Terminating Exploration of a Grid by an Optimal Number of Asynchronous Oblivous Robots

Stéphane Devismes* Anissa Lamani† Franck Petit† Pascal Raymond*
Sébastien Tixeuil†

Abstract

We propose optimal (*w.r.t.* the number of robots) deterministic solutions for the *terminating exploration* of an anonymous grid-shaped network by a team of asynchronous oblivious robots. We first consider the semi-synchronous model. We show that it is impossible to explore a grid of at least 3 nodes with less than 3 robots. Next, we show that it is impossible to explore a $(2, 2)$-Grid with less than 4 robots, and a $(3, 3)$-Grid with less than 5 robots, respectively. The two first results hold for both deterministic and probabilistic settings, while the latter holds only in the deterministic case. We then consider the asynchronous model. This latter being strictly weakest than the semi-synchronous model, all the aforementioned impossibility results still hold in that context. We then propose deterministic algorithms to exhibit the optimal number of robots allowing to explore of a given grid. Our results show that except in two particular cases, 3 robots are necessary and sufficient to deterministically explore a grid of at least 3 nodes. The optimal number of robots for the two remaining cases is: 4 for the $(2, 2)$-Grid and 5 for the $(3, 3)$-Grid, respectively.

1 Introduction

We consider swarms of *autonomous robots* that are endowed with motion actuators and visibility sensors. Many potential applications exist for such multi-robot systems in various contexts. For instance, patrolling in adversarial environments, environmental monitoring, exploration of awkward environments, intelligence activities, fighting fire in a rescue scenario, and many other risky tasks for humans can be solved by robotic swarms. Those applications require that the robots collaborate to succeed in task accomplishment that has been assigned to them. Devising distributed algorithms capable of coordinating robots without central control remains one of the main challenges to overcome [11].

In this paper, we address the problem of *exploring a finite discrete space* by autonomous mobile robots. Exploration is a basic building block for many applications. For instance, mapping of an unknown area 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 as a finite number of locations represented by a graph, where *nodes* represent indivisible locations that can be sensed by the robots, and where *edges* represent

*VERIMAG UMR 5104, Université Grenoble Alpes, France
†LIP6 UMR 7606, INRIA, UPMC Sorbonne Universités, France

the possibility for a robot to move from one location to the other, *e.g.*, a building, a town, a factory, and more generally, zoned areas. In such settings, a continuous two-dimensional Euclidean space is conveniently represented using a *grid-shaped network*.

The exploration problem has been addressed in the distributed computing community considering dual settings: probabilistic [7] *vs.* deterministic [10], and perpetual [1] *vs.* terminating [10]. In this paper, we focus on *deterministic terminating exploration*. In this problem, all robots, initially placed at different places, have to collectively visit all nodes (*i.e.*, every node to be visited by at least one robot) and then eventually stop.

We consider the problem of performing the terminating exploration with the constraint of using as less resources as possible. By minimizing the resources, we first mean weakening as much as possible the assumptions made on the robot capabilities. Actually, we assume robots that are *uniform* (all robots follow the same algorithm), *anonymous* (robots are identical), *oblivious* (robots do not remember the past), *disoriented* (robots have no mean to locally or globally orient themselves), and *deaf-mute* (robots have no means of communicating together). However, robots are endowed with visibility sensors enabling to see robots located on nodes. Arguments justifying such assumptions are varied and numerous. Reducing manufacturing costs, design, and operating expenses are some of them. Also, numerous realistic scenarios can easily be considered where devices are faulty, unusable, or even non-existent, for instance, sensors and guidance systems like GPS or compass may be faulty, communications may be scrambled or forbidden, physical space aboard robots may not allow to embed some devices, like sensors or storage memory. Another objective in assuming such robot weakness is to provide an answer to the following fundamental question: *What is computational complexity required for the robots to deterministically solve a given problem? (In our case the terminating exploration.)* Since we are addressing distributed solutions, such issue must be addressed globally, *i.e.,* over all the robots. In other words, the *number of robots* that are required to solve the problem is also a crucial parameter. The minimum number of robots in the accomplishment of a given task is also are important knowledge in order to achieve fault tolerant solutions as describe above in an environment where some robots may be totally destroyed or unusable.

1.1 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 [4] 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 stronger 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 [9]. Flocchini *et al.* [10] proved that no deterministic ring exploration is possible in the asynchronous model when the number of robots k divides the number of nodes n. However, they proposed a deterministic algorithm that solves the problem, still in the asynchronous model, using $k \geq 17$ robots provided that n and k are co-prime. Lamani *et al.* [13] proved that there exists no deterministic algorithm that can explore an even sized ring with $k \leq 4$ robots, even in the semi-synchronous (SSYNC) model [16]. Impossibility results in the

semi-synchronous model naturally extend to the asynchronous non-atomic model [14]. Lamani *et al.* [13] also provide an algorithm, working in the semi-synchronous model, which allows 5 robots to deterministically explore any ring whose size is co-prime with 5. By contrast, 4 robots are necessary and sufficient to *probabilistically* explore any ring of size at least 4 in the semi-synchronous model [7]. Still in the semi-synchronous model, 4 robots are necessary and sufficient to *probabilistically* explore any torus of size $\ell \times L$, where $7 \leq \ell \leq L$ [6].

Grid-shaped networks were considered in the context of anonymous and oblivious robot exploration [1, 2, 3] for a variant of the exploration problem where robots perpetually explore all nodes in the grid. The first two works [1, 2] consider robots that are endowed with a common sense of direction (that is, common, North, South, East, and West directions), while the last paper [3] considers the same model as ours: in this case, perpetually exploring a grid is as difficult as perpetually exploring a ring, as three robots are necessary and sufficient in both cases.

1.2 Contribution

In this paper, we propose optimal (*w.r.t.* the number of robots) solutions for the deterministic terminating exploration of an anonymous grid-shaped network by a team of k asynchronous oblivious robots in the asynchronous model. Moreover, our solution are asymptotically optimal in time, as they require $\Theta(n)$ moves to explore any grid of n nodes.

In more details, we first consider the semi-synchronous model. In this model, we show that it is impossible to explore a grid of at least 3 nodes with less than 3 robots. Next, we show that it is impossible to explore a $(2, 2)$-Grid with less than 4 robots, and a $(3, 3)$-Grid with less than 5 robots, respectively. The two first results hold for both deterministic and probabilistic explorations, while the latter holds only in the deterministic case. Note also that these impossibility results naturally extend to the asynchronous model.

Then, we propose several deterministic algorithms in the asynchronous model that use the optimal number of robots allowing to explore a given grid. Our results show that except in two particular cases, 3 robots are necessary and sufficient to deterministically explore a grid of at least 3 nodes. The optimal number of robots for the two remaining cases is: 4 for the $(2, 2)$-Grid and 5 for the $(3, 3)$-Grid, respectively.

The above results show that, contrary to the case of perpetual exploration, exploring a grid and terminating is easier than exploring a ring and terminating. In the ring, deterministic solutions essentially require 5 robots [13] while probabilities enable solutions with only 4 robots [7]. In the grid, 3 robots are necessary and sufficient in all but two cases even for deterministic algorithms, the two latter cases do require 4 or 5 robots. Also, deterministically exploring a grid requires no primality condition while deterministically exploring a ring expects the number k of robots to be co-prime with n, the number of nodes.

1.3 Roadmap

Section 2 presents the system model and the problem to be solved. Lower bounds are shown in Section 3. The deterministic general solution using 3 robots is given in Section 4, the special case with 5 robots is studied in Section 5. (Note that exploring a $(2, 2)$-Grid using 4 robots is trivially possible, henceforth not considered in this paper.) Section 6 gives some concluding remarks.

2 Preliminaries

2.1 Distributed Systems

We consider systems of autonomous entities called *robots* equipped of motion actuators and visibility sensors. These robots evolve 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. In G, nodes represent locations that can be sensed by robots and edges represent the possibility for a robot to move from one location to another. We assume that G is an (ℓ, L)-*Grid* (or a Grid, for short) where ℓ, L are two positive integers such that $\ell \times L = n$, *i.e.*, G satisfies the following condition: there exists an order on the nodes of V, $v_1, \ldots, v_n$, such that

1. $\forall x \in [1..n]$, $(x \bmod \ell) \neq 0 \Rightarrow \{v_x, v_{x+1}\} \in E$, and
2. $\forall y \in [1..\ell \times (L-1)]$, $\{v_y, v_{y+\ell}\} \in E$.

We denote by $\delta(v)$ the degree of node v in G.

Nodes of the grid are *anonymous*. (We may use indices, but for notation purposes only.) Moreover, given two adjacent nodes u and v, there is no explicit or implicit labelling allowing the robots to determine whether u is either on the left, on the right, above, or below v.

Remark that an (ℓ, L)-*Grid* and a (L, ℓ)-*Grid* are isomorphic. Hence, as the nodes are anonymous, we cannot distinguish an (ℓ, L)-*Grid* from a (L, ℓ)-*Grid*. So, without loss of generality, we always consider (ℓ, L)-*Grids*, where $\ell \leq L$. Note also that any $(1, L)$-*Grid* is isomorphic to a chain. In this case, either the grid consists of one single node, or two nodes are of degree 1 and all other nodes are of degree 2. When $\ell > 1$, 4 nodes in the grid are of degree 2 and all other nodes are of degree either 3 or 4. In any grid, the nodes of smallest degree are called *corners*. In any $(1, L)$-*Grid* with $L > 1$, the unique chain linking the two corners is called the *borderline*. In any (ℓ, L)-*Grid* such that $\ell > 1$, there exist four chains $v_1, \ldots, v_m$ of length at least 2 such that $\delta(v_1) = \delta(v_m) = 2$, and $\forall x, 1 < x < m$, $\delta(v_x) = 3$, these chains are also called the *borderlines*. We call *line* (of the grid) any maximal path of the grid that is parallel to a borderline (including the borderline itself). We call *L-borderline* any borderline constituted of L nodes (*n.b.*, there are 4, 2, or 1 L-borderlines depending on whether or not $\ell > 1$ and whether or not $\ell = L$).

2.2 Robots

Operating on the grid G are $k \leq n$ (deterministic) robots. The robots do not communicate in an explicit way; however they see the position of all other robots in their ego-centered coordinate system and can acquire knowledge from this information. We assume that the robots cannot remember any previous observation nor computation performed in any previous step. Such robots are said to be *oblivious* (or *memoryless*).

Each robot operates according to its (local) *program*. In general setting, an *algorithm* is a collection of k *programs*, each one operating on one single robot. However, here we assume that robots are *uniform* and *anonymous*, *i.e.*, they all have the same program using no local parameter (such as an identity) that could permit to differentiate them. The program of a robot consists in executing *Look-Compute-Move* (*LCM*) *cycles* infinitely many times. That is, the robot first observes its environment (Look phase). Then, based on its observation and according to its program, the robot then decides to move or stay idle (Compute phase). When the robot decides to move, it moves from its current node to an adjacent node during the Move phase.

2.3 Computational Models

We consider two models: the semi-synchronous (atomic) model [8, 16] (SSYNC model [11]), and the asynchronous (non-atomic) model [14] (ASYNC model [11]). In both models, time is represented by an infinite sequence of instants 0, 1, 2, ... No robot has access to this global time. 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* whose power depends on the considered model. We are interested in algorithms that correctly operate despite the choices of the adversary. In particular, our algorithms should also work even if the adversary forces the execution to be fully sequential or fully synchronous.

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

In the asynchronous model, LCM cycles are performed asynchronously by each robot: the time between Look, Compute, and Move operations is finite yet unbounded, and is decided by the adversary. The only constraint is that both Move and Look are instantaneous.[1]

Note 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 this neighbor at some later time $t'' > t'$ when some robots are at different nodes from those previously perceived by $\mathcal{R}$ because they moved meanwhile. Hence, in the asynchronous model, robots may move based on significantly outdated perceptions.

Of course, the semi-synchronous model is stronger than the asynchronous one. So, to be as general as possible, in this paper, our impossibility results are written assuming the semi-synchronous model, while our algorithms assume the asynchronous model.

2.4 Multiplicity

We assume that during the Look phase, each 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. We consider two kinds of (global) multiplicity detection: the *weak* and *strong* (global) multiplicity detections.

Under the *weak* multiplicity detection, for every node u_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 u_i contains none, one or several robots at time instant t. If $d_i(t) = \circ$, then we say that u_i is *free* at instant t, otherwise u_i is said *occupied* at instant t. If $d_i(t) = \top$, then we say that u_i contains a *tower* at instant t.

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 *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.

As previously, to be as general as possible, our impossibility results are written assuming the strong multiplicity detection, while our algorithms assume the weak multiplicity detection.

[1] As explained in [5], this assumption is not restrictive, rather a way to simplify the reasoning.

2.5 Configurations and Views

To define the notion of *configuration*, we need to 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 time t is $d_1(t), \ldots, d_n(t)$. We denote by *initial configurations* the configurations from which the system may start at time 0. Every configuration where all robots stay idle forever is said to be *terminal* (*i.e.*, in a terminal configuration there is no possibility, even probabilistic, for a robot to move). Two configurations $d_1, \ldots, d_n$ and $d'_1, \ldots, d'_n$ are *indistinguishable* (*distinguishable* otherwise) if there exists an automorphism f on G such that $\forall v_i \in V$, we have $d_i = d'_j$ where $v_j = f(v_i)$.

The *view* of robot $\mathcal{R}$ at time t is a labelled graph isomorphic to G, where every node u_i is labelled by $d_i(t)$, except the node where $\mathcal{R}$ is currently located, this latter node u_j is labelled 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.

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 the configuration. In this case, if the robot decides to take one of these edges, it may take any of them. As in related work (*e.g.*, [10, 9, 13]), 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.

2.6 Execution

We model the executions of our algorithm in G by the list of configurations by which the system goes. So, an *execution* is a maximal list of configurations $\gamma_0, \ldots, \gamma_i$ such that $\forall j > 0$, we have:

1. $\gamma_{j-1} \neq \gamma_j$,
2. γ_j is obtained from γ_{j-1} after some robots move from their locations in γ_{j-1} to an adjacent node, and
3. for every robot $\mathcal{R}$ that moves between γ_{j-1} and γ_j, there exists $0 \leq j' \leq j$, such that $\mathcal{R}$ takes its decision to move according to its program and its view in $\gamma_{j'}$. (In semi-synchronous model, we necessarily have $j' = j - 1$.)

An execution $\gamma_0, \ldots, \gamma_i$ is said to be *sequential* if $\forall j > 0$, exactly one robot moves between γ_{j-1} and γ_j. An execution *terminates* if it eventually reaches a terminal configuration.

2.7 Exploration

We consider the *exploration* problem, where k robots, initially placed at different nodes, collectively explore an (ℓ, L)-grid before stopping moving forever. By "collectively" explore we mean that every node is eventually visited by at least one robot. More formally, an algorithm $\mathcal{P}$ *deterministically* (resp. *probabilistically*) solves the exploration problem if every execution e of $\mathcal{P}$ starting from a *towerless* configuration[2] satisfies:

1. e terminates *in finite time* (resp. *with probability 1*), and

[2]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 [10] we assume that every possible initial configuration is towerless.

2. every node is visited by at least one robot during e.

Observe that the exploration problem is not defined for $k > n$ and is straightforward for $k = n$. (In this latter case the exploration is already accomplished in the initial towerless configuration.)

3 Bounds

In this section, we first show that, except for trivial case where $k = n$, if the model is *semi-synchronous* and the multiplicity is *strong*, then at least 3 (oblivious) robots are necessary to solve the (probabilistic or deterministic) exploration of any grid (Theorem 1). Moreover, in a $(2, 2)$-Grid, 4 robots are necessary (Theorem 2). Finally, at least 5 robots are necessary to solve the deterministic exploration of a $(3, 3)$-Grid (Theorem 4). In the two next sections, we show that all these bounds are also sufficient to solve the deterministic exploration in the asynchronous model.

Given that robots are oblivious, if there are more nodes than robots, then any terminal configuration should be distinguishable from any possible initial (towerless) configuration. So, we have:

Remark 1. *Any terminal configuration of any (probabilistic or deterministic) exploration algorithm for any graph of n nodes using $k < n$ oblivious robots contains at least one tower.*

Theorem 1. *There exists no (probabilistic or deterministic) exploration algorithm in the semi-synchronous model using $k \leq 2$ oblivious robots for any graph made of at least 3 nodes.*

Proof. By Remark 1, there is no algorithm allowing one robot to explore any graph made of at least 2 nodes. Indeed, any configuration is towerless in this case. Assume by contradiction, that there exists an algorithm $\mathcal{P}$ in the semi-synchronous model to explore with 2 oblivious robots a graph made of at least 3 nodes. Consider a sequential execution e of $\mathcal{P}$ that terminates.[3] Then, e starts from a towerless configuration (by definition) and eventually reaches a terminal configuration containing a tower (by Remark 1). As e is sequential, the two last configurations of e consist of a towerless configuration followed by a configuration containing one tower. These two configurations form a possible sequential execution that terminates where only two nodes are visited, thus a contradiction. □

Any $(2, 2)$-Grid is isomorphic to a ring of 4 nodes. It is shown in [7] that no (probabilistic or deterministic) exploration using less than 4 oblivious robots is possible for any ring of size at least 4 in the semi-synchronous model. So:

Theorem 2 ([7])**.** *There exists no (probabilistic or deterministic) exploration algorithm using $k \leq 3$ oblivious robots in the semi-synchronous model for a $(2, 2)$-Grid.*

Lemma 1. *Considering any deterministic exploration algorithm $\mathcal{P}$ in the semi-synchronous model using $1 < k < n$ oblivious robots for a $(3, 3)$-Grid, there exist finite sequential executions of $\mathcal{P}$, $e = \gamma_0, \ldots, \gamma_w$, where:*

- *For every x, y with $0 \leq x < y \leq w$, γ_x and γ_y are distinguishable.*
- *Only the first configuration γ_0 is towerless.*

[3] By definition, if we consider a deterministic exploration, then all executions should terminate; while if we consider a probabilistic exploration, some executions may not terminate, even though the overall probability that such executions occur should be 0.

Proof. Consider an exploration algorithm $\mathcal{P}$ in the semi-synchronous model using $1 < k < n$ oblivious robots for some graph G of n nodes. Consider any sequential execution e of $\mathcal{P}$. By definition of the exploration, e is finite and starts from a towerless configuration. Moreover, the terminal configuration of e contains a tower, by Remark 1. Take the last towerless configuration of e and all remaining configurations that follow in e (all of them contain a tower) and form e'. The sequence e' is a possible finite sequential execution of $\mathcal{P}$ where only the first configuration is towerless. Let $e' = \alpha^0, \ldots, \alpha^m$. Let two configurations $\alpha^x = d_1^x, \ldots, d_n^x$ and $\alpha^y = d_1^y, \ldots, d_n^y$ of e', that are indistinguishable with $0 \leq x < y \leq m$. Then, by definition, there exists an automorphism f on G satisfying the additional condition: let $v_1, \ldots, v_n$ be the nodes of V, for all $s \in [1..n]$, we have $d_s^x = d_u^y$ where $v_u = f(v_s)$. Then, $\alpha^0, \ldots, \alpha^x, \beta^{y+1}, \beta^m$ is a possible finite sequential execution of $\mathcal{P}$ such that $\forall z \in [y+1..m]$, we have $\beta^z = d_{g(1)}^z, \ldots, d_{g(n)}^z$ where g is a bijection such that $\forall s \in [1..n]$, $f(v_s) = v_{g(s)}$ and $\alpha^z = d_1^z, \ldots, d_n^z$. Moreover, in $\alpha^0, \ldots, \alpha^x, \beta^{y+1}, \beta^m$, the number of configurations indistinguishable from α^x decreases by at least one. Repeating the same construction, we eventually obtain a possible finite sequential execution $e'' = \rho_0, \ldots, \rho_w$ of $\mathcal{P}$ that starts from a towerless configuration only followed by configurations containing at least one tower such that for every x, y with $0 \leq x < y \leq w$, ρ_x and ρ_y are distinguishable. □

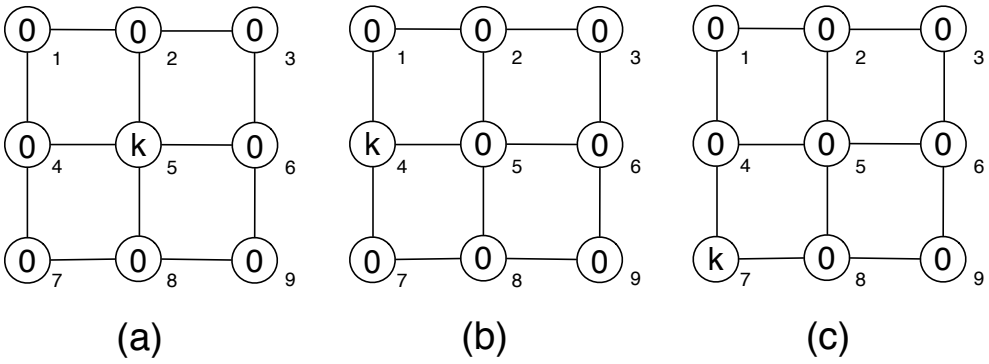

Figure 1: Three possible configurations in a $(3,3)$-Grid with a tower of k robots.

Lemma 2. *Considering any deterministic exploration algorithm $\mathcal{P}$ in the semi-synchronous model using $k > 1$ oblivious robots for a $(3,3)$-Grid, if there exists an execution of $\mathcal{P}$ $e = \gamma_0 \ldots \gamma_x \ldots$ where γ_x contains a tower of k robots, then there exists an execution e' starting with the prefix $e = \gamma_0 \ldots \gamma_x$ such that at most one new node can be visited after γ_x.*

Proof. Assume the existence of an execution of $\mathcal{P}$ $e = \gamma_0 \ldots \gamma_x \ldots$ where γ_x contains a tower of k robots. Then, γ_x is different from γ_0. Furthermore, γ_x is indistinguishable from one of the three configurations depicted in Figure 1—symbols inside the circles represent the multiplicity of the node and numbers next the circle are node's labels to help explanations only. Without loss of generality, assume that γ_x is either configuration (a), (b), or (c).

To visit a new node, one of the robots should eventually decide to move. Moreover, in γ_x, all robots have the same view. So, the adversary can choose any of them to move.

(1) Consider configuration (a). Then, all possible destinations for the robots are symmetric. So, the adversary can activate the robots in a way we retrieve configuration γ_{x-1}. Then, it can

activate robots in a way that the system return to γ_x, and so on. Hence, in this case, there exists a possible execution of $\mathcal{P}$ that is infinite, a contradiction. So, from (a), $\mathcal{P}$ cannot try to visit a new node.

(2) Consider configuration (b).
If robots synchronously move to node 5, node 5 may be unvisited. So, it is possible to visit a new node, but then we retrieve Case (1). So, we can conclude that in this case from (b) only one new node can be visited.
If robots synchronously move to node 1 (resp. 7), then this node may be unvisited. So, it is possible to visit a new node. But, in node 1, all possible destinations for the robots are symmetric. So, the adversary can activate the robots in a way that we retrieve the previous configuration, if we want to visit another node. So, as for Case (1), we can conclude that no new node can be visited, that is from (b) only one new node can be visited.

(3) Using a reasoning similar to case (1), we can conclude that from (c), $\mathcal{P}$ cannot try to visit a new node.

□

Proving Lemma 3 below is particularly tedious and error-prone because many cases must be considered (positions of robots, symmetry classes, *etc.*). The proof was thus completed as automatically as possible, by using model-checking techniques. The method is briefly sketched here, a detailed presentation, together with the source code and the necessary tools can be found on the web.[4]

Lemma 3. *Assume that there exists a deterministic exploration algorithm $\mathcal{P}$ in the semi-synchronous model using 3 oblivious robots for a $(3,3)$-Grid. Consider any suffix $\gamma_w, \ldots, \gamma_z$ of any sequential execution of $\mathcal{P}$ where:*

- *For every x, y with $w \leq x < y \leq z$, γ_x and γ_y are distinguishable.*
- *γ_w contains a tower of 2 robots.*

Then, at most 4 new nodes can be visited from γ_w before a robot of the tower moves.

Proof Outline. First, an operational model of the problem is built: this model is a reactive program that manages an abstract view of the grid and robots, according to a flow of (random) move commands. This model is restricted to the configurations relevant for the following property: an immobile two-robots tower and a mobile single robot. The reactive program (*i.e.*, the model) computes the consequences of the moves induced by the input commands; in particular, it takes trace of the *visited* nodes, and the encountered indistinguishable configuration classes. As soon as such a class has been reached twice, a flag *stuck* is raised. And, all along the execution, a *validity* flag is computed that way: *stuck* $\Rightarrow$ number of new *visited* nodes is ≤ 4. A model-checker tool is then used to check the following invariant: whatever be a sequence of input move commands, *valid* remains true. In other words, the invariance of *valid* is sufficient to establish that, starting from any configuration with a tower and a single moving robot, at most 4 new nodes can be visited before the configuration becomes indistinguishable from some already encountered configuration. Concretely, the model is written in the Lustre language [12, 15], and is itself partially generated by a "meta" program written in oCaml (which computes, in particular, the classes). The source is made of approximately 150 lines of oCaml, and 100 lines of Lustre. The invariance checking is performed by the model-checker from the Lustre distribution. □

[4] `http://www-verimag.imag.fr/~devismes/robots/`

Theorem 3. *There exists no deterministic exploration algorithm in the semi-synchronous model using $k \leq 3$ oblivious robots for a $(3,3)$-Grid.*

Proof. According to Theorem 1, we only need to consider the case of 3 robots.

Assume that there exists an exploration algorithm $\mathcal{P}$ in the semi-synchronous model for a $(3,3)$-Grid using 3 robots. By Lemma 1, there exists a (finite) sequential execution $e = \gamma_0, \ldots, \gamma_w$ that starts from a towerless configuration, only followed by configurations containing at least one towers, and such that for every x, y with $0 \leq x < y \leq w$, γ_x and γ_y are distinguishable.

In γ_0, 3 nodes are visited. The execution being sequential, no new node is visited in the first step where a tower of two robots is created. So, in γ_1, 3 nodes are visited and there exists a tower of two robots $\mathcal{R}_1$ and $\mathcal{R}_2$.

- Assume that $\mathcal{R}_1$ and $\mathcal{R}_2$ never moved after γ_1. Then, by Lemma 3, at most 4 new nodes are visited until the termination of e. So, at the termination of e, at most 7 distinct nodes have been visited, a contradiction.
- Assume that $\mathcal{R}_1$ or $\mathcal{R}_2$ eventually moved. Let γ_s the first configuration where $\mathcal{R}_1$ or $\mathcal{R}_2$ moves. From the previous case, at most 7 distinct nodes have been visited before γ_s. The execution being sequential, only one robot of the tower moves during the step from γ_s to γ_{s+1} and as in e only the first configuration is towerless, that robot moves to an occupied node. Now, the view of $\mathcal{R}_1$ and $\mathcal{R}_2$ are identical in γ_s. So, there exists an execution e' starting from the prefix $\gamma_0, \ldots, \gamma_s$ where both $\mathcal{R}_1$ and $\mathcal{R}_2$ move from γ_s to the same occupied node. As no new node is visited during the step, still at most 7 nodes are visited once the system is in the new configuration and this configuration contains a tower of 3 robots. By Lemma 2, at most one new node is visited from this latter configuration. So, at the termination of e', at most 8 distinct nodes have been visited, a contradiction.

□

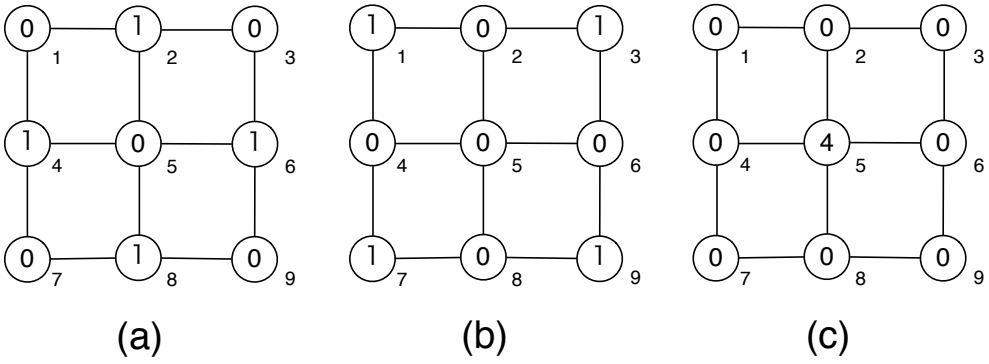

Figure 2: Three possible configurations in a $(3,3)$-Grid with 4 robots. Numbers inside the circles represent the multiplicity of the node. Numbers near the circles are node's labels that are used to ease the explanations only.

Theorem 4. *There exists no deterministic exploration algorithm in the semi-synchronous model using $k \leq 4$ oblivious robots for a $(3,3)$-Grid.*

Proof. According to Theorem 3, we only need to consider the case of 4 robots.

Assume, by the way of contradiction, that there exists an exploration algorithm $\mathcal{P}$ for a $(3,3)$-Grid with 4 robots in the semi-synchronous model.

Figure 2 depicts three possible configurations for a $(3,3)$-Grid with 4 robots. In Figure 2, symbols inside the circles represent the multiplicity of the node and numbers next the circle are node's labels to help explanations only. Note that both Configuration (a) and (b) can be initial configuration.

From now on, consider any synchronous execution of $\mathcal{P}$ (synchronous executions are possible in the semi-synchronous model) starting from configuration (a). By "synchronous" we mean that robots execute each operation of each cycle at the same time.

Configuration (a) is not a terminal configuration by Remark 1. So at least one robot move in the next Move operation. Moreover, the views of all robots are identical in (a). So, every robot moves in the next Move operation. Two cases are possible:

- Every robot moves to Node 5 and the system reaches Configuration (c). In this case, none of the corners has been visited, so Configuration (c) is not terminal and at least one robot moves in during the next Move operation. Moreover, the views of all robots are identical, so every robot moves in the next Move operation. Each robot cannot differentiate its four possible destinations. So, the adversary can choose destinations so that the system reaches configuration (a) again.
- Every robot moves to a corner node and as its view is symmetric, the destination corner is chosen be the adversary. In this case, the adversary can choose destinations so that the system reaches configuration (b). Configuration (b) being not terminal, at least one robot moves in during the next Move operation. Moreover, the views of all robots are identical, so every robot moves in the next Move operation. Each robot cannot differentiate its two possible possible destinations. So, the adversary can choose to destinations so that the system reaches configuration (a) again.

From the two previous cases, we can deduce that there exist executions of $\mathcal{P}$ that never terminates, so $\mathcal{P}$ is not an exploration algorithm, a contradiction. □

4 Deterministic solution using 3 robots

In this section, we focus on the deterministic exploration of a grid by 3 robots, in asynchronous model, and assuming weak multiplicity detection. Recall that there exists no deterministic solution for the exploration using 3 robots in a $(2,2)$- or $(3,3)$-grid assuming that model (Section 3). Moreover, exploring a $(1,3)$-grid using 3 robots is straightforward. So, we consider all remaining cases. We split our study in two cases. The deterministic solution for any (ℓ, L)-grid such that $L > 3$ is given in Subsection 4.1. The particular case of the $(2,3)$-grid is solved in Subsection 4.2.

4.1 The main algorithm

4.1.1 Overview

Our algorithm works according to the following successive three phases:

`Set-Up` **phase:** Starting from the initial (towerless) configuration, this phase aims at reaching a particular configuration called `Set-Up`, without creating any tower during the process. A

configuration is of type `Set-Up` if there is a single line of robots starting at a corner and along one of the L-borderlines of the grid, see Figure 3.

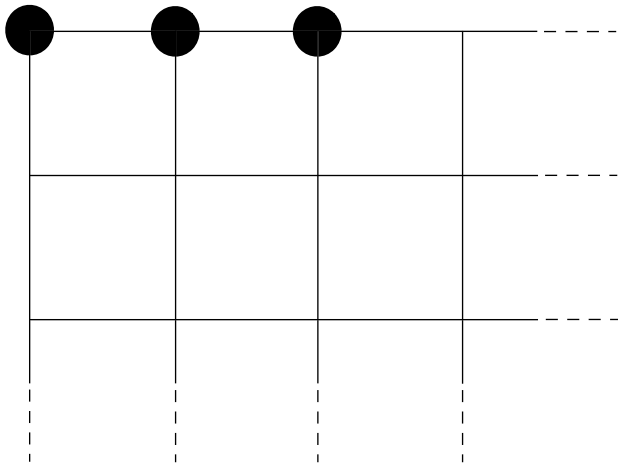

Figure 3: Configuration `Set-Up`

`Orientation` **phase:** Starting from a `Set-Up` configuration, this phase aims at giving an orientation to the grid. To achieve this, the robot that is at the corner moves to its adjacent occupied node creating then a tower. The position of the tower establishes a common coordinate system, see Figure 4. The resulting configuration is called an `Oriented` configuration.

`Exploration` **phase:** This phase starts from an `Oriented` configuration where exactly one node is occupied by one single robot, called *Explorer*. Based on the coordinate system defined during the `Orientation` phase, the explorer visits all the nodes, except three already visited ones, see Figure 5.

We now describe the three above phases in more details. We start with the two last ones. We focus then on the first phase, which is the most critical part of the algorithm.

4.1.2 Orientation Phase

This phase follows the `Set-Up` phase and consists of a single move where the robot that is at the corner move to its adjacent occupied node. Once it has moved, a tower is created. The resulting configuration is called an `Oriented` configuration, where the robots agree on a common coordinate system as shown in Figure 4. The node with coordinates $(0, 0)$ is the unique corner that is the closest to the tower. The x-axis is given by the vector linking the node $(0, 0)$ to the node where the tower is located. The y-axis is given by the vector linking the node $(0, 0)$ to its neighboring node that does not contain the tower.

The following lemma is straightforward:

Lemma 4. *Starting from a configuration of type* `Set-Up`*, the* `Orientation` *phase allows to reach a configuration of type* `Oriented` *in 1 move.*

4.1.3 Exploration Phase

This phase starts from an `Oriented` configuration. Note that, once this configuration is reached, nodes of coordinates $(0, 0)$, $(1, 0)$, and $(2, 0)$ have been necessarily visited. So, the goal is to visit all

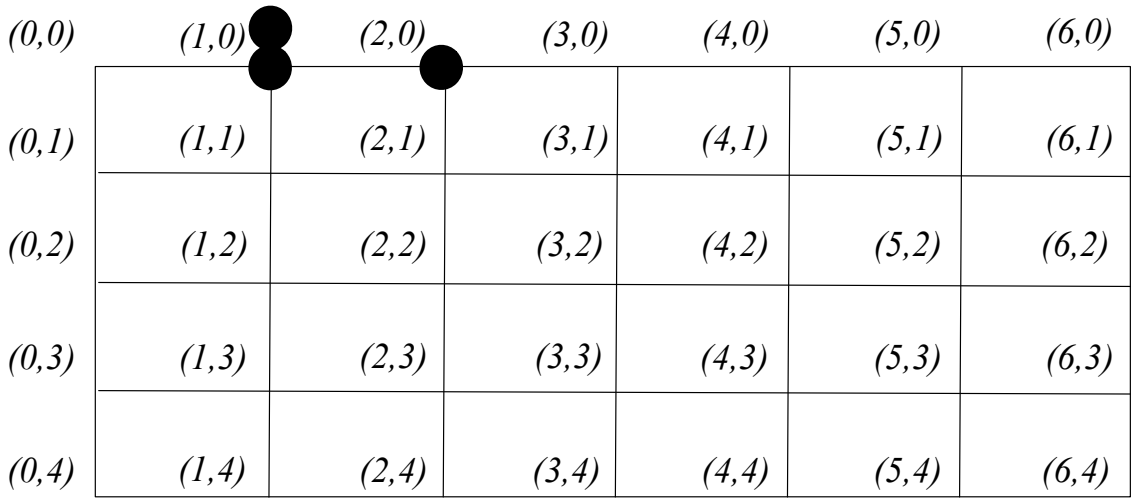

Figure 4: Coordinate system built by the `Orientation` phase.

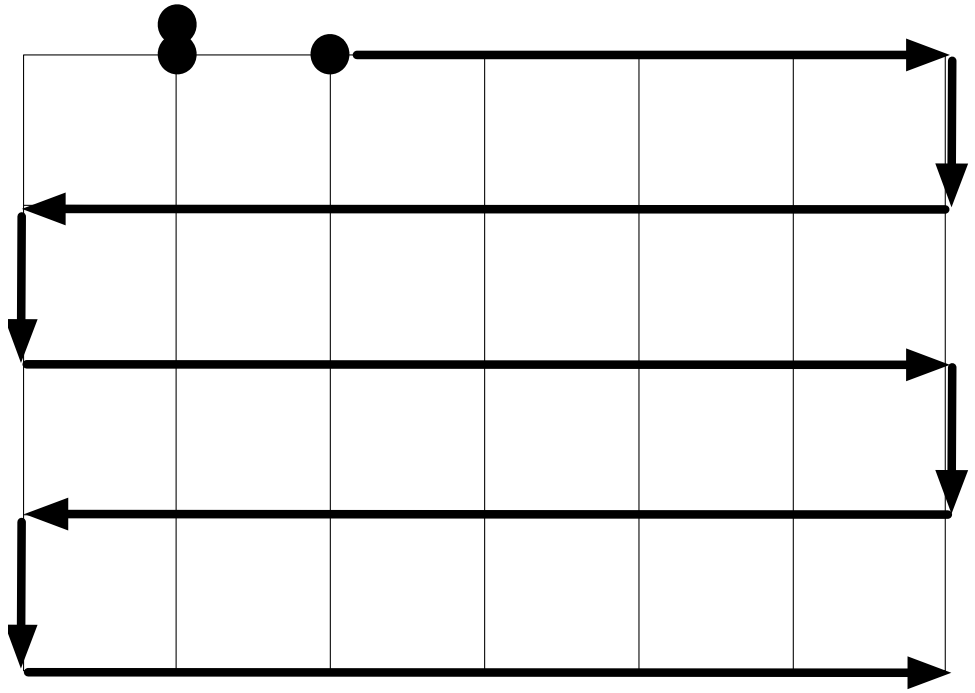

Figure 5: Exploration phase.

remaining nodes. To ensure that the exploration phase remains distinct from the previous phases and keep the coordinate system, we only authorize the robot that does not belong to the tower to move. This robot is called the *explorer*.

To explore all remaining nodes, the explorer should order all coordinates in such a way that (a) $(0,0)$ and $(0,1)$ are before its initial position (that is $(0,2)$) and all other coordinates are after; and (b) for all non-maximum coordinates (x,y), if (x',y') are successor of (x,y) in the order, then the nodes of coordinates (x,y) and (x',y') are neighbors. An example of such an order is $\preceq$, defined as follows: $(x,y) \preceq (x',y') \stackrel{\texttt{def}}{=} y < y' \vee [y = y' \wedge (x = x' \vee y \bmod 2 = 0 \wedge x < x' \vee y \bmod 2 = 1 \wedge x > x')]$

Using $\preceq$, the explorer moves as follows: While the explorer is not located at the node having the maximum coordinates according to $\preceq$, the explorer moves to the neighboring node whose coordinates are successors of the coordinates of its current position, as described in Figure 5.

The following lemma is straightforward:

Lemma 5. *Starting from an* `Oriented` *configuration, the* `Exploration` *phase is done in* $n-4$ *moves and once terminated all nodes have been visited.*

4.1.4 Set-Up Phase

Assuming $L > 3$, the `Set-Up` phase aims at reaching a `Set-Up` configuration from any towerless configuration, without creating any tower during the process. To that goal, we split the set of all non-`Set-Up` towerless configurations into several classes. The behavior of the robots is mainly determined according to these classes.

Beforehand, we define some terms and notations used in the sequel.

Definitions. We denote by $\|x,y\|$ the *Manhattan distance* between nodes x and y. By a slight abuse of notation, we also denote by $\|\mathcal{R}_1,y\|$, $\|x,\mathcal{R}_2\|$, or $\|\mathcal{R}_1,\mathcal{R}_2\|$ the Manhattan distance $\|x,y\|$ whenever $\mathcal{R}_1$ is a robot located on x and $\mathcal{R}_2$ is a robot located on y. In any configuration, we denote by $\texttt{dc}_{min}$ the minimal distance between a robot and any of the four corners. For any robot $\mathcal{R}$, we denote by $\#CC(\mathcal{R})$ the number of corners at distance $\texttt{dc}_{min}$ from $\mathcal{R}$. Conversely, we denote by $\#CR(c)$ the number of robots which are at distance $\texttt{dc}_{min}$ from c, for every corner c. When we say "the robot $\mathcal{R}$ moves toward a node x (resp. toward a robot $\mathcal{R}^\star$)" we mean that $\mathcal{R}$ moves to a neighboring node that is on a shortest path (in terms of Manhattan distance) linking the location of $\mathcal{R}$ to x (resp. to the location of $\mathcal{R}^\star$). When we say "the robot $\mathcal{R}$ moves toward the borderline *bl*" we mean that $\mathcal{R}$ moves to a neighboring node that is on a shortest path (in terms of Manhattan distance) linking the location of $\mathcal{R}$ to a closest node that belongs to *bl*.

Classes. We now split the set of all non-`Set-Up` towerless configurations into several classes.

Class `B`: every towerless configuration, which is not `Set-Up`, and where all robots are located on the same L-borderline.[5] In this case, we denote by EBL (for "Elected BorderLine") the borderline which contains the three robots. Among the configurations of `B`, we distinguish

- the configurations of type $\texttt{B}^{**}$ where a corner and its neighbor on EBL are occupied. (The second corner of EBL may be occupied too.)
- the configurations of type $\texttt{B}^{*}$ where
  - either a unique corner is occupied, but its neighbor on EBL is free, or

[5] If $\ell = 1$, then all non-`Set-Up` towerless configurations belong to Class `B`.

– the special case where $L = 5$ and EBL has two robots located on its extremities and the third one located on its middle node.

Let $\mathtt{B}^- = \mathtt{B} \setminus (\mathtt{B}^{**} \cup \mathtt{B}^*)$.

Class $\mathtt{C}_0$: every towerless configuration, which is not B, where all corners are free.

Class $\mathtt{C}_1$: every towerless configuration, which is neither Set-Up nor B, where exactly one corner is occupied. Among the configurations of $\mathtt{C}_1$, we distinguish the subclass $\mathtt{C}_1^*$ where, a second robot is on a L-borderline having an occupied corner, but not the third one. Let $\mathtt{C}_1^- = \mathtt{C}_1 \setminus \mathtt{C}_1^*$.

Class $\mathtt{C}_2$: every towerless configuration, which is not B, where two exactly corners are occupied. Among the configurations of $\mathtt{C}_2$, we distinguish the subclass $\mathtt{C}_2^*$ where the two occupied corners are extremities of the same L-borderline. Let $\mathtt{C}_2^- = \mathtt{C}_2 \setminus \mathtt{C}_2^*$.

Class $\mathtt{C}_3$: every towerless configuration, where three corners are occupied. Notice that Class $\mathtt{C}_3$ is not empty if and only if $\ell > 1$.

Remark 2. *$\{\mathtt{Set\text{-}Up}, \mathtt{B}, \mathtt{C}_0, \mathtt{C}_1, \mathtt{C}_2, \mathtt{C}_3\}$ is a disjoint partition of the set of towerless configurations.*

To simplify the design, we express below the algorithm of the Set-Up phase as a set of *prioritized* rules, *i.e.*, Rule (i) can be executed only if no Rule (j) with $j < i$ is enabled. The rules are discriminated according to the class where they apply. To accommodate the non-atomicity of the model, we try to prevent, as far as possible, situations where more than one robot is engaged to move. Actually, our algorithm never engage more than 2 robots simultaneously. Moreover:

Remark 3. *There are only two cases when two robots can be engaged simultaneously: the first case of either Rule (2) or Rule (16), respectively given in page 17 and 22.*

From Remark 3, it follows that the three robots always move sequentially (*i.e.*, one after another), except in the two above cases. However, even these two cases, we adopt a "*quasi-sequential*" behavior as follows:

Remark 4. *When two robots $\mathcal{R}_1$ and $\mathcal{R}_2$ are engaged in a configuration γ, but only one robot, say $\mathcal{R}_1$ moves, then in the reached configuration γ', no robot except $\mathcal{R}_2$ is engaged. Moreover, the only reachable configuration from γ' is the same as the configuration reached from γ when both $\mathcal{R}_1$ and $\mathcal{R}_2$ move simultaneously.*

According to Remark 3 and Remark 4, the correctness is established by showing that our algorithm actually achieves the transition system given in Figure 6.

Class B

When the current configuration belongs to this class, the Set-Up phase is in its final stage: the system leaves this class within $O(L)$ moves, and when it does, the reached configuration is of type Set-Up.

Rule (1): *If the configuration is of type $\mathtt{B}^{**}$, then let c be the corner and x its neighbor such that both c and x are occupied, the robot which is located neither on c nor on x moves toward x (on EBL).*

By definition of Rule (1), we immediately obtain the following lemma.

Lemma 6. *If the configuration is of type $\mathtt{B}^{**}$, then a configuration Set-Up is reached within at most $L - 3$ moves, and no tower is created during the process.*

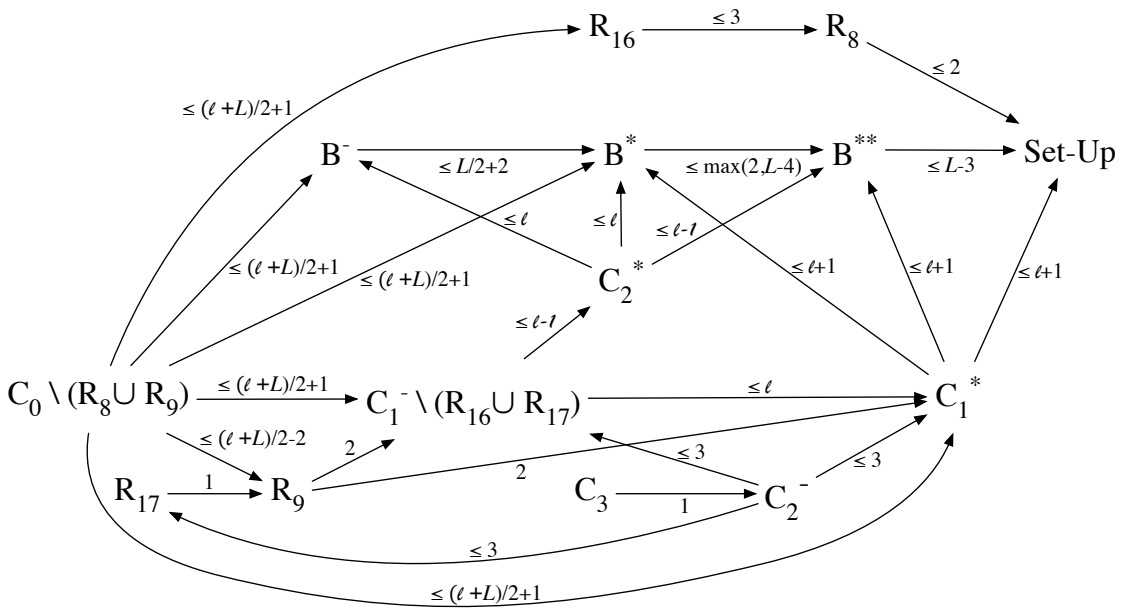

Figure 6: Transition system of the `Set-Up` phase. States represent classes of configurations, where R_i denote the class of configurations where Rule (i) is enabled. Each weighted arrow is the possibility of migrating from a class to another within a number of moves satisfying the bound given as weight.

Then, we have a special case that applies only if L is odd (in this case, $L \geq 5$). This special case is described in Figures 7 and 8 and solved by Rule (2). It allows to obtain a useful property used in Rules (5)-(7).

Rule (2): *If the configuration is of type* $\mathtt{B}$*, L is odd, and the node m in the middle of EBL is in one of the two situations described in Figures 7-8, then apply the moves depicted in the corresponding figure.*

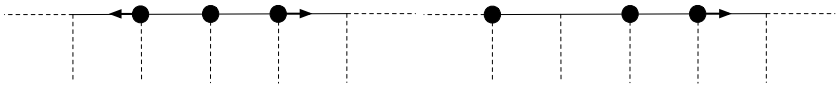

Figure 7: Case 1 for Rule (2).

Figure 8: Case 2 for Rule (2).

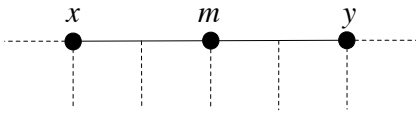

Figure 9: Result of Rule (2).

Observation 1. *If Rule (2) is enabled, then the system reaches within at most 2 moves the configuration described in Figure 9 where robots on x and y are not engaged, and no tower is created during the process.*

Rule (3): *If the configuration is of type* $\mathtt{B}^*$*, then the robot not located on a corner which is closest from an occupied corner moves toward a closest occupied corner (on EBL).*

Lemma 7. *If the configuration is of type* $\mathtt{B}^*$*, then a configuration* $\mathtt{B}^{**}$ *is reached within at most* $\max(2, L-4)$ *moves, without creating any tower during the process.*

Proof. Assume a configuration of type $\mathtt{B}^*$ with $L = 5$. Then, within at most 1 move, a configuration of type $\mathtt{B}^*$ like in Figure 9 is reached (if necessary, case 2 of Rule (2) is executed). Next, the robot m located at the middle node move using Rule (3) (its destination is chosen by the adversary), and then a configuration $\mathtt{B}^{**}$ is reached. No tower is created during these moves.

Assume now a configuration of type $\mathtt{B}^*$ with $L > 5$. Using Rule (3), a configuration $\mathtt{B}^{**}$ is reached within at most sequential $L-4$ moves, and without creating any tower during the process. □

Rule (4): *If the configuration is of type* $\mathtt{B}^-$*, no corner (of EBL) is occupied, and a robot $\mathcal{R}$ is the unique robot at distance* $\mathtt{dc}_{min}$ *from a corner,*[6] *then $\mathcal{R}$ moves (on EBL) toward its closest corner.*

If Rule 4 is enabled, $\mathtt{dc}_{min} \leq \frac{L}{2} - 2$, hence, by definition of Rule 4, follows.

Lemma 8. *If Rule 4 is enabled, then a configuration* $\mathtt{B}^*$ *is reached within at most $\frac{L}{2} - 2$ moves, and no tower is created during the process.*

We consider now the case where the configuration is of type $\mathtt{B}^-$ and Rule (2) is disabled whereas two robots are, each of them, at distance $\mathtt{dc}_{min}$ from a corner (including the case where those robots are located on the corners of EBL). In such a case, the third robots $\mathcal{R}$ allow to break ties.

First, consider the case where $\mathcal{R}$ is equidistant from the two corners of EBL. Notice that in that case, the two neighboring nodes of $\mathcal{R}$ are free since Rule (2) is disabled. Moreover, $\mathcal{R}$ can move to one of those free nodes without risking to create a tower, by Observation 1.

[6] This corner is necessarily on ABL.

Rules (5)-(7) below are, each of them, executed at most once. Rule (5) is followed by either Rule (6) or Rule (7). Rules (6) and (7) are mutually exclusive in the whole execution.

Rule (5): *If the configuration is of type* $\mathtt{B}^-$*, two robots are, each of them, at distance* $\mathtt{dc}_{min} \geq 0$ *from a corner (of EBL), and the other robot* $\mathcal{R}$ *is equidistant from the two corners of EBL, then* $\mathcal{R}$ *moves to any adjacent node on EBL (the destination node is chosen by the adversary).*

Rule (6): *If the configuration is of type* $\mathtt{B}^-$*, two robots* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *are, each of them, located on a corner (of EBL), and the other robot* $\mathcal{R}_3$ *satisfies* $\|\mathcal{R}_3, \mathcal{R}_1\| < \|\mathcal{R}_3, \mathcal{R}_2\|$*, then* $\mathcal{R}_2$ *moves to its adjacent (free) node on EBL.*

Rule (7): *If the configuration is of type* $\mathtt{B}^-$*, two robots* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *are, each of them, at distance* $\mathtt{dc}_{min} > 0$ *from a corner (of EBL), and the other robot* $\mathcal{R}_3$ *satisfies* $\|\mathcal{R}_3, \mathcal{R}_1\| < \|\mathcal{R}_3, \mathcal{R}_2\|$*, then* $\mathcal{R}_1$ *moves (on EBL) toward its closest corner.*

Lemma 9. *For any configuration* $\mathtt{B}^-$*, a configuration of type* $\mathtt{B}^*$ *is reached within at most* $\frac{L}{2} + 2$ *moves, without creating any tower during the process.*

Proof. Let γ be any configuration of type $\mathtt{B}^-$. If Rule 4 is enabled, we are done by Lemma 8. Otherwise, consider the following two cases:

1. Assume that two corners are occupied in γ. Then, $L > 5$ and a configuration of type $\mathtt{B}^*$ is reached within at most 2 moves (*i.e.*, Rule (6), maybe preceded by Rule (5).
2. Assume that two robots are, each of them, at distance $\mathtt{dc}_{min} > 0$ from a corner in γ. Then, within at most 4 moves (*n.b.*, the worst case consists in the following sequence: Rule (2) twice, Rule (5), and finally Rule (7)), either a configuration of type $\mathtt{B}^*$ or a configuration where Rule (4) is enabled is reached. In the latter case, up to $\frac{L}{2} - 2$ moves may be necessary to reach a configuration of type $\mathtt{B}^*$, by Lemma 8.

In all cases, no tower is created during the process, thanks to the property given in Observation 1. □

Class $\mathtt{C}_0$

Except for Rule (8) which allows to directly reach a configuration `Set-Up`, the rules below aim at migrating to either Class $\mathtt{C}_1$, Class $\mathtt{B}^-$, or $\mathtt{B}^*$, within at most $\frac{\ell+L}{2} + 1$ moves.

We begin by considering two special cases (Rules (8) and (9)) that help to solve the convergence from Class $\mathtt{C}_1$ when $\ell = L$ (*n.b.*, in that case, $L \geq 4$), refer to Rules (16) and (17).

Rule (8): *If the configuration is of type* $\mathtt{C}_0$*,* $\ell = L$*, and robots are placed like one the two situations described in Figures 10-13, apply the move depicted in corresponding figure.*

Notice that in Cases 1-3, the execution of Rule (8) leads to a configuration where Case 4 applies, hence follows.

Lemma 10. *If Rule (8) is enabled, then a configuration* `Set-Up` *is reached within at most* 2 *moves, and no tower is created during the process.*

The following rule can be executed at most once.

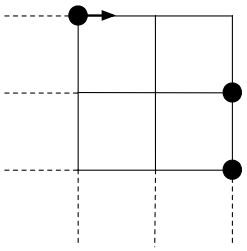

Figure 10: Case 1 for Rule (8) ($\ell = L$)

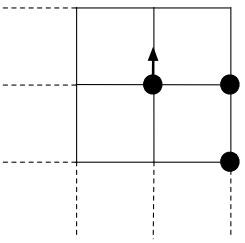

Figure 11: Case 2 for Rule (8) ($\ell = L$)

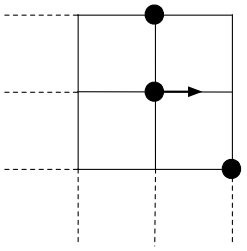

Figure 12: Case 3 for Rule (8) ($\ell = L$)

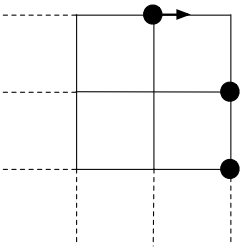

Figure 13: Case 4 for Rule (8) ($\ell = L$)

Rule (9): *If the configuration is of type* $\mathtt{C}_0$, $\ell = L$, *and exactly one robot* $\mathcal{R}_1$ *is a distance 1 from a corner, say* c, *and* $\|\mathcal{R}_2, c\| = \|\mathcal{R}_3, c\|$ *where* $\mathcal{R}_2$ *and* $\mathcal{R}_3$ *are the two other robots, then the robot, among* $\mathcal{R}_2$ *and* $\mathcal{R}_3$, *which is the closest from the borderline containing* $\mathcal{R}_1$, *moves toward* c *(the adversary breaks ties, if necessary).*

We now discriminate the remaining configurations $\mathtt{C}_0$ according to the number of robots at distance $\mathtt{dc}_{min}$ from a corner.

Rule (10): *If the configuration is of type* $\mathtt{C}_0$ *and there is a unique robot* $\mathcal{R}$ *at distance* $\mathtt{dc}_{min}$ *from a corner, then* $\mathcal{R}$ *moves toward a closest corner (the adversary breaks ties, if necessary).*

Lemma 11. *After an execution of Rule (9), either a configuration* $\mathtt{C}_1^*$ *or a configuration* $\mathtt{C}_1^-$ *where both Rules (16) and (17) are disabled is reached within 1 move.*

Proof. If Rule (9) is enabled, then Rule (8) is disabled and so both $\mathcal{R}_2$ and $\mathcal{R}_3$ are at distance at least 3 from c, and consequently, all their neighboring nodes are free. After the execution of Rule (9), Rule (10) is executed and the system reaches either a configuration $\mathtt{C}_1^*$ or a configuration $\mathtt{C}_1^-$. Now, if the reached configuration is of type $\mathtt{C}_1^-$, then

- the two robots which are not located on the unique occupied corner c are at different distances from c thanks to Rule (9),so Rule (17) is disabled, and
- one of them is at least at distance 3 from c as explained before, so Rule (16) is disabled.

Finally, since the two aforementioned moves are sequential, we can see that no tower is created during the process. □

The following four rules are executed at most once. More precisely, Rule (11) is necessarily followed by Rule (12), Rule (12) is necessarily followed by Rule (13), and Rules (11)-(13) and Rule (14) are mutually exclusive in the whole execution.

If exactly two robots are at distance $\mathtt{dc}_{min}$ from a corner, then the third robot is used to break ties.

Rule (11): *If the configuration is of type* $\mathtt{C}_0$*, there are exactly two robots* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *at distance* $\mathtt{dc}_{min}$ *from a corner,* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *are on the same line, and the third robot* $\mathcal{R}_3$ *is neighbor of both* $\mathcal{R}_1$ *and* $\mathcal{R}_2$*, then* $\mathcal{R}_3$ *moves to a neighboring free node (the adversary breaks ties, if necessary).*

Rule (12): *If the configuration is of type* $\mathtt{C}_0$*, there are exactly two robots* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *at distance* $\mathtt{dc}_{min}$ *from a corner, and* $\|\mathcal{R}_1, \mathcal{R}_3\| = \|\mathcal{R}_2, \mathcal{R}_3\|$*, where* $\mathcal{R}_3$ *is the third robot, then* $\mathcal{R}_3$ *moves to a neighboring free node* x *such that* $\|\mathcal{R}_1, x\| \neq \|\mathcal{R}_2, x\|$ *(the adversary breaks ties, if necessary).*

Rule (13): *If the configuration is of type* $\mathtt{C}_0$*, there are exactly two robots* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *at distance* $\mathtt{dc}_{min}$ *from a corner, but one of them, say* $\mathcal{R}_1$*, satisfies* $\|\mathcal{R}_1, \mathcal{R}_3\| < \|\mathcal{R}_2, \mathcal{R}_3\|$*, where* $\mathcal{R}_3$ *is the third robot, then* $\mathcal{R}_1$ *moves toward its closest corner (the adversary breaks ties, if necessary).*

If the three robots are at distance $\mathtt{dc}_{min}$ from a corner, we use $\#CC()$ and $\#CR()$ to "elect" one robot using Function *Leader*() given in Algorithm 1. In this case, $\#CC(\mathcal{R}) \geq 1$ for every robot $\mathcal{R}$. Notice also that the configuration being towerless, $\#CC(\mathcal{R}) \leq 2$ for every robot $\mathcal{R}$. Moreover, by definition, $\#CR(c) \leq 3$ for every corner c.

Rule (14): *If the configuration is of type* $\mathtt{C}_0$ *and all robots are at distance* $\mathtt{dc}_{min}$ *from a corner, then Leader() (see Algorithm 1) moves toward its closest corner.*

Lemma 12. *If the configuration is of type* $\mathtt{C}_0$ *and Rules (8) and (9) are disabled, then one of the two following situations occurs.*

1. *A configuration* γ *of type either* $\mathtt{C}_1$*,* $\mathtt{B}^-$*, or* $\mathtt{B}^*$ *is reached within at most* $\frac{\ell+L}{2}+1$ *moves. Moreover, in* γ*, if* $\ell = L$*, then the two robots* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *that are not on the unique occupied corner* c *satisfy* $\|\mathcal{R}_1, c\| \neq \|\mathcal{R}_2, c\|$*. Consequently, Rule (17) is disabled in* γ*.*
2. *A configuration of type* $\mathtt{C}_0$ *where Rule (9) is enabled is reached within at most* $\frac{\ell+L}{2}-2$ *moves.*

In both cases, no tower is created during the process.

Proof. In Class $\mathtt{C}_0$, moves are sequential only. So, if the system exits from $\mathtt{C}_0$, then it reaches either $\mathtt{C}_1$, $\mathtt{B}^-$, or $\mathtt{B}^*$ without creating tower during the process, by definition of Rules (8)-(14). Moreover, Rules (11)-(13) and Rule (14) are mutually exclusive, all of them always preceded Rule (10).

Assume Rule (13) is eventually executed. Then, Rule (13) may be preceded by at most two moves (Rules (11)-(12)). After Rule (13), Rule (10) becomes enabled and $\mathtt{dc}_{min} \leq \frac{\ell+L}{2} - 2$. Executing Rule (10), some robot $\mathcal{R}$ moves and Rule (9) cannot becomes enabled because Rule (13) has been executed before Rule (10), meaning the two robots other than $\mathcal{R}$ are always at different distance from the corner targeted by $\mathcal{R}$. So, after at most $\frac{\ell+L}{2}-2$ moves of Rule (10), a configuration of type either $\mathtt{C}_1$, $\mathtt{B}^-$, or $\mathtt{B}^*$ is reached: Case 1 of the lemma is satisfied.

Assume Rule (13) is never executed. Rule (14) may be executed once before Rule (10) becomes enabled. When Rule (10) is enabled, $\mathtt{dc}_{min} \leq \frac{\ell+L}{2} - 2$. Executing Rule (10), some robot $\mathcal{R}$ moves and a configuration where Rule (9) is enabled may be reached 1 move before a robot reaches a corner. So, either Rule (9) becomes enabled after at most one move of Rule (14) and at most $\frac{\ell+L}{2} - 3$ moves of Rule (10): case 2 of the lemma is satisfied. Or, Rule (9) never becomes enabled, meaning the two robots other than $\mathcal{R}$ are always at different distance from the corner targeted by

Algorithm 1 *Leader*()

1: **if** there is a unique robot $\mathcal{R}$ such that $\#CC(\mathcal{R}) = 1$ **then**
2: **return** $\mathcal{R}$
3: **else if** there is a unique robot $\mathcal{R}$ such that $\#CC(\mathcal{R}) = 2$ **then**
4: **return** $\mathcal{R}$
5: **else if** every robot $\mathcal{R}$ satisfies $\#CC(\mathcal{R}) = 1$ **then**
6: **if** there is a unique corner c such that $\#CR(c) = 1$ **then**
7: **return** $\mathcal{R}$, such that $\mathcal{R}$ is at distance $\mathtt{dc}_{min}$ from c
8: **else if** every corner c satisfies $\#CR(c) \leq 1$ **then**
9: Let c_1, c_2, and c_3 be the three corners such that $\#CR(c_i) = 1, \forall i \in \{1, 2, 3\}$.
10: Let c be the unique corner in $\{c_1, c_2, c_3\}$ which is the common extremity of two borderlines,
each one having one of the two other corners of $\{c_1, c_2, c_3\} \setminus \{c\}$ as other extremity.
11: Let $\mathcal{R}$ be the robot at distance $\mathtt{dc}_{min}$ from c.
12: **return** $\mathcal{R}$
13: **else**
14: Let c be the unique corner such that $\#CR(c) = 3$.
15: Let SRE be the smallest rectangle that encloses c and the three robots.
16: Let $\mathcal{R}$ be the unique robot which is not on the boundary of SRE.
17: **return** $\mathcal{R}$
18: **end if**
19: **else**
20: /* *Every robot* $\mathcal{R}$ *satisfies* $\#CC(\mathcal{R}) = 2$ */
21: There is a unique line ul that contains exactly two robots.
22: Let $\mathcal{R}$ be the robot which is not on ul.
23: **return** $\mathcal{R}$
24: **end if**

$\mathcal{R}$, and after at most one move of Rule (14) and at most $\frac{\ell+L}{2}-2$ moves of Rule (10): a configuration of type either $\mathtt{C}_1$, $\mathtt{B}^-$, or $\mathtt{B}^*$ is reached: Case 1 of the lemma is satisfied. □

Class $\mathtt{C}_1$

In this class, we have two particular cases. The first one (Rule (16)) allows to reach a configuration `Set-Up` within at most 5 moves. The second one (Rule (17)) is more intricate. Indeed, using Rule (17) the system migrates within one move to Class $\mathtt{C}_0$, and then goes back to a configuration γ of $\mathtt{C}_1$ within $O(\ell + L)$ moves, however from γ Rule (17) remains disabled forever. All other rules allow the system to migrate to Classes either `Set-Up`, $\mathtt{B}^*$, $\mathtt{B}^{**}$, or $\mathtt{C}_2^*$ within $O(\ell)$ moves (from this latter class, a configuration `B` is reached within $O(\ell)$ moves).

Rule (15): *If the configuration is of type $\mathtt{C}_1^*$, then the robot which is not on the L-borderline lbl containing two robots moves a neighboring free non-corner node toward the closest free non-corner node on lbl (the adversary breaks ties, if necessary).*

Lemma 13. *From Class $\mathtt{C}_1^*$, the system reaches either Class `Set-Up`, Class $\mathtt{B}^*$ or Class $\mathtt{B}^{**}$ within at most $\ell + 1$ moves, and without creating any tower during the process.*

Proof. Using Rule (15), always the same robot moves toward free nodes, without creating any tower until a configuration of either Class $\mathtt{B}^*$ or Class $\mathtt{B}^{**}$ is reached. Moreover, the robot may have to circumvent a robot on *lbl*, so in the worst case at most $\ell + 1$ moves are necessary to reach a configuration of either Class $\mathtt{B}^*$ or Class $\mathtt{B}^{**}$. □

We now consider two special cases when $\ell = L$ (in this case, $L \geq 4$).

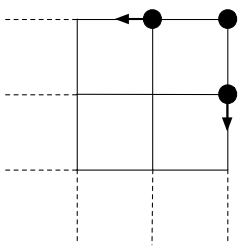

Figure 14: Case 1 for Rule (16) ($\ell = L$)

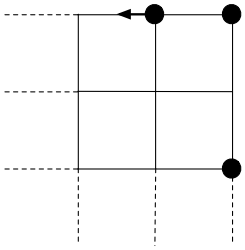

Figure 15: Case 2 for Rule (16) ($\ell = L$)

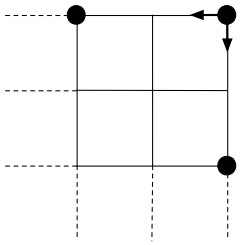

Figure 16: Case 3 for Rule (16) ($\ell = L$)

Rule (16): *If the configuration is of type $\mathtt{C}_1^-$, $\ell = L$, and robots are placed like one the two situations described in Figures 14-16, apply the move depicted in corresponding figure.*

Notice that in Case 3 (Figure 16), the symmetric is broken by the adversary. From the definition of Rule (16), we immediately deduce the following lemma.

Lemma 14. *If Rule (16) is enabled, then a configuration $\mathtt{C}_0$ where Rule (8) (Case 1) is enabled is reached within at most 3 moves, and no tower is created during the process.*

The following rule can be executed at most once.

Rule (17): *If the configuration is of type* $\mathtt{C}_1^-$, $\ell = L$, *and* $\|\mathcal{R}_3, \mathcal{R}_1\| = \|\mathcal{R}_2, \mathcal{R}_1\|$ *where* $\mathcal{R}_1$ *is the unique robot located at a corner and* $\mathcal{R}_2$ *and* $\mathcal{R}_3$ *are the two other ones, then* $\mathcal{R}_1$ *moves to a neighboring node (the adversary breaks the symmetry).*

Notice that, since Rule (16) is disabled, if Rule (17) is enabled, then the two robots not located on the unique occupied corner c are at least at distance 3 from c. In particular, the two neighbors of c are free. From the definition of Rule (17), we immediately deduce the following lemma.

Lemma 15. *If Rule (17) is enabled, then a configuration* $\mathtt{C}_0$ *where Rule (9) is enabled is reached within 1 move, and no tower is created during the process.*

Rules (18-20) below are executed, each of them, at most once. Moreover, they are mutually exclusive in the whole execution.

Rule (18): *If the configuration is of type* $\mathtt{C}_1^-$, $\ell < L$, *and* $\|\mathcal{R}_3, \mathcal{R}_1\| = \|\mathcal{R}_2, \mathcal{R}_1\|$ *where* $\mathcal{R}_1$ *is the unique robot located at a corner and* $\mathcal{R}_2$ *and* $\mathcal{R}_3$ *are the two other ones, then the robot, among* $\mathcal{R}_2$ *and* $\mathcal{R}_3$*, which is the closest from the L-borderline lbl having* $\mathcal{R}_1$ *located at one corner, moves to a neighboring free node toward the closest free node of lbl.*

If Rule (18) is enabled, then within on move the system reaches either Class $\mathtt{C}_1^*$, Class $\mathtt{C}_2^*$, a configuration of type $\mathtt{C}_1^-$ where Rule (21) is enabled.

Rule (19): *If the configuration is of type* $\mathtt{C}_1^-$ *and* $\|\mathcal{R}_3, \mathcal{R}_1\| > \|\mathcal{R}_2, \mathcal{R}_1\|$ *where* $\mathcal{R}_1$ *is the unique robot located at a corner and* $\mathcal{R}_2$ *and* $\mathcal{R}_3$ *are the two other ones, but both* $\mathcal{R}_2$ *and* $\mathcal{R}_3$ *are on two L-borderlines having an occupied corner*[7]*, then* $\mathcal{R}_2$ *moves to its neighboring node outside any borderline.*

An execution of Rule (19) leads to a configuration is type $\mathtt{C}_1^*$.

Rule (20): *If the configuration is of type* $\mathtt{C}_1^-$ *and the robots are in the situation described in Figure 17, then apply the move depicted in the figure.*

If Rule (20) is enabled, then $\ell > L$ and the borderline bl containing only one robot is a L-borderline (otherwise, the configuration is of type $\mathtt{C}_1^*$), and consequently bl contains at least 4 nodes ensuring that the destination exists.

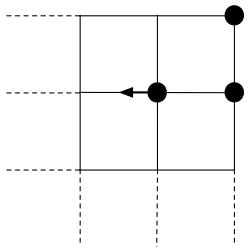

Figure 17: Rule (20), the configuration is of type $\mathtt{C}_1^-$

Rule (21): *If the configuration is of type* $\mathtt{C}_1^-$ *and* $\|\mathcal{R}_3, \mathcal{R}_1\| > \|\mathcal{R}_2, \mathcal{R}_1\|$ *where* $\mathcal{R}_1$ *is the unique robot located at a corner and* $\mathcal{R}_2$ *and* $\mathcal{R}_3$ *are the two other ones, then* $\mathcal{R}_2$ *moves to a free*

[7]This case can only occur if $\ell = L$.

neighboring node x such that (a) $\|\mathcal{R}_3, \mathcal{R}_1\| > \|x, \mathcal{R}_1\|$ and (b) x is on a shortest path (in terms of Manhattan distance) to a closest free node on a L-borderline having an occupied corner (the adversary breaks ties, if necessary).

Lemma 16. *From any configuration $\mathtt{C}_1^-$ where both Rule (16) and Rule (17) are disabled, a configuration of type either $\mathtt{C}_1^*$, or $\mathtt{C}_2^*$ is reached within, respectively, at most ℓ moves and at most $\ell - 1$ moves. Moreover, no tower is created during the process.*

Proof. Consider any configuration γ of type $\mathtt{C}_1^-$ where both Rule (16) and Rule (17) are disabled and study the following cases:

- *Assume Rule (18) is enabled in γ.* After the execution of Rule (18), the system is either Class $\mathtt{C}_1^*$, Class $\mathtt{C}_2^*$, a configuration of type $\mathtt{C}_1^-$ where Rule (21) is enabled. In the latter case, the fact that Rule (19) is disabled and the configuration is of type $\mathtt{C}_1^-$ implies that neither $\mathcal{R}_2$ nor $\mathcal{R}_3$ are on a L-borderline having one occupied corner. Moreover, since Rule (20) is disabled, there is a free node x satisfying Conditions (a) and (b). So, the unique robot that moves using Rule (21) always move to a free node and is at distance at most $\ell - 2$ from a free node of the L-borderline having an occupied corner. Hence, overall, in at most $\ell - 1$ moves, the system reaches a configuration of type either $\mathtt{C}_1^*$ or $\mathtt{C}_2^*$, without creating any tower during the process.
- *Assume Rule (19) is enabled in γ.* Within the next move, the system reaches a configuration of type $\mathtt{C}_1^*$ and we are done.
- *Assume Rule (20) is enabled in γ.* In this case, $\ell \geq 3$ and in three sequential moves (Rule (20) followed by Rule (21) twice), the system reaches a configuration of type $\mathtt{C}_1^*$ without creating any tower during the process, and we are done.
- *Assume Rule (21) is enabled in γ.* In this case, the fact that Rule (19) is disabled and the configuration is of type $\mathtt{C}_1^-$ implies that neither $\mathcal{R}_2$ nor $\mathcal{R}_3$ are on a L-borderline having one occupied corner. Moreover, since Rule (20) is disabled, there is a free node x satisfying Conditions (a) and (b). So, until a configuration of type $\mathtt{C}_1^*$ or $\mathtt{C}_2^*$ is reached, the same robot move using Rule (21) toward a free node, so no tower is created until a configuration of type $\mathtt{C}_1^*$ or $\mathtt{C}_2^*$. Moreover, we have two cases: either the final destination of the moving robot is a corner and in this case, a configuration of type $\mathtt{C}_2^*$ is reached within at most $\ell - 1$ moves; otherwise, a configuration of type $\mathtt{C}_1^*$ is reached within at most ℓ moves (the robot may have to circumvent an occupied corner).

□

Class $\mathtt{C}_2$

The aim here is to migrate to either $\mathtt{C}_1$ (within at most 3 moves) or $\mathtt{B}$ (within at most ℓ moves).

Rule (22): *If the configuration is of type $\mathtt{C}_2^*$, then the robot which is not located on a corner moves to a free neighboring node toward the closest free node located on the L-borderline having two occupied corners.*

By definition of Rule (22), follows.

Lemma 17. *If the configuration is of type $\mathtt{C}_2^*$, then a configuration of type Class $\mathtt{B}$ is reached within at most $\ell - 1$ moves, without creating any tower during the process.*

In the remaining cases, the robot which is not on a corner allows to break ties.

Rule (23): *If the configuration is of type* $\mathtt{C}_2^-$ *and* $\|\mathcal{R}_3, \mathcal{R}_1\| = \|\mathcal{R}_3, \mathcal{R}_2\|$ *where* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *are the two robots located on a corner and* $\mathcal{R}_3$ *the other one, and* $\mathcal{R}_3$ *is neighbor of two corners, then* $\mathcal{R}_3$ *moves to its unique non-corner neighbor.*

Rule (24): *If the configuration is of type* $\mathtt{C}_2^-$ *and* $\|\mathcal{R}_3, \mathcal{R}_1\| = \|\mathcal{R}_3, \mathcal{R}_2\|$ *where* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *are the two robots located on a corner and* $\mathcal{R}_3$ *the other one, then* $\mathcal{R}_3$ *moves to a neighboring free node* x *such that* x *is not a corner and* $\|x, \mathcal{R}_1\| \neq \|x, \mathcal{R}_2\|$ *(the adversary breaks ties, if necessary).*

Rule (25): *If the configuration is of type* $\mathtt{C}_2^-$ *and* $\|\mathcal{R}_3, \mathcal{R}_1\| \neq \|\mathcal{R}_3, \mathcal{R}_2\|$ *where* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *are the two robots located on a corner and* $\mathcal{R}_3$ *the other one, then the robot, among* $\mathcal{R}_1$ *and* $\mathcal{R}_2$*, which is the farthest from* $\mathcal{R}_3$ *moves to a neighboring non-corner free node (the adversary breaks ties if necessary).*

Lemma 18. *If the configuration is of type* $\mathtt{C}_2^-$*, then a configuration of type* $\mathtt{C}_1$ *where Rule (16) is disabled is reached within at most 3 moves, without creating any tower during the process.*

Proof. Each of the three rules (23), (24), and (25) are executed at most once and are sequential. So, from their definition, we can deduce that their execution do not create any tower.

Moreover, Rule (23) is necessarily followed by Rule (24), and Rule (24) is necessarily followed by Rule (25). Finally, when one of these rules is enabled, the configuration is not of type $\mathtt{C}_2^*$ and as using these rules, the robots located at corners do not move, the configuration reached after executing these rules cannot be of type $\mathtt{B}$. That is, the reached configuration is necessarily of type $\mathtt{C}_1$. Moreover, in this latter configuration, Rule (16) is disabled. Indeed, if $\ell = L$, then the two occupied corners are not extremities of the same borderline and so the unique robot that occupies a corner after at most 3 moves remains at distance at least 5 from the robot that has just left a corner. □

Class $\mathtt{C}_3$

The aim here is to migrate in 1 move to Class $\mathtt{C}_2^-$.

Rule (26): *If the configuration is of type* $\mathtt{C}_3$*, then the robot located on the occupied corner* c *that is the common extremity of two borderlines, each one having its two extremities occupied, moves toward an adjacent free node (the adversary breaks ties, if necessary).*

By definition of Rule (26), follows.

Lemma 19. *If the configuration is of type* $\mathtt{C}_3$*, then a configuration of type* $\mathtt{C}_2^-$ *is reached within 1 move, without creating any tower during the process.*

The transition system given in Figure 6 summarizes Lemmas 6-19. Its states represent a partition of the set of all towerless configurations (see Remark 2). Moreover, this system is acyclic and contains only one sink: the class `Set-Up`. So, we can deduce from any towerless configuration the system reaches within a finite number of moves a configuration of type `Set-Up`. Finally, we can remark that the weight of any directed path to `Set-Up` is in $O(\ell + L)$. Hence, follows.

Lemma 20. *Starting from any towerless configuration, the* `Set-Up` *phase allows to converge to a configuration of type* `Set-Up` *in* $O(\ell + L)$ *moves.*

By Lemmas 6-19, no tower is created during the `Set-Up` phase, hence Phases `Set-Up`, `Orientation`, and `Exploration` are unambiguous, and by Lemmas 4-20, we can deduce the following theorem.

Theorem 5. *The three phases* `Set-Up`*,* `Orientation`*, and* `Exploration` *allows to deterministically explore any (ℓ, L)-Grid such that $L > 3$ in $\Theta(n)$ moves using 3 robots, where $n = \ell \times L$ is the size of the grid.*

4.2 Exploring a (2,3)-Grid

The idea for the $(2, 3)$-Grid is quite simple. Let us consider the two L-borderlines of the grid. Since there are initially three isolated robots on the grid, there exists one of the two L-borderlines, say *lbl*, that contains either all the robots or exactly two robots. In the second case, the robot that is not part of *lbl* moves to the adjacent free node toward the free node of *lbl*. Thus, the three robots are located on *lbl* after at most 3 moves. Next, the robot not located at any corner moves to one of its two neighboring occupied nodes. (The destination is chosen by the adversary.) Thus, a tower is created after one additional move. Once the tower is created, the grid is oriented. Then, the single robot moves to the adjacent free node in the L-borderline that does not contain the tower. Next, it explores the nodes of this line by moving along the line toward the tower. When it becomes neighbor of the tower (after 3 moves), it stops and all the nodes of the $(2, 3)$-Grid have been explored.

Theorem 6. *In the asynchronous model, the deterministic exploration of a $(2, 3)$-Grid can be solved in at most 7 moves by 3 oblivious robots.*

5 Deterministic solution for a (3,3)-grid using 5 robots

In this section, we propose an algorithm that explores a $(3, 3)$-Grid using 5 asynchronous robots, assuming weak multiplicity detection. The algorithm works within two phases executed in sequence: the `Set-Up` phase, followed by the `Exploration` phase.

5.1 `Set-Up` Phase

The `Set-Up` phase starts from any towerless configuration. Let U be a maximal set of towerless configurations such that every two configurations of U are distinguishable.

First, since the number of robots and the size of the grid are small, we have automatically computed the set U of $\binom{9}{5} = 126$ possible towerless configurations. Then, for every pair of configurations g_1, g_2 in U, we have removed g_2 from U if g_1 and g_2 are indistinguishable. This process led to the set of 23 configurations shown in Figure 18.[8]

Observation 2. *Any set U contains exactly 23 configurations given in Figure 18.*

Figure 18 also shows the transition diagram of the `Set-Up` phase, with the robot's moves. The goal of `Set-Up` consists in reaching one of the configurations in `P-E` $\subseteq U$. They are called `Pre-Exploration` configurations, from which the `Exploration` phase starts. `P-E` $= \{$⑫, ⑬, ⑭, ⑮, ⑲, ⑳, ㉒, ㉓$\}$, where each ⓧ represents the ID of the configuration in Figure 18. The configuration in grey belongs to `P-E`, the set of `Pre-Exploration` configurations. The little arrows represent the possible

[8]The source code, written is `C`, can be found at `http://www-verimag.imag.fr/~devismes/robots/`.

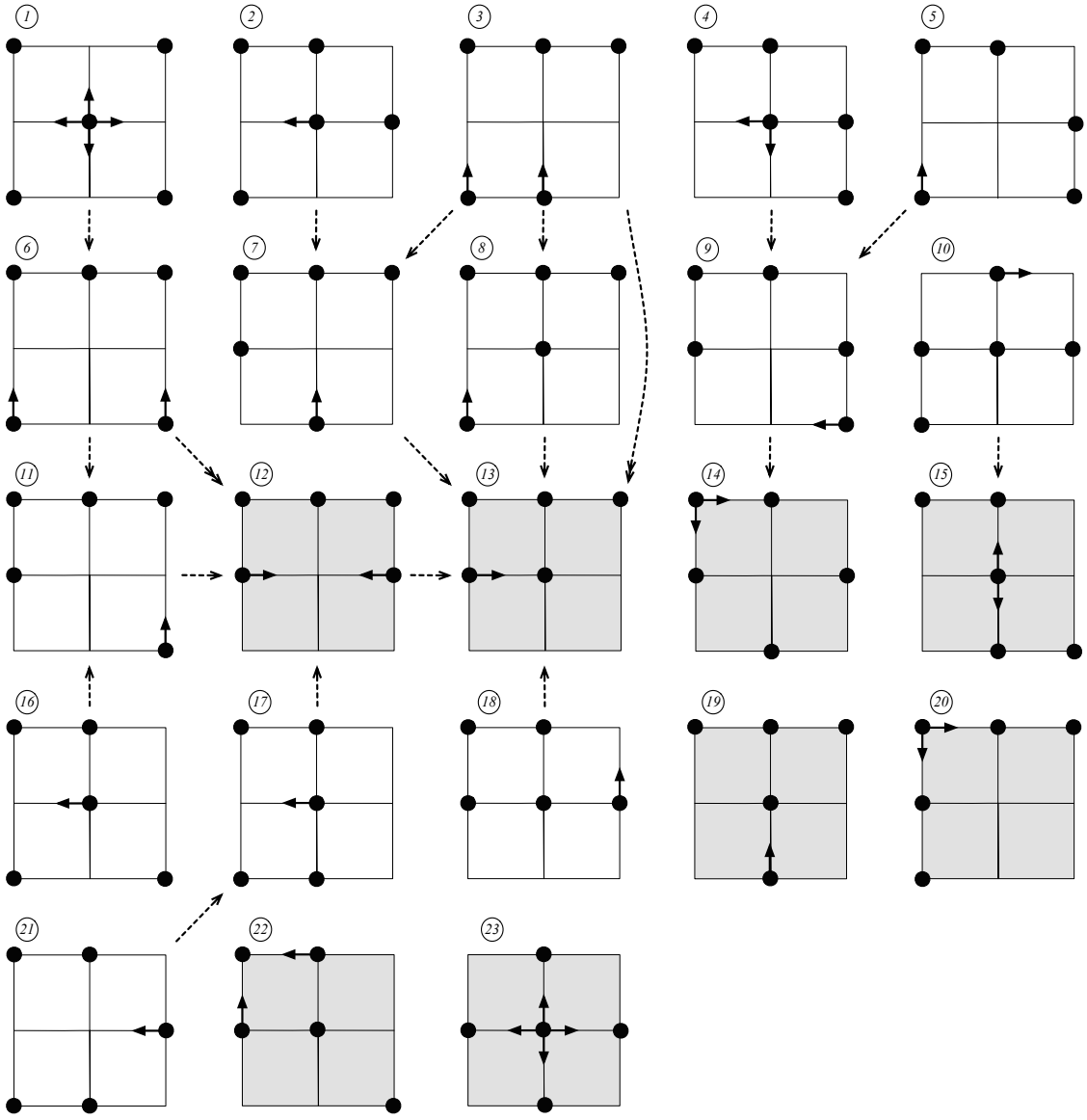

Figure 18: `Set-Up` built over U, a maximal set of distinguishable towerless configurations.

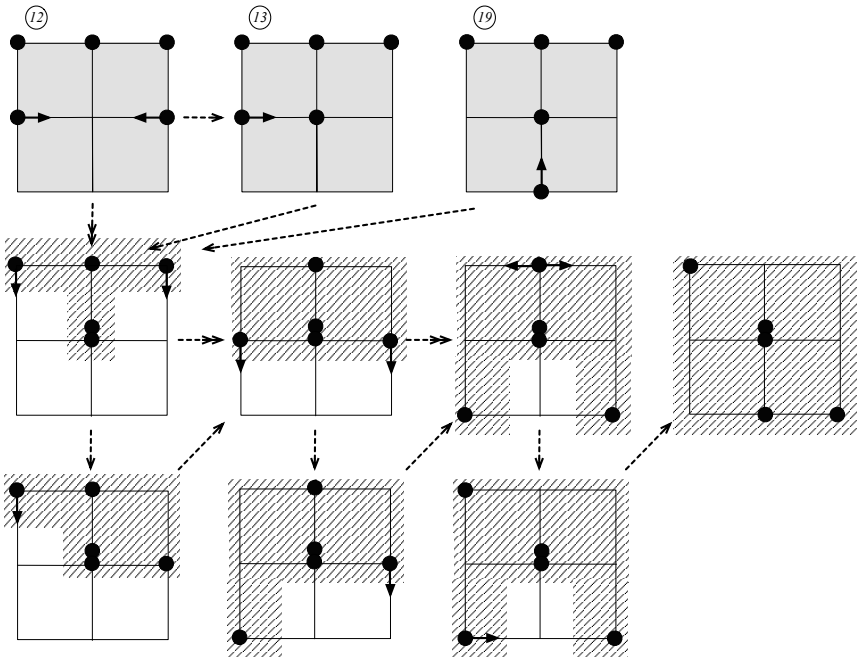

Figure 19: Exploration starting from the configurations ⑫, ⑬, and ⑲. Hatched areas show visited nodes.

moves. When several destinations are possible, the choice is left to the adversary. Transitions from a configuration to another are shown with dashed arrows. A single head arrow ⇢ (double head arrow ⇉) shows a transition with only one robot (resp., two robots) moving.

Observation 3. *Starting from any towerless configuration, the* `Set-Up` *phase leads to a configuration in* `P-E` *in at most* 3 *moves.*

5.2 `Exploration` Phase

There are five different sequences to explore the $(3, 3)$-grid, depending on the `Pre-Exploration` configuration where preparation phase terminated. The five sequences are shown from Figure 19 to Figure 23.

Explorations proposed in Figures 19 and 20 are respectively distinct from all other explorations thanks to the position of the unique tower. For the other explorations in Figures 21-23, the positions of the robots relative to the unique tower make all configurations of these explorations distinguishable from each other. So, explorations given in Figures 19-23 are unambiguous, and follows.

Observation 4. *Any configuration used during the* `Exploration` *phase appears only once and each of the five exploration sequences terminates after at most 9 moves.*

Theorem 7. *In the asynchronous model, the deterministic exploration of a* $(3, 3)$*-Grid can be solved in at most* 12 *moves by* 5 *oblivious robots.*

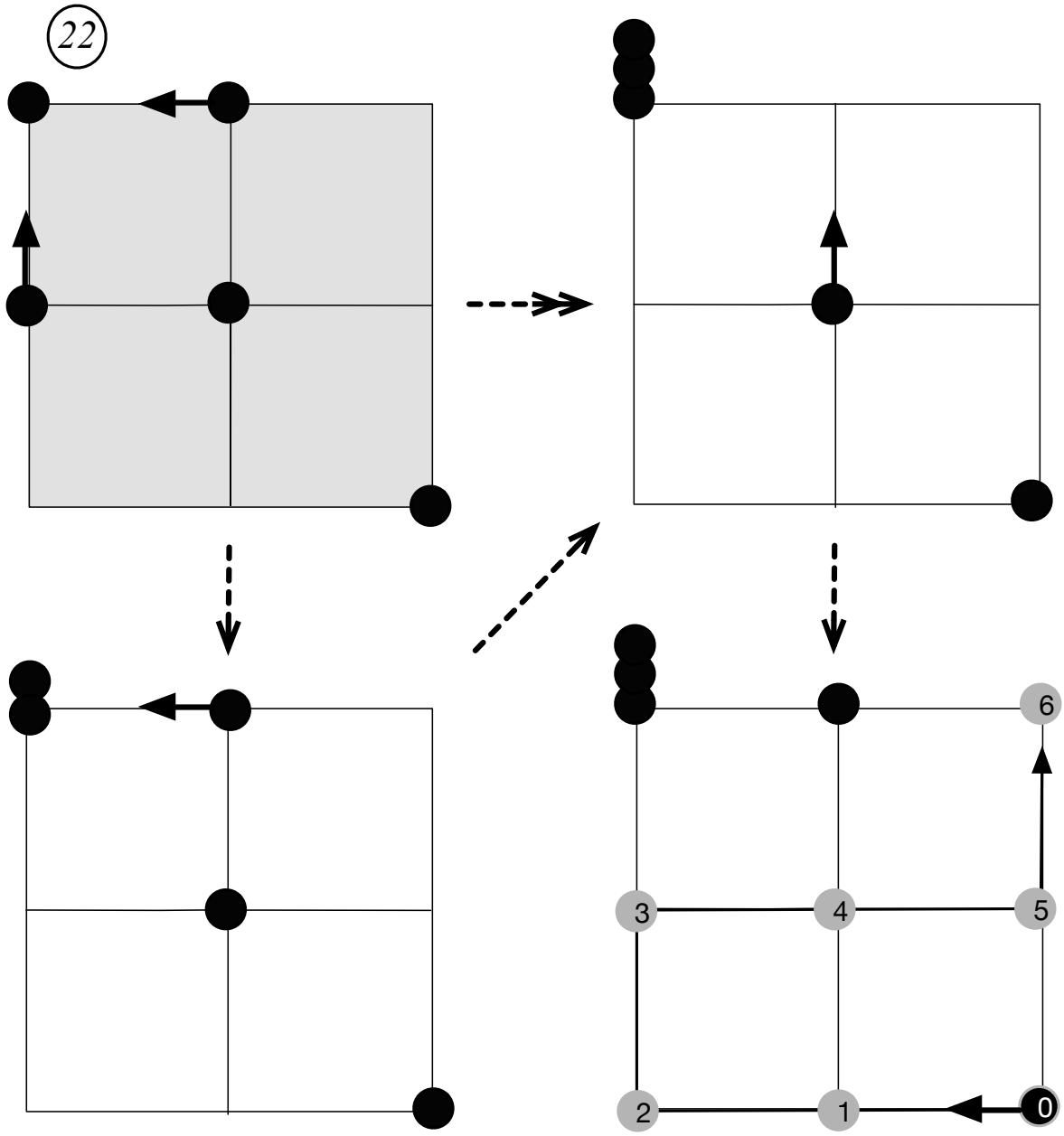

Figure 20: Exploration from Configuration ㉒.

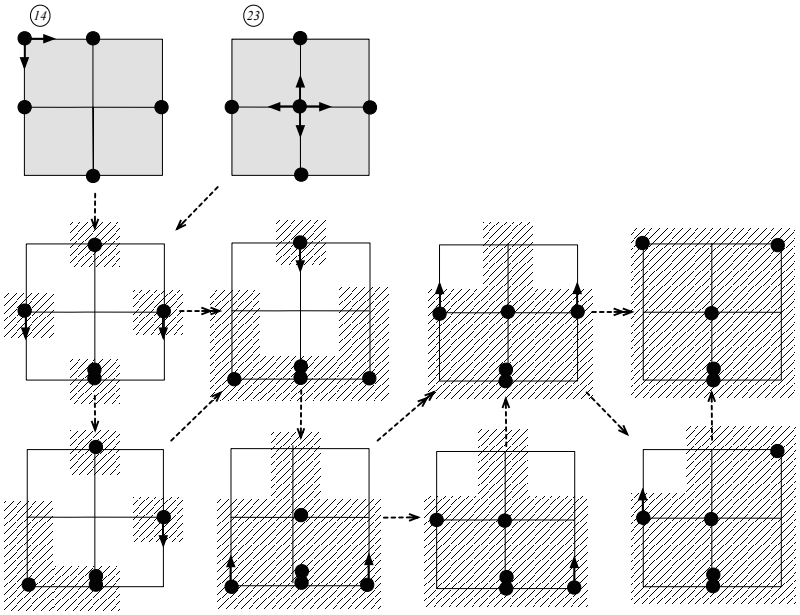

Figure 21: Exploration starting from (14) and (23). Hatched areas show visited nodes.

6 Conclusion

We presented necessary and sufficient conditions to explore an anonymous grid with a team of k asynchronous oblivious robots. Our results show that, perhaps surprisingly, exploring a grid is easier than exploring a ring. In the ring, deterministic (respectively, probabilistic) solutions essentially require 5 (resp., 4) robots. In the grid, 3 robots are necessary (even in the probabilistic case) and sufficient (even in the deterministic case) in the all but two cases, while the two remaining instances do require 4 and 5 robots, respectively. Note that the general algorithm given in this paper requires exactly 3 robots to solve the problem in $\Theta(n)$ moves, where n is the number of nodes in the grid. It is worth investigating whether exploration of a grid of n nodes can be achieved using any number k $(3 > k \geq n - 1)$ of robots, in particular when k is even.

References

[1] Roberto Baldoni, François Bonnet, Alessia Milani, and Michel Raynal. Anonymous graph exploration without collision by mobile robots. *Inf. Process. Lett.*, 109(2):98–103, 2008.

[2] Roberto Baldoni, François Bonnet, Alessia Milani, and Michel Raynal. On the solvability of anonymous partial grids exploration by mobile robots. In Theodore P. Baker, Alain Bui, and Sébastien Tixeuil, editors, *Principles of Distributed Systems, 12th International Conference, OPODIS 2008, Luxor, Egypt, December 15-18, 2008. Proceedings*, volume 5401 of *Lecture Notes in Computer Science*, pages 428–445. Springer, 2008.

[3] François Bonnet, Alessia Milani, Maria Potop-Butucaru, and Sébastien Tixeuil. Asynchronous exclusive perpetual grid exploration without sense of direction. In Antonio Fernández Anta,

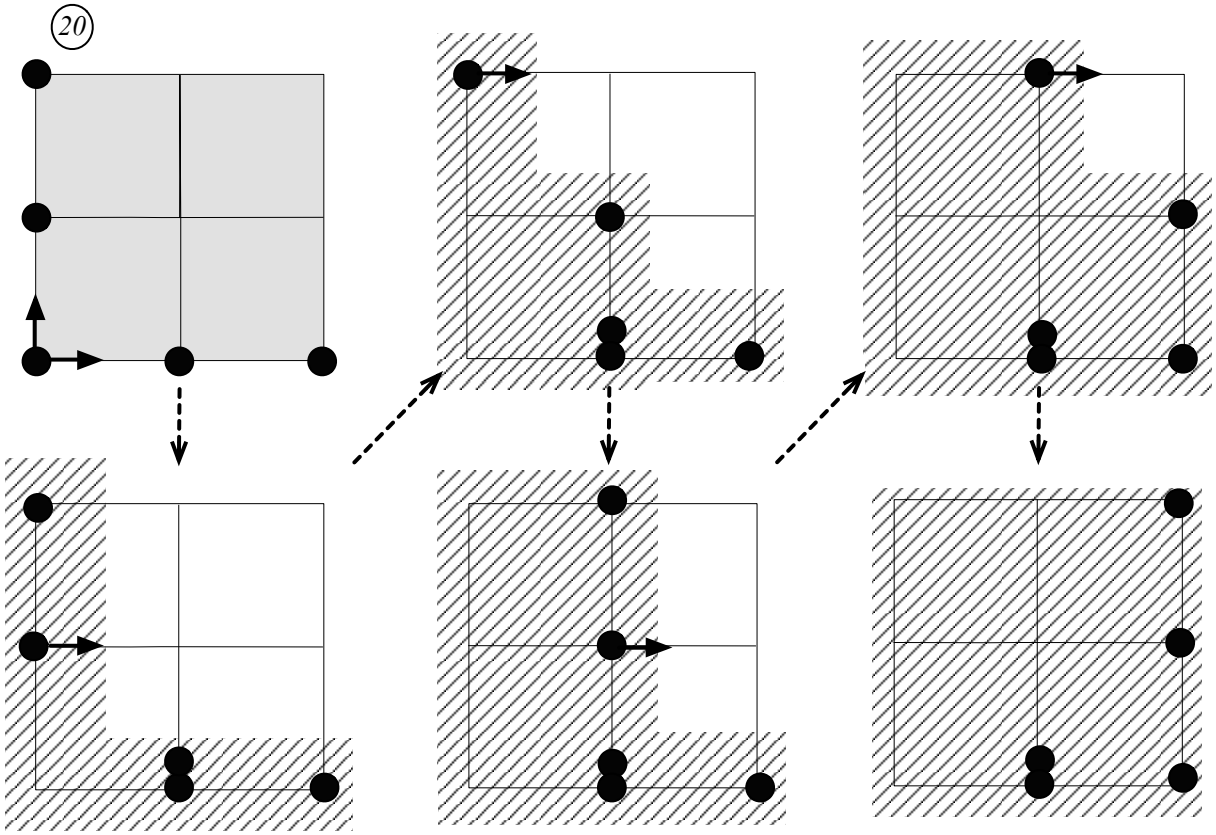

Figure 22: Exploration starting from ⑳. Hatched areas show visited nodes.

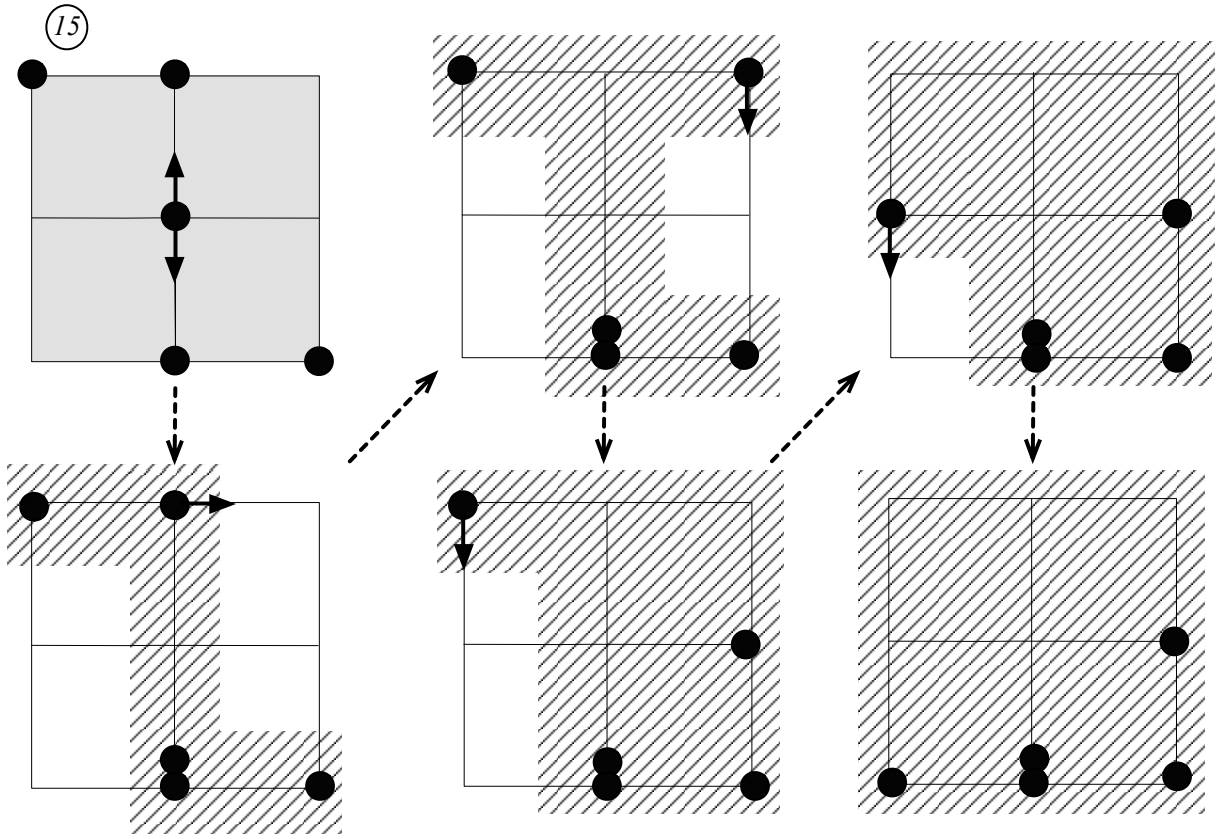

Figure 23: Exploration from ⑮. Hatched areas show visited nodes.

Giuseppe Lipari, and Matthieu Roy, editors, *Principles of Distributed Systems - 15th International Conference, OPODIS 2011, Toulouse, France, December 13-16, 2011. Proceedings*, volume 7109 of *Lecture Notes in Computer Science*, pages 251–265. Springer, 2011.

[4] Jérémie Chalopin, Paola Flocchini, Bernard Mans, and Nicola Santoro. Network exploration by silent and oblivious robots. In Dimitrios M. Thilikos, editor, *Graph Theoretic Concepts in Computer Science - 36th International Workshop, WG 2010, Zarós, Crete, Greece, June 28-30, 2010 Revised Papers*, volume 6410 of *Lecture Notes in Computer Science*, pages 208–219, 2010.

[5] Gianlorenzo D'Angelo, Gabriele Di Stefano, Alfredo Navarra, Nicolas Nisse, and Karol Suchan. A unified approach for different tasks on rings in robot-based computing systems. In *IPDPS Workshops*, pages 667–676. IEEE, 2013.

[6] Stéphane Devismes, Anissa Lamani, Franck Petit, and Sébastien Tixeuil. Optimal torus exploration by oblivious robots. In Ahmed Bouajjani and Hugues Fauconnier, editors, *Networked Systems - Third International Conference, NETYS 2015, Agadir, Morocco, May 13-15, 2015, Revised Selected Papers*, volume 9466 of *Lecture Notes in Computer Science*, pages 183–199. Springer, 2015.

[7] Stéphane Devismes, Franck Petit, and Sébastien Tixeuil. Optimal probabilistic ring exploration by semi-synchronous oblivious robots. *Theoretical Computer Science (TCS)*, 498:10–27, 2013.

[8] Asaf Efrima and David Peleg. Distributed algorithms for partitioning a swarm of autonomous mobile robots. *Theor. Comput. Sci.*, 410(14):1355–1368, 2009.

[9] Paola Flocchini, David Ilcinkas, Andrzej Pelc, and Nicola Santoro. Remembering without memory: Tree exploration by asynchronous oblivious robots. *Theor. Comput. Sci.*, 411(14-15):1583–1598, 2010.

[10] Paola Flocchini, David Ilcinkas, Andrzej Pelc, and Nicola Santoro. Computing without communicating: Ring exploration by asynchronous oblivious robots. *Algorithmica*, 65(3):562–583, 2013.

[11] Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. *Distributed Computing by Oblivious Mobile Robots*. Morgan & Claypool, 2000.

[12] N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud. The synchronous dataflow programming language lustre. *Proceedings of the IEEE*, 79(9):1305–1320, september 1991.

[13] Anissa Lamani, Maria Gradinariu Potop-Butucaru, and Sébastien Tixeuil. Optimal deterministic ring exploration with oblivious asynchronous robots. In *17th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 183–196, 2010.

[14] Giuseppe Prencipe. Instantaneous actions vs. full asynchronicity : Controlling and coordinating a set of autonomous mobile robots. In *Theoretical Computer Science, 7th Italian Conference (ICTCS 2001), Proceedings*, volume 2202 of *Lecture Notes in Computer Science*, pages 154–171, 2001.

[15] P. Raymond. Synchronous program verification with lustre/lesar. In S. Mertz and N. Navet, editors, *Modeling and Verification of Real-Time Systems*, chapter 6, page 171–206. ISTE/Wiley, 2008.

[16] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing*, 28(4):1347–1363, 1999.