Optimal torus exploration by oblivious robots

Stéphane Devismes, Anissa Lamani, Franck Petit, Sébastien Tixeuil

Optimal Torus Exploration by Oblivious Robots

Stéphane Devismes[1] Anissa Lamani[2] Franck Petit[3]
Sébastien Tixeuil[3]

[1] VERIMAG, Université Grenoble Alpes, France
[2] Kyushu University, Fukuoka, Japan
[3] LIP6, UPMC Sorbonne Universités, France

Abstract

We deal with a team of autonomous robots that are endowed with motion actuators and visibility sensors. Those robots are weak and evolve in a discrete environment. By weak, we mean that they are anonymous, uniform, unable to explicitly communicate, and oblivious. We first show that it is impossible to solve the terminating exploration of a simple torus of arbitrary size with less than 4 or 5 such robots, respectively depending on whether the algorithm is probabilistic or deterministic. Next, we propose in the SSYNC model a probabilistic solution for the terminating exploration of torus-shaped networks of size $\ell \times L$, where $7 \leq \ell \leq L$, by a team of 4 such weak robots. So, this algorithm is optimal *w.r.t.* the number of robots.
Keywords: Robots, Torus, Exploration, Obliviouness.

1 Introduction

We consider autonomous robots that are endowed with motion actuators and visibility sensors, but otherwise unable to communicate. They evolve in a discrete environment, *i.e.*, the space is partitioned into a finite number of locations, represented by a graph, where the nodes represent the possible locations that a robot can take and the edges the possibility for a robot to move from one location to another. Those robots must collaborate to solve a collective task despite being limited with respect to inputs from the environment, asymmetry, memory, *etc*. In particular, the robots we consider are anonymous, uniform, yet they can sense their environment and take decisions according to their own ego-centered view. In addition, they are oblivious, *i.e.*, they do not remember their past actions. Robots operate in *cycles* that include three phases: *Look*, *Compute*, and *Move* (L-C-M). The Look phase consists in taking a snapshot of the other robots positions using visibility sensors. During the Compute phase, a robot computes a destination based on the previous observation. The Move phase simply consists in moving to the destination using motion actuators. In this context, typical problems are *terminating exploration* [3, 6, 7, 8, 9], *exclusive perpetual exploration* [1, 4], *exclusive searching* [5, 4], and *gathering* [4, 11]. We address the *terminating exploration* (or simply *exploration*) problem, which requires that robots initially placed at different nodes, collectively explore the whole graph and stop upon completion. We focus on the case where the network is an *anonymous unoriented torus* (or simply *torus*). The terms *anonymous* and *unoriented* mean that no robot has access to any kind of device allowing to identify nodes or to determine any (global or local) direction, such as North-South/East-West.

A question naturally arises: "*Why addressing an abstract topology such as torus?*" To answer this question, we emphasize that robots are unable to communicate explicitly and have no persistent memory. So, they are unable to remember the various steps taken before. Therefore, the positions of the other robots are the only way to distinguish the different stages of the exploration process. Torus belongs to the class of *regular* graphs. Such graphs are of particular interest because they are topologies for which the symmetry of configurations with respect to robot positions is the most frequently observed, making the exploration problem hard to solve. So far, ring-shaped network is the only *regular* topology that has been studied [12, 9, 7]. As a result, an immediate question arises: "*Does the increase of the number of possible symmetries in the network (due to increasing dimensions) make the problem harder to solve?*" Terminating exploration has been also studied in other topologies than rings, namely the tree [8] and grid [6]. However, none of them are regular networks. Torus can be seen as a 2-dimensional ring. Compared to the ring, the difficulty lies in the

additional axes of symmetry. So, it appears to be the most natural candidate among regular graphs to study the impact of strong topological symmetry on the complexity to solve the problem. Furthermore, as previously stated, the exploration (with stop) process is intrinsically related to the ability to differentiate consecutive phases of the exploration. More possible symmetries hint that more robots than in rings are required to complete exploration: As robots have no way to distinguish and agree on some kind of orientation, *e.g.*, North-South/East-West, somehow the current robot configuration has to encode consistent information so that robots agree on both axes. Since numerous symmetric configurations induce a large number of required robots, minimizing the number of robots turns out to be a difficult problem.

Related Work. With respect to the (terminating) exploration problem, minimizing the number of robots for exploring particular classes of graphs led to contrasted results. The only result available for exploration in general graphs [3] considers that edges are labeled in such a way that the network configuration is asymmetric. In this extended model, three robots are not sufficient to explore all asymmetric configurations, and four robots are sufficient to explore all asymmetric configurations. Note that exploring the set of asymmetric configurations is strictly weaker than exploring the complete underlying graph, especially when the graph is highly symmetric. The rest of the literature is thus dedicated to a weaker model, where edges are not labeled. One extreme case in this weak model is the set of tree-shaped networks, as in general, $\Omega(n)$ robots are necessary and sufficient to explore a tree network of n nodes deterministically [8]. The other extreme case is the set of grid-shaped networks [6], where three robots are necessary and sufficient to explore deterministically any grid of at least three nodes (except for the grids of size 2×2 and 3×3, where four – respectively five – robots are necessary and sufficient). However, this result is mainly due to the fact that grids are not regular graphs: they contain nodes of degrees 2, 3, and 4. So, this topological property implies less symmetries.

By contrast, rings and tori are regular graphs, and consequently more intricate. So far, no research explored the feasibility of exploring a torus-shaped network with a team of k robots. In ring-shaped networks [9], the fact that the number k of robots and the ring size n must be coprime yields to the lower bound $\Omega(\log n)$ on the number of robots required to explore a n-size ring. Indeed, the smallest non-divisor of n evolves as $\log n$ in the worst case. However, notice that Lamani *et al.* also provide in [12] an algorithm that allows 5 robots to deterministically explore any ring whose size is coprime with 5. The large number of robots and the constraint on the ratio between the number of robots and the ring size induced by the deterministic setting in ring-shaped networks hinted at a possible more efficient solutions when robots can make use of probabilities [7]. As a matter of fact, four robots are necessary and sufficient to probabilistically explore any ring of size at least four. While the gain in going probabilistic is only one robot when n is not divisible by 5, a logarithmic factor is obtained in the general case. Aforementioned deterministic solutions operate in the model ASYNC [10] which assumes that robots execute L-C-M asynchronously, *i.e.*, in a fully independent manner. However, It is shown in [7] that randomization does not help in ASYNC, *i.e.*, there exists a scheduling such that random choices are all nullified. So, probabilistic algorithms require more synchronization, typically the model SSYNC (*semi-synchronous*) [10], in which robots are asynchronously activated to perform cycles, yet at each activation, a robot executes one L-C-M cycle atomically.

Contribution. We propose an optimal (*w.r.t.* the number of robots) solution for the terminating exploration of torus-shaped networks by a team of k robots. Precisely, we first show the impossibility to explore a simple torus of arbitrary size with less than four robots, even if the algorithm is probabilistic. If the algorithm is required to be deterministic, four robots are also insufficient. This negative result implies that the only way to obtain an optimal algorithm is to make use of randomization, and thus, within SSYNC [7]. We then propose a probabilistic algorithm designed in SSYNC that uses four robots to explore all tori of size $\ell \times L$, where $7 \leq \ell \leq L$. Hence, in such tori, four robots are necessary and sufficient to solve the probabilistic terminating exploration. As a torus can be seen as a 2-dimensional ring, our result shows, perhaps surprisingly, that increasing the number of possible symmetries in the network (due to increasing dimensions) does not necessarily bring an extra cost with respect to the number of robots that are necessary to solve the problem.

Roadmap. Section 2 defines the model and the exploration problem. Lower bounds are shown in Section 3. Our algorithm is presented in Section 4 and proven in Section 5. We conclude in Section 6.

2 Preliminaries

We consider teams of k autonomous mobile entities called *robots* evolving in a *simple undirected connected graph* $G = (V, E)$, where V is a finite set of n nodes and E a finite set of edges. Nodes represent locations that robots can take and edges represent the possibility for a robot to move from one location to another. A node is said to be *occupied* if at least one robot is located on it, otherwise the node is said to be *free*. If a node u is occupied by $x > 1$ robots, we say that u *contains a tower* (of x robots). Two nodes u and v are *neighbors* in G *iff* $\{u, v\} \in E$. We assume that G is an (ℓ, L)-*Torus* (or a *Torus*, for short), where ℓ and L are two positive integers, *i.e.*, G satisfies the following two conditions: (i) $n = \ell \times L$ and (ii) there exists an order $v_1, \ldots, v_n$ on the nodes of V such that $\forall i \in [1..n]$:

- If $i + \ell \leq n$, then $\{i, i + \ell\} \in E$, else $\{i, (i + \ell) \bmod n\} \in E$.
- If $i \bmod \ell \neq 0$, then $\{i, i + 1\} \in E$, else $\{i, i - \ell + 1\} \in E$.

Given the previous order $v_1, \ldots, v_n$, for every $i \in [0..(L-1)]$, the sequence $v_{1+i\times\ell}$, $v_{2+i\times\ell}, \ldots, v_{\ell+i\times\ell}$ is called an ℓ-ring. Similarly, for every $j \in [1..\ell]$, $v_j, v_{j+\ell}$, $v_{j+2\times\ell}$, $\ldots, v_{j+(L-1)\times\ell}$ is called an L-ring. Note that when $\ell = L$, any ℓ-ring is also an L-ring and conversely. More generally, we use the term *ring* to arbitrarily designate an ℓ-ring or an L-ring. Nodes are anonymous. Moreover, given two neighboring nodes u and v, there is no explicit or implicit labeling allowing robots to determine whether u is either on the left, on the right, above, or below v. However, for the purpose of explanations, we may use indices for nodes or robots. An *isomorphism* of graphs G and H is a bijection f between the vertex sets of G and H such that any two nodes u and v of G are neighbors in G *iff* $f(u)$ and $f(v)$ are neighbors in H. When G and H are one and the same graph, f is called an *automorphism* of G. An (ℓ, L)-*Torus* and an (L, ℓ)-*Torus* are isomorphic. Hence, as nodes are anonymous, an (ℓ, L)-*Torus* cannot be distinguished from an (L, ℓ)-*Torus*. So, without loss of generality, we will always consider (ℓ, L)-*Tori*, where $\ell \leq L$.

Remark 2.1 *Since an* (ℓ, L)*-torus is a simple graph, every node has four* distinct *neighbors, and consequently we have:* $3 \leq \ell \leq L$ *and* $n = \ell \times L \geq 9$.

The robots do not communicate in an explicit way; however they see the positions of all other robots in their ego-centered coordinate system. Each robot operates according to its (local) *program*. We call (*distributed*) *algorithm* a collection of k *programs*, each one operating on a single robot. Robots are *uniform* and *anonymous*, *i.e.*, they all have the same program using no parameter allowing to differentiate them. We assume that robots cannot remember any previous observation or computation. Such robots are called *oblivious*. The program of a robot consists in executing *Look-Compute-Move cycles* infinitely many times. That is, a robot $\mathcal{R}$ first observes its environment (Look phase). Based on its observation, $\mathcal{R}$ then (probabilistically or deterministically) decides to move or stay idle (Compute phase). If $\mathcal{R}$ decides to move, it moves toward its destination during the Move phase. During the Compute phase, the decision between moving or staying idle is either deterministic or probabilistic. In the latter case, $\mathcal{R}$ decides between moving and staying idle using some fixed probability $p \in (0, 1)$, and we say that $\mathcal{R}$ *tries to move*.

We consider the semi-synchronous model (SSYNC), where time is represented by an infinite sequence of instants 0, 1, 2, ... No robot has access to this global time. At each instant, a non-empty subset of robots is *activated*. Every robot that is activated at instant t *atomically* executes a full cycle between t and $t + 1$. Activations are determined by an *adversary*. Remark that, in this model, any robot performing a Look operation sees all other robots on nodes and not on edges.

We assume that during the Look phase, every robot can perceive whether several robots are located on the same node. This ability is called *(global) multiplicity detection*. We shall indicate by $d_i(t)$ the multiplicity of robots present in node v_i at instant t. We consider two versions of multiplicity detection: the *strong* and *weak* multiplicity detections. Under the *weak* multiplicity detection, for every node v_i, d_i is a function $\mathbb{N} \mapsto \{\circ, \bot, \top\}$ defined as follows: $d_i(t)$ is equal to either $\circ$, $\bot$, or $\top$ according to v_i contains none, one or several robots at instant t. Under the *strong* multiplicity detection, for every node v_i, d_i is a function $\mathbb{N} \mapsto \mathbb{N}$, where $d_i(t) = x$ indicates that there are x robots in node v_i at instant t.

To define the notion of *configuration*,we use an arbitrary order $\prec$ on nodes. The system being anonymous, robots do not know this order. Let $v_1, \ldots, v_n$ be the list of the nodes in G ordered by $\prec$. The configuration at instant t is $d_1(t), \ldots, d_n(t)$. We denote by *initial configurations* the configurations from which the system can start at instant 0. Every configuration from which no robot moves or tries to move if activated is said to be *terminal*. Two configurations $d_1, \ldots, d_n$ and $d'_1, \ldots, d'_n$ are *indistinguishable* (resp., *distinguishable* otherwise) *iff* there exists an automorphism on G, $f : V \mapsto V$ such that $\forall i \in \{1, \ldots, n\}$, $d_i = d'_j$ where $v_j = f(v_i)$.

The *view* of robot $\mathcal{R}$ at instant t is a labeled graph isomorphic to G, where every node v_i is labeled by $d_i(t)$, except the node where $\mathcal{R}$ is currently located, this latter node v_j is labeled by $d_j(t), \downarrow$. (Indeed, the coordinate system is ego-centered.) Hence, from its view, a robot can compute the view of each other robot, and decide whether some other robots have the same view as its own. The views $\mathcal{V}$ and $\mathcal{V}'$ are *identical iff* there exists an isomorphism f of $\mathcal{V}$ and $\mathcal{V}'$ such that every node v of $\mathcal{V}$ has the same label in $\mathcal{V}$ as node $f(v)$ in $\mathcal{V}'$. Every decision to move is based on the view obtained during the last Look action. However, it may happen that some edges incident to a node v currently occupied by the deciding robot look identical in its view, *i.e.*, v lies on a symmetric axis of its view. In this case, if the robot decides to take one of these edges, it may take any of them. We assume the worst-case decision in such cases, *i.e.*, the actual edge among the identically looking ones is chosen by the adversary. More generally, if several possible destinations are selected during the Compute phase, the final destination is also chosen by the adversary.

A *scheduling* is a list of activation's choices that can be made by the adversary, *i.e.*, a scheduling is any infinite list of non-empty subset of robots $\sigma_0, \sigma_1, \ldots$, where $\forall i \geq 0$, σ_i is the set of robots activated at instant i. An infinite list of configurations $\gamma_0, \gamma_1, \ldots$ can be *generated* from the scheduling $\sigma_0, \sigma_1, \ldots$ *iff* $\forall i \geq 0$, γ_{i+1} can be obtained from γ_i after each robot in σ_i is activated at instant i to atomically perform a cycle (in this case, $\gamma_i\gamma_{i+1}$ is *step*). We call *execution* any infinite list of configurations $\gamma_0, \gamma_1, \ldots$ that can be *generated* from an arbitrary scheduling and such that γ_0 is a possible initial configuration. An execution e *terminates* if e contains a terminal configuration. We restrict the power of the adversary by assuming that schedulings are *fair*: a scheduling $\sigma_0, \sigma_1, \ldots$ is *fair iff* for every robot $\mathcal{R}$, for every instant i, there exists an instant $j \geq i$ such that $\mathcal{R} \in \sigma_j$. An execution e is *fair iff* e can be generated by a fair scheduling. A particular case of fair scheduling is the *sequential* fair scheduling: a scheduling $\sigma_0, \sigma_1, \ldots$ that is fair and such that $\forall i \geq 0$, $|\sigma_i| = 1$. An execution e is *sequential fair* if it can be generated from a sequential fair scheduling.

A distributed algorithm $\mathcal{A}$ *deterministically* (resp., *probabilistically*) solves the exploration problem assuming a fair scheduling *iff* every fair execution e of $\mathcal{A}$ starting from a *towerless* configuration[1] satisfies: *(1)* e reaches a terminal configuration *in finite time* (resp., *with probability one*), and *(2)* every node is visited by at least one robot during e. Note that the previous definition implies that every initial configuration is *towerless*. Note also that the problem is not defined for $k > n$, and it is straightforward for $k = n$. Finally, in case of probabilistic exploration, termination is not certain, however the overall probability of non-terminating executions is 0.

3 Lower bound

We first generalize to arbitrary topologies a result from [7], that was initially given for rings. We then instantiate it to obtain a lower bound (actually 4) on the number of robots required to solve (deterministic or probabilistic) exploration in any torus. For purpose of generality, we assume here that the multiplicity detection of robots is *strong*. Moreover, we consider any (deterministic or probabilistic) exploration algorithm $\mathcal{A}$ using a team of k robots in an *arbitrary* topology $G = (V, E)$ of n nodes.

Assume $n > k$. The exploration is not (trivially) accomplished in an initial configuration. Since robots are oblivious, any terminal configuration of $\mathcal{A}$ in that case should be different from any possible initial configuration. As the set of possible initial configurations is exactly the set of towerless configurations, we have:

Remark 3.1 *If $n > k$, any terminal configuration of $\mathcal{A}$ contains at least one tower and, consequently $k > 1$.*

Lemma 3.1 *If $n > k$, then $k > 2$ and for every sequential fair execution e of $\mathcal{A}$ that terminates, e has at least $n - k + 1$ configurations containing a tower of less than k robots.*

Proof. First, by Remark 3.1, $k > 1$. Then, let e be a sequential fair execution of $\mathcal{A}$ that terminates. Consider the last configuration α without tower that appears in e and all remaining configurations that follow in e (all of them contains a tower) and form e' (e' necessarily exists, by Remark 3.1). By definition, $n - k$ new nodes (remember that k nodes are already visited in the initial configuration) must be visited before e' reaches its terminal configuration. Let $\beta\beta'$ be any step of e'.

1. If $\beta = \beta'$, then no node is visited during the step.
2. If $\beta \neq \beta'$, then there are three possible cases:

[1] Obliviousness requires the set of initial configurations to be a proper subset of the set of all configurations to make termination possible in our model; following the literature [9] we assume that every possible initial configuration is towerless.

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

If $k = 2$, then Case 2.(c) does not exist and so no node is visited, except the k initially visited nodes. As $n > k$, e' terminates without visiting all nodes, contradiction.

Assume now that $k > 2$. Initially, k nodes are visited. In Case 2.(a), which appears exactly once, β' contains a tower of less than k robots. Moreover, Case 2.(c) should appear at least $n - k$ times (only Case 2.(c) allows to visit new nodes). Hence, e', and so e, has at least $n - k + 1$ configurations containing a tower of less than k robots.

□

Lemma 3.2 *If $n > k$ and $k > 2$, then for every sequential fair execution e of $\mathcal{A}$ that terminates, e 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 fair execution $e = \gamma_0, \gamma_1, \ldots$ of $\mathcal{A}$ that terminates. As e terminates, e has a finite number of configurations. Let x be the number of configurations in e containing a tower of less than k robots. By Lemma 3.1, $x \geq n - k + 1$. We now show that (*) *if e contains at least two configurations having a tower of less than k robots that are indistinguishable, then there exists a sequential fair execution e' that terminates and such that e' has x' configurations containing a tower of less than k robots, where $x' < x$.*

Assume that there are two indistinguishable configurations $\gamma_t = d_1^t, \ldots, d_n^t$ and $\gamma_{t'} = d_1^{t'}, \ldots, d_n^{t'}$ in e having a tower of less than k robots. Without loss of generality, assume that $t' > t$. By definition, there exists an automorphism f on G such that $\forall i \in \{1, \ldots, n\}$, $d_i^t = d_j^{t'}$ where $v_j = f(v_i)$. Then, $e' = \gamma_0, \ldots, \gamma_t, \gamma'_{t'+1}, \gamma'_{t'+2}, \ldots$ is a sequential fair execution of $\mathcal{A}$, where $\forall z \geq t' + 1$, we have $\gamma'_z = d^z_{g(1)}, \ldots, d^z_{g(n)}$ where g is a bijection such that $\forall s \in [1..n]$, $f(v_s) = v_{g(s)}$ and $\gamma_z = d_1^z, \ldots, d_n^z$. Moreover, e' has x' configurations containing a tower of less than k robots, where $x' < x$.

By (*), if e contains less than $n-k+1$ distinguishable configurations altogether with a tower of less than k robots, it is possible to (recursively) construct a sequential fair execution e' of $\mathcal{A}$ that terminates such that e' has less than $n-k+1$ configurations containing a tower of less than k robots, a contradiction to Lemma 3.1. Hence, the lemma holds. □

From the two previous lemmas, follows:

Theorem 3.1 *Considering any (probabilistic or deterministic) exploration algorithm for k robots on a graph of $n > k$ nodes working under any fair scheduling, we have $k > 2$ and there exists a set $\mathcal{S}$ 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.*

A direct consequence if the previous theorem is the following:

Corollary 3.1 *Assuming fair schedulings, $\forall k, 0 \leq k < 3$, there is no algorithm to (deterministically or probabilistically) explore any torus of $n > k$ nodes using k robots.*

The previous corollary excludes that $\mathcal{A}$ works with $k < 3$ robots. Now, let assume that $k = 3$ and consider any arbitrary (ℓ, L)-torus (remember that by Remark 2.1, $n = \ell \times L \geq 9$). Then, by Theorem 3.1, we should be able to exhibit a set $\mathcal{S}$ of $n - 2$ 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 2 robots. Such configurations differ according to the relative positions of the tower and the robot which is alone. Two cases are then possible depending on whether $\ell = L$ or $\ell < L$. In the former case, the size of $\mathcal{S}$ is bounded by $\sum_{i=2}^{\lfloor \frac{L}{2} \rfloor + 1} i = \frac{\lfloor \frac{L}{2} \rfloor \times (\lfloor \frac{L}{2} \rfloor + 3)}{2}$. In the latter case, the size of $\mathcal{S}$ is bounded by $(\lfloor \frac{\ell}{2} \rfloor + 1)(\lfloor \frac{L}{2} \rfloor + 1) - 1$.

Let first study the case where $\ell = L$. Then, $\frac{\lfloor \frac{L}{2} \rfloor \times (\lfloor \frac{L}{2} \rfloor + 3)}{2}$ should be greater than or equal to $n - 2$, *i.e.*, $L^2 - 2$. From this inequality, we have: $7L^2 - 6L - 16 \leq 0$. $\Delta = 484 > 0$, so $7L^2 - 6L - 16 = 0$ has two solutions: $\frac{6 - \sqrt{484}}{14}$ and

$\frac{6+\sqrt{484}}{14}$; and $7L^2 - 6L - 16 \leq 0$ for $L \in [\frac{6-\sqrt{484}}{14}; \frac{6+\sqrt{484}}{14}]$. By Remark 2.1, $L \geq 3$. Moreover, $\frac{6+\sqrt{484}}{14} = 2$. So, we obtain a contradiction: there is neither probabilistic nor deterministic exploration algorithm in that case, even assuming a fair scheduling.

Let now study the case $\ell < L$. Then, $(\lfloor \frac{\ell}{2} \rfloor + 1)(\lfloor \frac{L}{2} \rfloor + 1) - 1$ should be greater than or equal to $n - 2$, *i.e.*, $\ell \times L - 2$. From this, we have: $2\ell + 2L + 8 \geq 3\ell \times L$. As $3 \leq \ell < L$ (Remark 2.1), $2\ell + 2L + 8 \geq 3\ell \times L$ has no solution: there is neither probabilistic nor deterministic exploration algorithm in that case, even assuming a fair scheduling.

An example is given in Fig. 3.1 for the case $\ell = L$ (the case $\ell < L$ is similar). In this example, for every value i inside a white node, every two configurations where (1) the black node contains the tower of two robots and (2) any white node of number i contains the single robot are indistinguishable. In the $(5, 5)$-torus, the size of $\mathcal{S}$ is at most 5.

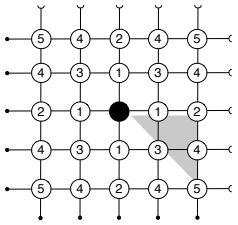

Figure 3.1: $(5, 5)$-torus.

Hence, there is neither probabilistic nor deterministic algorithm to explore any torus with 3 robots and, with Corollary 3.1, we can conclude:

Theorem 3.2 *Under fair schedulings, $\forall k, 0 \leq k < 4$, there is no algorithm to (deterministically or probabilistically) explore any torus of $n > k$ nodes using k robots.*

Consider now the *deterministic* exploration with $k = 4$ robots. Assume any (ℓ, L)-*Torus* such that $\ell = L$ and ℓ is even. Then, the four robots can be initially placed in such way that they have all identical views and all their possible destinations looked identical (just form a square whose adjacent sides have length $\frac{\ell}{2}$). Then, the adversary can choose to synchronously activate all robots at each step so that the initial symmetry continues: we obtain a non-terminating fair execution. Hence:

Theorem 3.3 *Under fair schedulings, $\forall k, 0 \leq k \leq 4$, there is no algorithm to* deterministically *explore all torus of $n \geq k$ nodes using k robots.*

4 Optimal Algorithm

In this section, we propose a probabilistic distributed algorithm to explore with 4 robots any (ℓ, L)-torus such that $7 \leq \ell \leq L$, assuming weak multiplicity detection. Before describing our algorithm, we first give some definitions.

4.1 Definitions

If v_1 and v_2 are two neighboring nodes respectively occupied by robots r_1 and r_2, then r_1 and r_2 are said to be neighbors. A *block* is a maximal elementary path along some ring $B = u_i, u_{i+1}, \ldots, u_{i+m}$ with $m > 0$, where each node is occupied by at least one robot. A robot that does not belong to any block is said to be *isolated*. A *hole* is any maximal non-empty elementary path of free nodes $H = u_i, u_{i+1}, \ldots, u_{i+m}$ that is along some ring. The *size* of a block (resp., a hole) is the number of nodes it contains. A block (resp. a hole) of size x is said to be an x-*block* (resp., a x-hole). Given the block B (resp., the hole H), the nodes u_i and u_{i+m} are termed as the *extremities* of B (resp., H). We call *neighbor* of a hole (resp. a block) any node that does not belong to the hole (resp. the block) but is neighbor of one of its nodes. In this case, we also say that the hole (resp. the block) is a *neighboring hole* (resp. *neighboring block*) of the node. By extension, any robot that is located at a neighboring node of a hole (resp. a block) is also called a neighbor of the hole (resp. the block). A node u is said to be *safe* if there is at most one robot located within distance one from u. We call *Couple* any ℓ-ring that contains exactly two robots.

4.2 Overview of the algorithm

Our algorithm works in three distinct successive phases, respectively called **SetUp**, **Tower**, and **Exploration**. Starting from any towerless configuration, the aim of the **SetUp Phase** is to arrange the robots in such a way that they eventually form a $\Diamond$.*Configuration* (see Fig. 4.2), without creating any tower during the process. This first phase is probabilistic. A $\Diamond$.Configuration is a towerless configuration, where (1) there are two distinct ℓ-rings of the torus that both contain a 2-block, and (2) there are two robots that have two robots in their neighborhood.

Once a ♢.Configuration is built, **Tower Phase** begins. This phase is also probabilistic and aims at creating a tower. Once the tower is created, the locations of robots give an explicit orientation to the torus; and the last phase, **Exploration Phase**, begins. This phase is deterministic. The two isolated robots collaborate together to deterministically explore the torus and eventually stop.

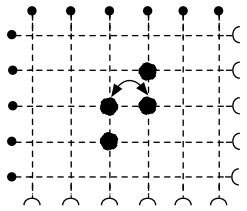

Figure 4.2: ♢.Configuration.

4.3 SetUp Phase

Phase SetUp is formally described in Alg. 1-6. The behavior of Phase SetUp is driven by several classes of configurations, which are defined below.

Algorithm 1 SetUp.

```
1.01  Let C be the current configuration
1.02  if C is Regular
1.03  then  try to move to a safe node
1.04  else  if C is Double-Trap1
1.05        then  Let R be the ring containing the 3-block
1.06              if I am the extremity of the 3-block that has only one neighboring robot
1.07              then move to the ring parallel to R that contains a single robot
1.08        else  if C is Double-Trap2
1.09              then  if I have one neighboring robot and there is a distance-2 robot r on the same ring
1.10                    then  move towards r
1.11              else  if C is Quadruplet
1.12                    then  execute Procedure Quadruplet
1.13                    else  if C is Triplet
1.14                          then  execute Procedure ♢-Creation
1.15                          else  if C is Isolated
1.16                                then  if L = ℓ
1.17                                      then  execute procedure Isolated-1
1.18                                      else  execute procedure Isolated-2
1.19                                else  if C is Twin
1.20                                      then  execute Procedure Twin
```

Configurations. A configuration is *Double-Trap1* (see Fig. 4.3) if there exists an ℓ-ring R that contains a 3-block having exactly one extremity with a neighboring robot that is not in R. A configuration is *Double-Trap2* (Fig. 4.4) if there is one isolated robot at some node z and two 2-blocks $B1 = u, v$ and $B2 = x, y$ such that (1) $v = x$, (2) $B1$ is on a ℓ-ring, (3) z and y are on a ℓ-ring parallel to the one containing $B1$, and (4) z is at distance 2 of both u and y.

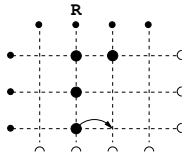

Figure 4.3: *Double-Trap1*

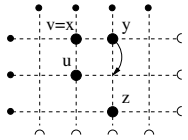

Figure 4.4: *Double-Trap2*

A configuration C is *Regular* if C is towerless and not a ♢.configuration, and the robots can be split in two pairs $\{r_1, r_2\}$ and $\{r_3, r_4\}$ such that the views of r_1 and r_2 (resp. of r_3 and r_4) are identical. (A particular case of Regular is a configuration where all robots have identical views.)

A configuration C is *Triplet* if C is towerless, not a Double-Trap1, and there is an ℓ-ring R that contains exactly 3 robots. When R contains neither a 3-block nor a 2-block, we define the *Wall* to be the ring perpendicular to R that contains the robot not in R, see Fig. 4.5.

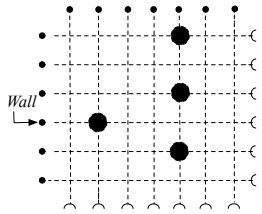

Figure 4.5: A *wall* in a *Triplet*.

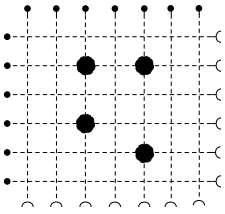

Figure 4.6: Twin Configuration.

A configuration C is *Twin* (Fig. 4.6) if C is towerless, contains a couple, but is neither Double-Trap1, nor Double-Trap2, nor ♢, nor Regular, nor Triplet. A configuration C is *Isolated* if C is towerless, not Regular, and there exists at most one robot on each ℓ-ring.

Finally, a configuration C is *Quadruplet* if C is not Regular and there exists an ℓ-ring R that contains 4 robots.

Overview. In Section 5, we prove that the SetUp phase achieves a convergence from any towerless configuration to a ♢.Configuration with probability one without creating any tower during the process. This proof actually consists in showing that each transition in Fig. 4.7 can be made, with positive probability, in finite

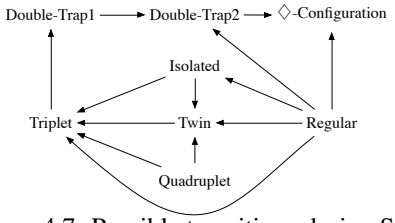

Figure 4.7: Possible transitions during SetUp.

number of steps. Some of these transitions are deterministic. Note also that there are many other possible transitions, not shown in Fig. 4.7. Details about this phase are given in Section 5.

Algorithm 2 Procedure Isolated-1 (Case $\ell = L$).

```
2.01 if there are several possible smallest enclosing rectangles (SERs)
2.02 then if I am neighbor of a safe node u such that if I move to u and I am the only one to move,
2.03      the number of SERs decreases
2.04      then try to move to u
2.05 else
2.06      Let s the unique smallest rectangle that encloses the four robots
2.07      We call internal robot any robot that is not on any side of s
2.08      if there is exactly one robot that is at a corner c of s
2.09      then if I am internal
2.10           then move towards the closest free node on a side of s having c as extremity
2.11      else if there are exactly two robots at two corners c_1, c_2 of s
2.12           then if I am a closest internal robot to an occupied corner of s
2.13                then move towards the closest free node on a side of s having c_1 or c_2 as extremity
2.14           else // there is no robot of any corner of s
2.15                Let d be the minimum distance between any occupied node and a corner of s
2.16                if there is a unique robot r that is at distance d from a corner of s
2.17                then if I am r
2.18                     then move towards the closest corner of s
2.19                else if there are two robots r_1, r_2 that are at distance d from the same corner of s
2.20                     then Let r_3 and r_4 be the two other robots
2.21                          For each i ∈ {3,4}, let d_i be the distance to unique corner c_i that is
2.22                          common to the side of s where r_i is and the side of s where r_1 or r_2 are
2.23                          Let min ∈ {3,4} such that d_min = min{d_3, d_4}
2.24                          if I am r_min
2.25                          then move towards c_min
2.26                     else if I am at distance d from a corner of s
2.27                          then move towards the closest corner of s
```

Algorithm 3 Procedure Isolated-2 (Case $\ell < L$).

```
3.01 if every L-ring contains at most one robot
3.02 then Let r_1, r_2, r_3, and r_4 be the four robots
3.03      if there are several possible smallest enclosing rectangles (SERs)
3.04      then if I am neighbor of a safe node u such that if I move to u and I am the only one to move,
3.05           the number of SERs decreases
3.06           then try to move to u
3.07      else
3.08           Let s the unique smallest rectangle that encloses the four robots
3.09           There are two robots, say r_1 and r_2, which are on two parallel sides of s that are on ℓ-rings
3.10           Let d be the minimal distance between r_3 or r_4 and an ℓ-ring which is a side of s
3.11           if I am r_3 or r_4, and I am at distance d from an ℓ-ring which is a side of s
3.12           then move along my L-ring toward the closest ℓ-ring which is a side of s
3.13 else if there is exactly one L-ring R that contains 2 or 3 robots
3.14      then if I am a robot alone in an L-ring that is closest to R
3.15           then move along my L-ring towards a closest free node having a node of R in its ℓ-ring
3.16      else if there are two L-rings that contain 2 robots
3.17           then if there is a safe neighboring node u outside my current L-ring
3.18                then try to move to u
3.19           else // there is one L-ring R that contains all robots
3.20                Apply Procedure Quadruplet on the unique L-ring R
```

4.4 Tower Phase

This phase starts from any $\Diamond$.Configuration— Fig. 4.2. Let u_1 and u_2 be the two occupied neighboring nodes having themselves two occupied neighboring nodes. Let r_1 and r_2 be the two robots located at u_1 and u_2, respectively. During this phase, r_1 and r_2 *try to move* towards each other anytime they are activated; the other two robots do not move if activated. If only one of them is activated, a tower is created. If both are activated simultaneously, there is a positive probability that only one of those moves; otherwise the system remains in a $\Diamond$.Configuration. As the scheduler is fair, both of them are activated regularly until a tower is created. So, the event "only one robot moves" eventually occurs with probability one. So, a tower $\mathtt{T}$ is created with probability one on either u_1 or u_2 and the Tower phase is done.

4.5 Exploration Phase

We first need further definitions. Given two nodes u and v, let R_{uv} be a *smallest enclosing rectangle* that includes both u and v. Let α_{uv} (β_{uv}) be the length in terms of hops of one of the smallest (resp., greatest) side of R_{uv}, R_{uv} is an

Algorithm 4 Procedure Twin.

```
4.01 if there is a unique Couple C
4.02 then if there is a unique robot r that is outside C and closest to C
4.03       then if I am r
4.04             then move to an adjacent free node on a shortest path to a free node of C
4.05       else if I am outside the Couple
4.06             then let R_1 be my ring perpendicular to C
4.07                  let R_2 be the ring parallel to C which contains the other robot outside C
4.08                  try to move to a neighboring free node on R_1 which does not belong to C or R_2
4.09 else // there are two or three Couples
4.10       if there are two Couples
4.11             if I am in a Couple and neighbor of a safe node u outside any couple
4.12             then try to move to u
4.13       else // there are three Couples
4.14             if I belong to two Couples
4.15             then Let C be the Couple containing the two robots that belong to two Couple
4.16                  try to move to a neighboring safe node on C
```

Algorithm 5 Procedure Quadruplet.

```
5.01 Let R be the ring that contains the 4 robots
5.02 if R contains exactly one smallest hole
5.03 then if I am neighbor to such a hole
5.04       then move to an adjacent free node outside R
5.05 else if R contains exactly one biggest hole
5.06       then if I am neighbor to such a hole
5.07             then move to an adjacent free node outside R
5.08       else if I am neighbor to the two smallest hole
5.09             then move to an adjacent free node outside R
```

Algorithm 6 Procedure $\Diamond$-Creation.

```
6.01 Let R be the ring of the torus that contains 3 robots
6.02 if R contains no 3-block
6.03 then if R contains no 2-block
6.04       then if R ∩ Wall = Free-Node
6.05             then if I am on R, but not neighbor of Free-Node
6.06                  then move along R towards Free-Node
6.07             else // R ∩ Wall = Occupied-Node
6.08                  if I am not on the Wall
6.09                  then move towards the Wall
6.10       else // R contains a 2-block
6.11             if I am on R, but not part of the 2-block
6.12             then move towards the 2-block
6.13 else // R contains a 3-block
6.14       if I am not part of R
6.15       then move towards a free node that closest to an extremity of the 3-block and not part of R
```

$(\alpha_{uv}, \beta_{uv})$-rectangle. The *Manhattan metric* between two nodes u and v, denoted by M_{uv}, is the tuple $(\alpha_{uv}, \beta_{uv})$. Manhattan metrics are ordered as follows: given four nodes u, v, u', and v', $M_{uv} \leq M_{u'v'}$ *iff* either $\alpha_{uv} < \alpha_{u'v'}$ or $\alpha_{uv} = \alpha_{u'v'}$ and $\beta_{uv} \leq \beta_{u'v'}$.

The exploration starts from the initial configuration built during the tower phase. Denote the node holding the tower by $\mathtt{T}$. The two rings passing through $\mathtt{T}$ are called *coordinate rings*. In the sequel, '$\circ$' (respectively, '$\star$') denotes the *nearest* (respectively, *farthest*) single node (or robot) from $\mathtt{T}$, *i.e.,* $M_{\circ T} < M_{\star T}$. Note that our algorithm ensures that both '$\circ$' and '$\star$' remain the same robots until the end of the exploration. Given a node u, if $\alpha_{Tu} < \beta_{Tu}$ and $\{\alpha_{Tu}, \beta_{Tu}\} \neq \{\lfloor \frac{\ell}{2} \rfloor, \lfloor \frac{L}{2} \rfloor\}$, then there exists an orientation of the coordinate rings such that $u = (\alpha_{Tu}, \beta_{Tu})$. In the following, when possible, we build a coordinate system over R_{Tu} by setting the x-axis (resp. the y-axis) as the coordinate rings that is parallel to the smallest (resp., greatest) side of $R_{T,*}$ and by orienting both axis to have positive coordinates for u.

The main idea of Phase Exploration is the following: both robots that are not part of the tower collaborate together in order to explore the whole torus. They alternate between two roles: *Explorer* and *Leader*. Leader $\mathcal{L}$ allows to build a coordinate system $S^{\mathcal{L}}$ over $R_{\mathtt{T}\mathcal{L}}$. The explorer is in charge of deterministically exploring the torus over $S^{\mathcal{L}}$. The exploration works in three phases, executed in sequence.

Phase 1. Fig. 4.8 illustrates that phase. Robot $\star$ (*i.e.,* the farthest robot of $\mathtt{T}$) plays the role of Leader. Starting from the configuration built by Phase Tower, Robot $\star$ first builds a $(1, 2)$-rectangle with $\mathtt{T}$ by moving along the ring parallel to the one containing both $\mathtt{T}$ and $\circ$ in the opposite direction to $\circ$, see Move #1 in Fig. 4.8. $R_{\mathtt{T}*}$ allows to build a coordinate system S^*, where Robot $\star$ occupies Node $(1, 2)$ *w.r.t.* S^*. Then, Robot $\circ$ initiates a spiral-shaped exploration. It visits the nodes that form the first surrounding square around $\mathtt{T}$ and stops at node $(-1, -1)$—Move #2. Next, Robot $\star$ moves to node $(2, 3)$ passing through node $(1, 3)$—Moves #3′ and #3″ in Fig. 4.8. Then, Robot $\circ$ visits the nodes that form the second surrounding square around $\mathtt{T}$ and stops at $(-2, -2)$—Move #4. Finally, Robot $\star$ moves back to $(1, 3)$, followed by Robot $\circ$ that moves back to $(-2, -1)$—Moves #5 and #6 in Fig. 4.8. Note that our method requires that Robot $\star$ must be able to move at least three lines away from the tower. Furthermore, Robot $\circ$ must be able to visit the two squares centered on the tower and the orientation built by Robot $\star$ must be unambiguous. These three conditions constrain the torus to be of size **at least $\mathbf{7 \times 7}$**.

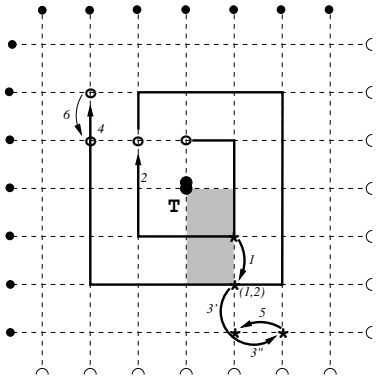

Figure 4.8: First phase of exploration. The integers show the move order.

Phase 2. In this phase, Robot $\circ$ is the leader. $R_{\mathtt{T}o}$ provides a coordinate system S°, where Robot $\circ$ is located at $(1, 2)$. Robot $\star$ now proceeds to the spiral exploration by visiting surrounding squares around $\mathtt{T}$ one after another, see Fig. 4.9 and 4.10. Robot $\star$ first explores the third surrounding square around $\mathtt{T}$, then the fourth, and so forth, until it visits the $(\lfloor \frac{\ell}{2} \rfloor - 1)$-th square. Then, there are two cases depending on the parity of ℓ: If ℓ is odd, then Robot $\star$ visits the whole $\lfloor \frac{\ell}{2} \rfloor$-th square and finish at the negative (*w.r.t* S°) corner of the square, see Fig. 4.9. Otherwise (ℓ is even), Robot $\star$ visits half of the $\lfloor \frac{\ell}{2} \rfloor$-th square only and stops at the positive corner (*w.r.t.* S°) of the square, see Fig. 4.9. In both cases, if $\ell = L$, then the exploration is done.

Phase 3. This last phase is performed only if $\ell \neq L$. In that case, Robot $\star$ terminates the exploration by going alternatively from the left to the right and from the right to the left among the nodes forming the remaining of the rectangle. If ℓ is odd, then Robot $\star$ progresses towards the negative (*w.r.t.* S°) side of the torus—Fig. 4.9. Otherwise (ℓ is even), the progression is made on the positive side—Fig. 4.10. In both cases, the exploration ends either on the positive side or the negative side of the L-th line, depending on either L is odd or even.

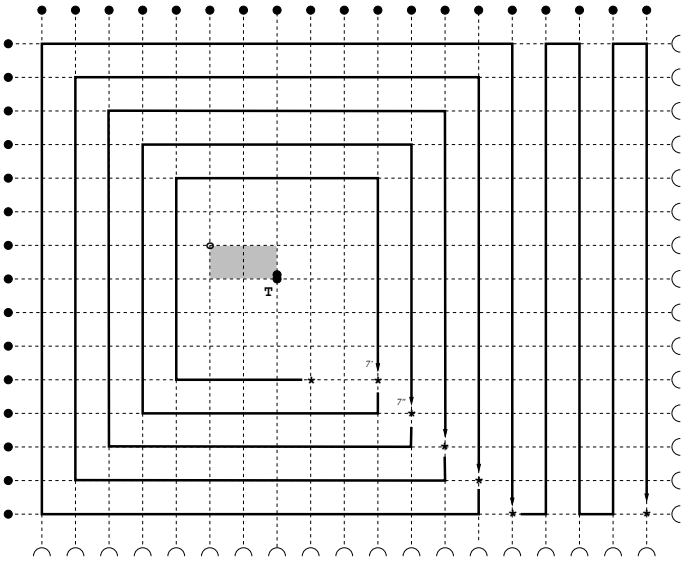

Figure 4.9: Second and third phase, odd case.

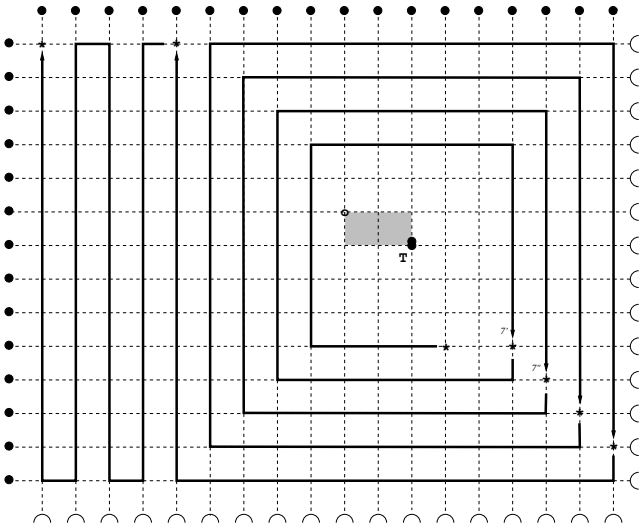

Figure 4.10: Second and third phase, even case.

5 Correctness Proof

Our algorithm is made of three phases. The first phase, SetUp, starts from any towerless configuration and ends in a $\Diamond$.Configuration. Moreover, this phase contains no configuration with a tower. The second phase, Tower, just consists in one probabilistic transition (that may require several steps) from a $\Diamond$.Configuration to the initial towered configuration of the Exploration phase. Finally, all configurations that appear during the Exploration phase contain a tower. Hence, we have the following property:

Remark 5.1 *The three phases (SetUp, Tower, and Exploration) are unambiguous.*

Assume the system is in a $\Diamond$.Configuration. Then, with probability 1, the initial (towered) configuration of the Exploration phase is eventually reached, see Subsection 4.4. From that configuration, robots $\star$ and $\circ$ explore the torus in a fully deterministic manner (Subsection 4.5). So, the termination of the exploration phase is certain. Consequently, starting from a $\Diamond$.Configuration, the torus is eventually explored with probability one. Hence, the critical part of the algorithm is the SetUp phase, which aims at converging from any initial (towerless) configuration to a $\Diamond$.Configuration. We now show that this phase eventually terminates with probability one.

First, the SetUp phase exactly deals with the set of all towerless configurations, and by definition, we have:

Remark 5.2 *The set of all towerless configurations is equal to the (disjoint) union of the following configuration sets: $\Diamond$.Configurations, Regular, Double-Trap1, Double-Trap2, Twin, Triplet, Quadruplet, and Isolated.*

The first part of the proof consists in showing that a $\Diamond$.Configuration is *deterministically* reached in finite time from any configuration Double-Trap1, Double-Trap2, or Triplet. In the following, we consider an arbitrary towerless configuration C. The scheduler being fair, we have the following property:

Remark 5.3 *Each robot is activated in finite time from C.*

Lemma 5.1 *If C is Double-Trap2, a $\Diamond$.Configuration is reached from C in finite time.*

Proof. Immediate from Alg. 1, Lines 1.08-1.10. (See also Fig. 4.4.) □

Lemma 5.2 *If C is Double-Trap1, $\Diamond$.Configuration is reached from C in finite time.*

Proof. From Alg. 1 (Lines 1.04-1.07), a configuration Double-Trap2 is reached in one step (Fig. 4.3), and we can conclude by Lemma 5.1. □

Lemma 5.3 *If C is Triplet, a $\Diamond$.Configuration is reached from C in finite time.*

Proof. Let R be the ℓ-ring that contains three robots in C. We have three main cases:

1. *R contains a 3-block.* In this case the robot that is not in R is the only one allowed to move (see Alg. 6, Lines 6.13-6.15). Its destination is the adjacent node on the shortest path towards the closest free node neighbor of one extremity of the 3-block. The target node is reached in finite time and then a Double-Trap1 configuration is created. By Lemma 5.2, we are done.
2. *R contains a 2-block.* In this case, the robot that is on R, but not part of the 2-block is the only one allowed to move (see Alg. 6, Lines 6.10-6.12). Its destination is the adjacent free node on the shortest path towards the 2-block on R. So, we retrieve Case 1 in finite time and we are done.
3. *All other cases.* (see Alg. 6, Lines 6.03-6.09)
   (a) If $R \cap$ wall = Occupied-Node. According to the choices of the fair scheduler, in finite time either a 3-block is created and we retrieve Case 1, or a 2-block is created and we retrieve Case 2.
   (b) If $R \cap$ wall = Free-Node, a 2-block is created on R in finite time and by Case 2, we are done.

□

From the other towerless configurations (Regular, Twin, Quadruplet, and Isolated), the convergence to a $\Diamond$.Configuration is *probabilistic*. Now, when considering any arbitrary fair execution, (1) the topology is fixed with, in particular, a *finite* number of nodes and (2) remind that the number of robots is *the constant 4*. So, the number of towerless configurations

is also *finite*. Now, while a configuration Double-Trap1, Double-Trap2, Triplet, or $\Diamond$.Configuration is not reached, the SetUp phase remains trapped in the finite subset of towerless configurations defined as the distinct union of Configuration Regular, Twin, Quadruplet, and Isolated, by Remark 5.2. So, in order to demonstrate that the system eventually reaches a $\Diamond$.Configuration with probability one, we show below that from any configuration Regular, Twin, Quadruplet, or Isolated, there always exists a positive probability to reach a configuration Double-Trap1, Double-Trap2, Triplet, or $\Diamond$ in finite time despite the choices of the fair scheduler. Then, we can conclude by Lemmas 5.1-5.3.

Lemma 5.4 *If C is Twin, with positive probability, a configuration Triplet is reached from C in finite time.*

Proof. When the configuration C is Twin, the cases below are possible:

1. *C contains a unique couple, R.*
   (a) *If there is a unique robot r that is outside R and the closest to R*, then r is the only robot allowed to move, and if activated, it moves to an adjacent free node on a shortest path to a free node of R (see Alg. 4, Lines 4.02-4.04). So, a Triplet configuration is reached, in this case, in finite time.
   (b) *Otherwise*, the two robots that are outside R. Let r be any of those robots. Let R_{per} be the ring of r perpendicular to R. Let R_{para} be the ring parallel to R which contains the other robot outside R. If activated, r *try to move* to a neighboring free node on R_{per} which does not belong to R or R_{para} (see Alg. 4, Lines 4.05-4.08). Then, at least one of these two robots is eventually activated, and when activated, with positive probability (actually, a probability greater than or equal to $p(1-p)$), exactly one of those robots moves and we retrieve the previous case. Hence, with positive probability, a configuration Triplet is reached in finite time.
2. *C contains two couples.* Then, some robots in a Couple have a neighboring safe node outside any couple. Only these robots are allowed to move (Alg. 4, Lines 4.10-4.12). If activated, they try to move to a neighboring safe node outside any couple. One of them is activated in finite time and when activated, with positive probability (actually, a probability greater than or equal to $p(1-p)^3$), exactly one of them moves. In this case, the system reaches a configuration Twin containing only one couple (Case 1). Hence, with positive probability, a configuration Triplet is reached in finite time.
3. *C contains three couples.* (In this case, the torus necessarily satisfies $\ell = L$.) Observe that, in this case, there exist two robots that belong to two couples. Only those robots are allowed to move. Let C be the Couple containing these two robots. If activated, they *try to move* to a neighboring safe node on C (see Alg. 4, Lines 4.13-4.16). At least one of them is activated in finite time and when activated, with positive probability (actually, a probability greater than or equal to $p(1-p)$), exactly one of them moves and the configuration reached is still Twin but with two couples, that is, one of the previous cases. Hence, we can conclude that, in this case, with positive probability, a configuration Triplet is reached in finite time.

□

Lemma 5.5 *If C is Quadruplet, a configuration either Twin or Triplet is reached from C in finite time.*

Proof. Let R be the ring that contains four robots in C. We have the three cases:

(1) *If R contains exactly one smallest hole.* In this case, the two robots neighbor to such a hole move to an adjacent free node outside R, if activated (Alg. 5, Lines 5.02-5.04). Then, if both robots move simultaneously, then the reached configuration is Twin. If only one robot moves, then the reached configuration is Triplet.

(2) *If R contains exactly one biggest hole.* This case is similar to the previous one, either a Twin or a Triplet configuration is reached in finite time.

(3) *Otherwise, there is one robot that is neighbor to the two smallest holes on R.* This robot is the only one allowed to move. Its destination is the adjacent free node outside R (see Alg. 5, Lines 5.08-5.09). After this step, a configuration Triplet is reached.

□

By Lemma 5.4, we have the following corollary from the previous lemma:

Corollary 5.1 *If C is Quadruplet, with positive probability, a configuration Triplet is reached from C in finite time.*

Lemma 5.6 *If C is Isolated, with positive probability, a configuration Triplet is reached from C in finite time.*

Proof. We split the study into two main cases:

1. $\ell = L$. There are two cases:
   - *There is a unique smallest enclosing rectangle (SER), s.* We call *internal robot* any robot that is not on a side of s. The following subcases are possible:
     (a) *There is exactly one robot that is at a corner c of s.* So, there is exactly one internal robot. It is the only robot allowed to move, its destination is the closest free node on a side of s having c as extremity (Alg. 2, Lines 2.08-2.10). Thus, a Twin configuration is reached in finite time and by Lemma 5.4, we are done.
     (b) *There are exactly two robots at two corners c_1, c_2 of s.* The other two robots are internal. The internal robots that are the closest to an occupied corner of s move towards the closest free node on a side of s having c_1 or c_2 as an extremity (Alg. 2, Lines 2.11-2.13). So, a Twin configuration is reached in finite time and by Lemma 5.4, we are done.
     (c) *There is no robot at the corner of s.* Observe that all robots are located at different sides of s. Let d be the minimum distance between any occupied node and a corner of s. The following subcases are possible:
        i. *There is a unique robot r at distance d from a corner of s.* r is the only robot allowed to move. Its destination is the adjacent free node towards the closest corner (Alg. 2, Lines 2.16-2.18). A Twin configuration is reached in finite time and by Lemma 5.4, we are done.
        ii. *There are exactly two robots r_1 and r_2 at distance d from a corner.* If r_1 and r_2 are the closest to the same corner c, then let r_3 and r_4 be the two other robots. For each $i \in \{3, 4\}$, let d_i be the distance to unique corner c_i that is common to the side of s where r_i is and the side of s where r_1 or r_2 are. (*n.b.*, $d_3 \neq d_4$ otherwise the configuration is regular.) Let $min \in \{3, 4\}$ such that $d_{min} = \min\{d_3, d_4\}$. Then, r_{min} moves toward c_{min} (Alg. 2, Lines 2.19-2.25), and a Twin configuration is reached in finite time. By Lemma 5.4, we are done.
        If r_1 and r_2 are closest to a corner of s, but not the same, then r_1 and r_2 are the only robots allowed to move. Their destination is the adjacent free node towards the closest corner (see Alg. 2, Lines 2.26-2.27). In finite time, either we retrieve case 1(c)i, or the system reaches a configuration Twin (depending on the choices of the scheduler). In the latter case, we can conclude with Lemma 5.4. In all other cases, we retrieve of the previous cases, and we are done.
        iii. *There are exactly three robots at distance d from a corner of s.* There are either one or three corners that have only one robot at distance d from them. In the former case, only one robot moves toward its closest corner (see Alg. 2, Lines 2.19-2.25). After the move, either we retrieve case 1(c)i and we are done, or the system is a configuration Twin and we conclude by Lemma 5.4. In the latter case, each of the three robots moves to its closest corner (Alg. 2, Lines 2.26-2.27), now all targets are different. So, in finite time, we retrieve either Case 1(c)i, or Case 1(c)ii, or a Twin configuration depending on the choices of the scheduler, and we conclude as in previous cases.
   - *There are several possible $SERs$.* Some of the robots have a safe neighboring node such that if they move to that node and they move alone, then the number of $SERs$ decreases. Only these robots are allowed to move. If activated, they try to move to such a safe node (Alg. 2, Lines 2.01-2.04). At least one of them is activated in finite time and when activated, with positive probability (actually, a probability greater than or equal to $p(1-p)^3$), exactly one of them moves. In this case, either the system reaches configuration Twin and we are done (Lemma 5.4), or the system is still in a configuration Isolated, yet the number of possible $SERs$ has strictly decreased.
     Hence, in the worst case, the system needs three consecutive sequential moves (which happen with a probability greater than or equal to $p^3(1-p)^9$) to reach either a Twin configuration, or a configuration Isolated containing a unique SER. Hence, with positive probability the system reaches in finite time a Twin configuration and by Lemma 5.4, we are done.
2. $\ell \neq L$. According to the number of robots on L-rings, we have 5 cases:
   (a) *Every L-ring contains at most one robot.* Let r_1, r_2, r_3, and r_4 be the robots.
   Assume there is a unique smallest enclosing rectangle (SER), s. Observe that there are two robots, say r_1 and r_2, that are on two parallel sides of s that are on ℓ-rings. Let d be the minimal distance between r_3 or r_4 and an ℓ-ring which is a side of s. If r_3 (resp. r_4) is at distance d from an ℓ-ring which is a side of s, it is allowed to move. Note that both r_3 and r_4 may move simultaneously. If allowed to move, the move is done

along the current L-ring toward the closest ℓ-ring which is a side of s (Alg. 3, Lines 3.01-3.12). Hence, Twin configuration is reached in finite time and by Lemma 5.4, we are done.
If there are several $SERs$, we proceed as in the case $\ell = L$. Hence, with positive probability the system reaches in finite time a Twin configuration and by Lemma 5.4, we are done.

(b) *There is exactly one L-ring R that contains 2 robots.* The robots (one or two robots) that are alone on their L-ring and closest to R move along their L-ring towards a closest free node having a node of R in its ℓ-ring (Alg. 3, Lines 3.13-3.15). So, in finite time, the system reaches either a Triplet, or a Twin configuration depending of the choices of the fair scheduler. In the latter case, with positive probability, a Triplet configuration is reached in finite time by Lemma 5.4 and we are done.

(c) *There is exactly one L-ring R that contains 3 robots.* The case is similar to the previous one, see Alg. 3, Lines 3.13-3.15: in finite time, the system reaches a Twin configuration and by Lemma 5.4, we are done.

(d) *There exist two L-rings that contain two robots.* In this case, some of the robots (at least one) have a safe neighboring node outside their current L-ring. Only these robots are allowed to move (Alg. 3, Lines 3.16-3.18). If activated, they try to move to safe neighboring node outside their current L-ring. One of them is activated in finite time and when activated, with positive probability (actually, a probability greater than or equal to $p(1-p)^3$), exactly one of them moves. In this case, we retrieve cases 2b or 2c, and we are done.

(e) *There is one L-ring R that contains all robots.* Similarly to Lemma 5.5, one or two robots leave R in finite time, and we retrieve one of the previous cases.

□

Lemma 5.7 *If C is Regular, with positive probability, a non-Regular towerless configuration is reached from C in finite time.*

Proof. If C is Regular, then every robot *tries to move* to a safe node, if activated. Now, in this case, with positive probability, exactly one of them moves during the next step and a non-Regular towerless configuration is reached. □

From all previous lemmas and corollary, we can deduce that, with probability one, the SetUp phase eventually terminates in a $\Diamond$.Configuration. Recall also that Phase 1 of the exploration requires that the torus satisfies the condition $7 \leq \ell \leq L$ (see page 10). Hence:

Theorem 5.1 *The SetUp, Tower, and Exploration phase allows four robots to probabilistically explore on any torus of size $\ell \times L$, where $7 \leq \ell \leq L$.*

6 Concluding Remarks

While the solution we provided for the torus exploration problem is optimal in terms of number of robots, there remain challenging open questions. First, we presented an algorithm for all tori of size $\ell \times L$, where $7 \leq \ell \leq L$. In [6], the authors stated that small grids require more robots. Determining if our results can be extended to smaller tori is an interesting problem. We expect mechanized approaches [2] to be valuable for investigating small size tori. Second, dealing with higher dimension (*e.g.*, from a ring to a torus) does not necessarily increase the robot number complexity of the exploration problem. The issue of the d-dimensional tori (with $d > 2$) remains open.

References

[1] Baldoni, R., Bonnet, F., Milani, A., Raynal, M.: Anonymous graph exploration without collision by mobile robots. Inf. Process. Lett. **109**(2), 98–103 (2008)

[2] Bonnet, F., Défago, X., Petit, F., Potop-Butucaru, M., Tixeuil, S.: Discovering and assessing fine-grained metrics in robot networks protocols. In: 33rd IEEE SRDS Workshops, Workshop on Self-organization in Swarm of Robots, pp. 50–59 (2014)

[3] Chalopin, J., Flocchini, P., Mans, B., Santoro, N.: Network exploration by silent and oblivious robots. In: WG, pp. 208–219 (2010)

[4] D'Angelo, G., Di Stefano, G., Navarra, A., Nisse, N., Suchan, K.: A unified approach for different tasks on rings in robot-based computing systems. In: IPDPS Workshops, pp. 667–676 (2013)

[5] D'Angelo, G., Navarra, A., Nisse, N.: Gathering and exclusive searching on rings under minimal assumptions. In: ICDCN, pp. 149–164 (2014)

[6] Devismes, S., Lamani, A., Petit, F., Raymond, P., Tixeuil, S.: Optimal grid exploration by asynchronous oblivious robots. In: SSS, pp. 64–76 (2012)

[7] Devismes, S., Petit, F., Tixeuil, S.: Optimal probabilistic ring exploration by semi-synchronous oblivious robots. Theor. Comput. Sci. **498**, 10–27 (2013)

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

[9] Flocchini, P., Ilcinkas, D., Pelc, A., Santoro, N.: Computing without communicating: Ring exploration by asynchronous oblivious robots. Algorithmica **65**(3), 562–583 (2013)

[10] Flocchini, P., Prencipe, G., Santoro, N.: Distributed Computing by Oblivious Mobile Robots. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers (2012)

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

[12] Lamani, A., Potop-Butucaru, M., Tixeuil, S.: Optimal deterministic ring exploration with oblivious asynchronous robots. In: SIROCCO, pp. 183–196 (2010)