Optimal probabilistic ring exploration by semi-synchronous oblivious robots☆

Stéphane Devismes[a,*], Franck Petit[b], Sébastien Tixeuil[b]

[a] *VERIMAG UMR 5104, Université Joseph Fourier, Grenoble, France*
[b] *LIP6 UMR 7606, UPMC Sorbonne Universités, France*

Abstract

We consider a team of k identical, oblivious, and semi-synchronous mobile robots that are able to sense (*i.e.*, view) their environment, yet are unable to communicate, and evolve on a constrained path. Previous results in this weak scenario show that initial symmetry yields high lower bounds when problems are to be solved by *deterministic* robots.

In this paper, we initiate research on probabilistic bounds and solutions in this context, and focus on the *exploration* problem of anonymous unoriented rings of any size n. It is known that $k = \Theta(\log n)$ deterministic robots are necessary and sufficient to solve the problem, provided that k and n are coprime. By contrast, we show that *four* identical probabilistic robots are necessary and sufficient to solve the same problem, also removing the coprime constraint. Our positive results are constructive.

Keywords: Mobile robots, obliviousness, ring exploration, probabilistic algorithm
2010 MSC: 68W15, 68M14

1. Introduction

Teams (or swarms) of mobile autonomous robots working together to learn and achieve cooperative tasks is an important open area of research. The robots are endowed with visibility sensors and motion actuators. Numerous potential applications exist for such multi-robot systems: environmental monitoring, large-scale construction, mapping, urban search and rescue, surface cleaning, risky area surrounding or surveillance, and the exploration of unknown environments, to only name a few. We address the *exploration* problem, where a team of robots cooperate to collectively explore the environment. Exploration is a basic building block for many of the aforementioned applications. For instance, mapping an unknown area

☆Preliminary versions of this paper appeared in [1, 2].

*Corresponding author.

Email addresses: Stephane.Devismes@imag.fr (Stéphane Devismes), Franck.Petit@lip6.fr (Franck Petit), Sebastien.Tixeuil@lip6.fr (Sébastien Tixeuil)

URL: http://www-verimag.imag.fr/~devismes/ (Stéphane Devismes), http://pagesperso-systeme.lip6.fr/Franck.Petit/ (Franck Petit), http://www-npa.lip6.fr/~tixeuil/ (Sébastien Tixeuil)

requires that the robots (collectively) explore the whole area. Similarly, to search and rescue people after a disaster, the team of robots potentially has to explore the whole area. The so-called "area" is often considered to be either the *continuous* Euclidean space (possibly with obstacles and objects) or a *discrete* space. In the latter case, space is partitioned into a finite number of locations represented by a graph, where nodes represent locations that can be sensed by the robots, and edges represent the possibility for a robot to move from one location to the other, *e.g.*, a building, a town, a factory, a mine, and more generally, zoned areas. In a discrete environment, the *(terminating) exploration* task requires every possible location to be visited by at least one robot, with the additional constraint that all robots stop moving after task completion.

The ability of a team of robots to succeed in accomplishing the assigned task greatly depends on the capabilities that the robots possess, namely, their sensing capabilities. Clearly, the type of viewing device has a great impact on the knowledge that the robots have of their environment. For example, endowed with a camera or a sonar, vision capabilities are limited to a certain distance. By contrast, if the robots have access to a global localization system (GPS, egocentric zone-based RFID technology), then their viewing capabilities are *a priori* unlimited. Note that GPS actually provides a global coordinate system of the environment. Can the robots achieve the same tasks if they are only equipped with a compass? What happens if they are devoid of any kind of orientation capabilities? What can they achieve if their sensing devices do not enable to differentiate between two of them? Obviously, the stronger the device capabilities are, the easier the problem is solved.

This paper falls into the category of works tackling the exploration problem from a computational point of view. In other words, our focus is to understand the relationship between the capabilities of the robots and the solvability of the exploration of a discrete environment. We consider autonomous robots that are endowed with visibility sensors but that are otherwise unable to communicate. We assume robots with weak capacities: they are *anonymous* (they are devoid of any visible ID), *oblivious* (that is, they cannot remember the past), and have no compass whatsoever.

1.1. Related Work

The vast majority of literature on robots (refer to [3] for an introductory survey) considers that they evolve in a continuous two-dimensional Euclidean space and use visual sensors with perfect accuracy that permit to locate other robots with infinite precision, *e.g.*, [4, 5, 6, 7, 8, 9]. Several works investigate restricting the capabilities of both visibility sensors and motion actuators of the robots. In [10, 11], robots visibility sensors are supposed to be accurate within a constant range, and sense nothing beyond this range. In [11, 12], the space allowed for the motion actuator is reduced to a one-dimensional continuous one: a ring in [11], an infinite path in [12].

A recent trend is to shift from the classical continuous model to the discrete model. The discrete model restricts both sensing and actuating capabilities of robots with respect to the previous works that assume a continuous two-dimensional Euclidean space and visual sensors with perfect accuracy. Indeed, for each location, a robot is able to sense if the location is empty or if robots are positioned on it. Also, a robot is not able to move from a position to another unless there is explicit indication to do so (*i.e.*, the two locations are connected by an edge in the representing graph). The discrete model permits to simplify many robot protocols by reasoning on finite structures rather than on infinite ones. However, as noted in most related papers [13, 14, 15, 16, 17, 18, 19, 20, 21, 22], this simplicity comes with the cost of extra symmetry possibilities, especially when the authorized paths are also symmetric. (Indeed, techniques to break formation such as those of [6] cannot be used in the discrete model.)

Assuming the same sensing capabilities—a robot is only able to sense if the location is empty or if anonymous robots are positioned on it—, the two main problems that have been studied in the discrete robot model are *gathering* [18, 19, 20, 21, 22] and *exploration* [13, 14, 15, 16, 17]. For gathering, both breaking symmetry [18, 20, 21, 22] and preserving symmetry [19] are meaningful approaches. For exploration, the fact that robots need to stop after exploring all locations requires robots to "remember" how much of the graph was explored, *i.e.*, to be able to distinguish between various stages of the exploration process since robots have no persistent memory. As configurations can be distinguished only by robot positions, the main complexity measure is then the number of robots that are needed to explore a given graph. The vast number of symmetric situations induces a large number of required robots. For tree networks, [14] shows that $\Omega(n)$ robots are necessary for most n-sized trees, and that sub-linear robot complexity (actually $\Theta(\log n/\log\log n)$) is possible only if the maximum degree of the tree is 3. In the case of line networks, the solvable cases have been fully characterized [17]: one or two robots are insufficient, 4 robots are sufficient if n is odd, and every other case is solvable for n greater than the number of robots. In uniform rings, [13] proves that the necessary and sufficient number of robots is $\Theta(\log n)$, although it proposes an algorithm that works with an additional assumption: the number k of robots and the size n of the ring are coprime. For the pairs of k and n that are coprime, five robots are necessary and sufficient in the deterministic case [15]. The case of general graphs has been investigated with the additional hypothesis that a local labeling is available to the robots at each node [16].

Note that all previous approaches in the discrete model are *deterministic*, *i.e.*, if a robot is presented twice the same situation, its behavior is the same in both cases.

1.2. Contribution

In this paper, we consider the *probabilistic exploration* of an anonymous unoriented ring in the *semi-synchronous model* [23] to lift constraints and to reduce bounds given in [13]. So, with respect to [13], we relax both the model (from asynchronous to semi-synchronous) and the specification of the problem (from deterministic to probabilistic exploration).

These two relaxations are mandatory. Indeed, we show that even using randomization, the exploration problem remains not solvable in the (fully) asynchronous model if k divides n. Moreover, it is straightforward that the necessary conditions and bounds exposed in [13] for the deterministic exploration still hold in the semi-synchronous model.

By contrast with [13], we show that *four* identical probabilistic robots are necessary and sufficient to solve the exploration problem in any anonymous unoriented ring in the semi-synchronous model, also removing the coprime constraint between the number of robots and the size of the ring. Our proof is constructive, as we present a probabilistic protocol for four robots to explore any ring of size at least four.

Outline. The remaining of the paper is divided as follows. Section 2 presents the system model that we use throughout the paper. In Section 3, we justify why we need to use the semi-synchronous model. In the same section, we provide evidence that no three probabilistic robots can explore any ring of more than three nodes, while Section 4 presents our protocol with four robots, and its correctness proof. Section 5 gives some concluding remarks.

2. Preliminaries

2.1. Distributed Systems

We consider systems of autonomous mobile entities called *agents* or *robots* evolving into a *graph*. We assume that the graph is a *ring* of n nodes, $u_0, \ldots, u_{n-1}$, *i.e.*, u_i is connected to both u_{i-1} and u_{i+1}.[1] The indices are used for notation purposes only: the nodes are *anonymous* (*i.e.*, every node is identical) and the ring is *unoriented* (*i.e.*, given two neighboring nodes u and v, there is no kind of explicit or implicit labeling allowing to determine whether u is on the right or on the left of v).

2.2. Robots

Operating on the ring are $k \leq n$ robots. The robots do not communicate in an explicit way; however they see the position of all other robots and can acquire knowledge from this information.

Each robot operates according to its (local) *program*. We call *protocol* a collection of k *programs*, each one operating on a single robot. Here we assume that robots are *uniform* and *anonymous*, *i.e.*, they all have the same program using no local parameter (such that an identity) allowing to differentiate them.

The program of a robot consists in executing *Look-Compute-Move cycles* infinitely many times. That is, the robot first observes its environment (Look phase). Based on its observation, a robot then (probabilistically or deterministically) decides to move or stay idle (Compute phase). When a robot decides to move, it moves toward its destination during the Move phase.

A robot can decide between moving and staying idle using some probability p —with $0 < p < 1$— during the Compute phase of one of its cycles. In this case, we say that the robot *tries to move*. Conversely, when a robot deterministically decides to move during its Compute phase, we simply say that the robot *moves*.

We assume that robots cannot remember any previous observation nor computation performed in any previous cycle. Such robots are said to be *oblivious* (or *memoryless*).

[1] Every computation over indices is assumed to be modulus n.

2.3. Computational Models

We consider two models: the *semi-synchronous* [9, 23] and *asynchronous* [13, 24] models. In both models, time is represented by an infinite sequence of instants 0, 1, 2, ... No robot has access to this global time. Moreover, every robot executes cycles infinitely many times. Each robot performs its own cycles in sequence. However, the time between two cycles of the same robot and the interleavings between cycles of different robots are decided by an *adversary*. As a matter of facts, we are interested in algorithms that correctly operate despite the choices of the adversary. In particular, our algorithms should work even if the adversary forces the execution to be fully sequential or fully synchronous.

In the *semi-synchronous* model, each Look-Compute-Move cycle execution is assumed to be *atomic*: every robot that is activated by the adversary at instant t atomically executes a full cycle between t and $t+1$.

In the *asynchronous* model, Look-Compute-Move cycles are performed asynchronously by each robot: the time between Look, Compute, and Move operations is finite yet unbounded, and decided by the adversary. The only constraint is that Look is instantaneous and Move is atomic: if a move starts at instant t, it is terminated at instant $t+1$.

Remark that in both models, any robot performing a Look operation sees all other robots on nodes and not on edges. However, in the *asynchronous* model, a robot $\mathcal{R}$ may perform a Look operation at some time t, perceiving robots at some nodes, then Compute a target neighbor at some time $t' > t$, and Move to that neighbor at some later time $t'' > t'$ in which some robots are at different nodes from those previously perceived by $\mathcal{R}$ because in the meantime they moved. Hence, robots may move based on significantly outdated perceptions. In the *asynchronous* model, a robot is said to be *engaged* if it has decided to move during its compute phase but not yet moved.

2.4. Multiplicity

We assume that during the Look phase, every robot can perceive whether several robots are located on the same node or not. This ability is called *Multiplicity Detection*. We shall indicate by $d_i(t)$ the multiplicity of robots present in node u_i at instant t.

In this paper, we consider two kinds of multiplicity detection: the *strong* and *weak* multiplicity detections.

Under the *strong* multiplicity detection, for every node u_i, d_i is a function $\mathbb{N} \mapsto \mathbb{N}$ where $d_i(t) = j$ indicates that there are j robots in node u_i at instant t. If $d_i(t) = 0$, then we say that u_i is *free* at instant t, otherwise u_i is said to be *occupied* at instant t. If $d_i(t) > 1$, then we say that u_i contains a *tower (of $d_i(t)$ robots)* at instant t.

Under the *weak* multiplicity detection, for every node u_i, d_i is a function $\mathbb{N} \mapsto \{\circ, \perp, \top\}$ defined as follows: $d_i(t)$ is equal to either $\circ$, $\perp$, or $\top$ according to u_i contains no, one or several robots at time instant t. As previously, if $d_i(t) = \circ$, then we say that u_i is *free* at instant t, otherwise u_i is said to be *occupied* at instant t. If $d_i(t) = \top$, then we say that u_i contains a *tower* at instant t.

2.5. Configurations

Given an arbitrary orientation of the ring and a node u_i, $\gamma^{+i}(t)$ (respectively, $\gamma^{-i}(t)$) denotes the sequence $\langle d_i(t)d_{i+1}(t)\ldots d_{i+n-1}(t)\rangle$ (resp., $\langle d_i(t)d_{i-1}(t)\ldots\ d_{i-(n-1)}(t)\rangle$). The sequence $\gamma^{-i}(t)$ is called *mirror* of $\gamma^{+i}(t)$ and conversely. The unordered pair $\{\gamma^{+i}(t), \gamma^{-i}(t)\}$ is called the *view*[2] of node u_i at instant t (we omit "at instant t" when it is clear from the context). The view of u_i is said to be *symmetric* if and only if $\gamma^{+i}(t) = \gamma^{-i}(t)$. Otherwise, the view of u_i is said to be *asymmetric*. Actually, the view represents the whole information a robot acquires during a Look phase.

By convention, we state that the *configuration* of the system at instant t is $\gamma^{+0}(t)$. Any configuration from which no robot moves nor tries to move is said to be *terminal*. Let $\gamma = \langle x_0x_1\ldots x_{n-1}\rangle$ be a configuration. The configuration $\langle x_ix_{i+1}\ldots x_{i+n-1}\rangle$ is obtained by *rotating* γ by $i \in [0\ldots n-1]$. Two configurations γ and γ' are said to be *indistinguishable* if and only if γ' can be obtained by rotating γ or its mirror. Two configurations that are not indistinguishable are said to be *distinguishable*. We designate by *initial configurations* the configurations from which the system can start at instant 0.

During the Look phase, it may happen that both edges incident to a node v currently occupied by the robot look identical in the snapshot, *i.e.*, v lies on a symmetric axis of the configuration. In this case, if the robot decides to move, it may traverse any of the two edges. We assume the worst case decision in such cases, *i.e.*, that the decision to traverse one of these two edges is taken by the adversary.

2.6. Computations

We call *computation* any infinite sequence of configurations $\gamma_0, \gamma_1, \gamma_2, \ldots$ such that (1) γ_0 is a possible initial configuration and (2) starting the system in γ_0 at instant 0, there exists a scheduling of cycle executions that can produce γ_1 at instant 1, γ_2 at instant 2, … Any transition γ_t, γ_{t+1} is called a *step* of the computation. A computation c *terminates* if c contains a terminal configuration.

A *scheduler* is a predicate over computations, that is, a scheduler defines a set of *admissible* computations.[3] Here we assume a *distributed fair* scheduler. Distributed means that robots can execute cycles concurrently. Fair means that every robot executes cycles infinitely often during a computation. A particular case of distributed fair scheduler is the *sequential* fair scheduler: every step consists in a full cycle execution of a *single* robot. In the following, we call *sequential computation* any computation that satisfies the sequential fair scheduler predicate.

2.7. Problem to be solved

We consider the *exploration* problem, where k robots, initially placed at different nodes, collectively explore an n-node ring before stopping moving forever. More formally, a protocol $\mathcal{P}$ *deterministically* (resp. *probabilistically*) solves the exploration problem if and only if every

[2]Since the ring is unoriented, the pair $\{\gamma^{+i}(t), \gamma^{-i}(t)\}$ cannot be ordered.

[3]The scheduler can be seen as a restriction of the adversary's power.

computation c of $\mathcal{P}$ starting from a *towerless* configuration satisfies: (1) c terminates *in finite time* (resp. *with probability 1*); (2) every node is visited by at least one robot during c.

Note that the previous definition implies that every initial configuration of the system in the problem we consider are *towerless*. Note also that using probabilistic solutions, termination is not certain, however the overall probability of non-terminating computations is 0. Finally, observe that the problem is not defined for $k > n$ and straightforward for $k = n$ (in this latter case the exploration is already accomplished in the initial configuration). Hence, throughout the paper, we always assume that $k < n$.

3. Negative Results

In this section, we present two impossibility results. The first one justifies the use of the semi-synchronous model. The second one gives a lower bound on the number of robots required to perform a probabilistic ring exploration. Note that to have results as general as possible we assume in this section that robots have strong multiplicity detection capabilities.

In the seminal work on ring exploration [13], authors consider deterministic solutions in the asynchronous model. As a preliminary results, they show that deterministic ring exploration is not always possible in the asynchronous model, in particular, if k (the number of robots) divides n (the ring-size). We now show that this result also holds for probabilistic solutions, hence justifying why we assume the semi-synchronous model.

Theorem 1. *Let $k < n$. In the asynchronous model, if k divides n, then the probabilistic exploration of an n-node ring is not possible.*

Proof. By contradiction, let $\mathcal{P}$ be a probabilistic ring exploration protocol. Consider an initial configuration γ where the k robots are equidistantly placed on the ring (it is possible because k divides n). As $k < n$, the exploration is not terminated in γ. Hence, there is at least one robot that has a strictly positive probability to decide to move if it performs its Look phase in γ. Now, as robots are uniform and the views of all robots are identical in γ, this implies that all robots have a strictly positive probability to decide to move if they perform their Look phase in γ. Moreover, the view of each robot is symmetric in γ, so if a robot moves in γ, then the incident edge it traverses is chosen by the adversary. Hence, from γ, a possible execution is then the following: (1) the adversary forces every not-yet-engaged robot to execute cycles until they all are engaged, (2) no move occurs before all robots are engaged, (3) once all robots are engaged, all moves occur synchronously, and (4) the edges traversed during the moves are chosen by the adversary in such a way that the full symmetry of the configuration is maintained. Then, as the next configuration is indistinguishable from γ and robots are oblivious, we can repeat the process indefinitely. Hence, with strictly positive probability, the adversary can force the computation to not terminate, despite the computation satisfies the distributed fair scheduler, a contradiction. □

We now show that the ring exploration is impossible to solve, even in a probabilistic manner, in our settings (*i.e.*, oblivious robots, anonymous ring, semi-synchronous model, distributed fair scheduler, ...) if there are less than four robots (Corollary 2). The proof is made in two steps:

- The first step is based on the fact that obliviousness constraints any exploration protocol to construct an implicit memory using the configurations. We show that if the scheduler behaves sequentially, then in any case except one, it is not possible to particularize enough configurations to memorize which nodes have been visited (Theorem 2 and Lemma 3).

- The second step consists in excluding the last case (Theorem 3).

First, as $n > k$ and robots are oblivious, any terminal configuration should be distinguishable from any possible initial (towerless) configuration. Hence, it follows:

Remark 1. *If $n > k$, any terminal configuration of any exploration protocol contains at least one tower.*

The next definition is used in Lemmas 1 and 2, proven afterward. These lemmas are technical results that lead to Corollary 1. The latter exhibits the minimal size of a subset of particular configurations required to solve the exploration problem on the ring.

Definition 1 (MRS). *Let s be a sequence of configurations. The* minimal relevant subsequence *of s, noted $\mathcal{MRS}(s)$, is the maximal subsequence of s where no two consecutive configurations are identical.*

Lemma 1. *Let $\mathcal{P}$ be any (probabilistic or deterministic) exploration protocol for k robots on a ring of $n > k$ nodes. For every sequential computation c of $\mathcal{P}$ that terminates, $\mathcal{MRS}(c)$ has at least $n - k + 1$ configurations containing a tower of less than k robots.*

Proof. Assume, by contradiction, that there is a sequential computation c of $\mathcal{P}$ that terminates and such that $\mathcal{MRS}(c)$ has less than $n - k + 1$ configurations containing a tower of less than k robots.

Take the last configuration α without tower that appears in c and all remaining configurations that follow in c (all of them contains a tower) and form c'. As α could be an initial configuration and c is an admissible sequential computation that terminates, c' is also an admissible sequential computation of $\mathcal{P}$ that terminates.

By definition, $\mathcal{MRS}(c')$ is constituted of a configuration with no tower only followed by configurations with tower and $n - k$ new nodes (remember that k nodes are already visited in the initial configuration) must be visited before c' reaches its terminal configuration.

Consider a step $\beta\beta'$ in c'.

1. If $\beta = \beta'$, then no node is visited during the step.

2. If $\beta \neq \beta'$, then there are three possible cases:

   (a) *β contains no towers.* In this case, $\beta = \alpha$ (the initial configuration of c') and β' contains a tower. As only one robot moves in $\beta\beta'$ to create a tower (c' is sequential), no node is visited during this step.

(b) *β contains a tower and β' contains a tower of k robots.* As c' is sequential and all robots are located at the same node in β', one robot moves to an already occupied node in $\beta\beta'$ and no node is visited during this step.

(c) *β contains a tower and β' contains a tower of less than k robots.* In this case, at most one node is visited in $\beta\beta'$ because c' is sequential.

To sum up, only the steps from a configuration containing a tower to a configuration containing a tower of less than k robots (Case 2.(c)) allow to visit at most one node each time. Now, in $\mathcal{MRS}(c')$ there are less than $n-k+1$ configurations containing a tower of less than k robots and the first of these configurations appearing into c' is consecutive to a step starting from the initial configuration (Case 2.(a)). Hence, less than $n-k$ nodes are dynamically visited during c' and, as exactly k nodes are visited in the initial configuration, less than n nodes are visited when c' terminates, a contradiction. □

Lemma 2. *Let $\mathcal{P}$ be any (probabilistic or deterministic) exploration protocol for k robots on a ring of $n > k$ nodes. For every sequential computation c of $\mathcal{P}$ that terminates, $\mathcal{MRS}(c)$ has at least $n-k+1$ configurations containing a tower of less than k robots and any two of them are distinguishable.*

Proof. Consider any sequential computation c of $\mathcal{P}$ that terminates.

By Lemma 1, $\mathcal{MRS}(c)$ has x configurations containing a tower of less than k robots where $x \geq n-k+1$.

We first show that (*) *if $\mathcal{MRS}(c)$ contains at least two configurations having a tower of less than k robots that are indistinguishable, then there exists a sequential computation c' that terminates and such that $\mathcal{MRS}(c')$ has x' configurations containing a tower of less than k robots where $x' < x$.* Assume that there are two indistinguishable configurations γ and γ' in $\mathcal{MRS}(c)$ having a tower of less than k robots. Without loss of generality, assume that γ occurs at time t in c and γ' occurs at time $t' > t$ in c. Consider the two following cases:

1. **γ' can be obtained by applying a rotation of i to γ.** Let p be the prefix of c from instant 0 to instant t. Let s be the suffix of c starting at instant $t'+1$. Let s' be the sequence obtained by applying a rotation of $-i$ to the configurations of s. As the ring and the robots are anonymous, ps' is an admissible sequential computation that terminates. Moreover, by construction $\mathcal{MRS}(ps')$ has x' configurations containing a tower of less than k robots where $x' < x$. Hence (*) is verified in this case.
2. **γ' can be obtained by applying a rotation of i to the mirror of γ.** We can prove (*) in this case by slightly modifying the proof of the previous case: we have just to apply the rotation of $-i$ to the *mirrors* of the configurations of s.

By (*), if $\mathcal{MRS}(c)$ contains less than $n-k+1$ distinguishable configurations with a tower of less than k robots, it is possible to (recursively) construct an admissible computation c' of $\mathcal{P}$ that terminates such that $\mathcal{MRS}(c')$ has less than $n-k+1$ configurations containing a tower of less than k robots, a contradiction to Lemma 1. Hence, the lemma holds. □

From Lemma 2, we can deduce the following corollary:

Corollary 1. *Considering any (probabilistic or deterministic) exploration protocol for k robots on a ring of $n > k$ nodes, there exists a subset $\mathcal{S}$ of at least $n - k + 1$ configurations such that:*

1. *Any two different configurations in $\mathcal{S}$ are distinguishable, and*
2. *In every configuration in $\mathcal{S}$, there is a tower of less than k robots.*

Theorem 2. $\forall k, 0 \leq k < 3, \forall n > k$, *there is no exploration protocol (even probabilistic) of an n-node ring with k robots.*

Proof. First, for $k = 0$, the theorem is trivially verified. Consider then the cases $k = 1$ and $k = 2$: with one robot it is impossible to construct a configuration with one tower; with two robots it is impossible to construct a configuration with one tower of less than k robots ($k = 2$). Hence, for $k = 1$ and $k = 2$, the theorem is a direct consequence of Corollary 1. □

Lemma 3. $\forall n > 4$, *there is no exploration protocol (even probabilistic) of an n-node ring with three robots.*

Proof. With three robots, the size of the maximal set of distinguishable configurations containing a tower of less than three robots is $\lfloor n/2 \rfloor$. By Corollary 1, we have then the following inequality:

$$\lfloor n/2 \rfloor \geq n - k + 1$$

From this inequality, we can deduce that n must be less or equal to four, and we are done. □

From this point on, we know that, assuming $k < 4$, Corollary 1 prevents the existence of any exploration protocol in any case except one: $k = 3$ and $n = 4$ (Theorem 2 and Lemma 3). Actually, assuming that the scheduler is sequential is not sufficient to show the impossibility in this latter case: Indeed, if we assume a sequential scheduler, then there is an exploration protocol for $k = 3$ and $n = 4$. The protocol works as shown in Figure 1.

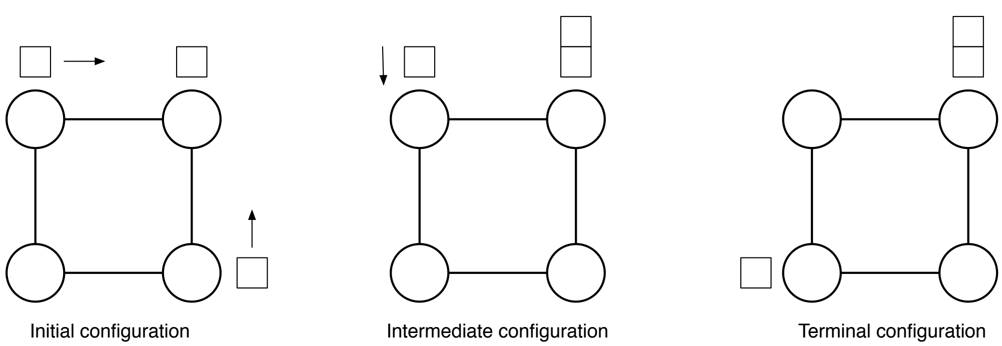

Figure 1: Protocol for $n = 4$, $k = 3$, and assuming a sequential scheduler. (The squares represent robots. The arrows show the destinations of the robots if they move.)

The theorem below is obtained by showing the impossibility for $k = 3$ and $n = 4$ using a (non-sequential) distributed fair scheduler. The proof of this theorem consists of a combinatorial study of all possible protocols for $k = 3$ robots and $n = 4$ nodes. In each case, we show that the protocol leads to one of the following contradictions:

- Either, the adversary can force with a strictly positive probability, an admissible computation to never terminate.
- Or, for every possible terminal configuration (*i.e.*, any configuration containing a tower, refer to Remark 1), there is an admissible computation that reaches the terminal configuration without visiting all nodes.

Theorem 3. *Assuming a distributed fair scheduler, there is no exploration protocol (even probabilistic) of an n-node ring with three robots for every $n > 3$.*

Proof. Lemma 3 excludes the existence of any exploration protocol for three robots on a ring of $n > 4$ nodes. Hence, to show this theorem, we just have to show that there is no exploration protocol for three robots working on a ring of four nodes.

Assume, by contradiction, that there exists an exploration protocol $\mathcal{P}$ for three robots on a ring of four nodes. Then, any possible initial configuration is indistinguishable with the configuration presented in Figure 2. Moreover, any possible terminal configuration contains a tower by Remark 1 and so is indistinguishable with one of the three configurations presented in Figure 3.

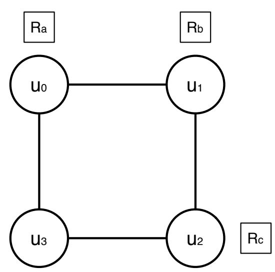

Figure 2: Initial configuration for $n = 4$ and $k = 3$. (The indices are used for notation purposes only.)

Consider that the system is initially in the configuration of Figure 2. Three cases are possible at instant 0 using $\mathcal{P}$:

- *There is a strictly positive probability that robot R_a (resp. robot R_c) moves to node u_3 if it executes a cycle.*[4] In this case, assume that the adversary can activate R_a to execute cycles until it moves. Then, the probability that R_a eventually moves is 1 (resp. R_a moves in one step if $\mathcal{P}$ is deterministic). Once R_a has moved, R_b has a strictly positive probability to move to node u_0 if it executes a cycle; indeed, R_b is in

[4] If $\mathcal{P}$ is deterministic, the probability is 1.

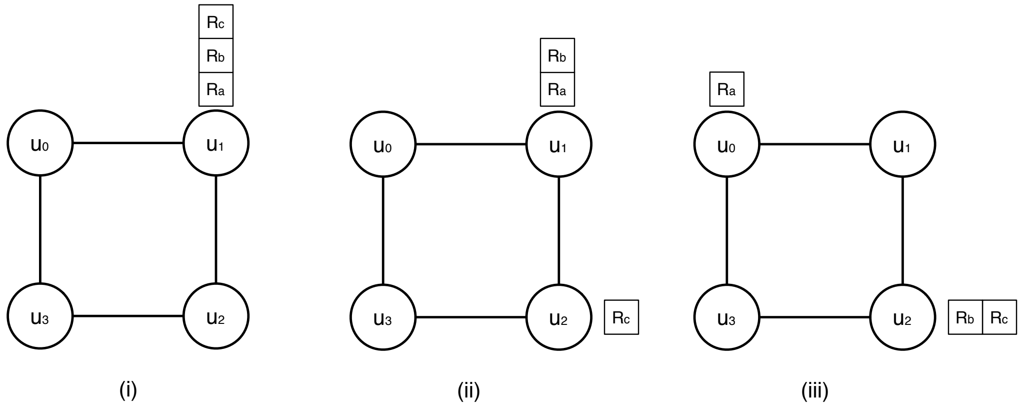

Figure 3: Terminal configurations for $n = 4$ and $k = 3$. (The indices are used for notation purposes only.)

the same situation as R_a at instant 0. Assume then that the adversary activates R_b until it moves. The probability that R_b eventually moves is 1. Repeating this scheme for R_c and so on, we can deduce that with a strictly positive probability, the adversary can force the computation to never terminate despite the computation is distributed and fair, a contradiction.

- *There is a strictly positive probability that robot R_a (resp. robot R_c) moves to node u_1 if it executes a cycle.* In this case, there is an admissible computation where R_a and R_c move to node u_1 in the first step. At instant 1, the system is in a configuration that is indistinguishable with configuration (i) of Figure 3. As node u_3 is still not visited in this case, any configuration that is indistinguishable with configuration (i) cannot be terminal. There is also an admissible computation where only R_a moves to node u_1 in the first step. At instant 1, the system is in a configuration that is indistinguishable with configuration (ii) of Figure 3. As node u_3 is still not visited in this case, any configuration that is indistinguishable with configuration (ii) cannot be terminal. Moreover, assuming that the system reaches a configuration indistinguishable from configuration (i) of Figure 3 at instant 1, there is a strictly positive probability that the three robots move (the configuration is not terminal and all robots have the same view). If they move, the adversary can choose which incident edge they traverse because the configuration is symmetric. Hence, we can obtain a configuration indistinguishable with configuration (iii) of Figure 3 and where node u_3 is still not visited. Thus, any configuration that is indistinguishable with configuration (iii) cannot be terminal. Hence, no configuration can be terminal, a contradiction.

- *There is a strictly positive probability that robot R_b moves if it executes a cycle.* Assume that the adversary activates R_b until it moves. Then, the probability that R_b eventually moves is 1. Once R_b decides to move, the adversary can choose the edge that R_b traverses because the view from R_b is symmetric. Hence, the system can reach the configuration γ: R_a is in node u_0, R_b and R_c are in node u_2. This configuration is indistinguishable with configuration (iii) in Figure 3 and node u_3 is still not visited. (Consequently, every configuration indistinguishable with configuration (iii) in Figure 3 cannot be terminal.) Consider the two following sub-cases:

- *The probability that R_a moves, if it executes a cycle, is 0.* Then, there is a strictly positive probability that R_c (resp. R_b) moves if it executes a cycle. Assume that the adversary activates R_a and then R_c until R_c moves. The probability that R_c eventually moves is 1 and as the the view from R_c is symmetric, the adversary can decide which edge R_c will traverse. Assume that the adversary forces R_c to go to node u_1, the system reaches a configuration indistinguishable with the initial configuration. We can repeat the same scheme infinitely often. So, with a strictly positive probability, the adversary can force the computation to never terminate despite the computation is distributed and fair, a contradiction.
- *The probability that R_a moves, if it executes a cycle, is strictly positive.* Assume that the adversary activates R_a until it moves. Then, the probability that R_a eventually moves is 1 and as the the view from R_a is symmetric, the adversary decides which edge R_a will traverse. Assume that R_a moves to node u_1, the system reaches the following configuration: R_a is in node u_1, R_b and R_c are in node u_2, and node u_3 is still not visited. This configuration is indistinguishable with configuration (ii) in Figure 3. (Consequently, every configuration indistinguishable with configuration (ii) in Figure 3 cannot be terminal.) Consider the two following sub-cases (these sub-cases are illustrated in Figure 4):

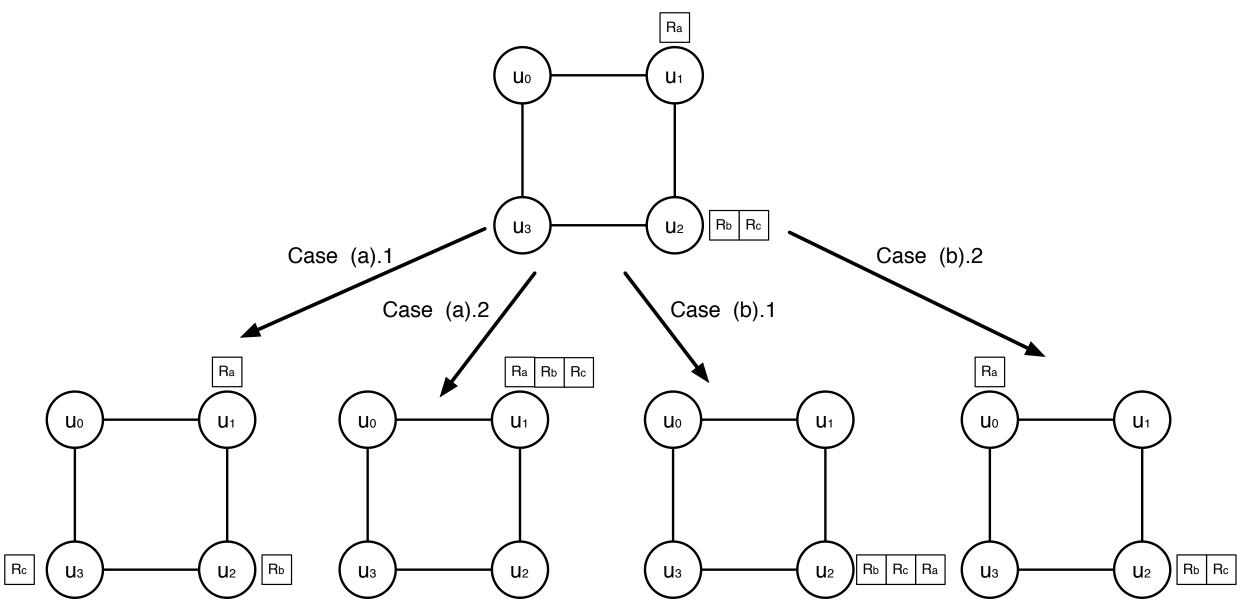

Figure 4: Illustration of sub-cases (a) and (b)

(a) *The probability that R_c (resp. R_b) moves, if it executes a cycle, is strictly positive.*

   1. *Assume that the destination of R_c, if R_c decides to move, is node u_3.* Then, the system reaches a configuration indistinguishable from initial configuration. We can repeat the same scheme infinitely often. So, with a strictly positive probability, the adversary can force the computation to never terminate despite the computation is distributed and fair, a contradiction.
   2. *Assume that the destination of R_c, if R_c decides to move, is node u_1.* Then, the destination of R_b, if R_b moves, is node u_1 too. Hence, there is an admissible computation where R_b and R_c move to node u_1. In this case, the system reaches a configuration that is not distinguishable from configuration (i) in Figure 3 while node u_3 is still not visited. In this case, no configuration can be terminal, a contradiction.

(b) *The probability that R_b (resp. R_c) moves, if it executes a cycle, is 0.* Then, the probability that R_a moves is strictly positive. Consider the two following sub-cases:

   1. *Assume that the destination of R_a, if R_a decides to move, is node u_2.* In this case, there is an admissible computation where R_a moves to node u_2: the system reaches a configuration that is not distinguishable from configuration (i) in Figure 3 while node u_3 is still not visited. In this case, no configuration can be terminal, a contradiction.
   2. *Assume that the destination of R_a, if R_a decides to move, is node u_0.* Assume that the adversary activates R_b, R_c, and then R_a until R_a moves. The probability that R_a eventually moves is 1 and we obtain a configuration that is indistinguishable with configuration γ. We can repeat the same scheme infinitely often. So, with a strictly positive probability, the adversary can force the computation to never terminate despite the computation is distributed and fair, a contradiction.

In all cases, we obtain a contradiction: there is no exploration protocol for three robots on a ring of $n > 4$ nodes and the theorem is proven. □

From Theorems 2 and 3, we can deduce the following corollary:

Corollary 2. *Assuming a distributed fair scheduler, $\forall k, 0 \leq k < 4, \forall n > k$, there is no exploration protocol (even probabilistic) of an n-node ring with k robots.*

4. Positive Result

In this section, we propose a probabilistic protocol for $\boldsymbol{k = 4}$ robots to explore any ring of $\boldsymbol{n > 4}$ nodes. We begin with some definitions in Section 4.1. Then, we present in Section 4.2 the main principles of our protocol. Finally, we prove its correctness in Section 4.3.

4.1. Definitions

Below, we give some definitions to characterize the configurations.

We call *segment* any maximal non-empty elementary path of occupied nodes. The *length of a segment* is the number of nodes that compose it. We call *x-segment* any segment of length x. In the segment $s = u_i, \ldots, u_k$ $(k \geq i)$ the nodes u_i and u_k are termed as the *extremities* of s. An *isolated node* is a node belonging to a 1-segment.

We call *hole* any maximal non-empty elementary path of free nodes. The *length of a hole* is the number of nodes that compose it. We call *x-hole* any hole of length x. In the hole $h = u_i, \ldots, u_k$ $(k \geq i)$ the nodes u_i and u_k are termed as the *extremities* of h. We call *neighbor* of a hole any node that does not belong to the hole but is neighbor of one of its extremities. In this case, we also say that the hole is a *neighboring hole* of the node. By extension, any robot that is located at a neighboring node of a hole is also referred to as a neighbor of the hole.

We call *arrow* a maximal elementary path $u_i, \ldots, u_k$ of length at least four such that (i) u_i and u_k are occupied by one robot, (ii) $\forall j \in [i+1 \ldots k-2]$, u_j is free, and (iii) there is a tower in u_{k-1}, the latter meaning occupied by at least two robots. The node u_i is called the *arrow tail* and the node u_k is called the *arrow head*. The *size* of an arrow is the number of free nodes that compose it, *i.e.*, it is the length of the arrow path minus 3. Note that the minimal size of an arrow is 1 and the maximal size is $n-3$. Note also that when there is an arrow in a configuration, the arrow is unique. An arrow is said to be *primary* if its size is 1. An arrow is said to be *final* if its size is $n-3$.

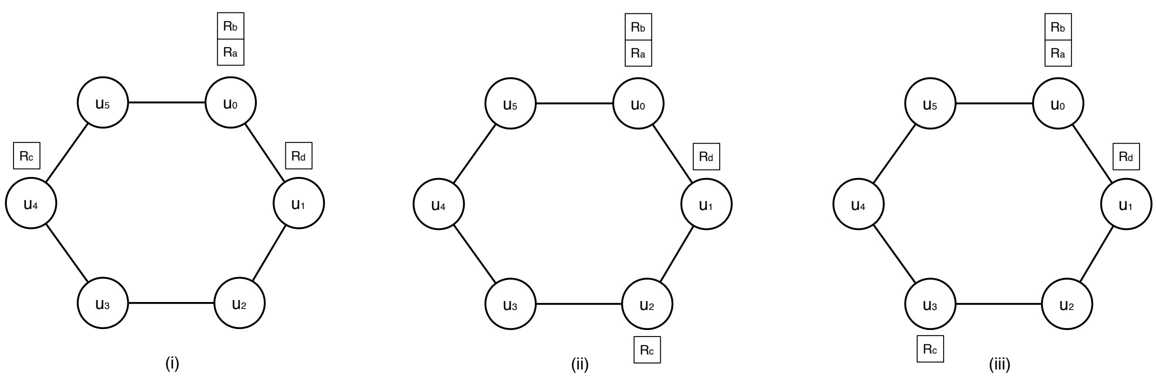

Figure 5: Arrows.

Figure 5 illustrates the notion of arrows: In Configuration (i) the arrow is formed by the path u_4, u_5, u_0, u_1; the arrow is primary; the node u_4 is the tail and the node u_1 is the head. In Configuration (ii), there is a final arrow (the path u_2, u_3, u_4, u_5, u_0, u_1). Finally, the size of the arrow in Configuration (iii) (the path u_3, u_4, u_5, u_0, u_1) is 2.

4.2. Overview of the algorithm

The main algorithm is given in Algorithm 1. To simplify the design, some specific cases are treated in Figures 6-9. These figures can be seen as an automaton:

Algorithm 1 The protocol.

```
if the configuration does not contain a final arrow and is distinguishable from (b) and (d) in Figure 6 then
    begin
    if the configuration contains an arrow then
        begin
        if I am the arrow tail then
            Move toward the arrow head through the hole having me and the arrow head as neighbors;
        end
    else
        if the configuration contains a 4-segment then
            begin
            if I am not located at an extremity of the 4-segment then
                Try to move toward my neighboring node that is not an extremity of the 4-segment;
            end
        else
            if the configuration contains a unique largest segment then   /* A unique 3- or 2- segment */
                begin
                if I am the isolated robot then
                    Move toward the unique largest segment through a smallest hole having me and an extremity of the largest segment as neighbors;
                end
            else
                if the ring-size is 6 then
                    See Figure 6, Configuration (a);
                else
                    if the ring-size is 7 then  /* there are two 2-segments */
                        begin
                        if I am neighbor of the largest hole then
                            Move through my neighboring hole;
                        end
                    else
                        if the ring-size is 8 then
                            begin
                            if there are two 2-segments then
                                See Figure 8, Configurations (a) and (e);
                            else  /* there are four isolated robots */
                                See Figure 9, Configuration (a);
                            end
                        else  /* the ring-size is more than 8 */
                            if the configuration contains (exactly) two 2-segments then
                                begin
                                if I am a neighbor of a largest hole then
                                    Try to move toward the other 2-segment through my neighboring hole;
                                end
                            else  /* the four robots are isolated */
                                begin
                                Let l_max be the length of the largest hole;
                                if every robot is neighbor of an l_max-hole then
                                    Try to move through a neighboring l_max-hole;
                                else
                                    if 3 robots are neighbors of an l_max-hole then
                                        begin
                                        if I am neighbor of only one l_max-hole then
                                            Move toward the robot that is neighbor of no l_max-hole through my shortest neighboring hole;
                                        end
                                    else  /* 2 robots are neighbors of the unique l_max-hole */
                                        if I am neighbor of the unique l_max-hole then
                                            Move through my shortest neighboring hole;
                                end
    end
else
    /* The exploration is terminated */
```

- Configurations are the states of the automaton.
- Bold arrows between configurations represent possible transitions. (More precisely, the transition $\gamma \mapsto \gamma'$ means that a configuration indistinguishable with γ' can be reached from a configuration indistinguishable with γ.)
- States without incoming arrow, except self-loops, are possible initial configuration.
- Below any configuration having no outgoing transition, we explain what robots have to do.
- In each node of each configuration, the symbols $\circ$, $\perp$, or $\top$ give the multiplicity of the node.
- In any configuration, we show how robots must behave using arrows: dashed arrows represent *try to move* actions, and bold arrows represent *(deterministic) moves*. When there are two possible directions for a robot, this means that if the robot is activated by the adversary to execute a cycle, the edge it will traverse is chosen by the adversary.

Except for two special cases where it terminates earlier (namely, Cases (b) and (d) in Figure 6, page 18), our protocol works in three main steps:

- **Alignment (Lines 15-57).** From an initial (towerless) configuration, the robots move along the ring in such a way that (1) they never form any arrow and (2) they eventually form a unique 4-segment with probability one.

  Actually, during this phase, we avoid as much as possible to create any tower.
- **Arrow Creation (Lines 9-13).** From any configuration containing a unique 4-segment, the four robots eventually form a primary arrow with probability one. The 4-segment is maintained until the primary arrow is formed.
- **Exploration (Lines 3-7).** From a configuration where the four robots form a primary arrow, the arrow tail deterministically moves toward the arrow head in such a way that the length of the arrow never decreases. The protocol terminates when robots form a final arrow. At the termination, all nodes have been visited.

Our protocol is probabilistic. As a matter of fact, as long as possible the robots move deterministically. Randomization is used to break the symmetry in some cases: When the system is in a symmetric configuration, the adversary may choose to synchronously activate some robots in such a way that the system stays in a symmetric configuration. To break the symmetry despite the choice of the adversary, some robots proceed as follows: If activated, they probabilistically decide whether or not they move during their Compute phase, that is, they perform a *try to move*.

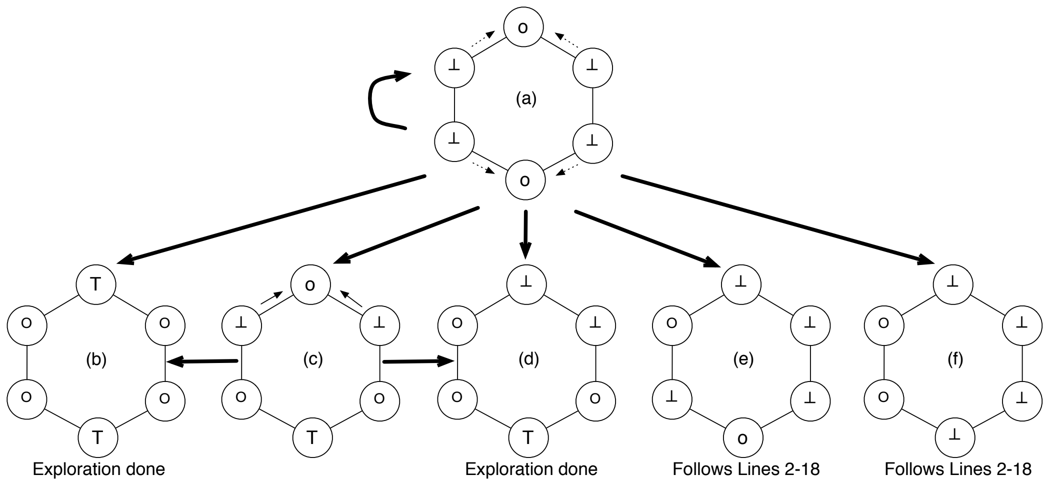

Figure 6: Symmetry breaking in a 6-size ring.

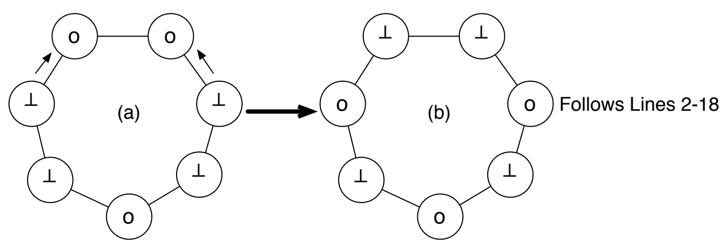

Figure 7: Symmetry breaking in a 7-size ring.

4.3. Correctness

We now show that, starting from any initial (towerless) configuration, Algorithm 1 explores any ring of size $n > 4$ using four robots with probability one. In other words, Algorithm 1 is a probabilistic exploration protocol for four robots in any ring of size $n > 4$.

We start the proof by giving in Section 4.3.1 some properties holding for all ring-sizes. Then, we prove the correctness of the algorithm in any ring of size greater than 8 in Section 4.3.2. In Section 4.3.3 we give dedicated proofs for each of the remaining cases (size 5 to 8). Finally, we provide the general result in Section 4.3.4.

4.3.1. Some Results

We first show several results holding for all ring-sizes. The first result (Lemma 4) is used to prove the correctness of the *alignment* phase. It shows that from some asymmetric towerless configurations the system deterministically converges to a 4-segment without creating any tower during the process.

Lemma 4. *Starting from any (towerless) configuration containing either a* 3*-segment or a unique* 2*-segment, the system reaches in finite time a configuration containing a* 4*-segment without creating any tower during the process.*

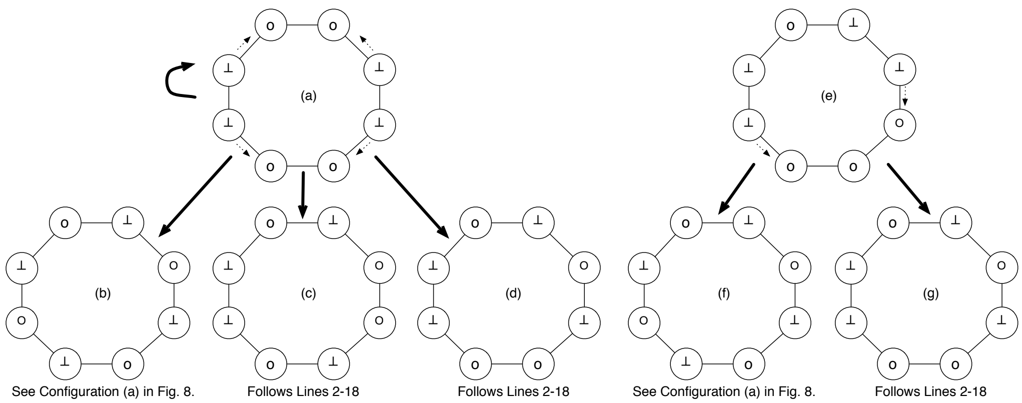

Figure 8: 2-segment symmetries in a 8-size ring.

Proof. Assume a towerless configuration where there is either (1) one 3-segment or (2) a unique 2-segment. Lines 15 to 19 in Algorithm 1 manage these two cases. In the first case, there is one isolated robot and it deterministically moves through its smallest neighboring hole until a 4-segment is formed.[5] In the second case, there are two isolated robots: the isolated robots deterministically move through their neighboring hole having an extremity of the 2-segment as neighbor until 4-segment is formed. Hence, a 4-segment is formed in finite time without creating a tower during the process and the lemma holds. □

The two next lemmas show that the *arrow creation* phase behaves as expected. This phase starts when the system has reached a configuration containing a 4-segment on nodes $u_i, u_{i+1}, u_{i+2}, u_{i+3}$. In this case, Lines 9-13 in Algorithm 1 are executed. Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be the robots located at the nodes u_{i+1} and u_{i+2} of the 4-segment. $\mathcal{R}_1$ and $\mathcal{R}_2$ try to move to u_{i+2} and u_{i+1}, respectively. Eventually only one of these robots moves, a primary arrow is formed on nodes $u_i, u_{i+1}, u_{i+2}, u_{i+3}$, and we obtain the two lemmas below:

Lemma 5. *Let γ be a configuration containing a 4-segment $u_i, u_{i+1}, u_{i+2}, u_{i+3}$. If γ is the configuration at instant t, then the configuration at instant $t+1$ is either identical to γ or the configuration containing the primary arrow u_i, u_{i+1}, u_{i+2}, u_{i+3}.*

Proof. Let $\mathcal{R}_1$ (resp. $\mathcal{R}_2$) be the robot located at node u_{i+1} (resp. u_{i+2}) in γ. In γ, all robots execute Lines 9-13 of Algorithm 1. So, from γ, only $\mathcal{R}_1$ and $\mathcal{R}_2$ can move: $\mathcal{R}_1$ can move to node u_{i+2} and $\mathcal{R}_2$ can move to node u_{i+1}. When one or both of these robots move, we obtain a configuration containing either a primary arrow or a 4-segment in $u_i, u_{i+1}, u_{i+2}, u_{i+3}$ and the lemma holds. □

Lemma 6. *From a configuration containing a 4-segment, the system eventually reaches a configuration containing a primary arrow with probability one.*

[5]Note that the first time the robot moves, its two neighboring holes may have the same length, in this case the adversary decides which edge to traverse.

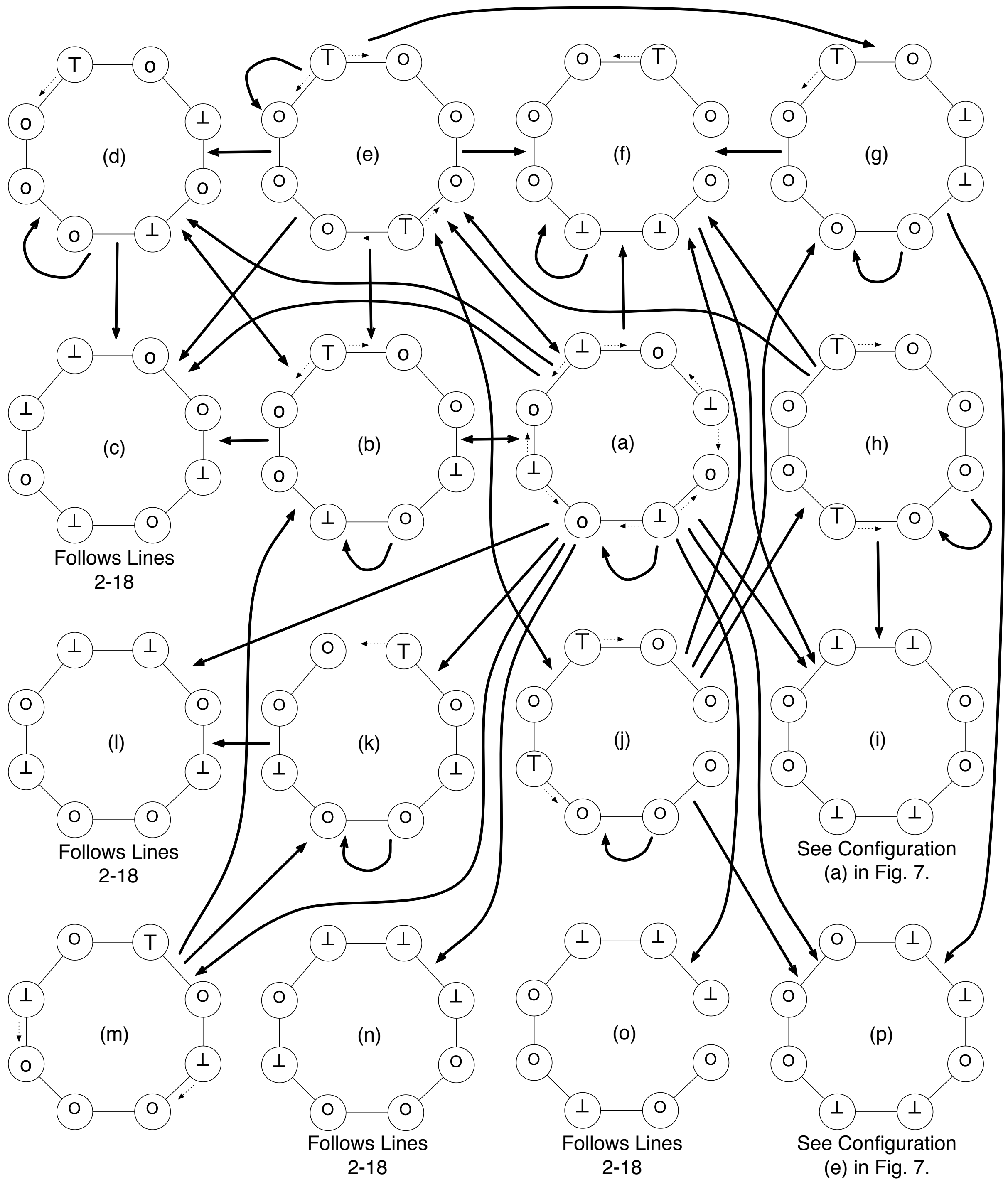

Figure 9: Isolated nodes symmetry in a 8-size ring.

Proof. By Lemma 5, we know that starting from a configuration γ containing a 4-segment, the system either remains in the same configuration or reaches a configuration containing a primary arrow. Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be the robots that are not located at the extremity of the 4-segment in γ. Only $\mathcal{R}_1$ and $\mathcal{R}_2$ can (probabilistically) decide to move in γ (refer to Lines 9-13). Also, by the fairness property, eventually one or both of them are activated. Now, despite the choice of the adversary, there is a strictly positive probability that only one of them probabilistically decides to move: in this case, the system reaches a configuration containing a primary arrow. Hence, considering only the steps where $\mathcal{R}_1$, $\mathcal{R}_2$, or both are activated, the system leaves γ to a configuration γ' containing a primary arrow following a geometric law. Consequently, γ' is eventually reached with probability one and we are done. □

Once the system has reached a configuration containing a primary arrow, robots execute Lines 3-7 in Algorithm 1. From such a configuration, the protocol is fully deterministic: Let $\mathcal{H}$ be the hole between the tail and the head of the primary arrow. We know that all nodes forming the primary arrow are already visited. So, the unvisited nodes can only be on $\mathcal{H}$ and the process just consists in traversing $\mathcal{H}$. To that goal, the robot located at the arrow tail traverses $\mathcal{H}$. When it is done, the system is in a terminal configuration containing a final arrow and all nodes have been visited. Hence, we can conclude with the following lemma:

Lemma 7. *From any configuration containing a* 4*-segment, the system reaches a terminal configuration containing a final arrow with probability one and when it is done, all nodes have been visited.*

Proof. The proof is based on the two following claims:

1. *Any configuration containing a final arrow is terminal.*
   Proof: Immediate, refer to Line 1 in Algorithm 1.
2. *From a configuration containing a non-final arrow of length x, the system eventually reaches a configuration containing a $x+1$-arrow.*
   Proof: In such a configuration, only the arrow tail can move. By the fairness property, the robot located at the arrow tail moves in finite time: it moves through its neighboring hole having the arrow head as other neighbor (refer to Lines 3-7). As a consequence, the size of the arrow is incremented to $x+1$, we are done.

Using the two previous claims, we now prove the lemma in two steps:

- **Termination.** *From any configuration containing a* 4*-segment, the system eventually reaches a terminal configuration containing a final arrow with probability one.*

  Proof: Immediate from Lemma 6, Claims 1 and 2.

- **Partial correctness.** *If a computation that starts from a configuration containing a* 4*-segment terminates, then any node has been visited.*

  Proof: Consider a configuration containing a 4-segment say $u_i, u_{i+1}, u_{i+2}, u_{i+3}$. By Lemmas 5 and 6, from this configuration the system eventually reaches a configuration containing a primary arrow on $u_i, u_{i+1}, u_{i+2}, u_{i+3}$ with probability one and nodes u_i, u_{i+1}, u_{i+2}, and u_{i+3} are already visited. By Claim 2, the robots execute then Lines 3-7 until the computation terminates. Let $\mathcal{H}$ be the path $u_{i-1}, \ldots, u_{i-n+4}$. By Claim 2, until the computation terminated, only the robot located at the arrow tail can move and it moves following $\mathcal{H}$. Hence, when the computation terminates all nodes of $\mathcal{H}$ have been visited (*i.e.*, nodes u_{i-1}, ..., u_{i-n+4}) and, as nodes u_i, u_{i+1}, u_{i+2}, u_{i+3} have also been visited, we are done.

□

4.3.2. Size $\boldsymbol{n > 8}$

Consider any ring of size $n > 8$. We already know that from any configuration containing a 4-segment, the robots perform the exploration as expected by Lemma 7. So, to prove the correctness of Algorithm 1 in such rings, we show that the *alignment phase* (Lines 15-57) works as expected. That is, starting from any towerless configuration, robots eventually form a 4-segment with probability one without creating any arrow during the process. More precisely, we will see here that robots perform this phase without even creating any tower.

Roughly speaking, in any ring of size greater than eight *alignment* phase works as follows: In asymmetric configurations, robots move deterministically (Lines 18, 52, and 56). Conversely, in symmetric configurations, some robots move probabilistically using *try to move* (Lines 41 and 47). Note that in all cases, we prevent the tower creation (and consequently the arrow creation) by applying the following constraint: a robot can move through a neighboring hole $\mathcal{H}$ only if its length is at least 2 or if the other neighboring robot cannot move through $\mathcal{H}$.

To show that starting from any initial (towerless) configuration, robots eventually form a 4-segment with probability one without creating any arrow during the process, we split the study into 3 cases:

- The initial configuration contains a 4-segment. Then, the result trivially holds.
- The initial configuration contains a 3-segment or a unique 2-segment. In this case, the result follows from Lemma 4.
- In either cases, that is the initial configuration contains either two 2-segments or four isolated robots, the result follows from Lemmas 8 and 9, given below.

Lemma 8. *In any ring of size greater than eight, if the configuration γ at instant t contains either two 2-segments or four isolated robots, then the configuration at instant $t+1$ contains no tower.*

Proof. First, note that the robots execute 38-57 in γ. Consider the two following cases:

- γ *contains two* 2*-segments.* In this case, as there are four robots and the size of the ring is greater than 8, the size of the largest hole is at least three. In such a configuration, the only possible moves are the moves where robots go through one of their neighboring holes of length at least three (refer to Line 41). Hence, all moving robots move to a different free node: no tower is created at instant $t + 1$.

- γ *contains four isolated robots.* Let l_{max} be the length of the largest hole in γ. In this case, as there is four robots and the size of the ring n is greater than 8, $l_{max} \geq 2$. Consider then the following three sub-cases:
  - *Every robot is neighbor of an l_{max}-hole.* In this case, every robot can move in the next step (refer to Line 47) but to a neighboring hole of size at least two. So, all moving robots move to a different free node. Hence, no tower is created at instant $t + 1$.
  - *Three robots are neighbors of an l_{max}-hole.* Let $\mathcal{R}$ be the robot that is not neighbor of any l_{max}-hole. In this case, the robots that may move (at most two) go through their neighboring hole having $\mathcal{R}$ as other neighbor (refer to Line 52). As $\mathcal{R}$ cannot move, no tower is created at instant $t + 1$.
  - *Two robots, say $\mathcal{R}_1$ and $\mathcal{R}_2$, are neighbors of the unique l_{max}-hole.* In this case, only $\mathcal{R}_1$ and $\mathcal{R}_2$ can move. If $\mathcal{R}_1$ (resp. $\mathcal{R}_2$) moves, then $\mathcal{R}_1$ (resp. $\mathcal{R}_2$) moves through its neighboring hole having not $\mathcal{R}_2$ (resp. $\mathcal{R}_1$) as other neighbor (refer to Line 56). So, all moving robots move to a different free node. As a consequence, no tower is created at instant $t + 1$.

In all cases, the configuration obtained at instant $t + 1$ contains no tower and the lemma holds. $\square$

Lemma 9. *Starting from any configuration containing either two 2-segments or four isolated robots on a ring of size greater than eight, the system eventually reaches a configuration containing a 4-segment with probability one.*

Proof. From any configuration containing either two 2-segments or four isolated robots, we know that the system remains in configurations containing no tower while the system does not reach a configuration containing a 4-segment by Lemmas 4 and 8. Moreover, if the system reaches a configuration containing either 3-segment or a unique 2-segment, then we can conclude by Lemma 4.

For a given n-size ring network, the number of configurations is *finite*. So, to prove the lemma, we have to show that from any configuration containing either two 2-segments or four isolated robots, there is always a strictly positive probability that the system eventually reaches a configuration containing either a 4-segment, or a 3-segment or a unique 2-segment (despite the choices of the adversary). To see this, consider a configuration γ satisfying one of the following cases:

1. γ *contains two 2-segments.* In this case, the robots that are neighbors of a largest hole (at least two) can *try* to move (refer to Line 41). So, by fairness property, a non-empty set of these robots, say S, is eventually activated by the adversary to execute a cycle.

Now, every robot in S probabilistically decides to move or not. So, there is a strictly positive probability that only one robot in S decides to move. In this case, we obtain a unique 2-segment and we are done.

2. γ *contains four isolated nodes.* Let l_{max} be the length of the largest hole in γ. Let us study the following sub-cases:
   (a) *Only two robots are neighbors of an* l_{max}*-hole.* In this case, the two robots that are neighbors of the unique l_{max}-hole can move (refer to Line 56). So, by fairness property, either one or both of them eventually move through their shortest neighboring hole. After such moves, either (i) the system is still in a configuration containing four isolated nodes and where two robots are neighbors of a unique largest hole but the size of the largest hole increased, or (ii) the system is in a configuration containing a unique 2-segment, or (iii) the system is in a configuration containing two 2-segments. Hence, the system reaches in finite time a configuration satisfying (ii) or (iii) and we are done.
   (b) *Exactly three robots are neighbors of an* l_{max}*-hole.* Let $\mathcal{R}_0$ be the robot that is not neighbor of any l_{max}-hole. Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be the two robots that are neighbors of exactly one l_{max}-hole. In this case, only $\mathcal{R}_1$ and $\mathcal{R}_2$ can move (refer to Line 52) and by fairness property at least one of them eventually does. If only one of them moves, then we obtain sub-case 2(a) or a unique 2-segment, and we are done. If both $\mathcal{R}_1$ and $\mathcal{R}_2$ move, then the system reaches either (i) a configuration where exactly three robots are neighbors of a largest hole of length $l_{max}+1$, or (ii) a configuration containing a unique 2-segment, or (iii) a configuration containing a 3-segment. If we repeat the argument, we eventually leave Case (i) to sub-case 2(a), (ii), or (iii), and we are done.
   (c) *The four robots are neighbors of an* l_{max}*-hole.* In this case, all activated robots *try* to move (refer to Line 47). Now, despite the choice of the adversary, there is a strictly positive probability that only one robot probabilistically decides to move. In this case, the robot moves through one of its neighboring l_{max}-hole of size at least two (to avoid any tower creation). As a consequence, we obtain sub-cases 2(a) or 2(b), and we are done.

□

By Lemmas 7, 4, 8, and 9, it follows:

Theorem 4. *Algorithm 1 is a probabilistic exploration protocol for* $\mathbf{4}$ *robots on a ring of* $\boldsymbol{n > 8}$ *nodes.*

4.3.3. Particular Cases

We now consider rings of size 5 to 8. The correctness for a ring of size 5 is straightforward because any initial configuration of a ring of size 5 contains a 4-segment. Then, any initial configuration in rings of size 6 to 8 matches one of the following cases: (1) the configuration contains a 4-segment; (2) the configuration contains a 3-segment and one isolated node; (3)

the configuration contains a 2-segment and two isolated nodes; (4) the configuration contains two 2-segments; (5) the configuration contains four isolated nodes.

In the three first cases, Lines 3-19 are executed and the correctness is obtained by Lemmas 4 and 7. Finally, note that case (4) is possible for size 6, 7, and 8 while case (5) is only possible on a ring of size 8.

Size 5. Any initial configuration of a ring of size 5 contains a 4-segment. So, by Lemma 7, we can conclude:

Theorem 5. *Algorithm 1 is a probabilistic exploration protocol for* **4** *robots on a ring of* **5** *nodes.*

Size 6. To show the correctness of the protocol in any ring of size 6, it remains to show that it correctly operates when the initial configuration contains two 2-segments. This configuration is indistinguishable with Configuration (a) in Figure 6, page 18. In Configuration (a), any robot tries to move toward its neighboring hole (dashed arrows). So, either the system stays in the same configuration or the system reaches Configuration (b), (c), (d), (e), or (f). However, with probability one, the system eventually leaves Configuration (a) to Configuration (b), (c), (d), (e), or (f). In Configuration (e) or (f), we retrieve a previous case, the robots execute Lines 3-19 in Algorithm 1. In cases (b) and (d), we have the guarantee that all nodes are visited and as configurations (b) and (d) cannot be obtained anywhere else, there is no ambiguity and the process can stop. In Configuration (c), the two isolated nodes move as shown by the bold arrow and the system reaches either Configuration (b) or Configuration (d). Once again, we have the guarantee that all nodes are visited and as configurations (b) and (d) cannot be obtained anywhere else, there is no ambiguity and the process can stop. So, we can conclude with the following theorem:

Theorem 6. *Algorithm 1 is a probabilistic exploration protocol for* **4** *robots on a ring of* **6** *nodes.*

Size 7. To show the correctness of the protocol in any ring of size 7, it remains to show that it correctly operates when the initial configuration contains two 2-segments. Such a configuration is indistinguishable with Configuration (a) in Figure 7, page 18. In this case, robots execute Lines 24-28 in Algorithm 1 and the system reaches in one step a configuration indistinguishable with configuration (b) in Figure 7, *i.e.*, the configuration contains one 2-segment and two isolated nodes. From that point, robots execute Lines 3-19 in Algorithm 1 and by Lemmas 4 and 7, we have:

Theorem 7. *Algorithm 1 is a probabilistic exploration protocol for* **4** *robots on a ring of* **7** *nodes.*

8(a)	$\mapsto$	8(d)				
8(b) = 9(a)	$\mapsto$	9(c)				
8(c)						
8(d)						
8(e)	$\mapsto$	8(g)				
8(f) = 9(a)	$\mapsto$	9(c)				
8(g)						
9(a)	$\mapsto$	9(c)				
9(b)	$\mapsto$	9(c)				
9(c)						
9(d)	$\mapsto$	9(c)				
9(e)	$\mapsto$	9(f)	$\mapsto$	9(i) = 8(a)	$\mapsto$	8(d)
9(f)	$\mapsto$	9(i) = 8(a)	$\mapsto$	8(d)		
9(g)	$\mapsto$	9(p) = 8(e)	$\mapsto$	8(g)		
9(h)	$\mapsto$	9(f)	$\mapsto$	9(i) = 8(a)	$\mapsto$	8(d)
9(i) = 8(a)	$\mapsto$	8(d)				
9(j)	$\mapsto$	9(g)	$\mapsto$	9(p) = 8(e)	$\mapsto$	8(g)
9(k)	$\mapsto$	9(l)				
9(l)						
9(m)	$\mapsto$	9(k)	$\mapsto$	9(l)		
9(n)						
9(o)						
9(p) = 8(e)	$\mapsto$	8(g)				

Table 1: Probabilistic Convergence to a configuration in $\mathcal{C}_{good}$.

Size 8. To show the correctness of the protocol in any ring of size 8, it remains to show that it correctly operates when the initial configuration contains either two 2-segments or four isolated nodes. Figures 8 and 9 (pages 19 and 20) describe the behavior of our protocol starting from a configuration that contains two 2-segments and four isolated nodes, respectively. Any configuration that contains either two 2-segments or four isolated nodes on a ring of size 8 is indistinguishable with Configurations (a), (e) in Figure 8, or Configuration (a) in Figure 9.

First, we can observe that there is no ambiguity between the process described in Figures 8 and 9 and the rest of the protocol. We can then remark that starting from Configurations (a), (e) in Figure 8, or Configuration (a) in Figure 9, the system leaves configurations of Figures 8 and 9 only when the system reaches a configuration containing either a 3-segment and one isolated node or a 2-segment and two isolated nodes: Configurations (c), (d), and (g) in Figure 8 as well as Configurations (c), (l), (n), (o) in Figure 9. Let $\mathcal{C}_{good}$ be the set of all these configurations.

From any configuration in $\mathcal{C}_{good}$, robots execute Lines 3-19 in Algorithm 1 and by Lemmas 4 and 7, the exploration is achieved with probability one.

Consider now a configuration γ in Figures 8 or 9 that is not in $\mathcal{C}_{good}$. In any configuration γ, there is at least one robot that executes a *try to move* if activated and every robot either stays idle or executes *try to move* if activated. Now, the scheduler being fair, in any configuration the adversary eventually chooses to activate robots that execute a *try to move*, and in that case, there is a strictly positive probability that only one robot moves despite the choice of the adversary. We can then remark (refer to Table 1) that from γ, there is path that leads to a configuration of $\mathcal{C}_{good}$ and any transition in this path is possible with a strictly positive probability: these transitions actually correspond to steps where exactly one robot moves. So, as the set of configurations in Figures 8 or 9 is finite, a configuration of $\mathcal{C}_{good}$ is eventually reached with probability one and we can conclude:

Theorem 8. *Algorithm 1 is a probabilistic exploration protocol for* $\mathbf{4}$ *robots on a ring of* $\mathbf{8}$ *nodes.*

4.3.4. General Result

By Theorems 4 to 8, it follows:

Theorem 9. *Algorithm 1 is a probabilistic exploration protocol for* $\mathbf{4}$ *robots on a ring of* $\boldsymbol{n}$ *nodes with* $\boldsymbol{n} > \mathbf{4}$.

5. Conclusion

We addressed the problem of exploring a discrete environment by a team of autonomous, oblivious, and mobile robots. One of the main challenges with such a distributed system is to overcome the weakness of the model by itself, mainly (i) the fact that the robots cannot remember past actions or positions and (ii) the lack of means to particularize robots or vertices, or to give orientation. In particular, the fact that robots need to stop after exploring all locations requires robots to find an implicit way to "remember" how much of the graph was explored, *i.e.*, to be able to distinguish between various stages of the exploration process since robots have no persistent memory. As configurations can be distinguished only by robot positions, the main complexity measure is then the number of robots that are needed to explore a given graph. The vast number of symmetric situations induces a large number of required robots.

We considered a semi-synchronous model of computation. In this model, we shown that for the exploration problem in uniform rings, randomization can shift complexity from $\Theta(\log n)$ to $\Theta(1)$ robots, since we proved that four probabilistic oblivious robots are necessary and sufficient to solve the problem.

Applying randomization to other problem instances is an interesting topic for further research. Then, an immediate open question raised by our work is the following. Our protocol is optimal with respect to the number of robots. However, the exploring time is only proven to be finite. We observed an average exploration time of $O(n)$ moves by making simulations. Computing the expected exploration time from our proof argument is feasible, however it would be more interesting to study the impact of the number of robots on the time complexity, since it seems natural that more robots should explore the ring faster.

References

[1] S. Devismes, F. Petit, S. Tixeuil, Optimal probabilistic ring exploration by asynchronous oblivious robots, in: Proceedings of the 16th International Colloquium on Structural Information and Communication Complexity (SIROCCO), Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2009, pp. 195–208.

[2] S. Devismes, Optimal exploration of small rings, in: Proceedings of the Third International Workshop on Reliability, Availability, and Security (WRAS), ACM, 2010, pp. 9:1–9:6.

[3] M. Potop-Butucaru, M. Raynal, S. Tixeuil, Distributed computing with mobile robots: an introductory survey, in: Proceedings of the 14th International Conference on Network-Based Information Systems (NBIS), IEEE Computer Society, Tirana, Albania, 2011, pp. 318–324.

[4] Y. Asahiro, S. Fujita, I. Suzuki, M. Yamashita, A self-stabilizing marching algorithm for a group of oblivious robots, in: Proceedings of 12th International Conference on Principles of Distributed Systems (OPODIS), Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2008, pp. 125–144.

[5] Y. Dieudonné, F. Petit, Scatter of weak mobile robots, Parallel Processing Letters 19 (1) (2009) 175–184.

[6] Y. Dieudonné, O. Labbani-Igbida, F. Petit, Circle formation of weak robots, ACM Transactions on Adaptive and Autonomous Systems (TAAS) 3 (4) (2008) 16:1–16:20.

[7] P. Flocchini, G. Prencipe, N. Santoro, P. Widmayer, Arbitrary pattern formation by asynchronous, anonymous, oblivious robots, Theor. Comput. Sci. 407 (1-3) (2008) 412–447.

[8] S. Souissi, X. Défago, M. Yamashita, Using eventually consistent compasses to gather memory-less mobile robots with limited visibility, ACM Transactions on Adaptive and Autonomous Systems (TAAS) 4 (1) (2009) 9:1–9:27.

[9] I. Suzuki, M. Yamashita, Distributed anonymous mobile robots: Formation of geometric patterns, SIAM J. Comput. 28 (4) (1999) 1347–1363.

[10] H. Ando, Y. Oasa, I. Suzuki, M. Yamashita, Distributed memoryless point convergence algorithm for mobile robots with limited visibility, IEEE Transactions on Robotics and Automation 15(5) (1999) 818–828.

[11] P. Flocchini, G. Prencipe, N. Santoro, P. Widmayer, Gathering of asynchronous robots with limited visibility, Theor. Comput. Sci. 337 (1-3) (2005) 147–168.

[12] Z. Bouzid, M. G. Potop-Butucaru, S. Tixeuil, Optimal byzantine-resilient convergence in unidimensional robot networks, Theoretical Computer Science (TCS) 411 (34-36) (2010) 3154–3168.

[13] P. Flocchini, D. Ilcinkas, A. Pelc, N. Santoro, Computing without communicating: Ring exploration by asynchronous oblivious robots, in: Proceedings of the 11th International Conference on Principles of Distributed Systems (OPODIS), Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2007, pp. 105–118.

[14] P. Flocchini, D. Ilcinkas, A. Pelc, N. Santoro, Remembering without memory: Tree exploration by asynchronous oblivious robots, Theor. Comput. Sci. 411 (14-15) (2010) 1583–1598.

[15] A. Lamani, M. Potop-Butucaru, S. Tixeuil, Optimal deterministic ring exploration with oblivious asynchronous robots, in: Proceedings of 17th International Colloquium on Structural Information and Communication Complexity (SIROCCO), Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2010, pp. 183–196.

[16] J. Chalopin, P. Flocchini, B. Mans, N. Santoro, Network exploration by silent and oblivious robots, in: Proceedings of 36th International Workshop on Graph Theoretic Concepts in Computer Science (WG), Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2010, pp. 208–219.

[17] P. Flocchini, D. Ilcinkas, A. Pelc, N. Santoro, How many oblivious robots can explore a line, Inf. Process. Lett. 111 (20) (2011) 1027–1031.

[18] R. Klasing, E. Markou, A. Pelc, Gathering asynchronous oblivious mobile robots in a ring, Theor. Comput. Sci. 390 (1) (2008) 27–39.

[19] R. Klasing, A. Kosowski, A. Navarra, Taking advantage of symmetries: Gathering of many asynchronous oblivious robots on a ring, Theor. Comput. Sci. 411 (34-36) (2010) 3235–3246.

[20] T. Izumi, T. Izumi, S. Kamei, F. Ooshita, Mobile robots gathering algorithm with local weak multiplicity in rings, in: Proceedings of the 17th International Colloquium on Structural Information and Communication Complexity (SIROCCO), Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2010, pp. 101–113.

[21] G. D'Angelo, G. D. Stefano, A. Navarra, Gathering of six robots on anonymous symmetric rings, in: Proceedings of 18th International Colloquium on the Structural Information and Communication Complexity (SIROCCO), Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2011, pp. 174–185.

[22] S. Kamei, A. Lamani, F. Ooshita, S. Tixeuil, Asynchronous mobile robot gathering from symmetric configurations without global multiplicity detection, in: A. Kosowski, M. Yamashita (Eds.), Proceedings of the 18th International Colloquium on Structural

Information and Communication Complexity (SIROCCO), Lecture Notes in Computer Science, Springer Berlin / Heidelberg, Gdansk, Poland, 2011, pp. 150–161.

[23] A. Efrima, D. Peleg, Distributed algorithms for partitioning a swarm of autonomous mobile robots, Theor. Comput. Sci. 410 (14) (2009) 1355–1368.

[24] G. Prencipe, Instantaneous actions vs. full asynchronicity : Controlling and coordinating a set of autonomous mobile robots, in: Proceedings of the 7th Italian Conference on Theoretical Computer Science, 2001, pp. 154–171.