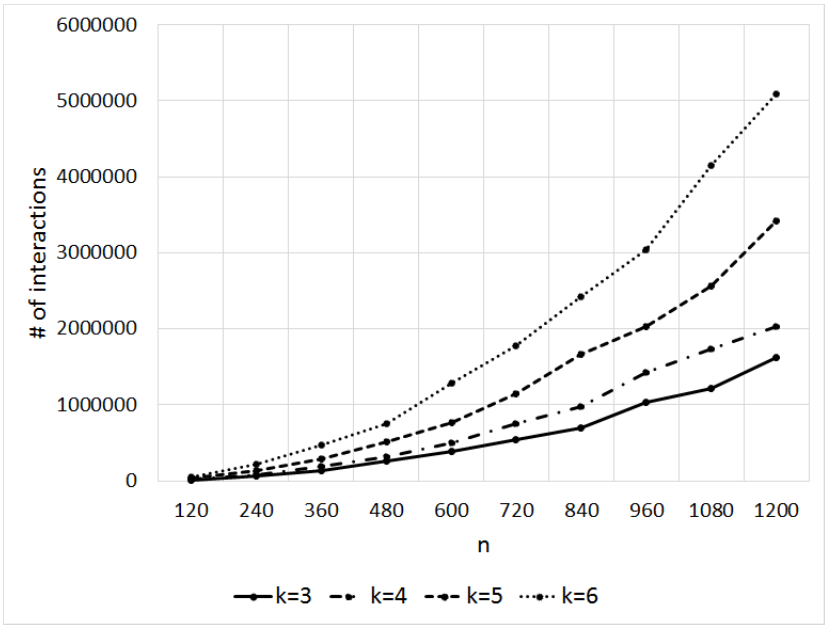


Figure 7. The number of interactions for $k \in \{3, 4, 5, 6\}$ with changing the population size n

addition, the number of interactions for the $(c+1)$-th grouping accounts for more than half of the total number of interactions for $n = c \cdot k + k$ and $n = c \cdot k + (k+1)$. These facts influence juggy forms of graphs in Figure 5.

Hereafter, to prevent the effect of $n \bmod k$, we execute simulations for the case where $n \bmod k = 0$ holds.

Figure 7 shows the number of interactions for $k \in \{3, 4, 5, 6\}$ with changing the population size n. We consider $n = 120 \cdot n'$ for $n' \in \{1, 2, \dots, 8\}$ so that $n \bmod k = 0$ holds. Figure 7 shows that, as n increases, the number of interactions also increases. The number of interactions seems to increase more than linearly but less than exponentially with n.

Varying the number of groups k

The logarithmic graph in Figure 8 shows the number of interactions for $n = 960$ with changing k. To avoid the effect of $n \bmod k$, we show the results only for the case where $n \bmod k = 0$ holds. Figure 8 shows that the number of interac-

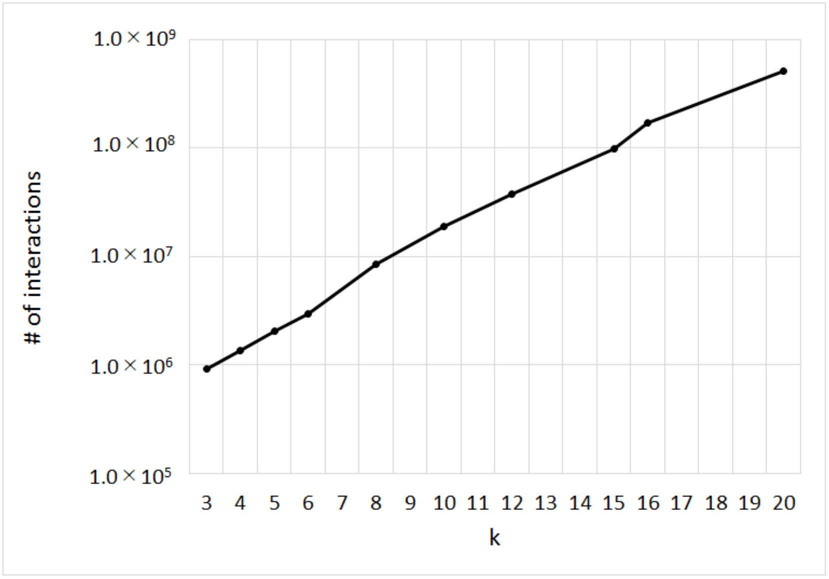


Figure 8. The number of interactions for $n = 960$ with changing k

tions seems to increase exponentially with k. This is because, to create a set of groups including agents with states g_1 to g_k, a m_2-state agent interacts $k - 2$ free agents (i.e., agents with state *initial* or *initial′*) without interacting other m-state agents. Since interaction of *initial* and *initial′* agents creates a m-state agent, a non-negligible number of m-state agents exist. Hence, the possibility that an agent interacts $k - 2$ free agents without interacting m-state agents becomes exponentially small when k becomes large. This increases the number of interactions exponentially with k.

6. Concluding Remarks

In this part, we focus on the uniform k-partition problem on complete graphs. We considered the problem under various assumptions such as the existence of a base station, fairness, symmetry of protocols, and initial states of agents. For the case of $k = 2$, we clarified the solvability for each combination of assumptions. Moreover, for each solvable case, we proposed a space-optimal protocol (i.e., we clarified tight upper and lower bounds on the number of states per agent for each case).

On the other hand, for the general case of an arbitrary number of partitions (the uniform k-partition), we clarified the solvability for most cases (23 out of 24 cases). Moreover, we clarified tight upper and lower bounds on the number of states per agent for many of solvable cases (10 out of 15 cases). Concretely, for 19 cases, we extended the results of the uniform 2-partition to results of the uniform k-partition, and we proposed a symmetric protocol with $3k-2$ states and designated initial states that solves the uniform k-partition problem under global fairness (this protocol works for 2 cases). This symmetric protocol is asymptotically space-optimal because $\Omega(k)$ states are necessary for any uniform k-partition protocol. Moreover, we evaluated the time complexity of the protocol by simulations. From the simulation results, we can observe that the time complexity increases exponentially with k but not exponentially with n.

We show that our most results can be applied to the k-partition. That is, even for the k-partition, we clarified the solvability and space-complexity in many cases. Concretely, for the 2-partition, we clarified solvability for all cases and clarified tight upper and lower bounds on the number of states for 22 out of 24 cases, and, for the k-partition, we clarified solvability for 23 out of 24 cases and clarified tight upper and lower bounds on the number of states for 8 out of 15 solvable cases.

Part IV

Uniform 2-partition on Arbitrary Graphs

1. Introduction

In this part, we aim to clarify the space complexity of the uniform 2-partition on arbitrary graphs. In Part III, we have only dealt with the uniform k-partition on complete graphs. In the original population protocol model, Angluin et al. mainly studied computability of the model on complete communication graphs (i.e., all agents can interact with each other). Then, many subsequent papers also studied various problems on complete communication graphs. However, in recent years, researchers have studied problems on various communication graphs [5, 11, 14, 26, 27, 37, 44]. Actually, in realistic systems, the communication graph may not be a complete graph because low-performance devices may not move so wide. For example, in a sensor network system, sensors may be scattered over a large area and some of them may move only a short distance because sensors move passively. Because of this, some pair of sensors never communicate with each other and thus the communication graph is not a complete graph.

1.1 Our Contributions

A summary of the results is presented in Table 9. Let us first observe that, as complete communication graphs are a special case of arbitrary communication graphs, the impossibility results in Section 3.1.1 remain valid in our setting. With a base station (be it initialized or non-initialized) under global fairness, we extend the three states protocol with designated initial states in Section 3.1.1 from complete communication graphs to arbitrary communication graphs. With a non-initialized base station and designated initial states, we propose a new symmetric protocol with $3P+1$ states that solves the problem under weak fairness with the assumption that an upper bound P of the number of agents is given to agents. These results yield identical upper bounds for the easier cases of

Table 9. Minimum number of states to solve the uniform 2-partition problem on *arbitrary* graphs under global fairness.

Base station	Initial states	Symmetry	Upper bound	Lower bound
Initialized base station	Designated	Asymmetric	3	3
		Symmetric	3	3
	Arbitrary	Asymmetric	-	4
		Symmetric	-	4
Non-initialized base station	Designated	Asymmetric	3	3
		Symmetric	3	3
	Arbitrary	Asymmetric	Unsolvable	
		Symmetric	Unsolvable	
No base station	Designated	Asymmetric	4	4
		Symmetric	5	5
	Arbitrary	Asymmetric	Unsolvable	
		Symmetric	Unsolvable	

Table 10. Minimum number of states to solve the uniform k-partition problem under weak fairness on complete graphs. P is a known upper bound of the number of agents, and $l \geq 3$ and h are positive integers.

Base station	Initial states	Symmetry	Upper bound	Lower bound
Initialized base station	Designated	Asymmetric/ Symmetric	$3P+1$ $3l+1$ for no $l \cdot h$ cycle	3
	Arbitrary	Asymmetric	-	P
		Symmetric	-	$P+1$
Non-initialized base station	Designated	Asymmetric/ Symmetric	$3P+1$ $3l+1$ for no $l \cdot h$ cycle	3
	Arbitrary	Asymmetric	Unsolvable	
		Symmetric	Unsolvable	
No base station	Designated	Asymmetric	Unsolvable	
		Symmetric	Unsolvable	
	Arbitrary	Asymmetric	Unsolvable	
		Symmetric	Unsolvable	

asymmetric protocols and/or initialized base station. In addition, we also show a condition of communication graphs where the number of states in the protocol can be reduced from $3P + 1$ to constant. Concretely, we show that the number of states in the protocol can be reduced to $3l + 1$ if we assume communication graphs such that every cycle either includes the base station or its length is not a multiple of l, where l is a positive integer greater than three. With no base station and designated initial states, we prove that four and five states are necessary and sufficient to solve uniform 2-partition under global fairness with asymmetric and symmetric protocols, respectively. In the same setting, in complete graphs, three and four states are necessary and sufficient. So, one additional state enables us to solve the problem on arbitrary graphs. On the other hand, by changing the fairness assumption from global fairness to weak fairness, we prove that the problem cannot be solved, by using a similar argument as in the impossibility result for leader election by Fischer and Jiang [30]. Overall, we show the solvability of uniform 2-partition in a variety of settings for a population of agents assuming arbitrary communication graphs.

1.2 Problem Definition

Let $Y = \{red, blue\}$ be an output set of the uniform 2-partition problem. Let $\gamma : Q_p \to \{red,\ blue\}$ be a function that maps a state of an agent to Y. We define the color of an agent a as $\gamma(s(a))$. Then, we say that agent a is *red* (resp., *blue*) if $\gamma(s(a)) = red$ (resp., $\gamma(s(a)) = blue$) holds. If an agent a has state s such that $\gamma(s) = red$ (resp., $\gamma(s) = blue$), we call a a *red* agent (resp., a *blue* agent). For some population V, the number of *red* agents (resp., *blue* agents) in V is denoted by $\#red(V)$ (resp., $\#blue(V)$). When V is clear from the context, we simply write $\#red$ and $\#blue$.

A configuration C is stable with respect to the uniform 2-partition if there exists a partition $\{H_r,\ H_b\}$ of V_p that satisfies the following conditions:

1. $||H_r| - |H_b|| \leq 1$ holds, and

2. For every configuration C' such that $C \xrightarrow{*} C'$, each agent in H_r (resp., H_b) remains *red* (resp., *blue*) in C'.

An execution $\Xi = C_0, C_1, C_2, \ldots$ solves the uniform 2-partition problem if Ξ includes a configuration C_t that is stable for uniform 2-partition. Finally, a protocol $\mathcal{P}$ solves the uniform 2-partition problem if every possible execution Ξ of protocol $\mathcal{P}$ solves the uniform 2-partition problem.

2. Upper Bounds with a Non-initialized Base Station and Designated Initial States

In this section, we prove some upper bounds on the number of states that are required to solve the uniform 2-partition problem on arbitrary graphs with designated initial states and a non-initialized base station. More concretely, with global fairness, we propose a symmetric deterministic protocol with three states by extending the protocol in Section 3.1.1 from a complete communication graph to an arbitrary communication graph. In the case of weak fairness, we present a symmetric protocol with $3P + 1$ states, where P is a known upper bound of the number of agents. Recall that we consider only deterministic protocols in this dissertation.

2.1 Upper Bound for Symmetric Protocols under Global Fairness

The state set of agents in this protocol is $Q_p = \{initial, red, blue\}$, and we assume that $\gamma(initial) = \gamma(red) = red$ and $\gamma(blue) = blue$ hold. The designated initial state of agents is *initial*. The idea of the protocol is as follows: the base station assigns *red* and *blue* to agents whose state is *initial* alternately. As the base station cannot meet every agent (the communication graph is arbitrary), the positions of state *initial* are moved throughout the communication graph using transitions. Thus, if an agent with *initial* state exists somewhere in the network, the base station has infinitely many chances to interact with a neighboring agent with *initial* state. This implies that the base station is able to repeatedly assign *red* and *blue* to neighboring agents with *initial* state unless no agent anywhere in the network has *initial* state. Since the base station assigns *red* and *blue* alternately, the uniform 2-partition is completed after no agent has *initial* state.

To make *red* and *blue* alternately, the base station has a state set $Q_b = \{b_{red}, b_{blue}\}$. Using its current state, the base station decides which color to use for the next interaction with a neighboring agent with *initial* state. Now, to move the position of an *initial* state in the communication graph, if an agent with *initial* state and an agent with *red* (or *blue*) state interact, they exchange their states. This implies that eventually an agent adjacent to the base station has *initial* state and then the agent and the base station interact (global fairness guarantees that such interaction eventually happens). Transition rules of the protocol are the following (for each transition rule $(p, q) \to (p', q')$, transition rule $(q, p) \to (q', p')$ exists, but we omit the description).

1. $(b_{red}, initial) \to (b_{blue}, red)$
2. $(b_{blue}, initial) \to (b_{red}, blue)$
3. $(blue, initial) \to (initial, blue)$
4. $(red, initial) \to (initial, red)$

From these transition rules, eventually each agent transitions to *red* or *blue*. Hence, there exist $\lceil n/2 \rceil$ *red* (resp., *blue*) agents, and $\lfloor n/2 \rfloor$ *blue* (resp., *red*) agents if the base station has b_{red} (resp., b_{blue}) as an initial state. When no agent has *initial* state, the protocol converges (indeed, no interaction is defined when no agent has *initial* state). Therefore, the protocol solves the uniform 2-partition problem.

We now prove the correctness of the protocol.

Theorem **18.** *In the population protocol model with a non-initialized base station, there exists a symmetric protocol with three states per agent that solves the uniform 2-partition problem with designated initial states assuming global fairness in arbitrary communication graphs.*

Proof. Recall that $\#red$ (resp., $\#blue$) denotes the number of agents in state s such that $\gamma(s) = red$ (resp., $\gamma(s) = blue$) holds, thus initially $\#red = n$ and $\#blue = 0$ hold. We show that eventually $|\#red - \#blue| \leq 1$ holds and no agent ever changes its state afterwards. Transition rules 1 and 2 indicate that, if the base station has state b_{red} (resp., b_{blue}), and an agent with *initial* state interact, the agent state becomes *red* (resp., *blue*), and the base station state becomes b_{blue} (resp., b_{red}). By repeating these transitions, agents are assigned *red* and *blue* alternately, and the number of agents in state *initial* decreases. This implies that, eventually $|\#red - \#blue| \leq 1$ holds by alternating transition rules 1 and 2 (transition rules 1 and 2 are never enabled simultaneously, and executing one disables it while enabling the other). Transition rules 3 and 4 indicate that, if an agent with *initial* state and an agent with *red* (or *blue*) state interact, they exchange their states. From the global fairness hypothesis and transition rules 3 and 4, if there exists an agent with *initial* state, the base station and a neighboring agent with *initial* state eventually interact. Thus, transition rules 1 and 2 occur repeatedly unless no agent has *initial*. After all agents have *red* or *blue* state, no agent ever changes its color. Recall that agents are assigned *red* and *blue* alternately and thus $|\#red - \#blue| \leq 1$ holds after those assignments. Moreover, the arguments do not depend on the initial state of the base station, and the proposed protocol is symmetric. Therefore, the theorem holds. □

Note that, under weak fairness, this protocol does not solve the uniform 2-partition problem. This is because we can construct a weakly-fair execution of this protocol such that some agents keep *initial* state infinitely often. For example, we can make an agent keep *initial* by constructing an execution as follows: If the agent (in *initial*) interacts with an agent in *red* or *blue*, the next interaction occurs between the same pair of agents.

2.2 Upper Bound for Symmetric Protocols under Weak Fairness

2.2.1 A protocol on arbitrary graphs

In this protocol, every agent x has variables $color_x \in \{ini, r, b\}$ and $depth_x \in \{\perp, 1, 2, \ldots, P\}$. Variable $color_x$ represents the color of agent x. That is, for an agent x, if $color_x = ini$ or $color_x = r$ holds, $\gamma(s(x)) = red$ holds. On the other hand, if $color_x = b$ holds, $\gamma(s(x)) = blue$ holds. The protocol is given in Algorithm 4.

The basic strategy of the protocol is the following.

Algorithm 4 Uniform 2-partition protocol with $3P+1$ states.

Variables at the base station:

```
    $RB \in \{r, b\}$: The state that the base station assigns next
```

Variables at an agent x:

```
    $color_x \in \{ini, r, b\}$: Color of the agent, initialized to $ini$
    $depth_x \in \{\bot, 1, 2, 3, \ldots, P\}$: Depth of agent $x$ in a tree rooted at the base station, initialized to $\bot$
 1: when an agent $x$ and the base station interact do
 2:     if $color_x = ini$ and $depth_x = 1$ then
 3:         $color_x \leftarrow RB$
 4:         $RB \leftarrow \overline{RB}$
 5:     end if
 6:     if $depth_x = \bot$ then
 7:         $depth_x \leftarrow 1$
 8:     end if
 9: end
10: when two agents $x$ and $y$ interact do
11:     if $depth_y \neq \bot$ and $depth_x = \bot$ then
12:         $depth_x \leftarrow depth_y + 1$
13:     else if $depth_x \neq \bot$ and $depth_y = \bot$ then
14:         $depth_y \leftarrow depth_x + 1$
15:     end if
16:     if $depth_x < depth_y$ and $color_y = ini$ then
17:         $color_y \leftarrow color_x$
18:         $color_x \leftarrow ini$
19:     end if
20:     if $depth_y < depth_x$ and $color_x = ini$ then
21:         $color_x \leftarrow color_y$
22:         $color_y \leftarrow ini$
23:     end if
24: end
```

Note: If $depth_x = \bot$ holds, $color_x = ini$ holds.

1. Create a spanning tree rooted at the base station. Concretely, agent x assigns its depth in a tree rooted at the base station into variable $depth_x$. Variable $depth_x$ is initialized to $\perp$. Variable $depth_x$ obtains the depth of x in the spanning tree as follows: If the base station and an agent p with $depth_p = \perp$ interact, $depth_p$ becomes 1. If an agent q with $depth_q \neq \perp$ and an agent p with $depth_p = \perp$ interact, $depth_p$ becomes $depth_q + 1$. By these behaviors, for any agent x, eventually variable $depth_x$ has a depth of x in a tree rooted at the base station.

2. Using the spanning tree, carry the initial color ini toward the base station and make the base station assign r and b to agents one by one. Concretely, if agents x and y interact and both $depth_y < depth_x$ and $color_x = ini$ hold, x and y exchange their colors (i.e., ini is carried from x to y). Hence, since ini is always carried to a smaller $depth$, eventually an agent z with $depth_z = 1$ obtains ini. After that, the base station and the agent z interact and the base station assigns r or b to z. Note that, if the base station assigns r (resp., b), the base station assigns b (resp., r) next.

Then, for any agent v, eventually $color_v \neq ini$ holds. Hence, there exist $\lceil n/2 \rceil$ *red* (resp., *blue*) agents, and $\lfloor n/2 \rfloor$ *blue* (resp., *red*) agents if variable RB in the base station has r (resp., b) as an initial value. Therefore, the protocol solves the uniform 2-partition problem.

From now, we demonstrate the correctness of the protocol. Let $\#ini = |\{x| color_x = ini\}|$, $\#r = |\{x|color_x = r\}|$, and $\#b = |\{x|color_x = b\}|$. First of all, by the pseudocode, we can observe that $\#r$, $\#b$, and $\#ini$ do not change except for an interaction between the base station and an agent v such that $depth_v = 1$ and $color_v = ini$ hold. This is because $\#r$, $\#b$, and $\#ini$ are changed only by lines 3–4.

Lemma **28.** *For any weakly-fair execution of Algorithm 4, unless an interaction occurs between the base station and an agent v such that $depth_v = 1$ and $color_v = ini$ hold, $\#r$, $\#b$, and $\#ini$ do not change.*

We now show three basic properties of variable *depth*. First, we have the following lemma because, in the pseudocode, agent v never changes $depth_v$ if $depth_v \neq \perp$ holds.

Lemma **29.** *For any agent v, when $depth_v \neq \bot$ holds in a configuration, $depth_v$ does not change afterwards.*

Then, we show the remaining properties of variable *depth*.

Lemma **30.** *For any agent v, $depth_v \neq \bot$ holds after some configuration in any weakly-fair execution of Algorithm 4.*

Proof. By the weak fairness assumption, for any agent v_0 adjacent to the base station, eventually v_0 and the base station interact. By lines 6–7 of the pseudocode, $depth_{v_0} \neq \bot$ holds after the interaction with the base station. From Lemma 29, for any agent v, once $depth_v \neq \bot$ holds, $depth_v \neq \bot$ holds perpetually. As the graph is connected, if there is a pair of agents v and v' such that $depth_v = \bot$ and $depth_{v'} \neq \bot$ hold, there is a pair of adjacent agents v_a and v_b such that $depth_{v_a} = \bot$ and $depth_{v_b} \neq \bot$ hold. Hence, eventually such agents v_a and v_b interact by the weak fairness assumption, and after the interaction occurs, $depth_{v_a} \neq \bot$ holds. By using a similar argument, for any agent v, eventually $depth_v \neq \bot$ holds. □

Lemma **31.** *For any agent v_1, when $depth_{v_1}$ is neither $\bot$ nor one, then v_1 is adjacent to an agent v_2 such that $depth_{v_1} = depth_{v_2} + 1$ holds. When $depth_{v_1}$ is 1, v_1 is adjacent to the base station.*

Proof. We focus on interactions such that an agent v assigns a value other than $\bot$ to $depth_v$. By Lemmas 29 and 30, the number of such interactions for a given agent is exactly one. We consider two cases.

First, we consider the case when $depth_v$ becomes one by the interaction. From the protocol, $depth_v$ can become one only if the base station and v interact. Hence, v is adjacent to the base station.

Next, we consider the case where $depth_v$ becomes neither $\bot$ nor one by the interaction. From the protocol, if the interaction happens between v and an agent w, $depth_v$ becomes $depth_w + 1$ by the interaction. By Lemma 29, $depth_w$ and $depth_v$ never change afterward. Thus, the lemma holds. □

From the above lemmas, we show that eventually $\#ini = 0$ holds in any weakly-fair execution of the protocol.

Lemma **32.** *For any weakly-fair execution of Algorithm 4, $\#ini = 0$ holds after finite time.*

Proof. For the purpose of contradiction, let us assume that, after some configuration C, $\#ini > 0$ never decreases. Let us consider a configuration C' such that $depth_v \neq \bot$ holds for any agent v in C', and C' appears after C. By Lemma 30, such C' exists. In C', let $Ini = \{x | color_x = ini\}$, and let $v_1 \in Ini$ be an agent such that $depth_{v_1} = \min\{depth_x | x \in Ini\}$ holds. By Lemma 31, either $depth_{v_1} = 1$ holds or v_1 is adjacent to an agent v_2 such that $depth_{v_2} = depth_{v_1} - 1$.

First, we consider the case when $depth_{v_1} = 1$ holds. By Lemma 31, v_1 is adjacent to the base station. Since $depth_{v_1} = 1$ holds, there is not an agent v_2' such that $depth_{v_2'} < depth_{v_1}$ holds. Hence, from the protocol, $color_{v_1}$ keeps ini unless v_1 and the base station interact. Thus, eventually v_1 with $color_{v_1} = ini$ and the base station interact. Then, $\#ini$ decreases in this case.

Next, we consider the case where v_1 is adjacent to an agent v_2 such that $depth_{v_2} = depth_{v_1} - 1$. From the protocol, since $depth_{v_1} \neq 1$ holds, $color_{v_1}$ keeps ini unless an interaction happens between v_1 and an agent v_2' such that $depth_{v_2'} < depth_{v_1}$ holds. From the weak fairness, eventually v_1 and such v_2' interact and $color_{v_2'}$ becomes ini. At that time, the smallest depth of agents with color ini decreases. By repeating this behavior similarly, eventually some agent v_h with $depth_{v_h} = 1$ obtains color ini. After that, v_h and the base station interact and $\#ini$ decreases. This is a contradiction.

□

Next, we prove that $|\#r - \#b| \leq 1$ always holds in any weakly-fair execution of the protocol.

Lemma **33.** *For any configuration in any weakly-fair execution of Algorithm 4,* (i) $0 \leq \#r - \#b \leq 1$ *holds if* $RB = b$ *holds, and* (ii) $0 \leq \#b - \#r \leq 1$ *holds if* $RB = r$ *holds.*

Proof. Let us consider an execution $\Xi = C_0, C_1, \ldots$ of the protocol. We prove the lemma by induction on the index of a configuration. In the base case (C_0), $\#r = \#b = 0$ holds and thus alternatives *(i)* and *(ii)* of the lemma hold immediately.

For the induction step, assume that there exists an integer $i \geq 0$ such that the lemma holds in C_i. Let us consider an interaction at $C_i \to C_{i+1}$.We consider two cases.

First, we consider the case where, for an agent v such that $depth_v = 1$ and $color_v = ini$ hold, the base station and v do not interact. By Lemma 28, when the base station and such v do not interact, $\#r$, $\#b$, and $\#ini$ do not change. Moreover, from the protocol, RB does not change at the interaction, and thus both alternatives *(i)* and *(ii)* of the lemma hold in this case.

Next, we consider the case where, for an agent v such that $depth_v = 1$ and $color_v = ini$ hold, the base station and v interact.

- In the case where $RB = r$ holds in C_i: After the interaction at $C_i \to C_{i+1}$, $\#r$ increases by one and RB becomes b. By the induction assumption, since $0 \le \#b - \#r \le 1$ holds in C_i, $0 \le \#r - \#b \le 1$ and $RB = b$ hold in C_{i+1}. Hence, alternatives *(i)* and *(ii)* of the lemma hold.

- In the case where $RB = b$ holds in C_i: After the interaction at $C_i \to C_{i+1}$, $\#b$ increases by one and RB becomes r. By the induction assumption, since $0 \le \#r - \#b \le 1$ holds in C_i, $0 \le \#b - \#r \le 1$ and $RB = r$ hold in C_{i+1}. Hence, alternatives *(i)* and *(ii)* of the lemma hold.

Thus, the lemma holds. □

Using Lemmas 32 and 33, we show that the protocol solves the uniform 2-partition problem.

Theorem **19.** *Algorithm 4 solves the uniform 2-partition problem. That is, there exists a protocol with* $3P+1$ *states and designated initial states that solves the uniform 2-partition problem under weak fairness assuming arbitrary communication graphs with a non-initialized base station.*

Proof. By Lemmas 32 and 33, $|\#r - \#b| \le 1$ holds in any weakly-fair execution of the protocol and eventually $\#ini = 0$ holds in the execution. Moreover, from the protocol, when there exists no agent v such that $color_v = ini$, any agent x does not change its $color_x$. Thus, the protocol solves the problem. Additionally, the protocol works with $2P + (P+1)$ states. This is because, if $depth_x = \perp$ holds, $color_x = ini$ holds. That is, $depth_x$ takes $\perp$, 1, 2, …, P for ini, and takes 1, 2, …, P for r and b.

Therefore, the theorem holds. □

2.2.2 A protocol with constant states on a restricted class of graphs

In this subsection, we show that the space complexity of Algorithm 4 can be reduced to constant for communication graphs such that every cycle either includes the base station or its length is not a multiple of l, where l is a positive integer at least three.

We modify Algorithm 4 as follows. Each agent maintains the distance from the base station by computing modulo l plus 1. That is, we change lines 12 and 14 in Algorithm 4 to $depth_x \leftarrow depth_y \mod l + 1$ and $depth_y \leftarrow depth_x \mod l + 1$, respectively. Now $depth_x \in \{\perp, 1, 2, 3, \ldots, l\}$ holds for any agent x. Then we define the relation $depth_x \prec depth_y$ as follow: $depth_x \prec depth_y$ holds if and only if $depth_y - depth_x \mod l = 1$ holds, and we use $depth_x \prec depth_y$ instead of $depth_x < depth_y$ in lines 16 and 20.

We can easily observe that these modifications do not change the essence of Algorithm 4. For two agents x and y, we say $x \prec y$ if $depth_x \prec depth_y$ holds. Each agent x eventually assigns a depth of x minus 1 modulo l plus 1 to $depth_x$, and at that time there exists a path $x_0, x_1, \ldots, x_h$ such that x_0 is a neighbor of the base station, $x = x_h$ holds, and $x_i \prec x_{i+1}$ holds for any $0 \le i < h$. In addition, there exists no cycle $x_0, x_1, \ldots, x_h = x_0$ such that $x_i \prec x_{i+1}$ holds for any $0 \le i < h$. This is because, from the definition of relation '$\prec$', the length of such a cycle should be a multiple of l, but we assume that underlying communication graphs do not include a cycle of agents in V_p whose length is a multiple of l. Hence, similarly to Algorithm 4, we can carry the initial color *ini* toward the base station and make the base station assign r and b to agents one by one.

Corollary **7.** *There exists a protocol with $3l + 1$ states and designated initial states that solves the uniform 2-partition problem under weak fairness assuming arbitrary communication graphs with a non-initialized base station if, for any cycle of the communication graphs, it either includes the base station or its length is not a multiple of l, where l is a positive integer at least three.*

Algorithm 5 Uniform 2-partition protocol with four states.

```
A state set
    $Q = \{r^\omega, b^\omega, r, b\}$
A mapping to colors
    $\gamma(r^\omega) = \gamma(r) = red$
    $\gamma(b^\omega) = \gamma(b) = blue$
Transition rules
    1. $(r^\omega, r^\omega) \to (r, b)$
    2. $(r^\omega, b^\omega) \to (b, b)$
    3. $(r^\omega, r) \to (r, r^\omega)$
    4. $(b^\omega, b) \to (b, b^\omega)$
    5. $(r^\omega, b) \to (r, b^\omega)$
    6. $(b^\omega, r) \to (b, r^\omega)$
```

3. Upper and Lower Bounds with No Base Station and Designated Initial States

In this section, we show upper and lower bounds of the number of states to solve the uniform 2-partition problem with no base station and designated initial states on arbitrary communication graphs. Concretely, under global fairness, we prove that the minimum number of states for asymmetric deterministic protocols is four, and the minimum number of states for symmetric protocols is five. Under weak fairness, we prove that the uniform 2-partition problem cannot be solved without a base station using proof techniques similar to those Fischer and Jiang [30] used to show the impossibility of leader election. Recall that we consider only deterministic protocols in this dissertation.

3.1 Upper Bound for Asymmetric Protocols under Global Fairness

In this subsection, on arbitrary graphs with designated initial states and no base station under global fairness, we give an asymmetric protocol with four states.

We define a state set of agents as $Q = \{r^\omega, b^\omega, r, b\}$, and function γ as follows: $\gamma(r^\omega) = \gamma(r) = red$ and $\gamma(b^\omega) = \gamma(b) = blue$. We say an agent has a token if its state is r^ω or b^ω. Initially, every agent has state r^ω, that is, every agent is *red* and has a token. The transition rules are given in Algorithm 5 (for each transition rule $(p,q) \to (p',q')$ except for transition rule 1, transition rule $(q,p) \to (q',p')$ exists, but we omit the description).

The basic strategy of the protocol is as follows. When two agents with tokens interact and one of them is *red*, a *red* agent transitions to *blue* and the two tokens are deleted (transition rules 1 and 2). Since n tokens exist initially and the number of tokens decreases by two in an interaction, $\lfloor n/2 \rfloor$ *blue* agents appear and $\lceil n/2 \rceil$ *red* agents remain after all tokens (except one token for the case of odd n) disappear. To make such interactions, the protocol moves a token when agents with and without a token interact (transition rules 3, 4, 5, and 6). Global fairness guarantees that, if two tokens exist, an interaction of transition rule 1 or 2 happens eventually. Therefore, the uniform 2-partition is achieved by the protocol.

From now, we prove the correctness of the protocol. We define $\#r$, $\#b$, $\#r^{\omega}$, $\#b^{\omega}$ as the number of agents that have state r, b, r^{ω}, b^{ω}, respectively. Let $\#red = \#r + \#r^{\omega}$ and $\#blue = \#b + \#b^{\omega}$ be the number of *red* and *blue* agents, respectively. Let $\#token = \#r^{\omega} + \#b^{\omega}$ be the number of agents with tokens.

Lemma **34.** *In any globally-fair execution of Algorithm 5, $\#r = \#b + 2 \cdot \#b^{\omega}$ holds in any configuration.*

Proof. Let us consider an execution $\Xi = C_0, C_1, \ldots$ of the protocol. We prove the equation by induction on the index of a configuration. The base case is the case of C_0. In this case, the equation holds because all agents have r^{ω} initially. For the induction step, assume that the equation holds in C_i $(0 \le i)$. Let us consider an interaction at $C_i \to C_{i+1}$ for each transition rule.

- Transition rule 1: When the transition rule 1 occurs at $C_i \to C_{i+1}$, $\#r$ and $\#b$ increase by one.

- Transition rule 2: When the transition rule 2 occurs at $C_i \to C_{i+1}$, $\#b^{\omega}$ decreases by one, and $\#b$ increases by two.

- Transitions rule 3 and 4: When the transition rule 3 or 4 occurs at $C_i \to C_{i+1}$, $\#r$, $\#b$, and $\#b^{\omega}$ do not change.

- Transition rule 5: When the transition rule 5 occurs at $C_i \to C_{i+1}$, $\#b$ decreases by one, and $\#r$ and $\#b^\omega$ increase by one.
- Transition rule 6: When the transition rule 6 occurs at $C_i \to C_{i+1}$, $\#b$ increases by one, and $\#r$ and $\#b^\omega$ decrease by one.

For every case, $\#r = \#b + 2 * \#b^\omega$ holds in C_{i+1}. Therefore, the lemma holds. □

Lemma **35.** *For any globally-fair execution of the Algorithm 5, $\#token \leq 1$ holds after finite time.*

Proof. First of all, there is no transition rule that increases $\#token$. This implies that, if $\#token \leq 1$ holds at some configuration, $\#token \leq 1$ holds thereafter. Hence, for the purpose of contradiction, we assume that there exists a globally-fair execution Ξ of the protocol where $\#token = x > 1$ continuously holds after some configuration.

Let us consider a configuration C that occurs infinitely often in Ξ. Note that C is stable and satisfies $\#token = x$. First, we show that a *blue* agent exists in C. In the initial configuration, all agents have r^ω. Thus, by the first interaction, an agent with b occurs by transition rule 1. Since there is no transition that decreases the number of *blue* agents, there exists a *blue* agent in C. In addition, since $\#r = \#b + 2 \cdot \#b^\omega$ holds by Lemma 34, a *red* agent exists in C. Let us consider two agents a_1 and a_2 such that a_1 is adjacent to a_2, and a_1 is *red*, and a_2 is *blue* in C.

Since tokens can move through a graph without changing colors by swapping states (transition rules 3,4,5, and 6), a configuration C' such that a_1 and a_2 have a token is reachable from C. When a_1 and a_2 interact at $C' \to C''$, $\#token = x - 2$ holds in C''. From the global fairness assumption, since C occurs infinitely often in Ξ, C'' also occurs infinitely often. Since $\#token = x$ continuously holds after some configuration, this is a contradiction. □

Next, by using these lemmas, we show that Algorithm 5 solves the problem under the assumptions.

Theorem **20.** *Algorithm 5 solves the uniform 2-partition problem. That is, there exists a protocol with four states and designated initial states that solves the* *uniform 2-partition problem under global fairness on arbitrary communication graphs.*

Proof. Since $\#token$ is reduced only by transition rules 1 and 2, $\#token$ is reduced by two in an interaction. This implies that, by Lemma 35, when n is even (resp., odd), $\#token = 0$ (resp., $\#token = 1$) holds after some configuration C.

First, we consider the case where n is even. By Lemma 34, if $\#token = 0$ holds, $\#r = \#b$ holds. Hence, since $n = \#r + \#b + \#token$ holds, $\#r = \#b = n/2$ holds at C. Moreover, since agents can change their colors only if $\#token \geq 2$ holds, they do not change their colors after C. Hence C is a stable configuration, and thus the uniform 2-partition is completed.

Next, we consider the case where n is odd. When $\#token = 1$ holds, we consider two cases. If an agent in state r^ω exists at C, by Lemma 34, $\#r = \#b$ holds and thus $\#red = \#r + \#r^\omega = \#b + 1 = \#blue + 1$ holds. If an agent in state b^ω exists at C, by Lemma 34, $\#r = \#b + 2$ holds and thus $\#red = \#r = \#b + \#b^\omega + 1 = \#blue + 1$ holds. Hence, in both cases, $\#red - \#blue = 1$ holds at C. Since agents can change their colors only if $\#token \geq 2$ holds, they do not change their colors after C. Hence C is a stable configuration, and thus the uniform 2-partition is completed. $\square$

3.2 Upper Bound for Symmetric Protocols under Global Fairness

In this subsection, with arbitrary communication graphs with designated initial states and no base station under global fairness, we give a symmetric protocol with five states. We obtain the symmetric protocol by applying a transformer proposed in [22] to the protocol in subsection 3.1. The transformer simulates an asymmetric protocol on a symmetric protocol. To do this, the transformer requires additional states. Moreover, the transformer works with a complete communication graphs. We show that one additional state is sufficient to transform from the asymmetric uniform 2-partition protocol to symmetric one even if we assume arbitrary graphs.

Observe that, with designated initial states and no base station, clearly no symmetric protocol can solve the problem if the number of agents n is two (the state of the two agents is the same in the initial state, so symmetry is never

Algorithm 6 Uniform 2-partition protocol with five states.

```
A state set
    $Q = \{r_0^\omega, r_1^\omega, b^\omega, r, b\}$
A mapping to colors
    $\gamma(r_0^\omega) = \gamma(r_1^\omega) = \gamma(r) = red$
    $\gamma(b^\omega) = \gamma(b) = blue$
Transition rules
     1. $(r_0^\omega, r_0^\omega) \to (r_1^\omega, r_1^\omega)$
     2. $(r_1^\omega, r_1^\omega) \to (r_0^\omega, r_0^\omega)$
     3. $(r_0^\omega, r_1^\omega) \to (r, b)$
     4. $(r_0^\omega, b^\omega) \to (b, b)$
     5. $(r_1^\omega, b^\omega) \to (b, b)$
     6. $(r_0^\omega, r) \to (r, r_0^\omega)$
     7. $(r_1^\omega, r) \to (r, r_0^\omega)$
     8. $(b^\omega, b) \to (b, b^\omega)$
     9. $(r_0^\omega, b) \to (r, b^\omega)$
    10. $(r_1^\omega, b) \to (r, b^\omega)$
    11. $(b^\omega, r) \to (b, r_0^\omega)$
```

broken and uniform 2-partition cannot occur). Thus, we assume that $n \geq 3$ holds.

We define a state set of agents as $Q = \{r_0^\omega, r_1^\omega, b^\omega, r, b\}$, and function γ as follows: $\gamma(r_0^\omega) = \gamma(r_1^\omega) = \gamma(r) = red$ and $\gamma(b^\omega) = \gamma(b) = blue$. We say an agent has a token if its state is r_0^ω, r_1^ω , or b^ω. Initially, every agent has state r_0^ω, that is, every agent is *red* and has a token. The transition rules are given in Algorithm 6.

The idea of Algorithm 6 is similar to Algorithm 5. That is, when two agents with tokens interact and one of them is *red*, a *red* agent transitions to *blue*, and the two tokens are deleted. Then, eventually $\lfloor n/2 \rfloor$ *blue* agents appear and $\lceil n/2 \rceil$ *red* agents remain after all tokens (except one token for the case of odd n) disappear. However, to make a *red* agent transition to a *blue* agent in the first place, Algorithm 5 includes transition rule 1 that makes agents with the same states transition to different states. This implies that Algorithm 5 is not symmetric. Hence, by borrowing the technique proposed in [22], we improve Algorithm 5 so that the new protocol (Algorithm 6) makes a *red* agent transition to a *blue* agent without such a transition (and two tokens are deleted at that time). Concretely, we achieve it as follows. In Algorithm 6, there are two states r_0^ω and r_1^ω that are *red* and have a token. When two agents with r_0^ω interact, they transition to r_1^ω, and vice versa (transition rules 1 and 2). Thus, under global fairness, eventually an agent with r_0^ω interacts with an agent with r_1^ω and then one of them transitions to *blue*, and two tokens are deleted (transition rule 3). Observe that these transitions do not affect the essence of Algorithm 5. This is because the numbers of *blue* agents, *red* agents, and tokens do not change after transition rules 1 and 2, and a *red* agent transitions to *blue*, and two tokens are deleted at transition rule 3.

We now prove the correctness of Algorithm 6 along the proof in subsection 3.1. We define $\#r$, $\#b$, $\#r_0^\omega$, $\#r_1^\omega$, $\#b^\omega$ as the number of agents that have state r, b, r_0^ω, r_1^ω, b^ω, respectively. Let $\#red = \#r + \#r_0^\omega + \#r_1^\omega$ and $\#blue = \#b + \#b^\omega$ be the number of *red* and *blue* agents, respectively. Let $\#token = \#r_0^\omega + \#r_1^\omega + \#b^\omega$ be the number of agents with tokens.

Recall that a mechanism of symmetry breaking (transition rules 1, 2, and 3) does not affect the essence of Algorithm 5. In particular, if $r_0^\omega = r_1^\omega$ $(= r^\omega)$ holds, Algorithm 6 is equal to Algorithm 5. Thus, since an equation of Lemma 34 holds in any configuration of any execution of Algorithm 6 and $\#r^\omega$ is not related to the equation, the following corollary holds.

Corollary **8.** *In any globally-fair execution of Algorithm 6, $\#r = \#b + 2\cdot\#b^\omega$ holds in any configuration.*

If a *blue* agent occurs, both *red* agent and *blue* agent exist afterwards with Algorithm 6. This is because, similarly to Algorithm 5, there is no transition rule that decreases the number of *blue* agents in Algorithm 6, and *red* agents also exist by Corollary 8.

Now, we simply show that eventually a *blue* agent occurs in any execution Ξ. When $n = \#r_0^\omega + \#r_1^\omega$ holds, only transition rules 1, 2, and 3 can occur. Hence, from the global fairness assumption, there is a configuration C such that, for some adjacent agents x and y, x has r_0^ω and y has r_1^ω in C and C occurs infinitely often in Ξ. Then, eventually x with r_0^ω and y with r_1^ω interact. By the interaction, transition rule 3 happens and thus one *blue* agent occurs.

Thus, we can observe that both *red* agent and *blue* agent exist after some configuration. Then, clearly we can obtain the following lemma by the same argument as in Lemma 35.

Lemma **36.** *For any globally-fair execution of Algorithm 6, $\#token \leq 1$ holds after some configuration.*

Finally, we show, similarly to Theorem 20, that Algorithm 6 solves the uniform 2-partition problem.

Theorem **21.** *Algorithm 6 solves the uniform 2-partition problem. That is, there exists a symmetric protocol with five states and designated initial states that solves the uniform 2-partition problem under global fairness with arbitrary communication graphs.*

Proof. Since $\#token$ is reduced only in transition rules 3, 4, and 5, $\#token$ is reduced by two in an interaction. This implies that, by Lemma 36, when n is even (resp., odd), $\#token = 0$ (resp., $\#token = 1$) holds after some configuration C.

After $\#token \leq 1$ holds, Algorithm 6 is substantially the same as Algorithm 5 because transition rules 1, 2, and 3 do not occur if $\#token \leq 1$ holds. Thus, we can prove the remaining part in the same way as Theorem 20 by using Corollary 8 instead of Lemma 34. □

3.3 Lower Bound for Asymmetric Protocols under Global Fairness

In this section, we show that, on arbitrary graphs with designated initial states and no base station under global fairness, there exists no asymmetric protocol with three states.

To prove this, we first show that, when the number of agents n is odd and no more than $P/2$, each agent changes its own state to another state infinitely often in any globally-fair execution Ξ of any uniform 2-partition protocol, where P is a known upper bound of the number of agents. This proposition holds regardless of the number of states in a protocol.

After that, we prove impossibility of an asymmetric protocol with three states. From the above proposition, in any globally-fair execution of any uniform 2-partition protocol, agents change their states infinitely often. Now, since the number of states is three, the number of *red* states or the number of *blue* states is one. This implies that some agents change their color after a stable configuration. Thus, no protocol can solve the uniform 2-partition problem.

From now, we show that, in any globally-fair execution Ξ of a protocol Alg solving uniform 2-partition on an arbitrary communication graph such that the number of agents $n < P/2$ is odd, all agents transition their own state to another state infinitely often.

Lemma **37.** *Assume that there exists a uniform 2-partition protocol Alg with designated initial states on arbitrary communication graphs assuming global fairness. Let us consider a graph* $G = (V, E)$ *such that the number of agents* n *is odd and no more than* $P/2$*. In any globally-fair execution* $\Xi = C_0, C_1, \dots$ *of* Alg *on* G*, each agent changes its state infinitely often.*

Proof. Let $V = \{v_0, v_1, v_2, v_3, \dots, v_{n-1}\}$. Assume, for the purpose of contradiction, that there exists v_α that does not change its state after some stable configuration C_h in globally-fair execution Ξ. Let s_α be a state that v_α has after C_h. Since the number of states is finite, in Ξ, there exists a stable configuration C_t that appears infinitely often after C_h. Without loss of generality, $\#red(V) - \#blue(V) = 1$ holds after C_h. Let v_β be an agent that is adjacent to v_α. Let S_β be a set of states that v_β has after C_h.

Next, consider a communication graph $G' = (V', E')$ that satisfies the following.

- $V' = V'_1 \cup V'_2$, where $V'_1 = \{v'_0, v'_1, v'_2, \dots, v'_{n-1}\}$ and $V'_2 = \{v'_n, v'_{n+1}, v'_{n+2}, \dots, v'_{2n-1}\}$.
- $E' = \{(v'_x, v'_y), (v'_{x+n}, v'_{y+n}) \in V' \times V' \mid (v_x, v_y) \in E\} \cup \{(v'_\alpha, v'_{n+\beta})\}$.

An example ($n = 5$) of G and G' is shown in Figure 9.

Over G', we construct an execution Ξ' such that, agents in V'_1 and V'_2 behave similarly to Ξ until states of agents in V'_1 and V'_2 converge, and then make interactions so that Ξ' satisfies global fairness. Concretely, consider a globally-fair execution $\Xi' = C'_0, C'_1, C'_2, C'_3, \dots$ on G' as follows:

- For $i \le t$, when v_x interacts with v_y at $C_i \to C_{i+1}$, v'_x interacts with v'_y at $C'_{2i} \to C'_{2i+1}$, and v'_{x+n} interacts with v'_{y+n} at $C'_{2i+1} \to C'_{2i+2}$.

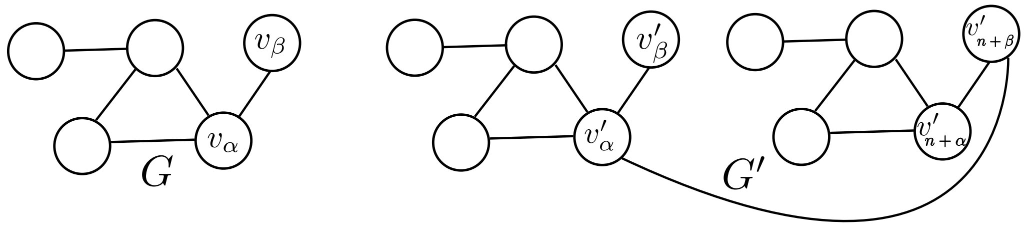


Figure 9. An example of communication graphs G and G' ($n = 5$).

- After C'_{2t}, agents make interactions so that Ξ' satisfies global fairness.

The outline of the remaining part of the proof is as follows. Since Ξ is globally-fair, we can show that the following facts hold after states of agents in V'_1 and V'_2 converge in Ξ'.

- v'_α has state s_α as long as $v'_{n+\beta}$ has a state in S_β.
- $v'_{n+\beta}$ has a state in S_β as long as v'_α has state s_α.

From these facts, in Ξ', v'_α continues to have state s_α and $v'_{n+\beta}$ continues to have a state in S_β. Hence, in Ξ', each agent in V'_1 cannot notice the existence of agents in V'_2, and vice versa. Thus, since we assume, without loss of generality, that $\#red(V) - \#blue(V) = 1$ holds in stable configuration C_h of Ξ, $\#red(V') - \#blue(V') = 2$ holds in a stable configuration of Ξ'.

Now, we show the details of the proof.

In the following, for configuration C' in Ξ' and configuration C in Ξ, we say C' of V'_1 (resp., V'_2) is equivalent to C if $s(v'_x, C') = s(v_x, C)$ (resp., $s(v'_{x+n}, C') = s(v_x, C)$) holds for any $v_x \in V$. Observe that, by the definition of Ξ', C'_{2t} of V'_1 and V'_2 is equivalent to C_t.

From now, by induction on the index of configuration, we prove the proposition that, for any configuration C'_m that occurs after C'_{2t}, there is a configuration C_a (resp., C_b) such that 1) C'_m of V'_1 (resp., V'_2) is equivalent to C_a (resp., C_b) and 2) C_a (resp., C_b) appears infinitely often in Ξ.

The base case is $C'_m = C'_{2t}$. Since C_t appears infinitely often in Ξ and C'_{2t} of V'_1 and V'_2 is equivalent to C_t, the base case holds.

For the induction step, assume that, for $C'_m (m \geq 2t)$, there are configurations C_a and C_b that satisfy the conditions. We consider two cases for an interaction at $C'_m \to C'_{m+1}$. The first case considers an interaction of v'_α and $v'_{n+\beta}$, and the second case considers other interactions.

First, we consider the case where v'_α and $v'_{n+\beta}$ interact at $C'_m \to C'_{m+1}$. By the assumption, we have $s(v'_\alpha, C'_m) = s(v_\alpha, C_a) = s_\alpha$, $s(v'_{n+\alpha}, C'_m) = s(v_\alpha, C_b) = s_\alpha$, and $s(v'_{n+\beta}, C'_m) = s(v_\beta, C_b)$. Hence, since v_α does not change its state even when it interacts with v_β at C_b, any transition rule is of the form $(s(v_\beta, C_b), s_\alpha) \to (s, s_\alpha)$, where s is some state. This implies that, when v'_α and $v'_{n+\beta}$ interact at C'_m, v'_α keeps the state s_α. Thus, C'_{m+1} of V'_1 is still equivalent to C_a. Additionally, since $s(v'_\alpha, C'_m) = s(v'_{n+\alpha}, C'_m) = s(v_\alpha, C_b) = s_\alpha$ and $s(v'_{n+\beta}, C'_m) = s(v_\beta, C_b)$ hold, $v'_{n+\beta}$ changes its state similarly to the case where v_β interacts with v_α in C_b. That is, letting $C_{b'}$ be a configuration immediately after v_α and v_β interact at C_b, C'_{m+1} of V'_2 is equivalent to $C_{b'}$. Since C_b occurs infinitely often in Ξ and Ξ is globally fair, $C_{b'}$ occurs infinitely often in Ξ. Thus, the proposition holds for C'_{m+1}.

Next, we consider the case where at least one agent other than v'_α and $v'_{n+\beta}$ joins an interaction at $C'_m \to C'_{m+1}$. By the definition, no edge other than $(v'_\alpha, v'_{n+\beta})$ connects V'_1 and V'_2. Hence, if v'_i and v'_j interact at $C'_m \to C'_{m+1}$, either $v'_i \in V'_1 \wedge v'_j \in V'_1$ or $v'_i \in V'_2 \wedge v'_j \in V'_2$ holds. In the former case, letting $C_{a'}$ be the configuration immediately after v_i and v_j interact at C_a, C'_{m+1} of V'_1 is equivalent to $C_{a'}$ and C'_{m+1} of V'_2 is still equivalent to C_b. In the latter case, letting $C_{b'}$ be the configuration immediately after v_{i-n} and v_{j-n} interact at C_b, C'_{m+1} of V'_1 is still equivalent to C_a and C'_{m+1} of V'_2 is equivalent to $C_{b'}$. Since C_a and C_b occur infinitely often in Ξ and Ξ is globally fair, such $C_{a'}$ and $C_{b'}$ occur infinitely often in Ξ and thus the proposition holds in the case.

Recall that, we assumed that $\#red(V) - \#blue(V) = 1$ after C_h in Ξ. Hence, $\#red(V'_1) - \#blue(V'_1) = 1$ and $\#red(V'_2) - \#blue(V'_2) = 1$ holds after C'_{2t} in Ξ'. Thus, $\#red(V') - \#blue(V') = 2$ holds after C'_{2t} in Ξ'. Since Ξ' is globally fair, this is a contradiction. □

From now, by using Lemma 37, we show the theorem.

Theorem **22.** *There exists no uniform 2-partition protocol with three states and designated initial states on arbitrary communication graphs assuming global fairness when the upper bound of the number of agents $P \geq 6$ is given.*

Proof. For the purpose of contradiction, we assume that such a protocol *Alg* exists.

Let $S = \{s_1, s_2, s_3\}$ be a state set of agents. Without loss of generality, $\gamma(s_1) = \gamma(s_2) = red$ and $\gamma(s_3) = blue$ hold. Let us consider a globally-fair execution Ξ of Alg on graph G such that the number of agents is odd and no more than $P/2$. By Lemma 37, after some stable configuration C_t in Ξ, each agent changes its state infinitely often. This implies that each agent with s_3 transitions to s_1 or s_2 after C_t. That is, each *blue* agent transitions to *red* state after C_t. Since C_t is stable, this is a contradiction. $\square$

Furthermore, by Lemma 37, we can obtain the following corollary.

Corollary **9.** *Assume that there exists a uniform 2-partition protocol with designated initial states on arbitrary communication graphs assuming global fairness (when the upper bound P of the number of agents is not given). Let us consider a graph $G = (V, E)$ such that the number of agents n is odd. In any globally-fair execution $\Xi = C_0, C_1, \ldots$ of the protocol on G, each agent changes its state infinitely often.*

By this corollary, we have the following theorem by the same argument as in Theorem 22.

Theorem **23.** *There exists no uniform 2-partition protocol with three states and designated initial states on arbitrary communication graphs assuming global fairness (when the upper bound P of the number of agents is not given).*

3.4 Lower Bound for Symmetric Protocols under Global Fairness

In this section, we show that, with arbitrary communication graphs, designated initial states, and no base station assuming global fairness, there exists no symmetric protocol with four states. Recall that, with designated initial states and no base station, clearly any symmetric protocol never solves the problem if the number of agents n is two. Thus, we assume that $3 \le n \le P$ holds, where P is a known upper bound of the number of agents. Note that the symmetric protocol proposed in subsection 3.2 solves the problem for $3 \le n \le P$.

In this subsection, we newly define $q \overset{sym}{\rightsquigarrow} q'$ as follows:

- For states q and q', we say $q \overset{sym}{\rightsquigarrow} q'$ if there exists a sequence of states $q = q_0, q_1, \cdots, q_k = q'$ such that, for any i $(0 \le i < k)$, transition rule $(q_i, q_i) \to (q_{i+1}, q_{i+1})$ exists.

Moreover, we say two agents are homonyms if they have the same state. Intuitively, $q \overset{sym}{\rightsquigarrow} q'$ means that an agent in state q can transition to q' by a sequence of interactions with homonyms.

Theorem **24.** *There exists no symmetric protocol for the uniform 2-partition with four states and designated initial states on arbitrary graph assuming global fairness when P is fourteen or more.*

For the purpose of contradiction, suppose that there exists such a protocol *Alg*. To begin with, we give a roadmap of the proof. In this proof, by using a property of symmetry, fist we prove some lemmas to show the existence of a state ini_r that satisfies the following conditions.

1. State ini_r is an initial state of each agent, and, for a state ini_b, there are transition rules $(ini_r, ini_r) \to (ini_b, ini_b)$ and $(ini_b, ini_b) \to (ini_r, ini_r)$.

2. There is no transition rule that increases the number of agents with ini_r or ini_b.

3. If a given graph is a complete graph, the number of agents with ini_r or ini_b converges to at most 1.

4. Agents can change their colors only if an agent with $s_1 \in \{ini_r, ini_b\}$ and an agent with $s_2 \in \{ini_r, ini_b\}$ interact.

5. If a given graph is a complete graph with three agents, there exists a stable configuration such that there exists exactly one agent with $s \in \{ini_r, ini_b\}$ and other agents have different color from a color of s.

Then, we consider a graph G' shown as the right side of Figure 10. On the graph, we consider four subgraphs G'_1, G'_2, G'_3, and G'_4. By making interactions on each subgraph, we make a configuration such that, agents on subgraph G'_5 have $s \in \{ini_r, ini_b\}$, and other agents have different color from a color of s and have neither ini_r nor ini_b (this is possible by the above conditions 3 and 5). Then,

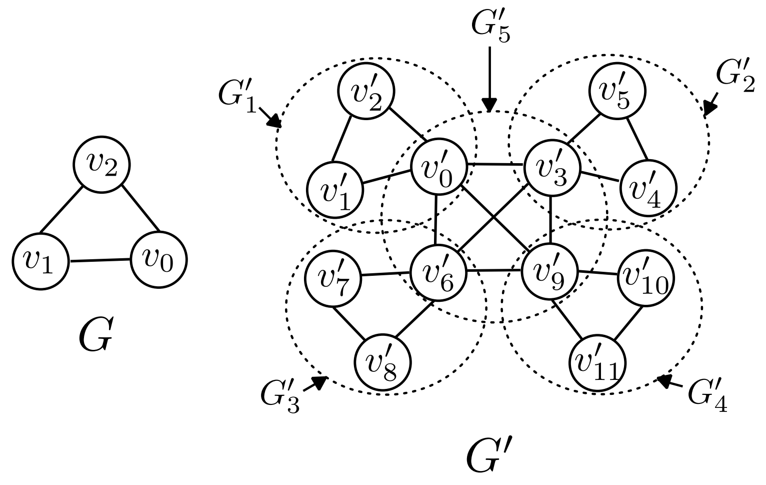


Figure 10. An image of graphs G and G'.

we make agents on G'_5 transition to an initial state (this is possible by the above condition 1). After that, we make interactions on G'_5 until agents converge. Let C be a resulting configuration. By the above condition 3, in C, the number of agents with ini_r or ini_b on G'_5 is at most 1. Moreover, in C, other agents (agents not on G'_5) have neither ini_r nor ini_b. Thus, by the above condition 2, the number of agents with ini_r or ini_b on G' is at most 1 after C. By the above condition 4, agents do not change their colors after C. Since agents not on G'_5 have different color from a color of s, they have the same color. Thus, after C, there are at least eight agents with the same color whereas the number of agents is twelve. Therefore, the uniform 2-partition problem cannot be solved.

Now, we show the details of the proof. Let R (resp., B) be a state set such that, for any $s \in R$ (resp., $s' \in B$), $\gamma(s) = red$ (resp., $\gamma(s') = blue$) holds. First, we show that the following lemma holds from Lemma 37.

Lemma **38.** *$|R| = |B| = 2$ holds.*

Proof. Assume, for the purpose of contradiction, that $|R| \neq |B|$ holds. Without loss of generality, assume that $|R| = 1$ and $|B| = 3$ hold (clearly $R \neq \varnothing$ holds and thus only this combination is valid), and let r be the state belonging to R.

Let us consider a graph G such that the number of agents is odd and no more than $P/2$. At least one agent stabilizes to r in a globally-fair execution of *Alg* with G. By Lemma 37, the agent changes its state to a state in B afterwards. This is a contradiction. $\square$

Let ini_r and r (resp., ini_b and b) be states belonging to R (resp., B). In addition, without loss of generality, assume that ini_r is the initial state of agents.

From the property of symmetry, the following lemma holds.

Lemma **39.** *Let us consider a symmetric transition sequence* $(ini_r, ini_r) \to (p_1, p_1)$, $(p_1, p_1) \to (p_2, p_2)$, $(p_2, p_2) \to (p_3, p_3)$, $\ldots$ *starting from* ini_r. *For any* i, $p_i \neq p_{i+1}$ *holds.*

Proof. For the purpose of contradiction, suppose that $p_i = p_{i+1}$ holds for some i. Let us consider a complete communication graph $G = (V, E)$ such that the number of agents n is four, where $V = \{v_1, v_2, v_3, v_4\}$.

Let us consider a globally-fair execution Ξ as follows:

- v_1 (resp., v_3) interacts with v_2 (resp., v_4) i times.
- After that, make interactions so that Ξ satisfies global fairness.

Since the initial state of agents is ini_r, all agents have p_i after the i interactions. By the assumption, since $(p_i, p_i) \to (p_i, p_i)$ holds, every agent keeps state p_i after that. Hence, Ξ cannot reach a stable configuration. Since Ξ is globally fair, this is a contradiction. $\square$

Additionally, we can extend Lemma 39 as follows (we show the detail of this proof later):

Lemma **40.** *There exists some state* $s_b \in B$ *such that* $(ini_r, ini_r) \to (s_b, s_b)$ *and* $(s_b, s_b) \to (ini_r, ini_r)$ *hold.*

Without loss of generality, assume that (ini_r, ini_r) $\to$ (ini_b, ini_b) and (ini_b, ini_b) $\to$ (ini_r, ini_r) exist. For some population V, we denote the number of agents with ini_r (resp., ini_b) belonging to V as $\#ini_r(V)$ (resp., $\#ini_b(V)$). Moreover, let $\#ini(V)$ be the sum of $\#ini_r(V)$ and $\#ini_b(V)$. When V is clear from the context, we simply denote them as $\#ini_r$, $\#ini_b$, and $\#ini$, respectively.

From now, we show that Theorem 24 holds if the following lemmas and corollary hold. We show the detail of these intermediate proofs later (To show Lemma 41, P must be fourteen or more).

Lemma **41.** *There does not exist a transition rule such that $\#ini$ increases after the transition.*

Lemma **42.** *Let us consider a globally-fair execution Ξ of Alg with some complete communication graph G. After some configuration in Ξ, $\#ini \le 1$ holds.*

Corollary **10.** *Let us consider a state set $Ini = \{ini_r,\ ini_b\}$. When $s_1 \notin Ini$ or $s_2 \notin Ini$ holds, if transition rule $(s_1, s_2) \to (s'_1, s'_2)$ exists then $\gamma(s_1) = \gamma(s'_1)$ and $\gamma(s_2) = \gamma(s'_2)$ hold.*

We consider a globally-fair execution $\Xi = C_0, C_1, C_2, \ldots$ of Alg with a complete communication graph $G = (V, E)$ such that the number of agents is three, where $V = \{v_0,\ v_1,\ v_2\}$. In a stable configuration of Ξ, either $\#blue(V) - \#red(V) = 1$ or $\#red(V) - \#blue(V) = 1$ holds.

First, we consider the case of $\#blue(V) - \#red(V) = 1$.

Since $|V| \le P/2$ holds and $|V|$ is odd, by Lemma 37 red agents keep exchanging r for ini_r in Ξ. Hence, there exists a stable configuration C_t of Ξ such that there exists exactly one agent that has ini_r. Without loss of generality, we assume that the agent is v_0.

We consider the communication graph $G' = (V', E')$ that includes four copies of G. The details of G' are as follows:

- Let $V' = \{v'_0,\ v'_1,\ v'_2,\ v'_3,\ \ldots,\ v'_{11}\}$. We define a partition of V' as $V'_1 = \{v'_0,\ v'_1,\ v'_2\}$, $V'_2 = \{v'_3,\ v'_4,\ v'_5\}$, $V'_3 = \{v'_6,\ v'_7,\ v'_8\}$, and $V'_4 = \{v'_9,\ v'_{10},\ v'_{11}\}$. Let $V'_{red} = \{v'_0,\ v'_3,\ v'_6,\ v'_9\}$.
- $E' = \{(v'_x, v'_y),\ (v'_{x+3}, v'_{y+3}),\ (v'_{x+6}, v'_{y+6}),\ (v'_{x+9}, v'_{y+9}) \in V' \times V' \mid (v_x, v_y) \in E\} \cup \{(v'_x, v'_y) \in V' \times V' \mid x,\ y \in \{0,\ 3,\ 6,\ 9\}\}$.

An image of G and G' is shown in Figure 10.

We consider the following execution $\Xi' = C'_0,\ C'_1,\ C'_2,\ \ldots$ of Alg with $G' = (V', E')$.

- For $i \le t$, when v_x interacts with v_y at $C_i \to C_{i+1}$, v'_x interacts with v'_y at $C'_{4i} \to C'_{4i+1}$, v'_{x+3} interacts with v'_{y+3} at $C'_{4i+1} \to C'_{4i+2}$, v'_{x+6} interacts with v'_{y+6} at $C'_{4i+2} \to C'_{4i+3}$, and v'_{x+9} interacts with v'_{y+9} at $C'_{4i+3} \to C'_{4i+4}$.

- After C'_{4t}, make interactions between agents in V'_{red} until states of agents in V'_{red} converge and $\#ini(V'_{red}) \leq 1$ holds. We call the configuration $C'_{t'}$.
- After $C'_{t'}$, make interactions so that Ξ' satisfies global fairness.

Until C'_{4t}, agents in V'_1, V'_2, V'_3, and V'_4 behave similarly to agents in V from C_0 to C_t. This implies that, in C'_{4t}, every agent in V'_{red} has state ini_r. From Lemma 42, since ini_r is the initial state of agents, it is possible to make interactions between agents in V'_{red} until states of agents in V'_{red} converge and $\#ini(V'_{red}) \leq 1$ holds. Moreover, since v_0 is the only agent that has ini_r in C_t, no agent in $V'_i \backslash V'_{red} (1 \leq i \leq 4)$ has state ini_r or ini_b in C'_{4t}. Hence, $\#ini \leq 1$ holds in $C'_{t'}$. By Corollary 10, if $\#ini \geq 2$ does not hold, no agent can change its color. Thus, since $\#ini \leq 1$ holds after $C'_{t'}$ by Lemma 41, no agent can change its color after $C'_{t'}$. Since v_1 and v_2 are *blue* in C_t, v'_1, v'_2, v'_4, v'_5, v'_7, v'_8, v'_{10}, and v'_{11} are *blue* in $C'_{t'}$. In addition, $\#blue(V'_{red}) = \#red(V'_{red})$ holds. Hence, $\#blue(V') - \#red(V') = 8$ holds. Since no agent can change its color after $C'_{t'}$ and Ξ' is globally-fair, this is a contradiction.

Next, consider the case of $\#red(V) - \#blue(V) = 1$. In this case, we can prove in the same way as the case of $\#blue(V) - \#red(V) = 1$. However, in the case, we focus on ini_b instead of ini_r. That is, we assume that agents in V'_{red} (i.e., v'_0, v'_3, v'_6, and v'_9) have ini_b in C'_{4t}. From C'_{4t}, we make v'_0 (resp., v'_6) interact with v'_3 (resp., v'_9) once. Then, by Lemma 40, all of them transition to ini_r. After that, since all agents in V'_{red} have ini_r, we can construct an execution such that only agents in V'_{red} interact and eventually $\#ini(V'_{red}) \leq 1$ holds. As a result, we can lead to contradiction in the same way as the case of $\#blue(V) - \#red(V) = 1$.

Proofs of Lemmas 40, 41, and 42, and Corollary 10

From now, we prove Lemmas 40, 41, and 42, and Corollary 10. First, we show the proof of Lemma 40.

Lemma **40.** *There exists some state $s_b \in B$ such that $(ini_r, ini_r) \rightarrow (s_b, s_b)$ and $(s_b, s_b) \rightarrow (ini_r, ini_r)$ hold.*

Proof. Let us consider a globally-fair execution $\Xi = C_0, C_1, C_2, \ldots$ of *Alg* with a complete communication graph G such that the number of agents n is six. First, we consider a state q such that there are two or more agents with q in a stable configuration C_t of Ξ. Observe that, for any state q' such that $q \overset{sym}{\rightsquigarrow} q'$ holds, $\gamma(q) = \gamma(q')$ holds. This is because, if such equation does not hold, it contradicts the definition of the stable configuration (i.e., it contradicts the fact that each agent cannot change its own color after a stable configuration). Using this fact, we show that the lemma holds.

Since the number of states is four and Lemma 39 holds, there are three possible symmetric transition sequences starting from ini_r as follows:

1. For distinct states ini_r, p_1, p_2, and p_3, there exists a transition sequence $(ini_r, ini_r) \to (p_1, p_1)$, $(p_1, p_1) \to (p_2, p_2)$, $(p_2, p_2) \to (p_3, p_3)$, $(p_3, p_3) \to (x, x)$, $\ldots$, where $x \in \{ini_r, p_1, p_2\}$.

2. For distinct states ini_r, p_1, and p_2, there exists a transition sequence $(ini_r, ini_r) \to (p_1, p_1)$, $(p_1, p_1) \to (p_2, p_2)$, $(p_2, p_2) \to (y, y)$, $\ldots$, where $y \in \{ini_r, p_1\}$.

3. For distinct states ini_r and p_1, there exists a transition sequence $(ini_r, ini_r) \to (p_1, p_1)$, $(p_1, p_1) \to (ini_r, ini_r)$, $(ini_r, ini_r) \to (p_1, p_1)$, $\ldots$.

Case 1: Assume, for the purpose of contradiction, that the transition sequence 1 holds. Let $p \in \{p_1, p_2, p_3\}$ be a state such that $\gamma(p) = red$ holds. By the assumption, in a stable configuration C_t of Ξ, $\#red$ is three. This implies that there exist two agents with ini_r or p. If there exist two agents with ini_r in C_t, they can transition to $p' \in \{p_1, p_2, p_3\}$ such that $\gamma(p') = blue \neq \gamma(ini_r)$ holds by a sequence of interactions with homonyms. Hence, by the definition of stable configurations, there exists at most one agent with ini_r in C_t and thus there exist two or more agents with p.

We consider cases of $p = p_3$, $p = p_2$, and $p = p_1$. In the case of $p = p_3$, $\gamma(p_1) = \gamma(p_2) = blue$ holds. By the transition sequence 1, p can transition to p_2 by a sequence of interactions with homonyms. By the definition of stable configurations, since $\gamma(p) \neq \gamma(p_2)$ holds, this case this case cannot occur. In the case of $p = p_2$ or $p = p_1$, $\gamma(p_3) = blue$ holds. By the transition sequence 1, p can transition to p_3 by a sequence of interactions with homonyms. By the definition of stable configurations, since $\gamma(p) \neq \gamma(p_3)$ holds, this case cannot occur. Thus, the transition sequence 1 does not hold.

Case 2: Assume, for the purpose of contradiction, that the transition sequence 2 holds. We show that 1) $\gamma(p_1) = \gamma(p_2) = \textit{blue}$ holds and 2) $y = p_1$ holds. After that, from these facts, we show that 3) the transition sequence 2 does not hold.

First we show 1) $\gamma(p_1) = \gamma(p_2) = \textit{blue}$ holds. Assume, for the purpose of contradiction, that either p_1 or p_2 is a *red* state and the other is a *blue* state (since ini_r is *red* state, either p_1 or p_2 is *blue* state). In this case, since ini_r can transition to *blue* state by a sequence of interactions with homonyms, there exists at most one agent with ini_r in C_t of Ξ. Hence, in C_t, there exists two *red* agents with p_1 or p_2. By the transition sequence 2, p_1 (resp., p_2) can transition to p_2 (resp., p_1) by a sequence of interactions with homonyms. Since $\gamma(p_1) \neq \gamma(p_2)$ holds, this contradicts the definition of stable configuration.

Next, we show 2) $y = p_1$ holds. For the purpose of contradiction, assume that $y = ini_r$ holds. In this case, p_1 and p_2 can transition to ini_r by a sequence of interactions with homonyms. In addition, $\gamma(ini_r) = \textit{red}$ holds. From 1) $\gamma(p_1) = \gamma(p_2) = \textit{blue}$, since $\#\textit{blue}$ is three in a stable configuration C_t of Ξ, there exist two agents with p_1 or p_2. These facts contradict the definition of stable configuration.

Finally, we show 3) the transition sequence 2 does not hold.

Let us consider a complete communication graph $\hat{G} = (\hat{V}, \hat{E})$ such that the number of agents n is six, where $\hat{V} = \{\hat{v_0}, \hat{v_1}, \hat{v_2}, \ldots, \hat{v_5}\}$. Moreover, consider a globally-fair execution $\hat{\Xi}$ as follows:

- $\hat{v_0}$, $\hat{v_1}$, and $\hat{v_2}$ interact with $\hat{v_3}$, $\hat{v_4}$, and $\hat{v_5}$ once, respectively.
- After that, make interactions so that $\hat{\Xi}$ satisfies global fairness.

After the first item, all agents have p_1. Hence, from 1) $\gamma(p_1) = \gamma(p_2) = \textit{blue}$ and 2) $y = p_1$ (i.e., $(p_1, p_1) \to (p_2, p_2)$ and $(p_2, p_2) \to (p_1, p_1)$ hold), there exists transition rule $(p_1, p_2) \to (s_{r1}, s_{r2})$ such that $\gamma(s_{r1}) = \gamma(s_{r2}) = \textit{blue}$ does not hold. Since transition rules $(p_1, p_1) \to (p_2, p_2)$ and $(p_2, p_2) \to (p_1, p_1)$ exist and only p_1 and p_2 are *blue* states, a stable configuration $\hat{C_t}$ of $\hat{\Xi}$ can reach $\hat{C_{t'}}$ such that both p_1 and p_2 exist in $\hat{C_{t'}}$. Additionally, since $(p_1, p_2) \to (s_{r1}, s_{r2})$ exists, some agent can change its color after $\hat{C_{t'}}$. Since $\hat{C_t}$ is stable, this is a contradiction. Thus, the transition sequence 2 does not hold.

Case 3: Since Cases 1 and 2 do not hold, Case 3 holds. Thus, by proving that $\gamma(p_1) = \textit{blue}$ holds, we can obtain the lemma. We assume, for the purpose of contradiction, that $\gamma(p_1) = red$ holds. Let us consider a globally-fair execution Ξ' of Alg with a complete communication graph $G' = (V', E')$ such that the number of agents is six. Since transition rules $(ini_r, ini_r) \to (p_1, p_1)$ and $(p_1, p_1) \to (ini_r, ini_r)$ exist and $\gamma(ini_r) = \gamma(p_1) = red$ holds, there exists transition rule $(ini_r, p_1) \to (s_{b1}, s_{b2})$ such that $\gamma(s_{b1}) = \gamma(s_{b2}) = red$ does not hold. Since transition rules $(ini_r, ini_r) \to (p_1, p_1)$ and $(p_1, p_1) \to (ini_r, ini_r)$ exist and only ini_r and p_1 are red state (and the number of red agents is three in any stable configuration), a stable configuration C'_t of Ξ' can reach $C'_{t'}$ such that both p_1 and ini_r exist in $C'_{t'}$. In addition, since $(ini_r, p_1) \to (s_{b1}, s_{b2})$ exists, some agent can change its color from $C'_{t'}$. Since C'_t is stable, this is a contradiction. Therefore, $\gamma(p_1) = blue$ holds and thus the lemma holds. □

Next, from Lemmas 37 and 40, we prove the following lemmas.

Lemma **43.** *Let us consider a globally-fair execution Ξ of Alg with some complete communication graph $G = (V, E)$. In any stable configuration of Ξ, there is at most one agent with ini_r and at most one agent with ini_b.*

Proof. For the purpose of contradiction, assume that there exists a stable configuration C_t of Ξ such that there are more than one agent with ini_r (or more than one agent with ini_b). Since $G = (V, E)$ is a complete communication graph, two agents with ini_r (or two agents with ini_b) can interact. By Lemma 40, ini_r (resp., ini_b) transitions to ini_b (resp., ini_r). Since $\gamma(ini_r) \neq \gamma(ini_b)$ holds and C_t is stable, this is a contradiction. □

Lemma **44.** *Let us consider a globally-fair execution Ξ of Alg with a complete communication graph $G = (V, E)$ such that the number of agents n is odd and no more than $P/2$. There exists a stable configuration C_{t_1} (resp., C_{t_2}) such that there exists exactly one agent with ini_r (resp., ini_b), and C_{t_1} (resp., C_{t_2}) occurs infinitely often in Ξ.*

Proof. Let us consider a globally-fair execution Ξ of Alg with a complete communication graph $G = (V, E)$ such that the number of agents n is odd and no more than $P/2$. From Lemma 43, if some agent change its state to ini_r (resp., ini_b) in a stable configuration of Ξ, there exists exactly one agent with ini_r (resp., ini_b) in the resulting configuration. Thus, from Lemma 37, since red agents (resp., *blue* agents) keep exchanging r for ini_r (resp., b for ini_b) in Ξ, there is a stable configuration C_{t_1} (resp., C_{t_2}) of Ξ such that there exists exactly one agent with ini_r (resp., ini_b) and C_{t_1} (resp., C_{t_2}) occurs infinitely often in Ξ. □

Lemma **45.** *There exist transition rules $(r,r) \to (r,r)$, $(b,b) \to (b,b)$, and $(r,b) \to (r,b)$.*

Proof. Let us consider a globally-fair execution Ξ of Alg with a complete communication graph $G = (V,E)$ such that the number of agents n is seven. Let us consider a stable configuration C_{t_1} (resp., C_{t_2}) of Ξ such that there exists exactly one agent with ini_r (resp., ini_b) and C_{t_1} (resp., C_{t_2}) occurs infinitely often in Ξ. From Lemma 44, such C_{t_1} and C_{t_2} exist. Note that, since we assume that $P \geq 14$ holds, we can use Lemma 44 (because $n = 7 \leq P/2$ holds and n is odd). Since the number of agents n is seven, there exist at least two agents with r and at least two agents with b in C_{t_1} and C_{t_2}. Hence, since G is a complete graph and C_{t_1} and C_{t_2} are stable, we can say the following.

- If $(r,r) \to (r',r')$ exists, $\gamma(r) = \gamma(r')$ holds.
- If $(b,b) \to (b',b')$ exists, $\gamma(b) = \gamma(b')$ holds.
- If $(r,b) \to (r'',b'')$ exists, $\gamma(r) = \gamma(r'')$ and $\gamma(b) = \gamma(b'')$ hold.

In addition, by Lemma 43, r' and r'' are not ini_r, and, b' and b'' are not ini_b. Therefore, the lemma holds. □

From these lemmas, we can prove Lemma 41.

Lemma **41.** *There does not exist a transition rule such that #ini increases after the transition.*

Proof. For the purpose of contradiction, assume that such a transition rule exists. By Lemma 45, transition rules $(r,r) \to (r,r)$, $(b,b) \to (b,b)$, and $(r,b) \to (r,b)$ exist and these transition rules do not increase $\#ini$. Hence, there exists $(rb, ini) \to (ini_1, ini_2)$ such that $rb \in \{r,b\}$, and ini, ini_1, $ini_2 \in \{ini_r, ini_b\}^3$ hold.

Let us consider a globally-fair execution Ξ of Alg with a complete graph $G = (V,E)$ such that the number of agents n is five. Moreover, consider a stable configuration C_{t_1} (resp., C_{t_2}) such that there exists exactly one agent with ini_r

(resp., ini_b) and C_{t_1} (resp., C_{t_2}) occurs infinitely often in Ξ. From Lemma 44, such C_{t_1} and C_{t_2} exist. Note that, because $n = 5 \leq P/2$ and n is odd, we can use Lemma 44. Since the number of agents n is five, there exists at least one agent with r and at least one agent with b in C_{t_1} and C_{t_2}. This implies that, an agent with ini_r (resp., ini_b) can interact with an agent with r in C_{t_1} (resp., C_{t_2}). Similarly, an agent with ini_r (resp., ini_b) can interact with an agent with b in C_{t_1} (resp., C_{t_2}). Moreover, since G is a complete graph and Ξ is globally fair, those interactions happen infinitely often.

First, we consider the case where $ini = ini_r$ and $rb = r$ hold. Let us consider an interaction between an agent with ini_r and an agent with r in C_{t_1}. Since any agent cannot change its color in a stable configuration, both agents transition to ini_r by the interaction. However, by Lemma 43, two agents cannot have ini_r in any stable configuration. This is a contradiction. Thus, $ini = ini_r$ and $rb = r$ do not hold. In a similar way, $ini = ini_b$ and $rb = b$ do not hold.

Next, we consider the case where $ini = ini_r$ and $rb = b$ hold. Let us consider an interaction between an agent with ini_r and an agent with b in C_{t_1}. Let $C_{t'_1}$ be configuration that can be obtained from C_{t_1} by the interaction. Since any agent cannot change its color after a stable configuration, one ini_r and one ini_b occur by the interaction. By Lemma 43, there exist exactly one ini_r and exactly one ini_b in $C_{t'_1}$. This implies that, since $C_{t'_1}$ is stable and n is five, there exists at least one agent with b and thus an agent with ini_r can interact with an agent with b in $C_{t'_1}$. Hence, we can obtain $C_{t''_1}$ from $C_{t'_1}$ by making interaction between an agent with ini_r and an agent with b. Clearly, in $C_{t''_1}$, there exist two agents with ini_b. This contradicts Lemma 43 and thus $ini = ini_r$ and $rb = b$ do not hold. In a similar way, we can prove that $ini = ini_b$ and $rb = r$ do not hold. Hence, for any $rb \in \{r, b\}$ and $ini \in \{ini_r, ini_b\}$, transition rule $(rb, ini) \to (ini_1, ini_2)$ with ini_1, $ini_2 \in \{ini_r, ini_b\}$ does not exist. Consequently the lemma holds. □

By the existence of $(ini_r, ini_r) \to (ini_b, ini_b)$ and $(ini_b, ini_b) \to (ini_r, ini_r)$, we can prove the following lemma.

Lemma **46.** *There exists a transition rule $(ini_r, ini_b) \to (x, y)$ such that $x \in \{r, b\}$ or $y \in \{r, b\}$ holds.*

Proof. For the purpose of contradiction, assume that, if there is $(ini_r, ini_b) \to$

(x, y), $x \in \{ini_r, ini_b\}$ and $y \in \{ini_r, ini_b\}$ hold.

Let us consider a globally-fair execution Ξ of Alg with a complete communication graph $G = (V, E)$ such that the number of agents is three. By Lemma 43, there exists at most one agent with ini_r and at most one agent with ini_b in a stable configuration of Ξ. However, by the assumption and the existence of $(ini_r, ini_r) \to (ini_b, ini_b)$ and $(ini_b, ini_b) \to (ini_r, ini_r)$, all agents have ini_r or ini_b permanently in Ξ. This is a contradiction. □

By Lemmas 41, 43, and 46, we show the proof of Lemma 42.

Lemma **42.** *Let us consider a globally-fair execution Ξ of Alg with some complete communication graph G. After some configuration in Ξ, $\#ini \le 1$ holds.*

Proof. Let us consider C_t such that $\#ini \le 2$ holds in C_t and C_t occurs infinitely often in Ξ. By Lemma 43, such a configuration exists. Moreover, by Lemma 41, $\#ini \le 2$ holds even after C_t. First, we consider the case where $\#ini \le 1$ holds in C_t. By Lemma 41, $\#ini \le 1$ holds after C_t and thus the lemma holds immediately in this case. Next, consider the case where $\#ini = 2$ holds in C_t. By Lemma 43, in C_t, there exists an agent v_1 (resp., v_2) with ini_r (resp., ini_b). By Lemma 46, when v_1 interacts with v_2 at $C_t \to C_{t+1}$, $\#ini \le 1$ holds in C_{t+1}. By global fairness, C_{t+1} occurs infinitely often in Ξ. By Lemma 41, $\#ini \le 1$ holds after C_{t+1} and thus the lemma holds. □

From Lemmas 43 and 44, we can obtain the following lemma.

Lemma **47.** *Let $ini \in \{ini_r, ini_b\}$ and $rb \in \{r, b\}$. If $(ini, rb) \to (x, y)$ exists, $\gamma(ini) = \gamma(x)$ and $\gamma(rb) = \gamma(y)$ hold.*

Proof. We assume, for the purpose of contradiction, that there exists $(ini, rb) \to (x, y)$ such that $\gamma(ini) \ne \gamma(x)$ or $\gamma(rb) \ne \gamma(y)$ holds.

Let us consider a globally-fair execution Ξ of Alg with a complete communication graph $G = (V, E)$ such that the number of agents n is five. Because $n = 5 \le P/2$ and n is odd, we can use Lemma 44. Hence, by Lemmas 43 and 44, there exists a stable configuration C_{t_1} (resp., C_{t_2}) of Ξ such that there is v_r (resp., v_b), which is the only agent with ini_r (resp., ini_b) in C_{t_1} (resp., C_{t_2}) and there is v_{rb} with rb. By the assumption, when v_r (resp., v_b) interacts with v_{rb} at

$C_{t_1} \to C_{t_1+1}$ (resp., $C_{t_2} \to C_{t_2+1}$), v_r (resp., v_b) or v_{rb} changes its color. Since C_{t_1} and C_{t_2} are stable, this is a contradiction. □

From Lemmas 45 and 47, we can obtain the following corollary.

Corollary **10.** *Let us consider a state set $Ini = \{ini_r,\ ini_b\}$. When $s_1 \notin Ini$ or $s_2 \notin Ini$ holds, if transition rule $(s_1, s_2) \to (s'_1, s'_2)$ exists then $\gamma(s_1) = \gamma(s'_1)$ and $\gamma(s_2) = \gamma(s'_2)$ hold.*

3.5 Impossibility under Weak Fairness

In this subsection, assuming arbitrary communication graphs and designated initial states and no base station, we show that there is no protocol that solves the problem under weak fairness. Fischer and Jiang [30] proved the impossibility of leader election for a ring communication graph. We borrow their proof technique and apply it to the impossibility proof of a uniform 2-partition problem.

Theorem **25.** *There exists no protocol that solves the uniform 2-partition problem with designated initial states and no base station under weak fairness assuming arbitrary communication graphs.*

Proof. For the purpose of contradiction, let us assume that there exists such a protocol *Alg*.

First, we consider a complete graph G with three agents v_0, v_1, and v_2. Let (v_0, v_1), (v_1, v_2), and (v_2, v_0) be the edges of G. Furthermore, let $\Xi = C_0$, C_1, C_2, …, C_t, … be an execution of *Alg*, where C_t is a stable configuration. Without loss of generality, we assume that $\#red = 1$ and $\#blue = 2$ hold in C_t.

Next, consider a ring G' with six agents such that two copies of G are combined to form G'. Let v'_0, v'_1, v'_2, v'_3, v'_4, and v'_5 be agents of G', and let (v'_0, v'_1), (v'_1, v'_5), (v'_5, v'_3), (v'_3, v'_4), (v'_4, v'_2), and (v'_2, v'_0) be edges of G' (see Figure 11).

Now, let us construct the following execution $\Xi' = C'_0$, C'_1, C'_2, C'_3 ….

- For x and y such that either $x = 0$ or $y = 0$ holds, when v_x interacts with v_y at $C_i \to C_{i+1}$, v'_x interacts with v'_y at $C'_{2i} \to C'_{2i+1}$, and v'_{x+3} interacts with v'_{y+3} at $C'_{2i+1} \to C'_{2i+2}$.

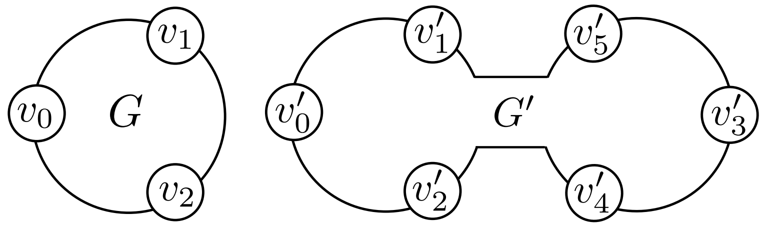


Figure 11. Graphs G and G'.

- When v_1 interacts with v_2 at $C_i \to C_{i+1}$, v'_1 interacts with v'_5 at $C'_{2i} \to C'_{2i+1}$, and v'_4 interacts with v'_2 at $C'_{2i+1} \to C'_{2i+2}$. Similarly, when v_2 interacts with v_1 at $C_i \to C_{i+1}$, v'_5 interacts with v'_1 at $C'_{2i} \to C'_{2i+1}$, and v'_2 interacts with v'_4 at $C'_{2i+1} \to C'_{2i+2}$.

If configurations C of G and C' of G' satisfy the following condition, we say that those configurations are *equivalent*.

- For i $(0 \le i \le 2)$, $s(v_i, C) = s(v'_i, C') = s(v'_{i+3}, C')$ holds.

From now, by induction on the index of configuration, we show that C_r and C'_{2r} are equivalent for any $r \ge 0$. Clearly C_0 and C'_0 are equivalent, so the base case holds immediately. For the induction step, we assume that C_l and C'_{2l} are equivalent, and then consider two cases of interaction at $C_l \to C_{l+1}$.

First we consider the case where, for x and y such that either $x = 0$ or $y = 0$ holds, v_x interacts with v_y at $C_l \to C_{l+1}$. In this case, at $C'_{2l} \to C'_{2l+1}$, v'_x interacts with v'_y and, at $C'_{2l+1} \to C'_{2l+2}$, v'_{x+3} interacts with v'_{y+3}. By the induction assumption, $s(v_x, C_l) = s(v'_x, C'_{2l}) = s(v'_{x+3}, C'_{2l})$ and $s(v_y, C_l) = s(v'_y, C'_{2l}) = s(v'_{y+3}, C'_{2l})$ hold. Thus, agents v'_x and v'_{x+3} change their state similarly to v_x, and agents v'_y and v'_{y+3} change their state similarly to v_y. Hence, C_{l+1} and C'_{2l+2} are equivalent in this case.

Next, we consider the case where v_1 and v_2 interact at $C_l \to C_{l+1}$. When v_1 interacts with v_2 at $C_l \to C_{l+1}$, v'_1 interacts with v'_5 at $C'_{2l} \to C'_{2l+1}$ and v'_4 interacts with v'_2 at $C'_{2l+1} \to C'_{2l+2}$. When v_2 interacts with v_1 at $C_l \to C_{l+1}$, v'_5 interacts with v'_1 at $C'_{2l} \to C'_{2l+1}$ and v'_2 interacts with v'_4 at $C'_{2l+1} \to C'_{2l+2}$. By the induction assumption, $s(v_1, C_l) = s(v'_1, C'_{2l}) = s(v'_4, C'_{2l})$ and $s(v_2, C_l) = s(v'_2, C'_{2l}) = s(v'_5, C'_{2l})$ hold. Thus, agents v'_1 and v'_4 change their state similarly to v_1, and agents v'_2 and

v'_5 change their state similarly to v_2. Hence, C_{l+1} and C'_{2l+2} are equivalent in this case.

Thus, C_r and C'_{2r} are equivalent for any $r \geq 0$.

This implies that, after C'_{2t}, $\#red = 2$ and $\#blue = 4$ hold. Moreover, since Ξ is weakly fair, clearly Ξ' is weakly fair. This is a contradiction. □

4. Concluding Remarks

In this part, we considered the uniform 2-partition problem assuming arbitrary communication graphs. We investigated the problem solvability under various assumptions such as the existence of the base station, initial states of agents, fairness, and symmetry. Concretely, with no base station under global fairness, we proved that four is the minimum number of states per agent to enable asymmetric protocols, and five is the minimum number of states per agent to enable symmetric protocols. With the base station, we showed that three is the minimum number of states per agent. Hence, we clarified that, for the uniform 2-partition under global fairness, there is a difference of just two states between the protocol with and without the base station.

Under weak fairness, we proved the impossibility to obtain an asymmetric protocol with no base station. With a base station, we proposed a symmetric protocol with $3P + 1$ states under weak fairness.

Part V

Graph Class Identification Protocols

1. Introduction

In this part, we aim to clarify the space complexity of the graph class identification problems under various assumptions such as initial states of agents, fairness of the execution, and an initial knowledge of agents. In the population protocol model, Angluin et al. proposed various graph class identification protocols [6]. However, they focused on graph class identification protocols for directed graphs. In this part, we propose graph class identification protocols for undirected graphs. To the best of our knowledge, this is first research that deals with graph class identification problems for undirected graphs.

We remark that some protocols in [6] for directed graphs can be easily extended to undirected graphs with designated initial states under global fairness (see Table 11). 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

Table 11. 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. [6]

opposite directed edges, the two directed edges compose a directed cycle.

1.1 Our Contributions

In this part, we clarify the computability of graph class identification problems for undirected graphs under various assumptions. A summary of our results is given in Table 11. 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.

2. Graph Properties and Problem Definitions

We define graph properties treated in this dissertation 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.

A protocol $\mathcal{P}$ solves the *gp* identification problem if every possible execution Ξ of protocol $\mathcal{P}$ solves the *gp* identification problem.

3. Graph Class Identification Protocols

3.1 Tree Identification Protocol with No Initial Knowledge under Global Fairness

In this section, we give a tree identification protocol (hereinafter referred to as "TI protocol") with 18 states and designated initial states under global fairness.

The basic strategy of the protocol is as follows. First, agents elect one leader token, one right token, and one left token. Agents carry these tokens on a graph by interactions as if each token moves freely on the graph. After the election, agents repeatedly execute a trial to detect a cycle by using the tokens. The trial starts when two adjacent agents x and y have the right token and the left token, 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. Clearly, when the trial succeeds, there is a cycle. In this case, an agent with the leader token recognizes the success of the trial and decides that there is a cycle and thus the given graph is not a tree. Then, the decision is conveyed to all agents by the leader token and thus all agents decide that the given graph is not a tree. 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. Therefore, the protocol solves the problem.

Before we explain the details of the protocol, first we introduce variables at agent a.

- $LF_a \in \{L_{se}, L_l, L_r, L^t_{se}, L^t_{se'}, L_{se'}, L^t_l, L^t_r, \phi\}$: 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). We show the details of them later. 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.

From now, we explain the details of the protocol. The protocol is given in Algorithms 7 and 1.

Election of three tokens (lines 2–8) Initially, each agent has a right token. When two agents with right tokens interact, the agents change one of the tokens to a left token (lines 2–3). When two agents with left tokens interact, the agents change one of the tokens to a leader token (lines 4–5). When two agents with leader tokens interact, the agents delete one of the tokens (lines 6–7). As we explain later, agents carry a token on a graph by interactions as if a token moves freely on the graph. Thus, by the above behaviors, eventually agents elect one right token, one left token, and one leader token.

In the cycle detection part, we will just show behaviors after agents complete the token election (i.e., agents elect one right token, one left token, and one leader token). However, in this protocol, agents may make a wrong decision before agents complete the token election. Agents overcome this problem by the following behaviors.

- Agents behave as if the leader token has the decision, and agents follow the decision. Concretely, when agent a moves the leader token to agent b by an interaction, agent b copies tre_a to tre_b. Since the leader token moves freely on the graph, finally all agents follow the decision of the leader token.
- When two agents with leader tokens interact and agents delete one of them, the agents reset tre of the remaining leader token. That is, if agent a has the remaining leader token, it assigns yes to tre_a (line 8).

Note that the last token is elected by an interaction between agents with the leader tokens (i.e., the last interaction in this election part occurs between agents with the leader tokens). By this interaction, the elected leader token resets its tre to yes. Hence, tre of the leader token is yes just after agents complete the token election, and all agents follow tre of the leader token. Thus, because agents

Algorithm 7 A TI 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 TI protocol (2/2)

```
31:     else if $LF_a \neq \phi \wedge LF_b \neq \phi$ then
32:         if $LF_{ab} \in \{L_{se}, L^t_{se}, L^t_{se'}, L_{se'}\}$ for $ab \in \{a,b\}$ then
33:             $tre_a \leftarrow tre_{ab}$, $tre_b \leftarrow tre_{ab}$
34:         end if
35:         if $LF_{ab} = L^t_\kappa$ for $ab \in \{a,b\}$ and $\kappa \in \{l,r\}$ then
36:             $LF_{ab} \leftarrow L_\kappa$
37:         end if
38:         if $LF_{ab} = L_{se'} \vee LF_{ab} = L^t_{se} \vee LF_{ab} = L^t_{se'}$ for $ab \in \{a,b\}$ then
39:             $LF_{ab} \leftarrow L_{se}$
40:         end if
41:         $LF_a \leftrightarrow LF_b$
42:     end if
43: end
```

correctly detect a cycle after the token election (we will show this later), agents are not affected by the wrong decision.

Movement of tokens (lines 9–18) When an agent having a token interacts with an agent having no token, the agents move the token (lines 9–18). Concretely, the token moves by a behavior of line 18. In lines 10–12, *tre* of the leader token is conveyed. We will explain the behavior of lines 13–17 after the explanation of the trial of the cycle detection.

The trial of the cycle detection (lines 19–43) In this paragraph, we show that, by a trial of the cycle detection, agents correctly detect a cycle after agents complete the token election. To begin with, we explain the start of the trial. 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. Thus, when agents place the right token and the left token, agents make the right token and the left token transition to the trial mode. An L^t_r token (resp., an L^t_l 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 image of the start of the trial is shown in Figure 12. Figures 12(a) and 12(b) show the behavior such that agents make the left token and the right token transition to the trial mode. First, an agent with an L_{se} token changes an L_l token to an L_l^t token by an interaction (Figure 12(a)), where the L_{se} token represents the default leader token. By the interaction, the agents exchange their tokens and the L_{se} token transitions to an $L_{se'}$ token, where the $L_{se'}$ token represents the leader token next to the L_l^t token. This behavior appears in lines 19–21. Then, an agent with the $L_{se'}$ token changes the right token to a trial mode by an interaction (Figure 12(b)). By the interaction, the agents exchange their tokens. Thus, since the $L_{se'}$ token represents the leader token next to the L_l^t token, agents place an L_r^t token next to the L_l^t token by the interaction. Hence, by the interaction, agents place the tokens in the following order: the L_l^t token, the L_r^t token, the leader token (Figure 12(c)). Moreover, by the interaction, the $L_{se'}$ token transitions to an L_{se}^t token, where the L_{se}^t token represents the leader token trying to detect a cycle. This behavior appears in lines 22–24. When agents place all tokens as shown in Figure 12(c), a trial of the cycle detection starts.

From now, we explain the main behavior of the cycle detection (Figure 13 and 14). Let x (resp., y) be an agent having the L_r^t token (resp., the L_l^t token). Let $\mathcal{X}$ (resp., $\mathcal{Y}$) be a set of agents adjacent to x (resp., y). Let $\mathcal{X}' = \mathcal{X}\backslash\{y\}$ and $\mathcal{Y}' = \mathcal{Y}\backslash\{x\}$. In a trial, agents try to carry the leader token from an agent in $\mathcal{X}'$ to an agent in $\mathcal{Y}'$ without using the edge between x and y.

First, we explain the case where a trial succeeds (Figure 13). In the trial, agents carry the L_{se}^t token while the L_r^t token and the L_l^t token are placed at x and y, respectively. Concretely, if the following procedure occurs, the trial succeeds.

1. Agents carry the L_{se}^t token from an agent in $\mathcal{X}'$ to an agent in $\mathcal{Y}'$ without using the edge between x and y (Figure 13(c)).

2. An agent having the L_{se}^t token interacts with agent y having the L_l^t token. By the interaction, agents exchange their tokens and the L_{se}^t token transitions to an $L_{se'}^t$ token (Figure 13(d)). In addition, by the interaction, agents confirm that the L_l^t token was placed at y while agents move the leader token to an agent in $\mathcal{Y}'$. The $L_{se'}^t$ token represents the leader token that confirmed it. This behavior appears in lines 25–27.

3. Agent y having the $L_{se'}^t$ token interacts with agent x having the L_r^t token

(Figure 13(e)). By the interaction, agents confirm that the L_r^t token was placed at x while agents move the leader token to an agent in $\mathcal{Y}'$. This behavior appears in lines 28–30.

Clearly, if there is no cycle, agents do not perform this procedure. Thus, if agents perform this procedure, an agent with the leader token decides that there is a cycle and thus the given graph is not a tree (Figure 13(f)). Concretely, the agent with the leader token changes its tre to no (line 30).

Next, we explain the case where a trial fails (Figure 14). There are three cases where the trial fails: (1) An agent having the $L_{se'}$ or $L_{se'}^t$ token fails to interact with the right token, (2) an agent having the L_l^t or L_r^t token fails to wait for the leader token, and (3) an agent having the L_{se}^t token fails to interact with an agent having the L_l^t token. Case (1) is that an agent having the $L_{se'}$ token (resp., the $L_{se'}^t$ token) interacts with an agent that does not have the L_r token (resp., the L_r^t token). Figure 14(A-1) and (B-1) shows an example of case (1). By the interaction, agents make the token transition to the L_{se} token (lines 9–17 and 31–43). If agents make the $L_{se'}$ token transition to the L_{se} token, the condition in line 22 is never satisfied in the trial. If agents make the $L_{se'}^t$ token transition to the L_{se} token, the condition in line 28 is never satisfied in the trial. Case (2) is that an agent having an L_l^t token (resp., an L_r^t token) interacts with an agent that does not have the L_{se}^t token (resp., the $L_{se'}^t$ token). Figure 14(A-2) and (B-2) shows an example of case (2). By the interaction, agents make the L_l^t token (resp., the L_r^t token) transition to an L_l token (resp., an L_r token) by the behavior of lines 13–14 or 31–37, and thus the condition in line 25 or 28 is never satisfied in the trial. Case (3) is that an agent having the L_{se}^t token interacts with an agent having a token that is not the L_l^t token. Figure 14(A-3) and (B-3) shows an example of case (3). By the interaction, agents make the L_{se}^t token transition to the L_{se} token (lines 31–43). If agents make the L_{se}^t token transition to the L_{se} token, the condition in line 25 is never satisfied in the trial.

Agents have an infinite number of chances of the trial. This is because agents can make the leader token, the left token, and the right token transition to the L_{se} token, the L_l token, and the L_r token, respectively, from any configuration (lines 9–18 and 31–43). Hence, from global fairness, eventually agents make the left and right tokens transition to the trial mode on the cycle and then agents

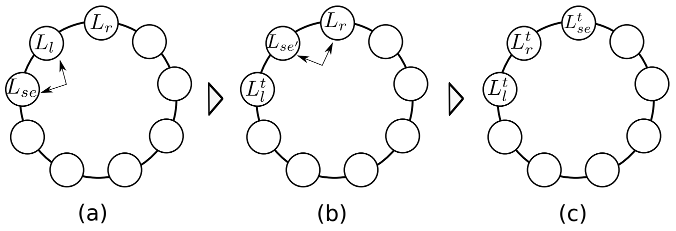


Figure 12. An image of the start of the trial

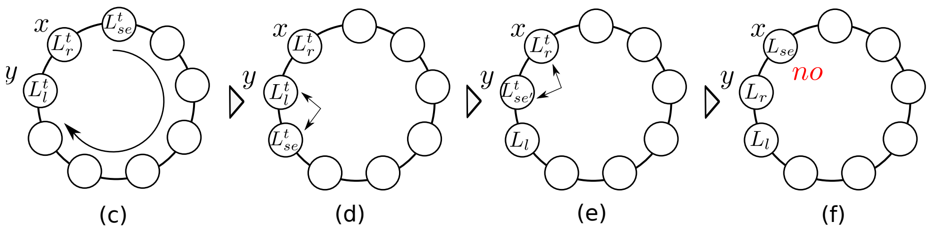


Figure 13. An image of the success of the trial

find the cycle by the leader token. Thus, eventually a trial succeeds if there is a cycle.

By the behaviors of the trial, since *tre* of the leader token is *yes* just after agents complete the token election, *tre* of the leader token converges to a correct value. Since eventually all agents follow the decision of the leader token, all agents correctly decide whether the given graph is a tree or not.

Correctness

First of all, if the number of agents n is less than 3, clearly a leader token is not generated in Algorithm 7. Hence, in the case, tre_a of each agent a converges to *yes*. Thus, since the given graph with $n < 3$ is a tree, each agent make a correct decision in this case. From now on, we consider the case where the number of agents n is at least 3.

To begin with, we define some notions for the numbers of the leader, left, and right tokens as follows:

Definition **19.** *The number of agents with L_r or L_r^t tokens is denoted by $\#L_r$.*

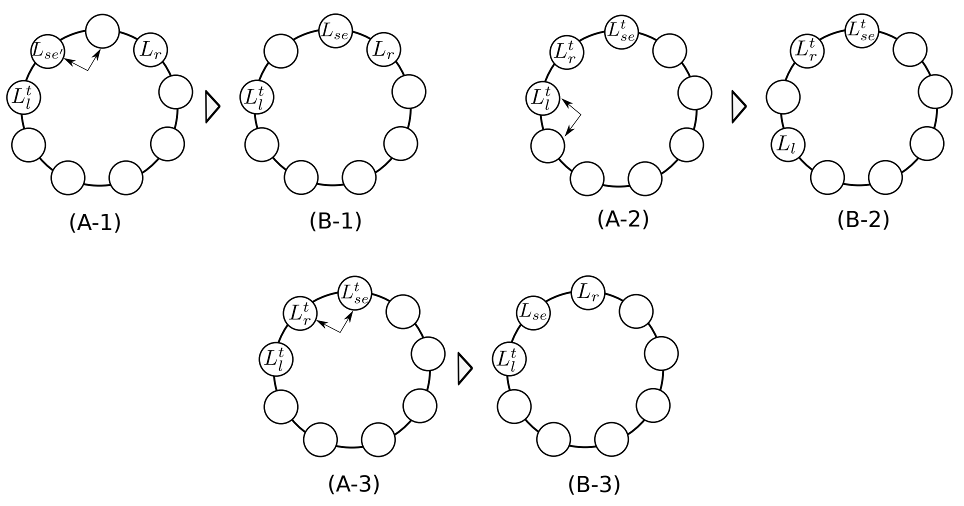


Figure 14. Images of the fail of the trial

The number of agents with L_l or L_l^t tokens is denoted by $\#L_l$. The number of agents with L_{se}, L_{se}^t, $L_{se'}^t$, or $L_{se'}$ tokens is denoted by $\#L_{se}$.

Next, we define a configuration where agents complete the token election.

Definition **20.** *For an execution $\Xi = C_0, C_1, \ldots$, we say that agents complete the token election at C_i if $\#L_r > 1$, $\#L_l > 1$, or $\#L_{se} > 1$ holds in C_{i-1}, and $\#L_r = 1$, $\#L_l = 1$, and $\#L_{se} = 1$ hold in C_i.*

From now, we show that agents eventually complete the token election, and, for agent a with the leader token, $tre_a = yes$ hold just after the election.

Lemma **48.** *For any globally-fair execution $\Xi = C_0, C_1, \ldots$, there is a configuration C_i at which agents complete the token election.*

In C_i, there exists an agent a that has an L_{se} token and tre_a is yes. Moreover, in any configuration after C_i, $\#L_r = 1$, $\#L_l = 1$, and $\#L_{se} = 1$ hold.

Proof. Consider a globally-fair execution $\Xi = C_0, C_1, C_2, \ldots$. From the pseudocode, when an agent having a leader token interacts with an agent having no leader token, agents move the leader token. Similarly, when an agent having a left token (resp., a right token) interacts with an agent having no left token (resp., no right token), agents move the token. Only if an agent having a leader token interacts with an agent having a leader token, the number of leader tokens decreases. Similarly, only if an agent having a left token (resp., a right token) interacts with an agent having a left token (resp., a right token), the number of the tokens decreases. These imply that, from global fairness, if there are two or more tokens of the same type (leader, left, or right), eventually adjacent agents have the tokens and then they interact.

Hence, from global fairness, because there is no behavior to increase $\#L_r$, $\#L_r$ continues to decrease as long as $\#L_r \geq 2$ holds, by the behavior of lines 2–3. Thus, after some configuration, $\#L_r = 1$ holds and the behavior of lines 2–3 does not occur. After that, because there is no behavior to increase $\#L_l$ except for the behavior of lines 2–3, $\#L_l$ continues to decrease as long as $\#L_l \geq 2$ holds, by the behavior of lines 4–5. Thus, after some configuration, $\#L_l = 1$ holds and the behavior of lines 4–5 does not occur. After that, because there is no behavior to increase $\#L_l$ except for the behavior of lines 4–5, $\#L_{se}$ continues to decrease as long as $\#L_{se} \geq 2$ holds, by the behavior of lines 6–7. Thus, after some configuration, $\#L_{se} = 1$ holds. Hence, there exists a configuration C_i such that $\#L_r = 1$, $\#L_l = 1$, and $\#L_{se} = 1$ hold after C_i and $\#L_r > 1$, $\#L_l > 1$, or $\#L_{se} > 1$ holds in C_{i-1}.

If $n > 3$ holds, agents execute lines 6–8 of the pseudocode at transition $C_{i-1} \to C_i$ because only the behavior of lines 6–8 decreases the number of leader tokens. For an agent a with the leader token, the leader token transitions to an L_{se} token and tre_a transitions to yes when agents execute lines 6–8. If $n = 3$ holds, agents execute lines 4–5 of the pseudocode at transition $C_{i-1} \to C_i$, and the first leader token is generated by this transition (and hence the leader token is the L_{se} token and $tre_a = yes$ for an agent a with the L_{se} token). These imply that, in C_i, there exists an agent a that has the L_{se} token and tre_a is yes. Therefore, the lemma holds. □

From Lemma 48, after agents complete the token election, only one leader token remains. From now on, we define tre of the leader token as tre_a such that agent a has the leader token.

From the pseudocode, tre of the leader token is conveyed to all agents. This implies that, after tre of the leader token converges, tre of each agent also converges to the same value as tre of the leader token. Hence, from now, we show that *tre* of the leader token converges to *no* (resp., *yes*) if there is a cycle (resp., no cycle) on the graph.

First, we show that *tre* of the leader token converges to *yes* if there is no cycle on the graph.

Lemma **49.** *For any globally-fair execution Ξ, if there is no cycle on a given communication graph, tre of the leader token converges to yes.*

Proof. Variable *tre* of the leader token transitions to *no* only if agents execute lines 28–30. From Lemma 48, when agents complete the token election, *tre* of the leader token transitions to *yes*. Thus, for the purpose of contradiction, we assume that, for a globally-fair execution Ξ with a graph G containing no cycle, agents execute lines 28–30 after agents complete the token election. From now, let us consider the configuration after agents complete the token election. We first prove that, to execute lines 28–30, agents execute the following procedure.

1. By executing lines 19–21, an $L_{se'}$ token and an L_l^t token are generated.

2. By executing lines 22–24, an L_{se}^t token and an L_r^t token are generated.

3. By executing lines 25–27, an $L_{se'}^t$ token is generated.

4. Agents execute lines 28–30.

From now, we show why agents execute the above procedure to execute lines 28–30. To execute lines 28–30, an $L_{se'}^t$ token is required (line 28). Recall that, when agents complete the token election, the leader token is L_{se}. Hence, to generate an $L_{se'}^t$ token, agents need to execute lines 25–27 (i.e., the item 3 of the procedure is necessary). This is because the behavior of lines 25–27 is the only way to generate an $L_{se'}^t$ token. To execute lines 25–27, an L_{se}^t token is required (line 25). To generate an L_{se}^t token, agents need to execute lines 22–24 (i.e., the item 2 of the procedure is necessary) because the behavior of lines 22–24 is the only way to generate an L_{se}^t token. Similarly, to execute lines 22–24, an $L_{se'}$ token is required (line 22), and, to generate an $L_{se'}$ token, agents need to execute lines 19–21 (i.e., the item 1 of the procedure is necessary) because the behavior of lines 19–21 is the only way to generate an $L_{se'}$ token.

In the procedure, agents may perform the behaviors of some items multiple times by resetting the leader token to a L_{se} token (e.g., agents may perform the behaviors of items 1, 2, 3, and 4 after performing the behaviors of items 1 and 2). However, we can observe that agents finally execute a procedure such that agents perform the behavior of each item only once in the procedure. From now on, we consider only such a procedure.

From the pseudocode, to execute lines 28–30, the following three conditions should hold during the procedure. Note that, after agents complete the token election, $\#L_r = 1$, $\#L_l = 1$, and $\#L_{se} = 1$ hold.

- After executing lines 19–21, an agent having an $L_{se'}$ token does not interact with other agents until the agent interacts with an agent having an L_r token (i.e., the agent interacts only when agents execute lines 22–24). Otherwise, agents make the $L_{se'}$ token transition to an L_{se} token and cannot execute lines 22–24 (i.e., the item 2 of the procedure cannot be executed).

- After executing lines 19–21, an agent having an L_l^t token does not interact with other agents until the agent interacts with an agent having an L_{se}^t token (i.e., the agent interacts only when agents execute lines 25–27). Otherwise, agents make the L_l^t token transition to an L_l token and cannot execute lines 25–27 (i.e., the item 3 of the procedure cannot be executed).

- After executing lines 22–24, an agent having an L_r^t token does not interact with other agents until the agent interacts with an agent having an $L_{se'}^t$ token (i.e., the agent interacts only when agents execute lines 28–30). Otherwise, agents make the L_r^t token transition to an L_r token and cannot execute lines 28–30 (i.e., the item 4 of the procedure cannot be executed).

From items 1 and 2, an L_l^t token exists next to an $L_{se'}$ token when agents execute lines 22–24. Hence, from the pseudocode, an L_r^t token and an L_l^t token are next to each other just after agents execute lines 22–24. In addition, an L_{se}^t token and the L_r^t token are also next to each other just after agents execute lines 22–24. To execute lines 25–27, an agent having the L_{se}^t token must interact with the agent having the L_l^t token without meeting the agent having the L_r^t token. Furthermore, the agent having the L_r^t token must not interact with other agents until agents execute lines 28–30. By the assumption, since agents execute lines 28–30, there are two paths from the agent having the L^t_{se} token to the agent having the L^t_l token just after agents execute lines 22–24. One of the paths is the path via the agent having the L^t_r token. The other is the path without passing through the agent having the L^t_r token. Therefore, there is a cycle in G. This is a contradiction. □

Next, we show that *tre* of the leader token converges to *no* if there is a cycle on the graph.

Lemma **50.** *For any globally-fair execution Ξ, if there is a cycle on a given communication graph, tre of the leader token converges to no.*

Proof. Consider a globally-fair execution Ξ with a graph G containing a cycle. In Ξ, let C be a configuration such that C occurs infinitely often. From Lemma 48, eventually agents complete the token election and thus C occurs infinitely often after agents complete the token election.

Clearly, each condition in lines 2–8 is not satisfied after C. Thus, from the pseudocode, a token moves by any interaction (except for null transitions) after C. This implies that tokens can move freely on G after C. Hence, from global fairness, a configuration C' such that all tokens are on a cycle occurs. Moreover, there exists a configuration C'' such that C'' is reachable from C' and L_{se}, L_l, and L_r tokens are on the cycle in C''. This is because C'' occurs if the following behaviors occur from C'.

1. Making L_l and L_r tokens: If an agent having an L^t_l (or L_l) token and an agent having an L^t_r (or L_r) token can interact in C', they interact and then an L_l token and an L_r token are generated by the behavior of lines 31–43. Otherwise, since the left token and the right token are on a cycle in C' (and hence an agent with the token has at least two edges), each agent having the token can interact with an agent having no token. In the case, an agent having an L^t_l token (resp., an L^t_r token) interacts with an agent having no token, and an L_l token (resp., an L_r token) is generated.

2. Making an L_{se} token: If the leader token is an $L_{se'}$ token or an $L^t_{se'}$ token, an agent having the token interacts with an agent having an L_l token or no

token. As a result, an L_{se} token is generated. If the leader token is an L_{se}^t token, the L_{se}^t token moves to an agent that is on a cycle and is adjacent to an agent with an L_l token (or an L_r token). Then, an agent having the L_{se}^t token interacts with an agent having the L_l token (or the L_r token) and then an L_{se} token is generated.

There exists a configuration such that the configuration is reachable from C'' and, on a cycle, an agent having an L_{se} token is adjacent to an agent having an L_l token in the configuration. This is because tokens can move freely on a graph. In the configuration, agents can execute lines 19–21. If agents execute lines 19–21, the configuration transitions to a configuration such that $L_{se'}$, L_l^t, and L_r tokens exist in a cycle. From the configuration, the L_r token can move to an agent next to an agent with the $L_{se'}$ token while an agent with the $L_{se'}$ token and an agent with the L_l^t token do not interact with any agent. This is because they are on a cycle and the L_r token can move along the cycle. Then, an agent having the $L_{se'}$ token can interact with an agent having the L_r token and then agents execute lines 22–24. Such behavior causes a configuration such that L_{se}^t, L_l^t, and L_r^t tokens are on a cycle. From the configuration, the L_{se}^t token can move to an agent next to an agent having the L_l^t token while an agent having the L_r^t token and an agent having the L_l^t token do not interact with any agent. This is because they are on a cycle and the L_{se}^t token can move along the cycle. Then, an agent having the L_{se}^t token can interact with an agent having the L_l^t token and agents can execute lines 25–27. After that, an agent having an $L_{se'}^t$ token can interact with an agent having the L_r^t token and agents can execute lines 28–30. Hence, from global fairness, since each of the configurations occurs infinitely often, agents execute lines 28–30 infinitely often and agents assign *no* to *tre* of the leader token infinitely often. Although *tre* of the leader token transitions to *yes* if agents execute line 8, agents does not execute line 8 after C. Therefore, the lemma holds. □

From Lemmas 48, 49, and 50, we prove the following theorem.

Theorem **26.** *Algorithms 7 and 1 solve the tree identification problem. That is, there exists a protocol with constant states and designated initial states that solves the tree identification problem under global fairness.*

Proof. From Lemma 48, there is a configuration C such that *tre* of the leader token is *yes* in C and agents complete the token election at C. Hence, from Lemmas 49 and 50, if there is a cycle (resp., no cycle) in a given communication graph, *tre* of the leader token converges to *no* (resp., *yes*). From the pseudocode, since each token can move freely on the graph, *tre* of each agent converges to the same value of *tre* of the leader token. Thus, if there is a cycle (resp., no cycle) in a given communication graph, *tre* of each agent converges to *no* (resp., *yes*). Therefore, the theorem holds. □

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

In this subsection, we give a k-regular identification protocol (hereinafter referred to as "kRI protocol") with $O(k \log P)$ states and designated initial states under global fairness. In this protocol, the upper bound P of the number of agents is given. However, we also show that the protocol solves the problem with $O(k \log n)$ states if the number of agents n is given.

From now, we explain the basic strategy of the protocol. 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 (we explain later how to increase the level), 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. Hence, 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. Concretely, each agent examines whether its degree is at least k, and whether its degree is at least $k + 1$. If an 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. Each agent 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 stores this information locally by making 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.

Before we explain the details of the protocol, first we introduce other variables at agent a.

- $LF_a \in \{L_0, L_1, \ldots, L_k, \phi, \phi'\}$: 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.

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.

Now, we explain the details of the protocol. The protocol is given in Algorithm 2.

The election of leader tokens with levels (lines 2–7 and 26–30 of the pseudocode) Initially, each agent has the leader token and the level of each agent is 0. If two agents with leader tokens at the same level interact, agents delete one of the leader tokens and increase the level of the agent with the remaining leader token by one (lines 2–5). Moreover, loc_a and reg_a transition to *no* (lines 6–7), where agent a is the agent with the remaining leader token. Next, we consider the case where two agents at different levels interact. If an agent a at

Algorithm 2 A kRI 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 larger level interacts with an agent b at the smaller level, agent b update its level to the same level as the larger level (regardless of possession of the leader token). This behavior appears in lines 26–27. Furthermore, at the interaction, agent b resets loc_b to *no* (line 28), and agent b deletes its leader token if agent b has the leader token (line 29). We can observe that there is level *lev_last* such that all agents converge to level *lev_last*, because agents update their levels by only above behaviors and there is no behavior that increases the number of leader tokens. Since an agent at the largest level updates its level only if the agent has the leader token, there is an agent with the leader token at the largest level in any configuration. Thus, since each agent converges to level *lev_last* and the leader token moves freely among agents at the same level (we will show this movement behavior later), eventually agents elect a leader token by above behaviors.

Agents at the largest level delete the leader token only by the behavior of lines 3–7. This implies that, if at least two agents at the largest level have the leader token, eventually agents at the largest level with the leader tokens interact and then the largest level is updated. Hence, only one leader token can obtain level *lev_last*. When an agent a with the leader token updates its level to level *lev_last* by the behavior of lines 3–7, agent a resets loc_a and reg_a to *no*. Since other agents are at levels smaller than level *lev_last* just after the interaction, other agents will reset their *loc* to *no* by the behavior of line 28. From these facts, the interaction causes a configuration such that 1) the number of agents with the leader token at level *lev_last* is one, 2) $reg_a = no$ and $loc_a = no$ hold for the agent a with the leader token at level *lev_last*, and 3) other agents will reset their *loc* after the configuration.

Note that, since agents delete one leader token by the behavior of lines 3–7, at most half of leader tokens at level i update their level to $i+1$ for $0 \leq i$. Thus, since there is no behavior that increases the number of leader tokens, the maximum level is at most $\lfloor \log n \rfloor$. In this protocol, since only the upper bound P of the number of agents is given, the maximum level is at most $\lfloor \log P \rfloor$.

Search for an agent whose degree is at least k or at least $k+1$ with levels (lines 8–18 of the pseudocode) First of all, this search behavior is performed only on the same level (line 2). Recall that eventually all agents converge to the same level (and agents discard the information at other levels).

In this behavior, to examine degrees of agents, agents use the leader token. An agent having the leader token confirms whether the agent can interact with k different agents, so that the agent confirms that its degree is at least k. To confirm it, the agent marks adjacent agents one by one and counts how many times the agent interacts with a non-marked agent. Concretely, when an agent a having the L_i token interacts with an agent b having ϕ, agent a marks agent b (i.e., LF_b transitions to ϕ'). At the interaction, a makes the L_i token transition to the L_{i+1} token. These behaviors appear in lines 8–9. If agent a obtains the L_j token by such an interaction, agent a has interacted with j different agents because a has marked j non-marked agents. Thus, when agent a having the L_{k-1} token interacts with an agent with ϕ, agent a notices that a has at least k edges and thus a updates loc_a and reg_a to yes (lines 13–18). Similarly, when agent a having the L_k token interacts with an agent with ϕ, agent a notices that a has at least $k+1$ edges and thus a updates reg_a to no (lines 19–22).

When agent a having the L_i token interacts with a marked agent b (i.e., agent b with ϕ'), agent a resets the L_i token to the L_0 token. Moreover, at the interaction, a deletes a mark of b and carries the L_0 token to b (i.e., $LF_b = \phi'$ transitions to L_0 and LF_a transitions to ϕ). These behaviors appear in lines 10–12. By these behaviors, the leader token can move freely on a graph because an agent having the leader token can mark any adjacent agent by an interaction. Note that, after the leader token moves, some agents may remain as marked agents. However, even in the case, eventually agents correctly detect an agent whose degree is at least k or $k+1$ because an agent with the leader token can delete marks of adjacent agents freely. Concretely, an agent a having the leader token deletes a mark of the adjacent agent b by making interaction between a and b three times. Figure 15 shows the example of the three interactions. By the interactions, the agents carry the leader token from a to b and then agent b returns the leader token to a. As a result, agent a has the leader token again and the marked agent b transitions to a non-marked agent.

By the above behaviors, eventually each agent with degree k makes its loc transition to yes. If agents find some agent a with $loc_a = no$, agents make reg of the leader token transition to no (lines 23–25). Thus, if there is an agent whose

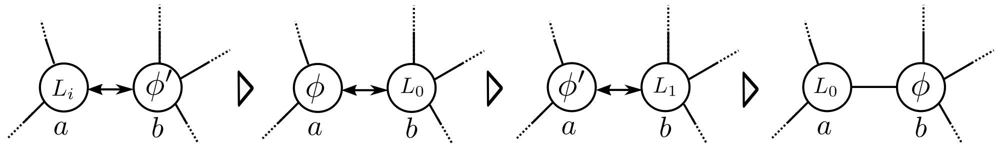


Figure 15. The example of deleting a mark

degree is not k, *reg* of the leader token converges to *no*. On the other hand, if the degree of each agent is k, eventually each agent makes its *loc* transition to *yes*, and there is no behavior that makes *reg* of the leader token transition to *no* afterwards. Hence, since *loc* and *reg* of the leader token transition to *yes* simultaneously by lines 15–18, *reg* of the leader token converges to *yes* in the case. When agents move the leader token, agents convey *reg* of the leader token (line 12). Therefore, eventually each agent makes a correct decision.

Correctness

First of all, we define some notations for the level. Let $lev(a, C)$ be level of agent a in a configuration C. For a set of all agents $V = \{v_1, v_2, \ldots, v_n\}$ and a configuration C, let $lev\_max(C) = \max_{a \in V}\{lev(a, C)\}$.

To begin with, we show that, in any execution of Algorithm 2, the behavior of line 4 is not performed by agent a such that $level_a = \lfloor \log n \rfloor$ holds. This implies that the domain of variable *level* is valid. Note that each agent a increases its $level_a$ one by one.

Lemma **51.** *In any execution, agent a does not increase $level_a$ if $level_a = \lfloor \log n \rfloor$ holds.*

Proof. First of all, from the pseudocode, there is no behavior that decreases the number of leader tokens, and there is no behavior that decreases the level of an agent.

From now, we show that, in any execution, at most one leader token at level $\lfloor \log n \rfloor$ can occur. From the pseudocode, the number of leader tokens at some level increases only if the behavior of lines 3–7 occurs. Hence, by the behavior of lines 3–7, if the number of leader tokens at level i increases by one, agents delete two leader tokens at level $i - 1$. Thus, since an initial level of each agent is 0 and the initial number of leader tokens is n, at most one leader token at level $\lfloor \log n \rfloor$ can occur.

From the pseudocode, the behavior of line 4 occurs only if two agents with leader tokens at the same level interact. Hence, if $level_a = \lfloor \log n \rfloor$ holds, agent a cannot increase $level_a$ and thus the lemma holds. $\square$

From now, we show that Algorithm 2 solves the problem. First, we prove that, in any configuration C of any execution, there exists an agent with the leader token at level $lev\_max(C)$.

Lemma **52.** *Let us consider a graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$. For any configuration C of any execution Ξ with G, there exists an agent v_m with the leader token such that $lev(v_m, C) = lev\_max(C)$ holds.*

Proof. For any graph $G = (V, E)$, let us consider an execution $\Xi = C_0, C_1, C_2, \ldots$ of the protocol, where $V = \{v_1, v_2, \ldots, v_n\}$. We prove the lemma by induction on the index of a configuration. In the base case (C_0), clearly $level_{v_1} = level_{v_2} = \cdots = level_{v_n} = lev\_max(C_0) = 0$ holds. For the induction step, we assume that, there exists an agent v_m with the leader token such that $lev(v_m, C_k) = lev\_max(C_k)$. Let us consider an interaction at $C_k \to C_{k+1}$ for four cases.

- Case where the behavior of lines 3–7, 10–12, or 26–30 does not occur at $C_k \to C_{k+1}$: From the pseudocode, by the interaction, agents do not update $level_{v_i}$ for $1 \le i \le n$. In addition, by the interaction, agents do not move the leader token, and the number of leader tokens is not changed. Hence, in this case, agent v_m with the leader token satisfies that $lev(v_m, C_{k+1}) = lev\_max(C_{k+1})$.

- Case where the behavior of lines 3–7 occurs at $C_k \to C_{k+1}$: From the pseudocode, if an agent increases its level by the interaction, the agent has the leader token after the interaction. Hence, if $lev\_max(C_{k+1}) = lev\_max(C_k)$ holds, agent v_m does not join the interaction and thus $lev(v_m, C_{k+1}) = lev\_max(C_{k+1})$ holds and agent v_m has the leader token in C_{k+1}. On the other hand, if $lev\_max(C_{k+1}) > lev\_max(C_k)$ holds, one of the interacting agents v'_m with the leader token satisfies $lev(v_{m'}, C_{k+1}) = lev\_max(C_{k+1})$.

- Case where the behavior of lines 10–12 occurs at $C_k \to C_{k+1}$: From the pseudocode, by the interaction, agents move the leader token from an interacting agent to the other interacting agent. The interacting agents have the same level before the interaction. Moreover, by the interaction, agents do not change their level and thus $lev\_max(C_{k+1}) = lev\_max(C_k)$ holds. Hence, if agent v_m does not join the interaction, $lev(v_m, C_{k+1}) = lev\_max(C_{k+1})$ holds and agent v_m has the leader token in C_{k+1}. If agent v_m joins the interaction, other interacting agent $v_{m'}$ with the leader token satisfies $lev(v_{m'}, C_{k+1}) = lev\_max(C_{k+1})$.
- Case where the behavior of lines 26–30 occurs at $C_k \to C_{k+1}$: From the pseudocode, by the interaction, an interacting agent a at the larger level do not change its variables, and the other interacting agent b at the smaller level becomes the same level as the level of agent a. Hence, in this case, agent v_m with the leader token satisfies that $lev(v_m, C_{k+1}) = lev\_max(C_{k+1})$ (whether v_m joins the interaction or not).

In each case, there exists an agent v_m with the leader token such that $lev(v_m, C_{k+1}) = lev\_max(C_{k+1})$. Therefore, the lemma holds. □

Next, we show that, in any execution, the level of each agent converges to the same value.

Lemma **53.** *For any execution Ξ with some graph G, there exist a configuration C and lev_last ($0 \le lev\_last \le \lfloor \log n \rfloor$) such that $level_a = lev\_last$ holds for each agent a after C.*

Proof. Let us consider a graph $G = (V, E)$ and an execution $\Xi = C_0, C_1, \ldots$ with G, where $V = \{v_1, v_2, \ldots, v_n\}$. From the pseudocode, since there is no behavior that increases the number of leader tokens, the number of leader tokens does not change after some configuration C_i. This implies that, after C_i, the behavior of lines 3–7 does not occur. Thus, no agent can obtain the level larger than $lev\_max(C_i)$. Let us consider two agents v_x and v_y that interact at $C_j \to C_{j+1}$ for $j \ge i$. If $lev(v_x, C_j) > lev(v_y, C_j)$ holds, $lev(v_y, C_{j+1}) = lev(v_x, C_{j+1}) = lev(v_x, C_j)$ holds by the behavior of lines 26–30. Hence, since there is no behavior that decreases level of an agent, $lev(v_1, C_k) = lev(v_2, C_k) = \cdots = lev(v_n, C_k) =$

$lev\_max(C_i)$ holds for a configuration C_k $(k \geq i)$ from global fairness. Since each agent maintains $lev\_max(C_i)$ after C_k, the lemma holds. □

Let us consider some execution Ξ^* with some graph G. Let lev_last be level such that $level_a = lev\_last$ holds for each agent a after some configuration of Ξ^*. From Lemma 53, such lev_last exists. From now, we show some properties about Ξ^* and lev_last. First, we prove that, in Ξ^*, after some of agents obtains level lev_last, there is only one leader token that is at level lev_last.

Lemma **54.** *In Ξ^*, after some of agents obtains level lev_last, there is only one leader token that is at level lev_last.*

Proof. Let C be a configuration in Ξ^* such that the first agent at level lev_last appears (i.e., there is an agent at level lev_last in C and there is no agent that is at level lev_last before C). We show that there is only one leader token that is at level lev_last after C. First of all, from Lemma 52, after C, there is at least one leader token that is at level lev_last. Thus, for the purpose of contradiction, we assume that there exists a configuration C' of Ξ^* such that there are two or more leader tokens that are at level lev_last in C' and C' occurs after C. Let ω_1 and ω_2 be the leader tokens in C'. Let us consider a configuration C'' such that $level_a = lev\_last$ holds for each agent a after C'' and C'' occurs after C'. If agents delete ω_1 or ω_2, the behavior of lines 3–7 or 26–30 occurs. However, both behaviors must not occur from the definition of lev_last. Hence, there also exist ω_1 and ω_2 in C''.

After C'', when an agent having a leader token interacts with an agent having no leader token, agents move the leader token, or, by making an additional interaction between them, agents move the leader token. This implies that ω_1 and ω_2 can move to any agent after C''. Hence, from global fairness, eventually an agent having ω_1 interacts with an agent having ω_2 and then they update their levels to $lev\_last + 1$ by the behavior of lines 3–7. This contradicts the definition of lev_last. □

In addition, since agents at level lev_last do not delete the leader token, Lemma 54 can be extended as follow.

Lemma **55.** *In Ξ^*, only one leader token can be at level lev_last.*

Next, we show properties about an agent at level *lev_last* whose degree is less than $k+1$.

Lemma **56.** *In Ξ^*, an agent a at level lev_last does not perform the behavior of lines 15–18 if the degree of agent a is less than k. Moreover, in Ξ^*, an agent b at level lev_last does not perform the behavior of lines 19–22 if the degree of agent b is less than $k+1$.*

Proof. In Ξ^*, when an agent c with the leader token updates its level from $lev\_last - 1$ to *lev_last* by the behavior of lines 3–7, LF_c transitions to L_0. Note that, from Lemma 55, only agent c with the leader token at level *lev_last* updates its level from $lev\_last - 1$ to *lev_last* by the behavior of lines 3–7 in Ξ^*. Hence, when other agents update its level to *lev_last*, the agents reset their LF to ϕ.

From now, we consider interactions between agents at level *lev_last*. To execute lines 15–18, it needs to occur k times that an agent having the leader token interacts with a non-marked agent without moving the leader token. This is because the leader token transitions to L_0 when it moves. When an agent having the leader token interacts with a non-marked agent, the agent with the leader token marks the non-marked agent. From the pseudocode, unless the leader token moves, a marked agent at level *lev_last* never transitions to a non-marked agent. From these facts, since there is only one leader token, agents never execute lines 15–18 if there is no agent whose degree is at least k. Similarly, agents never execute lines 19–22 if there is no agent whose degree is at least $k+1$. Therefore, the lemma holds. □

Now, we show properties about an agent whose degree is at least k. We prove that, if there is an agent whose degree is at least k (resp., $k+1$), the agent performs the behavior of lines 13–18 (resp., lines 19–22) infinitely often.

Lemma **57.** *In Ξ^*, an agent a performs the behavior of lines 13–18 infinitely often if the degree of agent a is at least k. Moreover, in Ξ^*, an agent b performs the behavior of lines 19–22 infinitely often if the degree of agent b is at least $k+1$.*

Proof. From Lemmas 53 and 55, there exists a configuration C in Ξ^* such that there is only one leader token and each agent is at level *lev_last* after C. We consider configurations after C. From the pseudocode, the leader token can move to any agent after C. Thus, from global fairness, there exists a configuration C' such that C' occurs infinitely often and agent a whose degree is at least k has the leader token in C'.

In C', since there exists only one leader token, each agent adjacent to a has ϕ or ϕ'. To make all ϕ' adjacent to a transition to ϕ, we consider the following procedure.

1. Agent a having the leader token interacts with an agent c having ϕ' three times. From the pseudocode, after the interactions, agent a has the L_0 token and agent c has ϕ (and other agents are the same states as before the interactions).

2. Agents repeat the above behavior until no agent adjacent to a has ϕ'.

From global fairness, eventually agents perform the above procedure. Hence, there exists a configuration such that, for agent a with the leader token, each agent adjacent to a has ϕ and the configuration occurs infinitely often. Moreover, from the configuration, eventually agent a interacts with each adjacent agent in a row. From the pseudocode, when such interactions occur, agents execute lines 13–18. Thus, if the degree of agent a is at least k, agent a performs the behavior of lines 13–18 infinitely often. Similarly, if the degree of agent b is at least $k+1$, agent b performs the behavior of lines 19–22 infinitely often. □

From Lemmas 53 and 55, eventually agents complete the election of leader tokens in Ξ^*. From now on, we define *reg* of the leader token as reg_a such that agent a has the elected leader token. We show that *reg* of the leader token transitions to *yes* only a finite number of times in Ξ^*.

Lemma **58.** *In Ξ^*, reg of the leader token transitions to yes only a finite number of times.*

Proof. From the pseudocode, *reg* of the leader token transitions to *yes* only by the behavior of lines 15–18. Hence, we prove that this behavior occurs only a finite number of times. From Lemmas 53 and 55, there exists a configuration C in Ξ^* such that there is only one leader token and each agent is at level lev_last.

From the pseudocode, loc_a of an agent a transitions from *yes* to *no* only by the behaviors of lines 3–7 and 26–30. The behavior of lines 3–7 occurs only if there are multiple leader tokens, and the behavior of lines 26–30 occurs only if there are agents at different levels. From these facts, loc_a of each agent a transitions from *yes* to *no* a finite number of times in Ξ^*. Hence, from the condition of line 15, the behavior of lines 15–18 occurs a finite number of times in Ξ^*. Therefore, the lemma holds. □

By using the above lemmas, we show that the protocol solves the k-regular identification problem. First, we show that, if the given graph is not k-regular, reg_a of each agent a converges to *no*.

Lemma **59.** *If the given communication graph is not k-regular, reg_a of each agent a converges to no in any globally-fair execution.*

Proof. Let Ξ be an execution with a non k-regular graph. Let lev_last be level such that the level of each agent converges to level lev_last in Ξ. From Lemma 53, such lev_last exists.

First, we show that, if there is an agent a whose degree is less than k on the graph, *reg* of the leader token converges to *no* in Ξ. From Lemma 56, if there is an agent a whose degree is less than k on the graph, agent a never updates its loc_a from *no* to *yes* after agent a obtains level lev_last. When agent a obtains level lev_last, agent a updates its loc_a to *no* by the behavior of lines 3–7 or 26–30. Hence, since loc_a converges to *no*, *reg* of the leader token transitions to *no* infinitely often by the behavior of lines 23–25. From Lemma 58, *reg* of the leader token transitions from *no* to *yes* only a finite number of times in Ξ. Thus, if there is an agent whose degree is less than k on the graph, *reg* of the leader token converges to *no* in Ξ.

Next, we show that, if there is an agent whose degree is at least $k+1$ on the graph, *reg* of the leader token converges to *no* in Ξ. From Lemma 57, if there is an agent whose degree is at least $k+1$ on the graph, *reg* of the leader token transitions to *no* infinitely often in Ξ. Thus, if there is an agent whose degree is at least $k+1$ on the graph, *reg* of the leader token converges to *no* in Ξ.

Now, we show that reg_a of each agent a converges to *no* in Ξ. Let C be a configuration in Ξ such that there is only one leader token and each agent is at level *lev_last* after C. From Lemmas 53 and 55, such C exists in Ξ. From the pseudocode, after C, the leader token moves freely on the graph and the decision of the leader token is conveyed to all agents. Thus, since *reg* of the leader token converges to *no* in Ξ, reg_a of each agent a converges to *no* in Ξ from global fairness. □

Next, we show that, if the given graph is k-regular, reg_a of each agent a converges to *yes*.

Lemma **60.** *If the given communication graph is k-regular, reg_a of each agent a converges to yes in any globally-fair execution.*

Proof. Let Ξ be an execution with a k-regular graph. In Ξ, let us consider a configuration C such that 1) there is only one leader token, 2) each agent has some level *lev_last* after C, and 3) loc_a of some agent a is *no* in C. From the pseudocode, when an agent b increases its level, agent b updates its loc_b to *no*. Thus, from Lemmas 53 and 55, such C exists in Ξ.

From Lemma 57, each agent performs the behavior of lines 13–18 infinitely often after C. Hence, eventually each agent a, such that $loc_a = no$ in C, makes loc_a transition to *yes* by the behavior of lines 15–18 after C. Since there is only one leader token and each agent is at level *lev_last* after C, the behaviors of lines 3–7 and 26–30 do not occur after C. This implies that, after C, loc_a of each agent a keeps *yes* if $loc_a = yes$.

Let us consider the last agent that performs the behavior of lines 15–18. From Lemma 58, since *reg* of the leader token transitions to *yes* by the behavior of lines 15–18, the behavior of lines 15–18 occurs a finite number of times and thus such an agent exists. From the pseudocode, when the agent performs the behavior, *reg* of the leader token transitions to *yes*. After that, since loc_a of each agent a is *yes*, the behavior of lines 23–25 does not occur. Furthermore, from Lemma 56, the behavior of lines 19–22 is not performed. Thus, *reg* of the leader token does not transition to *no* afterwards and thus reg_a of each agent a converges to *yes*. □

From Lemma 59, if the given communication graph is not k-regular, reg_a of each agent a converges to *no* in any execution of the protocol. From Lemma 60, if the given communication graph is k-regular, reg_a of each agent a converges to *yes* in any execution of the protocol. Thus, we can obtain the following theorem.

Theorem **27.** *Algorithm 2 solves the k-regular identification problem. That is, 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.*

Clearly, when the number of agents n is given, the protocol works even if the protocol uses variable $level_a = \{0, 1, 2, \ldots, \lfloor \log n \rfloor\}$ instead of $level_a = \{0, 1, 2, \ldots, \lfloor \log P \rfloor\}$. Therefore, we have the following theorem.

Theorem **28.** *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 Protocol with Knowledge of n under Weak Fairness

In this subsection, we give a star identification protocol (hereinafter referred to as "SI protocol") with $O(n)$ states and designated initial states under weak fairness. In this protocol, the number of agents n is given. Recall that, in this protocol under weak fairness, if a transition $(p, q) \to (p', q')$ exists for $p \neq q$, a transition $(q, p) \to (q', p')$ also exists. 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 explain the details, first we introduce variables at an agent a.

- $LF_a \in \{F, F', l', L_2, L_3, \ldots, L_{n-1}\}$: Variable LF_a, initialized to F, 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. $LF_a = F$ means that agent a is a non-marked agent. $LF_a = F'$ 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 $star_b$.

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.

Now, we show the details of the protocol.

Examination of the subgraph (lines 5–14 and 20–22 of the pseudocode) First, agents elect a central agent. Initially, LF_a of each agent a is F. When two F-agents interact, both F-agents transition to l'-agents (lines 5–6). When an l'-agent interacts with an F-agent, the l'-agent transitions to an L_2-agent (lines 7–8). By these behaviors, we can observe that the L_2-agent is adjacent to at least two agents. Clearly, if the given graph is a star, agents correctly elect the central agent. If the given graph is not a star, agents may elect no central agent or multiple central agents. However, in both cases, $star$ of each agent keeps no (we explain the details later).

Then, to confirm that the central agent is adjacent to $n-1$ agents, the central agent marks adjacent agents one by one and counts how many times the agent interacts with a non-marked agent. Concretely, for $2 \le i \le n-3$, when the L_i-agent interacts with an F-agent, the L_i-agent marks the F-agent (i.e., the F-agent transitions to an F'-agent), and the L_i-agent transitions to the L_{i+1}-agent (lines

Algorithm 3 A SI protocol

```
A variable at an agent $a$:
    $LF_a \in \{F, F', l', L_2, L_3, \ldots, L_{n-1}\}$: States that represent roles of agents, initialized to $F$. $L_i$ represents a central agent, $l'$ represents a candidate of the central agent, $F'$ represents a marked agent, and $F$ 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 $LF_a = F \land LF_b = F$ then
 6:             $LF_a \leftarrow l'$, $LF_b \leftarrow l'$
 7:         else if $LF_a = l' \land LF_b = F$ then
 8:             $LF_a \leftarrow L_2$, $LF_b \leftarrow F'$
    { Counting the number of adjacent agents by the central agent }
 9:         else if $LF_a = L_i \land LF_b = F$ $(2 \leq i \leq n-2)$ then
10:             $LF_a \leftarrow L_{i+1}$, $LF_b \leftarrow F'$
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 $LF_a = F' \land LF_b = F'$ then
16:             $star_a \leftarrow never$, $star_b \leftarrow never$
17:         else if $LF_a = F' \land LF_b = l'$ 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
```

9–11). Note that, when an agent becomes the central agent, the agent has already interacted two different agents. Thus, in this protocol, the agent marked the two agents by the interactions, and the agent starts as an L_2-agent (lines 5–8). Recall that an l'-agent is a marked agent.

When an agent becomes an L_{n-1}-agent, the L_{n-1}-agent notices that the agent is adjacent to $n-1$ agents. Thus, agents notice that there is a star subgraph in the given graph such that the subgraph consists of n agents. Hence, when an agent becomes an L_{n-1}-agent, the interacting agents make their *star* transition to *yes* (lines 12–14). When an agent a with $star_a = yes$ and an agent b with $star_b = no$ interact, $star_b$ transitions to *yes* (lines 20–22). By this behavior, eventually *yes* is conveyed to all agents.

When agents elect multiple central agents (resp., no central agent), *star* of each agent cannot transition to *yes* because the central agents cannot mark $n-1$ agents (resp., there is no agent that marks agents). Thus, since initially *star* of each agent is *no*, agents make their *star* keep *no*. In both cases, since clearly the given graph is not a star, agents make a correct decision.

Decision of *never* (lines 2–3 and 15–19 of the pseudocode) When two F'-agents interact, they make their *star* transition to *never* (lines 15–16). When an F'-agent interacts with an l'-agent, they also make their *star* transition to *never* (lines 17–19).

From now, we show that 1) agents do not perform the above behaviors if the given graph is a star, and 2) agents perform the above behaviors or agents make their *star* keep *no* if the given graph is not a star.

If the given graph is a star, agents correctly elect the central agent. Thus, since agents other than the central agent can interact only with the L_i-agent, agents do not perform the above behaviors.

Next, let us consider the case where the given graph is not a star. In this case, if agents do not confirm the star subgraph that consists of n agents, agents do not make their *star* transition to *yes*. Thus, agents make a correct decision because, if $star_a = no$ or $star_a = never$ holds, $\gamma(s_a) = no$ holds. On the other hand, if agents confirm the star subgraph that consists of n agents, there is an agent that is not elected as the central agent and is adjacent to two or more agents.

Moreover, after agents confirm the subgraph, the agent is an F'- or l'-agent and is adjacent to another F'- or l'-agent. Hence, eventually two F'-agents interact, or the F'-agent interacts with the l'-agent. By the interaction, the interacting agents make their *star* transition to *never*. When an agent a with $star_a = never$ and an agent b interact, $star_b$ transitions to *never* (lines 2–3). By this behavior, eventually *never* is conveyed to all agents. Note that, when this behavior occurs, the behavior of lines 20–22 does not occur. Thus, *never* has priority over *yes*.

From these facts, we can observe that agents make a correct decision in any case.

Correctness

First, we define some notations.

Definition **21.** *The number of F-agents is denoted by $\#F$. The number of F'-agents is denoted by $\#F'$. The number of L_i-agents for $2 \leq i \leq n-1$ is denoted by $\#L_i$. Let $\#L = \#L_2 + \#L_3 + \cdots + \#L_{n-1}$.*

Definition **22.** *A function $L(a, C)$ represents the number of agents counted by an agent a in configuration C. Concretely, if $LF_a = L_i$ holds in C, $L(a, C) = i$ holds. If $LF_a \in \{F,\ F',\ l'\}$ holds in C, $L(a, C) = 0$ holds.*

From now, we show an equation that holds in any execution of Algorithm 3.

Lemma **61.** *Let us consider an execution Ξ with some graph $G = (V, E)$, where $V = \{v_1,\ v_2,\ \ldots,\ v_n\}$. In any configuration C of Ξ, $L(v_1, C) + L(v_2, C) + \cdots + L(v_n, C) = \#F' + \#L$ holds.*

Proof. Let us consider an execution Ξ with a graph $G = (V, E)$, where $V = \{v_1,\ v_2,\ \ldots,\ v_n\}$. We prove the lemma by induction on the index of a configuration. In the base case (C_0), clearly $L(v_1, C_0) + L(v_2, C_0) + \cdots + L(v_n, C_0) = \#F' + \#L = 0$ holds. For the induction step, we assume that the equation holds in C_k. Let us consider an interaction at $C_k \rightarrow C_{k+1}$ for three cases.

- Case where the behavior of lines 7–8 or 9–11 does not occur at $C_k \rightarrow C_{k+1}$: In this case, clearly $L(v_i, C_{k+1}) = L(v_i, C_k)$ holds for $1 \leq i \leq n$. Moreover, by the interaction, $\#F'$ and $\#L$ do not change. Thus, $L(v_1, C_{k+1}) + L(v_2, C_{k+1}) + \cdots + L(v_n, C_{k+1}) = \#F' + \#L$ holds in this case.

- Case where the behavior of lines 7–8 occurs at $C_k \rightarrow C_{k+1}$: From the pseudocode, an l'-agent (resp., an F-agent) transitions to an L_2-agent (resp., an F'-agent) by the interaction. Hence, by the interaction, $L(v_1, C_{k+1}) + L(v_2, C_{k+1}) + \cdots + L(v_n, C_{k+1}) = L(v_1, C_k) + L(v_2, C_k) + \cdots + L(v_n, C_k) + 2$ holds. In addition, $\#F'$ increases by one and $\#L$ increases by one. Thus, $L(v_1, C_{k+1}) + L(v_2, C_{k+1}) + \cdots + L(v_n, C_{k+1}) = \#F' + \#L$ holds in this case.
- Case where the behavior of lines 9–11 occurs at $C_k \rightarrow C_{k+1}$: From the pseudocode, an L_i-agent (resp., an F-agent) transitions to an L_{i+1}-agent (resp., an F'-agent) by the interaction. Hence, by the interaction, $L(v_1, C_{k+1}) + L(v_2, C_{k+1}) + \cdots + L(v_n, C_{k+1}) = L(v_1, C_k) + L(v_2, C_k) + \cdots + L(v_n, C_k) + 1$ holds, and $\#F'$ increases by one. Thus, $L(v_1, C_{k+1}) + L(v_2, C_{k+1}) + \cdots + L(v_n, C_{k+1}) = \#F' + \#L$ holds in this case.

In each case, the equation holds after the interaction. Therefore, the lemma holds. ☐

Next, by using Lemma 61, we show that, if the given communication graph is a star, agents make a correct decision.

Lemma **62.** *If the given communication graph is a star, $star_a$ of each agent a converges to yes for any weakly-fair execution Ξ.*

Proof. First of all, by the property of the star graph, an agent a with degree $n - 1$ joins any interaction. In the first interaction, agent a and another agent b interact. From the pseudocode, agents a and b transition to l'-agents. After that, since agents other than a and b are F-agents and agent a with degree $n - 1$ is the l'-agent, agents can transition to other states only by the behavior of lines 7–8. From weak fairness, eventually the behavior of lines 7–8 occurs, and agent a transitions to an L_2-agent.

After that, agent a does not become an F-, F'-, or l'-agent from the pseudocode. Thus, since agent a always joins interactions, agent b is always the l'-agent, and any agent other than a and b is F- or F'-agent. Hence, from Lemma 61, if agent a is the L_i-agent, $\#F'$ is $i - 1$ and $\#F$ is $n - 2 - (i - 1) = n - i - 1$. Then, from the pseudocode, agent a and F-agents can update their LF only by the behavior of line 9–11. Thus, since agent a is adjacent to each agent, the behavior of line 9–11 occurs repeatedly from weak fairness until agent a transitions to an L_{n-1}-agent. Note that, since $\#F$ is $n - i - 1$ when $LF_a = L_i$ holds, eventually agent a transitions to the L_{n-1}-agent. When agent a transitions to the L_{n-1}-agent, $star_a$ transitions to *yes* from the pseudocode. Then, agent a is always the L_{n-1}-agent. Hence, since agents other than agent a are adjacent only to agent a, *star* of any agent does not transition to *never* because the behaviors of lines 15–19 do not occur. Thus, from the pseudocode, *star* of each agent transitions to *yes* by the behavior of lines 20–22, and *star* of each agent does not update afterwards. Therefore, the lemma holds. □

From now, we show that, even if the given communication graph is not a star, agents make a correct decision. To show this, we first show the property of a configuration such that there is an L_{n-1}-agent in the configuration.

Lemma **63.** *If there is an L_{n-1}-agent in a configuration, there is an l'-agent and other $n - 2$ agents are F'-agents in the configuration. Furthermore, after the configuration, each agent a does not update its LF_a.*

Proof. Let C be a configuration in which there is an agent a that is an L_{n-1}-agent.

First of all, clearly $\#F' + \#L > n$ does not hold in any configuration. Hence, from Lemma 61, any agent other than a is not an L_i-agent for $2 \le i \le n-1$ in C (otherwise $\#F' + \#L = n - 1 + i > n$ holds). Thus, since $\#F' + \#L = n - 1$ holds in C, $n - 2$ agents are F'-agents and one agent b is an F- or l'-agent.

Now, we show that agent b is an l'-agent in C. From the pseudocode, if an agent is an F'-agent in C, the agent is not an l'-agent before and after C. Since an l'-agent occurs only by the behavior of lines 5–6 and an L_{n-1}-agent exists in C, the behavior of lines 5–6 occurs before C. From these facts, agents a and b performed the behavior of lines 5–6 before C. From the pseudocode, if an agent is an l'-agent, the agent is not an F-agent afterwards. Hence, agent b is an l'-agent in C and thus agent a is the L_{n-1}-agent, agent b is the l'-agent, and other $n - 2$ agents are F'-agents in C.

From the pseudocode, clearly each agent a does not update its LF_a after C. Therefore, the lemma holds. □

Now, by using Lemma 63, we show that agents make a correct decision even if the given communication graph is not a star.

Lemma **64.** *If the given communication graph is not a star, $star_a$ of each agent a converges to no or never for any weakly-fair execution Ξ.*

Proof. Let Ξ be a weakly-fair execution with a non-star graph. Let us consider two cases: (1) $star_a$ of some agent a transitions to *yes* in Ξ, and (2) *star* of any agent does not transition to *yes* in Ξ.

In case (1), since the behavior of lines 12–14 occurs, an agent a becomes an L_{n-1}-agent. Moreover, from the pseudocode, agent a is always the L_{n-1}-agent afterwards. From now on, let us consider configurations after agent a becomes the L_{n-1}-agent. Since the given graph is not a star, there are two agents whose degree is two or more. Hence, there is an agent b that is adjacent to an agent other than a. From weak fairness, eventually agent b interacts with the agent other than a. By the interaction, from Lemma 63, the behavior of lines 15–16 or 17–19 occurs and $star_b$ transitions to *never*. After that, by the behavior of lines 2–3, *star* of each agent transitions to *never*. Furthermore, after agents update their *star* to *never*, agents do not update their *star*. Thus, *star* of each agent converges to *never* in this case.

In case (2), if *star* of an agent transitions to *never*, *star* of each agent converges to *never* similarly to case (1); otherwise, since initially *star* of each agent is *no*, *star* of each agent converges to *no*. Therefore, the lemma holds. □

From Lemma 62, if the given communication graph is a star, $star_a$ of each agent a converges to *yes* in any execution of the protocol. From Lemma 64, if the given communication graph is not a star, $star_a$ of each agent a converges to *no* or *never* in any execution of the protocol. Thus, we can obtain the following theorem.

Theorem **29.** *Algorithm 3 solves the star identification problem. That is, 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.

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).

Lemma **65.** *Let us consider a communication graph $G = (V, E)$, where $V = \{v_1, v_2, v_3, \dots, v_n\}$. Let $G' = (V', E')$ be a communication graph that satisfies the following, where $V' = \{v'_1, v'_2, v'_3, \dots, v'_{2n}\}$.*

- *$E' = \{(v'_x, v'_y), (v'_{x+n}, v'_{y+n}) \in V' \times V' \mid (v_x, v_y) \in E\} \cup \{(v'_1, v'_{z+n}), (v'_{1+n}, v'_z) \in V' \times V' \mid (v_1, v_z) \in E\}$ (Figure 16 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. First, we define a term that represents a relation between configurations. Let C_i and C'_j be configurations with G and G', respectively. If $s(v_x, C_i) = s(v'_x, C'_j) = s(v'_{x+n}, C'_j)$ holds for $1 \le x \le n$, we say that C_i and C'_j are equivalent.

Let Ξ be a weakly-fair execution of $\mathcal{P}$ with G such that there exists a configuration C_t of Ξ after which $\forall v \in V : \gamma(s(v)) = yn \in \{yes,\ no\}$ holds. Without loss of generality, we assume $yn = yes$.

Since Ξ is weakly fair, Ξ can be represented by the following $\Xi = C_0, C_1, C_2, \dots, C_t, \dots, C_{t'_0}, \xi_1, C_{t'_1}, \xi_2, C_{t'_2}, \xi_3, \dots$.

- For $u \ge 0$, $C_{t'_u}$ is a stable configuration such that $C_{t'_0} = C_{t'_1} = C_{t'_2} = \cdots$ holds and $C_{t'_u}$ occurs infinitely often.

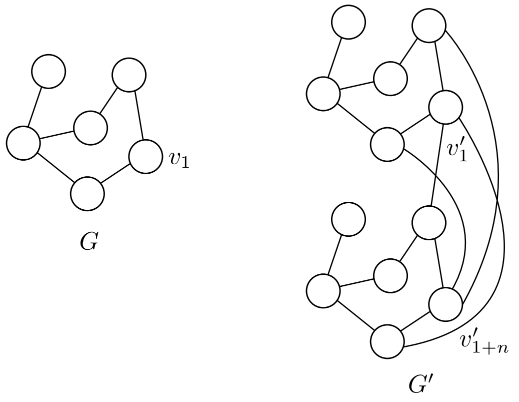
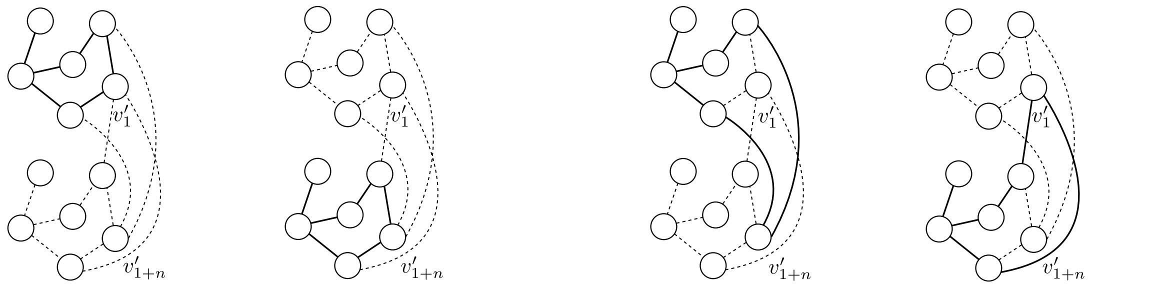


Figure 16. An example of graphs G and G'

- For $j \geq 1$, ξ_j is a sub-execution such that, in $C_{t'_{j-1}}, \xi_j, C_{t'_j}$, for each pair (a, b) in E, agent a interacts with agent b and agent b interacts with agent a, at least once.

Next, let us consider the following execution $\Xi' = C'_0$, C'_1, C'_2, $\ldots$, $C'_{2t'_0}$, ξ'_1, $C'_{2t'_1}$, ξ'_2, $C'_{2t'_2}$, ξ'_3, $\ldots$ of $\mathcal{P}$ with G'.

- For $0 \leq i < t'_0$, when v_x interacts with v_y at $C_i \rightarrow C_{i+1}$, v'_x interacts with v'_y at $C'_i \rightarrow C'_{i+1}$, and v'_{x+n} interacts with v'_{y+n} at $C'_{i+t'_0} \rightarrow C'_{i+t'_0+1}$. Clearly, v'_1, $\ldots$, v'_n and v'_{1+n}, $\ldots$, v'_{2n} behave similarly to v_1, $\ldots$, v_n in Ξ and thus $C_{t'_0}$ and $C'_{2t'_0}$ are equivalent.
- For $j \geq 1$, by using $\xi_j = C^j_1$, C^j_2, C^j_3, $\ldots$, C^j_m, we define $\xi'_j = \hat{C}^j_1$, $\hat{C}^j_2$, $\hat{C}^j_3$, $\ldots$, $\hat{C}^j_{2m+1}$ as follows (Figure 17 shows images of $C'_{2t'_{j-1}}$, ξ'_j, $C'_{2t'_j}$).
  - Case where j is an even number: v'_1, $\ldots$, v'_n and v'_{1+n}, $\ldots$, v'_{2n} behave similarly to v_1, $\ldots$, v_n in Ξ. Concretely, for $1 \leq i < m$, when v_x interacts with v_y at $C^j_i \rightarrow C^j_{i+1}$, v'_x interacts with v'_y at $\hat{C}^j_i \rightarrow \hat{C}^j_{i+1}$, and v'_{x+n} interacts with v'_{y+n} at $\hat{C}^j_{i+m+1} \rightarrow \hat{C}^j_{i+m+2}$. When v_x interacts with v_y at $C_{t'_{j-1}} \rightarrow C^j_1$ (resp., $C^j_m \rightarrow C_{t'_j}$), v'_x interacts with v'_y at $C'_{2t'_{j-1}} \rightarrow \hat{C}^j_1$ (resp., $\hat{C}^j_m \rightarrow \hat{C}^j_{m+1}$), and v'_{x+n} interacts with v'_{y+n} at $\hat{C}^j_{m+1} \rightarrow \hat{C}^j_{m+2}$ (resp., $\hat{C}^j_{2m+1} \rightarrow C'_{2t'_j}$).



(i) $C'_{2t'_{j-1}}$, $\hat{C}^j_1$, …, $\hat{C}^j_m$, $\hat{C}^j_{m+1}$ (ii) $\hat{C}^j_{m+1}$, $\hat{C}^j_{m+2}$, …, $\hat{C}^j_{2m+1}$, $C'_{2t'_j}$

(a) j is even

(i) $C'_{2t'_{j-1}}$, $\hat{C}^j_1$, …, $\hat{C}^j_m$, $\hat{C}^j_{m+1}$ (ii) $\hat{C}^j_{m+1}$, $\hat{C}^j_{m+2}$, …, $\hat{C}^j_{2m+1}$, $C'_{2t'_j}$

(b) j is odd

Figure 17. Images of $C'_{2t'_{j-1}}$, ξ'_j, $C'_{2t'_j}$. A solid line represents an edge on which interactions occur in $C'_{2t'_{j-1}}$, ξ'_j, $C'_{2t'_j}$, and a dashed line represents an edge on which interactions do not occur in $C'_{2t'_{j-1}}$, ξ'_j, $C'_{2t'_j}$

In this case, clearly v'_1, …, v'_n and v'_{1+n}, …, v'_{2n} make transitions similarly to v_1, …, v_n in Ξ. Hence, if $C_{t'_{j-1}}$ and $C'_{2t'_{j-1}}$ are equivalent, $C_{t'_j}$ and $C'_{2t'_j}$ are equivalent.

– Case where j is an odd number: $\{v'_{1+n}\} \cup \{v'_2, \ldots, v'_n\}$ and $\{v'_1\} \cup \{v'_{2+n}, \ldots, v'_{2n}\}$ behave similarly to v_1, …, v_n in Ξ. Concretely, for $1 \le i < m$ and $x, y \ne 1$, when v_x interacts with v_y at $C^j_i \to C^j_{i+1}$, v'_x interacts with v'_y at $\hat{C}^j_i \to \hat{C}^j_{i+1}$, and v'_{x+n} interacts with v'_{y+n} at $\hat{C}^j_{i+m+1} \to \hat{C}^j_{i+m+2}$. When v_x interacts with v_y at $C_{t'_{j-1}} \to C^j_1$ (resp., $C^j_m \to C_{t'_j}$), v'_x interacts with v'_y at $C'_{2t'_{j-1}} \to \hat{C}^j_1$ (resp., $\hat{C}^j_m \to \hat{C}^j_{m+1}$), and v'_{x+n} interacts with v'_{y+n} at $\hat{C}^j_{m+1} \to \hat{C}^j_{m+2}$ (resp., $\hat{C}^j_{2m+1} \to C'_{2t'_j}$).
For $1 \le i < m$, when v_1 interacts with v_x (resp., v_x interacts with v_1) at $C^j_i \to C^j_{i+1}$, v'_{1+n} interacts with v'_x (resp., v'_x interacts with v'_{1+n}) at $\hat{C}^j_i \to \hat{C}^j_{i+1}$, and v'_1 interacts with v'_{x+n} (resp., v'_{x+n} interacts with v'_1) at $\hat{C}^j_{i+m+1} \to \hat{C}^j_{i+m+2}$. When v_1 interacts with v_x (resp., v_x interacts with v_1) at $C_{t'_{j-1}} \to C^j_i$, v'_{1+n} interacts with v'_x (resp., v'_x interacts with v'_{1+n}) at $C'_{2t'_{j-1}} \to \hat{C}^j_i$, and v'_1 interacts with v'_{x+n} (resp., v'_{x+n} interacts with v'_1) at $\hat{C}^j_{m+1} \to \hat{C}^j_{m+2}$. When v_1 interacts with v_x (resp., v_x interacts with v_1) at $C^j_m \to C_{t'_j}$, v'_{1+n} interacts with v'_x (resp., v'_x interacts with

v'_{1+n}) at $\hat{C}^j_m \to \hat{C}^j_{m+1}$, and v'_1 interacts with v'_{x+n} (resp., v'_{x+n} interacts with v'_1) at $\hat{C}^j_{2m+1} \to C'_{2t'_j}$.
In this case, $\{v'_{1+n}\} \cup \{v'_2, \ldots, v'_n\}$ and $\{v'_1\} \cup \{v'_{2+n}, \ldots, v'_{2n}\}$ make transitions similarly to $v_1, \ldots, v_n$ in Ξ. Hence, if $C_{t'_{j-1}}$ and $C'_{2t'_{j-1}}$ are equivalent, $C_{t'_j}$ and $C'_{2t'_j}$ are equivalent because $s(v_1, C_{t'_{j-1}}) = s(v'_1, C'_{2t'_{j-1}}) = s(v'_{1+n}, C'_{2t'_{j-1}})$ holds.

Since Ξ is weakly fair, clearly each pair of agents in E' interacts infinitely often in Ξ' and thus Ξ' satisfies weak fairness. By behaviors of Ξ', since C'_{2t} and C_t are equivalent and $\forall v \in V : \gamma(s(v)) = yes$ holds after C_t, $\forall v' \in V' : \gamma(s(v')) = yes$ holds after C'_{2t}. From these facts, the lemma holds. □

4.2 Impossibility with the Known Upper Bound of the Number of Agents 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 65 to $\mathcal{P}_{P=x}$ because we can apply the same protocol $\mathcal{P}_{P=x}$ to both G and G' in Lemma 65. Clearly, we can construct G and G' in Lemma 65 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 **30.** *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 30, 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 the Known Number of Agents 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, we first define a particular execution Ξ with a line graph.

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 s_0 be an initial state of agents. Let us consider a transition sequence $T = (s_0, s_0) \rightarrow (s_{a_1}, s_{b_1}), (s_{b_1}, s_{a_1}) \rightarrow (s_{b_2}, s_{a_2}), (s_{a_2}, s_{b_2}) \rightarrow (s_{a_3}, s_{b_3}), (s_{b_3}, s_{a_3}) \rightarrow (s_{b_4}, s_{a_4}), \ldots$. 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.

We define an execution $\Xi = C_0, \xi_1, C_{u_1}, \xi_2, C_{u_2}, \xi_3, C_{u_3}, \xi_4, C_{u_4}, \xi_5, \ldots$ of a protocol $\mathcal{P}$ with G as follows, where ξ_m is a sub-execution ($m \geq 1$).

- In C_0, ξ_1, C_{u_1}, until $s(v_1) = sa$ and $s(v_2) = sb$ hold, agents repeat the following interactions: v_1 interacts with v_2, v_2 interacts with v_1, v_1 interacts with v_2 .... From the definition of the transition sequence T, this is possible.
- In C_{u_1}, ξ_2, C_{u_2}, until $s(v_3) = sa$ and $s(v_4) = sb$ hold, agents repeat the following interactions: v_3 interacts with v_4, v_4 interacts with v_3, v_3 interacts with v_4 .... Thus, $s(v_1, C_{u_2}) = s(v_3, C_{u_2}) = sa$ and $s(v_2, C_{u_2}) = s(v_4, C_{u_2}) = sb$ hold.
- For $i \geq 2$, we construct the execution as follows:
  - Case where $i \bmod 3 = 0$ holds: In C_{u_i}, ξ_{i+1}, $C_{u_{i+1}}$, until $s(v_1) = sa$ and $s(v_2) = sb$ hold, agents repeat the following interactions: v_1 interacts with v_2, v_2 interacts with v_1, v_1 interacts with v_2 .... To satisfy weak

fairness, we construct the interactions so that v_1 and v_2 interact at least twice.

  - Case where $i \bmod 3 = 1$ holds: In C_{u_i}, ξ_{i+1}, $C_{u_{i+1}}$, until $s(v_2) = sb$ and $s(v_3) = sa$ hold, agents repeat the following interactions: v_3 interacts with v_2, v_2 interacts with v_3, v_3 interacts with v_2 .... To satisfy weak fairness, we construct the interactions so that v_2 and v_3 interact at least twice.
  - Case where $i \bmod 3 = 2$ holds: In C_{u_i}, ξ_{i+1}, $C_{u_{i+1}}$, until $s(v_3) = sa$ and $s(v_4) = sb$ hold, agents repeat the following interactions: v_3 interacts with v_4, v_4 interacts with v_3, v_3 interacts with v_4 .... To satisfy weak fairness, we construct the interactions so that v_3 and v_4 interact at least twice.

For $i \geq 2$, if $s(v_1, C_{u_i}) = s(v_3, C_{u_i}) = sa$ and $s(v_2, C_{u_i}) = s(v_4, C_{u_i}) = sb$ hold, we can construct such interactions from the definition of the transition sequence T. Thus, since $s(v_1, C_{u_2}) = s(v_3, C_{u_2}) = sa$ and $s(v_2, C_{u_2}) = s(v_4, C_{u_2}) = sb$ hold, $C_{u_i} = C_{u_{i+1}}$ holds for $i \geq 2$.

Since each pair of agents interact infinitely often in Ξ, Ξ is weakly-fair. Since Ξ is weakly-fair, $\gamma(sa) = \gamma(sb) = yn \in \{yes, no\}$ holds in a stable configuration of Ξ.

Now, we show that there is a case where a line graph and a ring graph are not distinguishable.

Lemma **66.** *Let $G' = (V', E')$ be a ring graph with four agents, where $V' = \{v'_1, v'_2, v'_3, v'_4\}$ and $E' = \{(v'_1, v'_2), (v'_2, v'_3), (v'_3, v'_4), (v'_4, v'_1)\}$. 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'.*

Proof. Let us consider the following execution $\Xi' = C'_0$, ξ'_1, $C'_{u'_1}$, ξ'_2, $C'_{u'_2}$, ξ'_3, $C'_{u'_3}$, ξ'_4, $C'_{u'_4}$, ξ'_5, ... of $\mathcal{P}$ with G', where ξ'_m is a sub-execution ($m \geq 1$).

- In C'_0, ξ'_1, $C'_{u'_1}$, until $s(v'_1) = sa$ and $s(v'_2) = sb$ hold, agents repeat the following interactions: v'_1 interacts with v'_2, v'_2 interacts with v'_1, v'_1 interacts with v'_2 .... From the definition of the transition sequence T, this is possible.

- In $C'_{u'_1}$, ξ'_2, $C'_{u'_2}$, until $s(v'_3) = sa$ and $s(v'_4) = sb$ hold, agents repeat the following interactions: v'_3 interacts with v'_4, v'_4 interacts with v'_3, v'_3 interacts with v'_4 .... Thus, $s(v'_1, C'_{u'_2}) = s(v'_3, C'_{u'_2}) = sa$ and $s(v'_2, C'_{u'_2}) = s(v'_4, C'_{u'_2}) = sb$ hold.

- For $i \geq 2$, we construct the execution as follows:

  - Case where $i \bmod 4 = 0$ holds: In $C'_{u'_i}$, ξ'_{i+1}, $C'_{u'_{i+1}}$, until $s(v'_1) = sa$ and $s(v'_2) = sb$ hold, agents repeat the following interactions: v'_1 interacts with v'_2, v'_2 interacts with v'_1, v'_1 interacts with v'_2 .... To satisfy weak fairness, we construct the interactions so that v'_1 and v'_2 interact at least twice.

  - Case where $i \bmod 4 = 1$ holds: In $C'_{u'_i}$, ξ'_{i+1}, $C'_{u'_{i+1}}$, until $s(v'_2) = sb$ and $s(v'_3) = sa$ hold, agents repeat the following interactions: v'_3 interacts with v'_2, v'_2 interacts with v'_3, v'_3 interacts with v'_2 .... To satisfy weak fairness, we construct the interactions so that v'_2 and v'_3 interact at least twice.

  - Case where $i \bmod 4 = 2$ holds: In $C'_{u'_i}$, ξ'_{i+1}, $C'_{u'_{i+1}}$, until $s(v'_3) = sa$ and $s(v'_4) = sb$ hold, agents repeat the following interactions: v'_3 interacts with v'_4, v'_4 interacts with v'_3, v'_3 interacts with v'_4 .... To satisfy weak fairness, we construct the interactions so that v'_3 and v'_4 interact at least twice.

  - Case where $i \bmod 4 = 3$ holds: In $C'_{u'_i}$, ξ'_{i+1}, $C'_{u'_{i+1}}$, until $s(v'_4) = sb$ and $s(v'_1) = sa$ hold, agents repeat the following interactions: v'_1 interacts with v'_4, v'_4 interacts with v'_1, v'_1 interacts with v'_4 .... To satisfy weak fairness, we construct the interactions so that v'_1 and v'_4 interact at least twice.

  For $i \geq 2$, if $s(v'_1, C'_{u'_i}) = s(v'_3, C'_{u'_i}) = sa$ and $s(v'_2, C'_{u'_i}) = s(v'_4, C'_{u'_i}) = sb$ hold, we can construct such interactions from the definition of the transition sequence T. Thus, since $s(v'_1, C'_{u'_2}) = s(v'_3, C'_{u'_2}) = sa$ and $s(v'_2, C'_{u'_2}) = s(v'_4, C'_{u'_2}) = sb$ hold, $C'_{u'_i} = C'_{u'_{i+1}}$ holds for $i \geq 2$.

Since each pair of agents interact infinitely often in Ξ', Ξ' is weakly-fair. From these facts, since $\gamma(sa) = \gamma(sb) = yn$ holds, $\forall v' \in V' : \gamma(s(v')) = yn$ holds in a stable configuration in Ξ'. Thus, the lemma holds. $\square$

Note that, even if the number of agents is given, Lemma 66 holds because $|V| = |V'| = 4$ holds in the lemma. In Lemma 66, 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 66, there is no protocol that solves the *line*, *ring*, *tree*, or *k-regular* identification problem, and thus we have the following theorem.

Theorem **31.** *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.*

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 18 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 [30]. In [30], Fischer and Jiang proved the impossibility of leader election for a ring graph.

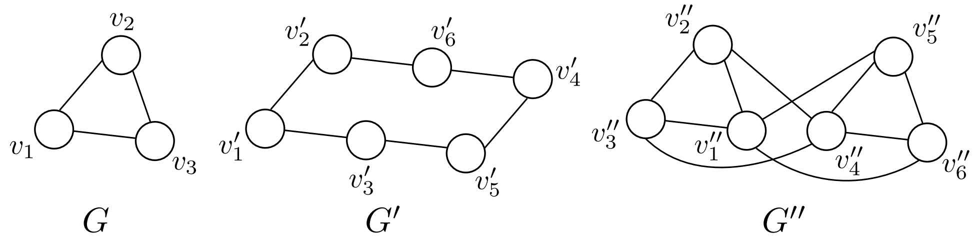


Figure 18. Graphs G, G', and G''

Lemma **67.** *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. Let $\Xi = C_0$, C_1, C_2 be a weakly-fair execution of $\mathcal{P}_{n=6}$ with G. Let us consider the following execution $\Xi' = D_0$, D'_0, D_1, D'_1 ... of $\mathcal{P}_{n=6}$ with G'.

- For x and y such that either $x = 1$ or $y = 1$ holds, when v_x interacts with v_y at $C_i \to C_{i+1}$, v'_x interacts with v'_y at $D_i \to D'_i$, and v'_{x+3} interacts with v'_{y+3} at $D'_i \to D_{i+1}$.
- For $x \in \{2, 3\}$ and $y \in \{2, 3\}$ such that $x \neq y$ holds, when v_x interacts with v_y at $C_i \to C_{i+1}$, v'_x interacts with v'_{y+3} at $D_i \to D'_i$, and v'_{x+3} interacts with v'_y at $D'_i \to D_{i+1}$.

For a configuration C of G and a configuration D of G', if $s(v_i, C) = s(v'_i, D) = s(v'_{i+3}, D)$ holds for i $(1 \leq i \leq 3)$, we say that C and D are *equivalent*.

From now, we show, by induction on the index of configuration, that C_r and D_r are equivalent for any $r \geq 0$. Since clearly C_0 and D_0 are equivalent, the base case holds. For the induction step, we assume that C_l and D_l are equivalent, and then consider two cases of interaction at $C_l \to C_{l+1}$.

First we consider the case where, for x and y such that either $x = 1$ or $y = 1$ holds, agents v_x interacts with v_y at $C_l \to C_{l+1}$. In this case, v'_x interacts with v'_y at $D_l \to D'_l$, and v'_{x+3} interacts with v'_{y+3} at $D'_l \to D_{l+1}$. By the induction assumption, $s(v_x, C_l) = s(v'_x, D_l) = s(v'_{x+3}, D_l)$ and $s(v_y, C_l) = s(v'_y, D_l) = s(v'_{y+3}, D_l)$ hold. Hence, agents v'_x and v'_{x+3} (resp., v'_y and v'_{y+3}) update their states similarly to v_x (resp., v_y), and thus C_{l+1} and D_{l+1} are equivalent in this case.

Next, we consider the case where, for $x \in \{2,3\}$ and $y \in \{2,3\}$ such that $x \neq y$ holds, agents v_x interacts with v_y at $C_l \to C_{l+1}$. In this case, v'_x interacts with v'_{y+3} at $D_l \to D'_l$, and v'_{x+3} interacts with v'_y at $D'_l \to D_{l+1}$. By the induction assumption, $s(v_x, C_l) = s(v'_x, D_l) = s(v'_{x+3}, D_l)$ and $s(v_y, C_l) = s(v'_y, D_l) = s(v'_{y+3}, D_l)$ hold. Hence, agents v'_x and v'_{x+3} (resp., v'_y and v'_{y+3}) update their states similarly to v_x (resp., v_y), and thus C_{l+1} and D_{l+1} are equivalent in this case. Thus, C_r and D_r are equivalent for any $r \geq 0$.

In Ξ', since the number of agents is given correctly, a stable configuration exists, and $\forall v' \in V' : \gamma(s(v')) = yes$ holds in the configuration because G' satisfies *bipartite*. Since C_r and D_r are equivalent for any $r \geq 0$, $\forall v \in V : \gamma(s(v)) = yes$ holds after some configuration in Ξ. Thus, the lemma holds. □

Now, we show that there exists execution Ξ'' of $\mathcal{P}_{n=6}$ with G'' such that all agents converge to *yes*.

Lemma **68.** *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.*

Proof. By Lemma 67, there exists a weakly-fair execution of $\mathcal{P}_{n=6}$ with G such that all agents converge to *yes* in the execution even if $2n$ is given as the number of agents whereas the number of agents is n. This implies that we can apply Lemma 65 to protocol $\mathcal{P}_{n=6}$ and graphs G and G''. This is because G and G'' satisfy the condition of G and G' in Lemma 65, and the protocol $\mathcal{P}_{n=6}$ satisfies the condition of protocol $\mathcal{P}$ in Lemma 65. Hence, there exists a weakly-fair execution Ξ'' of $\mathcal{P}_{n=6}$ with G'' such that $\forall v'' \in V' : \gamma(s(v'')) = yes$ holds in a stable configuration of Ξ''. Thus, the lemma holds. □

Graph G'' does not satisfy *bipartite*. Thus, from Lemma 68, $\mathcal{P}_{n=6}$ is incorrect. Therefore, we have the following theorem.

Theorem **32.** *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 **69.** *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, $E' = \{(v'_x, v'_y) \in V' \times V' \mid (v_x, v_y) \in E\} \setminus \{(v'_\alpha, v'_\beta)\}$.*

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. Let C_t be a stable configuration in Ξ. 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' as follows:

- For $1 \leq i \leq n$, $s(v'_i, C'_0) = s(v_i, C_t)$ holds.
- $C'_{t'}$ 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 $C'_{i-t-1} \rightarrow C'_{i-t}$, v_x interacts with v_y at $C''_{i-1} \rightarrow C''_i$. 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, and a non-tree graph by adding an edge to a line graph, a ring graph, a star graph, and a tree graph, respectively. Moreover, we can construct a bipartite graph, and a k-regular graph by adding an edge to some non-bipartite graph, and some non-k-regular graph, respectively. From Lemma 69, 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 **33.** *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 part, we considered the graph class identification problems on various assumptions such as initial states of agents, fairness, and initial knowledge of agents. With designated initial states, we proposed graph class identification protocols for trees, k-regular graphs, and stars under global fairness. In particular, the star identification protocol works even under weak fairness. Moreover, we showed 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. Overall, we clarified the solvability of the graph class identification problems for each combination of assumptions except for one case of k-regular graph.

Part VI

Conclusion

In this dissertation, we dealt with a distributed system for low performance devices. Concretely, we used the population protocol model as a model of a distributed system for low-performance devices. In the population protocol model, we tackled with two significant challenges: (1) Handling multiple tasks and (2) Efficient task execution. Concretely, we studied the uniform k-partition problem to achieve handling multiple tasks, and studied graph class identification problems to achieve efficient task execution.

In Part III and IV, we dealt with the uniform k-partition problem which aims to divide a population into k groups of equal size. We considered the problem on complete communication graphs in Part III, and considered the problem on arbitrary communication graphs in Part IV. On complete communication graphs, all agents can interact with each other. This implies that this case is for an environment in which the devices (agents) have a wide range of movement (because, on the population protocol model, agents can communicate only when the agents sufficiently close). While such an environment is feasible, there may be cases where devices do not move so wide. Thus, we also considered the problem on arbitrary communication graphs.

In Part III, as a first step, we considered the uniform 2-partition problem on complete graphs. As a result, we clarified the solvability of the uniform 2-partition for each combination of assumptions (24 out of 24 cases). Moreover, if it is solvable, we clarified tight upper and lower bounds on the number of states per agent. On the other hand, for the uniform k-partition problem, we clarified the solvability for most combinations of assumptions (23 out of 24 cases), and we clarified tight upper and lower bounds on the number of states per agent for 10 out of 15 solvable cases.

In Part IV, we clarified the solvability of the uniform 2-partition on arbitrary graphs in variety of cases (20 out of 24 cases), and we clarified tight upper and lower bounds on the number of states per agent for 6 out of 10 solvable cases.

In Part V, we focused on graph class identification problems which aim to decide whether the given communication graph is in desired class (e.g., whether the given graph is a ring graph). We consider graph class identification problems for basic graphs under various assumptions. Concretely, with designated initial states, we proposed graph class identification protocols for trees and k-regular graphs under global fairness, and we proposed the star identification protocol under weak fairness. As impossibility results, for lines, rings, k-regular graphs, trees, or bipartite graphs, we proved that there is no protocol under weak fairness. With arbitrary initial states, we proved that there is no graph class identification protocol for lines, rings, k-regular graphs, stars, trees, or bipartite graphs.

From now, we show future directions of our works. One of the important tasks is to study the time complexity of the problems. In our works, except for the uniform k-partition protocol in Part III, we only considered the space complexity. In order to increase the versatility and applicability, we should study the time complexity of the problems. Another important future task is to investigate the relationship with other basic problems. Concretely, we should consider what is possible by combining our protocols with other protocols, and investigate whether there is a problem which can be solved efficiently by using our protocols as subroutine.

Acknowledgements

Thankfully, I have had the support of many people. First of all, I deeply would like to appreciate Professor Michiko Inoue for her guidance and continuous support. She gave me useful comments to improve my papers and presentations. Secondly, I am grateful to Professor Shoji Kasahara for precious comments on this dissertation. He has left very valuable comments on the dissertation even though it is a bit far from his research field. I express our sincere thanks to Associate Professor Fukuhito Ooshita. He always cooperated with me and gave me various technical comments and encouragements. I would like to greatly thank Assistant Professor Michihiro Shintani for guidance and useful comments on this dissertation. Despite his busy schedule, he provides effective comments on this dissertation. I would like to extremely appreciate Professor Sébastien Tixeuil for his kind supports and thoughtful comments. He has devotedly supported me, and he always cared about my research. Finally, I would also like to thank the members of Dependable System Laboratory for helpful supports.

References

[1] Dan Alistarh, James Aspnes, David Eisenstat, Rati Gelashvili, and Ronald L Rivest. Time-space trade-offs in population protocols. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2560–2579, 2017.

[2] Dan Alistarh, James Aspnes, and Rati Gelashvili. Space-optimal majority in population protocols. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2221–2239, 2018.

[3] 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.

[4] Dan Alistarh and Rati Gelashvili. Recent algorithmic advances in population protocols. *ACM SIGACT News*, 49(3):63–73, 2018.

[5] Dan Alistarh, Rati Gelashvili, and Joel Rybicki. Fast graphical population protocols. *arXiv preprint arXiv:2102.08808*, 2021.

[6] 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.

[7] 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.

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

[9] Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007.

[10] Dana Angluin, James Aspnes, Michael J Fischer, and Hong Jiang. Self-stabilizing population protocols. In *International Conference On Principles Of Distributed Systems*, pages 103–117, 2005.

[11] 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.

[12] 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.

[13] James Aspnes and Eric Ruppert. An introduction to population protocols. In *Middleware for Network Eccentric and Mobile Applications*, pages 97–120, 2009.

[14] Joffroy Beauquier, Peva Blanchard, and Janna Burman. Self-stabilizing leader election in population protocols over arbitrary communication graphs. In *International Conference on Principles of Distributed Systems*, pages 38–52, 2013.

[15] 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.

[16] 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.

[17] Stav Ben-Nun, Tsvi Kopelowitz, Matan Kraus, and Ely Porat. An o (log3/2 n) parallel time population protocol for majority with o (log n) states. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, pages 191–199, 2020.

[18] Petra Berenbrink, Robert Elsässer, Tom Friedetzky, Dominik Kaaser, Peter Kling, and Tomasz Radzik. A population protocol for exact majority with o(log5/3 n) stabilization time and theta(log n) states. In *Proceedings of International Symposium on Distributed Computing*.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] Janna Burman, Joffroy Beauquier, and Devan Sohier. Brief announcement: Space-optimal naming in population protocols. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 479–481, 2018.

[24] 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.

[25] Ioannis Chatzigiannakis, Othon Michail, and Paul G. Spirakis.

[26] 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.

[27] 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.

[28] Carole Delporte-Gallet, Hugues Fauconnier, Rachid Guerraoui, and Eric Ruppert. When birds die: Making population protocols fault-tolerant. *Distributed Computing in Sensor Systems*, pages 51–66, 2006.

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

[30] 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, 2006.

[31] 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.

[32] 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, 2019.

[33] Taisuke Izumi. On space and time complexity of loosely-stabilizing leader election. In *Proceedings of International Colloquium on Structural Information and Communication Complexity*, pages 299–312, 2015.

[34] 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.

[35] Adrian Kosowski and Przemyslaw Uznanski. Brief announcement: Population protocols are fast. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 475–477, 2018.

[36] Anissa Lamani and Masafumi Yamashita. Realization of periodic functions by self-stabilizing population protocols with synchronous handshakes. In *Proceedings of International Conference on Theory and Practice of Natural Computing*, pages 21–33, 2016.

[37] George B Mertzios, Sotiris E Nikoletseas, Christoforos L Raptopoulos, and Paul G Spirakis. Determining majority in networks with local interactions and very small local memory. In *International Colloquium on Automata, Languages, and Programming*, pages 871–882, 2014.

[38] Othon Michail, Ioannis Chatzigiannakis, and Paul G Spirakis. Mediated population protocols. *Theoretical Computer Science*, 412(22):2434–2450, 2011.

[39] 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.

[40] Yuichi Sudo, Ryota Eguchi, Taisuke Izumi, and Toshimitsu Masuzawa. Time-optimal loosely-stabilizing leader election in population protocols. *arXiv preprint arXiv:2005.09944*, 2020.

[41] Yuichi Sudo, Toshimitsu Masuzawa, Ajoy K Datta, and Lawrence L Larmore. The same speed timer in population protocols. In *Proceedings of International Conference on Distributed Computing Systems*, pages 252–261, 2016.

[42] Yuichi Sudo, Junya Nakamura, Yukiko Yamauchi, Fukuhito Ooshita, Hirotsugu Kakugawa, and Toshimitsu Masuzawa. Loosely-stabilizing leader election in a population protocol model. *Theoretical Computer Science*, 444:100–112, 2012.

[43] Yuichi Sudo, Fukuhito Ooshita, Taisuke Izumi, Hirotsugu Kakugawa, and Toshimitsu Masuzawa. Time-optimal leader election in population protocols. *IEEE Transactions on Parallel and Distributed Systems*, 2020.

[44] Yuichi Sudo, Fukuhito Ooshita, Hirotsugu Kakugawa, and Toshimitsu Masuzawa. Loosely-stabilizing leader election on arbitrary graphs in population protocols. In *International Conference on Principles of Distributed Systems*, pages 339–354, 2014.

[45] Yuichi Sudo, Fukuhito Ooshita, Hirotsugu Kakugawa, and Toshimitsu Masuzawa. Loosely stabilizing leader election on arbitrary graphs in population protocols without identifiers or random numbers. *IEICE Transactions on Information and Systems*, 103(3):489–499, 2020.

[46] Yuichi Sudo, Fukuhito Ooshita, Hirotsugu Kakugawa, Toshimitsu Masuzawa, Ajoy K Datta, and Lawrence L Larmore. Loosely-stabilizing leader election with polylogarithmic convergence time. *Theoretical Computer Science*, 806:617–631, 2020.

[47] 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.

Publication list

Peer-Reviewed Journal Papers

[1] Hiroto Yasumi, Fukuhito Ooshita, Ken'ichi Yamaguchi, and Michiko Inoue, Space-optimal population protocols for uniform bipartition under global fairness, *IEICE Transactions on Information and Systems*, 102(3):454–463, 2019.

[2] Hiroto Yasumi, Naoki Kitamura, Fukuhito Ooshita, Taisuke Izumi, and Michiko Inoue, A population protocol for uniform k-partition under global fairness, *International Journal of Networking and Computing*, 9(1):97–110, 2019.

[3] Hiroto Yasumi, Fukuhito Ooshita, Michiko Inoue, and Sebastien Tixeuil, Uniform bipartition in the population protocol model with arbitrary graphs, *Theoretical Computer Science*, 892:187–207, 2021.

Peer-Reviewed Conference Papers

[1] Hiroto Yasumi, Fukuhito Ooshita, Ken'ichi Yamaguchi, and Michiko Inoue, Constant-space population protocols for uniform bipartition, *Proceedings of International Conference on Principles of Distributed Systems*, 2017.

[2] Hiroto Yasumi, Naoki Kitamura, Fukuhito Ooshita, Taisuke Izumi, and Michiko Inoue, A population protocol for uniform k-partition under global fairness, *Proceedings of Workshop on Advances in Parallel and Distributed Computational Models*, pages 813–819, 2018.

[3] Hiroto Yasumi, Fukuhito Ooshita, and Michiko Inoue, Uniform partition in population protocol model under weak fairness, *Proceedings of International Conference on Principles of Distributed Systems*, 2019.

[4] Hiroto Yasumi, Fukuhito Ooshita, Michiko Inoue, and Sébastien Tixeuil, Uniform Bipartition in the Population Protocol Model with Arbitrary Communication Graphs, *Proceedings of International Conference on Principles of Distributed Systems*, 2020.

[5] Hiroto Yasumi, Fukuhito Ooshita, and Michiko Inoue, Population Protocols for Graph Class Identification Problems, *Proceedings of International Conference on Principles of Distributed Systems*, 2021.