Optimal grid exploration by asynchronous oblivious robots

Stéphane Devismes, Anissa Lamani, Franck Petit, Pascal Raymond, Sébastien Tixeuil

Optimal Grid Exploration by Asynchronous Oblivious Robots

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

Abstract

In this paper, we propose optimal (*w.r.t*, the number of robots) solutions for the deterministic exploration of a grid shaped network by a team of k asynchronous oblivious robots. In more details, we first show that no exploration protocol exists with less than three robots for any grid with more than three nodes, less than four robots for the $(2,2)$-Grid, and less than five robots for the $(3,3)$-Grid. Next, we show that the problem is solvable using only 3 robots for any (i,j)-Grid, provided that $j > 3$. Our result is constructive as we present a deterministic algorithm that performs in the non-atomic CORDA model. We also present specific deterministic protocols for the $(3,3)$-Grid using five robots.

Keyword: Exploration, grid, oblivious robots, CORDA model.

*VERIMAG UMR 5104, Université Joseph Fourier, Grenoble (France)
†MIS, Université de Picardie Jules Verne (France)
‡LIP6, UPMC Sorbonne Universités (France)

1 Introduction

We consider autonomous robots that are endowed with visibility sensors (but that are otherwise unable to communicate) and motion actuators. Those robots must collaborate to solve a collective task, here the *terminating grid exploration* (or *exploration* for short), despite being limited with respect to input from the environment, asymmetry, memory, etc.

So far, two universes have been studied: the *continuous two-dimensional Euclidian space* and the *discrete universe.* In the former, robot entities freely move on a plane using visual sensors with perfect accuracy that permit to locate all other robots with infinite precision, *e.g.*, [5, 8, 14, 15, 2]. In the latter, the space is partitioned into a finite number of locations, conventionally 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 the other, *e.g.*, [6, 7, 1, 4, 3, 9, 10, 11, 12].

In this paper, we pursue research in the discrete universe and focus on the *exploration problem* when the network is an anonymous unoriented grid, using a team of autonomous mobile robots. Exploration requires that the robots explore the grid and stop when the task is accomplished. In other words, every node of the grid must be visited by at least one robot and the protocol eventually terminates — every robot eventually stays idle forever.

The robots we consider are unable to communicate, however they can sense their environment and take decisions according to their local view. We assume anonymous and uniform robots (*i.e.*, they execute the same protocol and there is no way to distinguish between them using their appearance). In addition they are oblivious, *i.e*, they do not remember their past actions. In this context, robots asynchronously operate in cycles of three phases: Look, Compute, and Move. In the first phase, robots observe their environment in order to get the position of all other robots in the grid. In the second phase, they perform a local computation using the previously obtained view and decide their target destination to which they will move during the last phase.

Related Works. The fact that the robots have to stop after the exploration process implies that the robots somehow have to remember which part of the graph has been explored. Nevertheless, in this weak scenario, robots have no memory and thus are unable to remember the various steps taken before. In addition, they are unable to communicate explicitly. Therefore the positions of the other robots remain the only way to distinguish different stages of the exploration process. The main complexity measure then is the minimal number of required robots. Since numerous symmetric configurations induce a large number of required robots, minimizing the number of robots turns out to be a difficult problem. As a matter of fact, in [7], it is shown that, in general, $\Omega(n)$ robots are necessary to explore a tree network of n nodes deterministically.

In [6], authors proved that no deterministic exploration is possible on a ring when the number of robots k divides the number of nodes n. In the same paper, the authors proposed a deterministic algorithm that solves the problem using at least 17 robots provided that n and k are co-prime. In [12], Lamani *et al.* proved that there exists no deterministic protocol that can explore an even sized ring with $k \leq 4$ robots, even in the ATOM model [15] where robots execute their Look, Compute and Move phases in an atomic manner, and thus extend naturally in the non-atomic CORDA model. They also provide a deterministic protocol using five robots and performing in the fully asynchronous non-atomic CORDA model [13] (provided that five and n are co-prime). By contrast, [4] presents a probabilistic exploration algorithm for a ring topology of size $n > 8$. Four probabilistic robots are proved optimal since the same paper shows that no protocol (probabilistic or deterministic) can explore a ring with three robots, in the atomic model — so called, SYm or ATOM model. Note that the remaining cases where the ring size is less or equal to 8 have been then solved, still using 4 robots in the atomic model, in [3].

To our knowledge, grid shaped networks were only considered in the context of anonymous and oblivious robot exploration [1] for a variant of the exploration problem where robots perpetually explore all nodes in the grid (instead of stopping after exploring the whole network). Also, the protocols presented in [1] make use of a common sense of direction for all robots (common north, south, east, and west directions) and assume an essentially synchronous scheduling.

Contribution. In this paper, we propose optimal (*w.r.t*, the number of robots) solutions for the deterministic exploration of a grid shaped network by a team of k asynchronous oblivious robots in the asynchronous and non-atomic CORDA model. In more details, we first show that no exploration protocol exists with less than three robots for any grid with more than three nodes, less than four robots for the $(2,2)$-Grid, and less than five robots for the $(3,3)$-Grid. Next, we show that the problem is solvable using only 3 robots for any (i,j)-Grid, provided that $j > 3$. Our result is constructive as we present a deterministic algorithm that performs in the non-atomic CORDA model. We also present specific deterministic protocols for the $(3,3)$-Grid using five robots.

The above results show that, perhaps surprisingly, exploring a grid is easier than exploring a ring, even when robots do not have any common orientation. In the ring, deterministic solutions essentially require five robots [12] while probabilities enable solutions with only four robots [4, 3]. In the grid, three robots are necessary and sufficient in the general case even for deterministic protocols, while particular instances of the grid do require four or five robots. Also exploring a general grid requires no primality condition while exploring a ring expects the number k of robots to be co-prime with n the number of nodes.

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 three robots is given in Section 4, the special case with five robots is proposed in Section 5. Section 6 gives some concluding remarks.

2 Preliminaries

Distributed Systems. We consider systems of autonomous mobile entities called *agents* or *robots* evolving in a *simple unoriented connected graph* $G = (V, E)$, where V is a finite set of 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 (i,j)-*Grid* (or a Grid, for short) where i, j are two positive integers, *i.e.*, G satisfies the following two conditions: (a) $|V| = i \times j$, and (b) there exists an order on the nodes of V, $v_1, \dots, v_{i\times j}$, such that:

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

Nodes of the grid are anonymous (we may use indices, but for notation purposes only). We denote by $n = i \times j$ the number of nodes in G. We denote the degree of node v in G by $\delta(v)$. Given two neighboring nodes u and v, there is no explicit or implicit labeling allowing to determine whether u is either on the left, on the right, above, or below v. Remark that an (i,j)-*Grid* and an (j,i)-*Grid* are isomorphic. Hence, as the nodes are anonymous, we cannot distinguish an (i,j)-*Grid* from a (j,i)-*Grid*. So, without loss of generality, we always consider (i,j)-*Grids*, where $i \leq j$. Note also that any $(1,j)$-*Grid* is isomorphic to a chain. In any (i,j)-*Grid*, if $i = 1$, then either the grid consists of one node, or two nodes are of degree 1 and all other nodes are of degree 2; otherwise four nodes 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 $(1,j)$-*Grid* with $j > 1$, the unique chain linking the two corners is called the *borderline*. In any (i,j)-*Grid* such that $i > 1$, there exist four chains $v_1, \dots, v_m$ 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*.

Robots. Operating in G are $k \leq n$ robots. The robots do not communicate in an explicit way; however they see the position of the other robots and can acquire knowledge based on 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*. We call *protocol* a collection of k *programs*, each one operating on a single robot. Here we assume that robots are *uniform* and *anonymous*, *i.e.*, they all have the same program using no local parameter (such that an identity) that could permit to differentiate them. The program of a robot consists in executing *Look-Compute-Move cycles* infinitely many times. That is, the robot first observes its environment (Look phase). Based on its observation, a robot then decides to move or stay idle (Compute phase). When a robot decides to move, it moves from its current node to a neighboring node during the Move phase.

Computational Model. We consider an *asynchronous* model similar to the one in [6]. In this model, time is represented by an infinite sequence of instants 0, 1, 2, ... No robot has access to this global time. Look-Compute-Move cycles are performed asynchronously by each robot: the time between Look, Compute, and Move operations is finite yet unbounded, and is decided by an adversary. The only constraint is that both Move and Look are instantaneous, hence any robot performing a Look operation sees all other robots at nodes and not on edges. However, 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 $t'' > t'$ in which some robots are in different nodes from those previously perceived by $\mathcal{R}$ because in the meantime they moved. Hence, robots may move based on significantly outdated perception.

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

In this paper, we consider the *weak* multiplicity detection. Under the *weak* multiplicity detection, for every node u_i, d_i is a function $\mathbb{N} \mapsto \{\circ, \perp, \top\}$ defined as follows: $d_i(t)$ is equal to either $\circ$, $\perp$, or $\top$ according to u_i contains no, one or several robots at time instant t. 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.

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 (actually, this order is used in the reasoning only). 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 can start at time 0. Every configuration where all robots stay idle forever is said to be *terminal*. Two configurations $d_1, \ldots, d_n$ and $d'_1, \ldots, d'_n$ are *indistinguishable* if and only if there exists an automorphism f on G satisfying the additional condition: $\forall v_i \in V$, 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 $d_j(t), *$ (any robot knows the multiplicity of the node where it is located). Hence, from its view, a robot can compute the view of all other robots, 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 this 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. We assume the worst-case decision in such cases, *i.e.* the actual edge among the identically looking ones is chosen by an adversary.

Execution. We model the executions of our protocol in G by the list of configurations through 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: (*i*) $\gamma_{j-1} \neq \gamma_j$, (*ii*) γ_j is obtain from γ_{j-1} after some robots move from their locations in γ_{j-1} to a neighboring node, and (*iii*) 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 $\gamma_{j'}$. An execution $\gamma_0, \ldots, \gamma_i$ is said to be *sequential* if and only if $\forall j > 0$, exactly one robot moves between γ_{j-1} and γ_j.

Problem to be solved. We consider the *exploration* problem, where k robots, initially placed at different nodes, collectively explore a (i, j)-grid before stopping moving forever. By "collectively" explore we mean that every node is eventually visited by at least one robot. Observe that the problem is not defined for $k > n$ and straightforward for $k = n$. (In this latter case the exploration is already accomplished in the initial configuration.)

3 Bounds

In this section, we first show that, except for some trivial cases (where $k = n$), when robots are oblivious and the model is atomic, at least three robots are necessary to solve the exploration in any grid (Theorem 1), even if their protocol is deterministic. Moreover, in a $(2, 2)$-Grid, 4 robots are necessary (Theorem 2) and sufficient (the exploration of a $(2, 2)$-Grid by 4 robots is trivial). Finally, at least 5 robots are necessary to solve the exploration in a $(3, 3)$-Grid (Theorem 4). Impossibility results for the ATOM model naturally extend to the CORDA model. In the two next sections, we show that all these bounds are also sufficient to solve the exploration in the asynchronous and non-atomic CORDA model. Note that, all results presented here still holds even when robots are endowed with strong multiplicity (that permits to exactly count the number of robots at a particular node).

First, when robots are oblivious and there are more nodes than robots, any terminal configuration should be distinguishable from any possible initial (towerless) configuration. So, we have:

Remark 1 *Any terminal configuration of any exploration protocol for a grid of $n > k$ nodes using k oblivious robots contains at least one tower.*

Theorem 1 *There exists no (probabilistic or deterministic) exploration protocol in the ATOM model using* 1 *or* 2 *oblivious robots for any (i, j)-Grid with at least* 3 *nodes.*

Proof. By Remark 1, it is straightforward to see that there is no exploration protocol for any (i, j)-Grid with more than 2 nodes and 1 robot. Indeed any configuration is towerless.

Assume now, by contradiction, there exists an exploration protocol in the ATOM model $\mathcal{P}$ for a (i, j)-Grid with more than 2 nodes and 2 oblivious robots.

Then, any execution of $\mathcal{P}$ starts from a towerless configuration (by definition) and eventually reaches a terminal configuration containing a tower (by Remark 1). Let consider a sequential execution. Then, the two last configurations consist in a towerless configuration followed by a configuration containing a tower. These two configurations form a possible sequential execution that terminates while only two nodes are visited, thus a contradiction. □

Any $(2, 2)$-Grid is isomorphic to a 4-size ring. It has been shown in [4] that no exploration using less than 4 oblivious robots is possible for any ring of size at least 4 in the ATOM model. So, the following theorem holds:

Theorem 2 *There exists no deterministic exploration protocol using* 1*,* 2*, or* 3 *oblivious robots in the ATOM model for a* $(2, 2)$*-Grid.*

Theorem 3 *There exists no deterministic exploration protocol in the ATOM model using* 1*,* 2*, or* 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 protocol $\mathcal{P}$ in the ATOM model for a $(3, 3)$-Grid using 3 robots. By Lemma **??**, there exists a 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 $0 \leq x < y$, 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 $\mathcal{R}_2$.

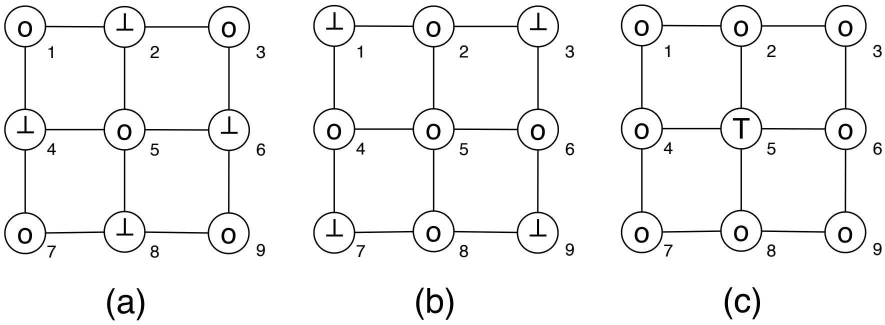

Figure 1: Three possible configurations in a $(3, 3)$-Grid with 4 robots.

- Assume that $\mathcal{R}_1$ and $\mathcal{R}_2$ never moved after γ_1. Then, by Lemma **??**, 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_ℓ the first configuration from which $\mathcal{R}_1$ or $\mathcal{R}_2$ moves. From the previous case, at most 7 distinct nodes have been visited before γ_ℓ. The execution being sequential, only one robot of the tower moves during the step from γ_ℓ to γ_{i+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_ℓ. So, there exists an execution e' starting from the prefix $\gamma_0, \ldots, \gamma_\ell$ where both $\mathcal{R}_1$ and $\mathcal{R}_2$ move from γ_ℓ 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 **??**, 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.

□

Theorem 4 *There exists no deterministic exploration protocol in the ATOM model using 1, 2, 3, or 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 protocol $\mathcal{P}$ for a $(3, 3)$-Grid with 4 robots in the ATOM model.

Figure 1 depicts three possible configurations for a $(3, 3)$-Grid with 4 robots. 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. 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 asynchronous 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 during 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 possible destinations. So, the adversary can choose destinations so that the system reaches configuration (a) again.

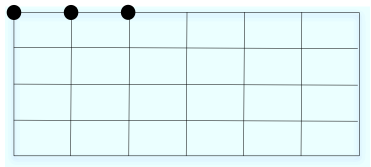

Figure 2: `Set-Up` Configuration

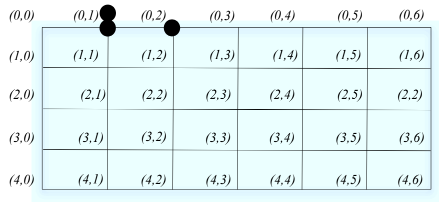

Figure 3: Coordinate system built by the `Orientation` phase

- 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 case, we can deduce that there exist executions of $\mathcal{P}$ that never terminates, so $\mathcal{P}$ is not an exploration protocol, a contradiction. □

4 Deterministic solution using three robots

In this section, we focus on solutions for the exploration problem that use three robots only. We recall that there exists no deterministic solution for the exploration using three robots in a $(3,3)$- or $(2,2)$-grid (Section 3). Moreover, exploring a $(3,1)$-grid using using three robots is straightforward. So, we consider all remaining cases. We split our study into 2 cases. A general deterministic solution for any (i,j)-grid such that $j > 3$ is given in Subsection 4.1. The particular case of the $(2,3)$-grid is solved in Subsection 4.2.

4.1 General Solution

Overview. Our deterministic protocol has three main phases:
- `Set-Up` phase: The aim of this phase is to create a single line of robots starting from a corner and along a longest borderline of the grid — refer to Figure 2. Let us refer to this configuration as the `Set-Up` configuration. Note that the starting configuration of this phase can be any arbitrary towerless configuration that does not contain a line of robots on the longest line. Moreover, we do not create any tower during this phase.
- `Orientation` phase: The starting configuration of this phase is the `Set-Up` configuration. The aim of this phase is to give an orientation of the grid. In order to achieve that, one tower is created on the longest line allowing the robots to establish a common coordinate system— refer to Figure 3. Let refer to the such a configuration by `Oriented` configuration.
- `Exploration` phase: Starting from the `Oriented` configuration, exactly one node is occupied by a single robot, called *explorer*. This robot will perform the exploration task by visiting all the nodes, except two already visited ones, based on the coordinate system defined in the `Orientation` phase — refer to Figure 5 (page 14).

Set-Up phase. Starting from any towerless configuration, the `Set-Up` phase constructs a single line of robots along a longest borderline of the grid, and such that one extremity is located at a corner. In order to do so, we distinguish three main configurations as follows:
- Configuration of type `Leader`: In a such configuration there is exactly one robot located at one corner of the grid.
- Configuration of type `Choice`: In such a configuration, there are at least two robots that are

located at a corner of the grid. Thus, we have to choose one of these robots to remain at a corner. The other ones have to leave the corner.
- Configuration of type `Undefined`: In such a configuration, there is no robot in any corner of the grid. The idea is then to elect one robot that will move to join one of the corners of the grid.

In the following, we present the behavior of robots, referred to as $R1, R2$, and $R3$, in each of the main configurations. Note that such configurations are declined into several subconfigurations.

1. The configuration is of type `Leader`: In such a configuration, there is exactly one robot that is in a corner of the grid. Let $R1$ be this robot. We consider the following subcases:

   - The configuration is of type `Strict-Leader`: In such a configuration, there is no robot on any borderline having the corner where $R1$ is located as extremity. In this case, the robots that are the closest to $R1$ are the ones allowed to move. Their destination is their adjacent empty node on the shortest path towards the closest empty node that is on the longest borderline having the corner where $R1$ is located as extremity.

   - The configuration is of type `Half-Leader`: In such a configuration there is only one robot $R2$ that is on a borderline having the corner where $R1$ is located as extremity. Two subcases are possible:

     - The configuration is of type `Half-Leader1`: $R2$ is on the longest borderline. In this case the third robot $R3$ is the one allowed to move. Its destination is its adjacent empty node towards the closest empty node on the borderline that contains both $R1$ and $R2$.
     - The configuration is of type `Half-Leader2`: $R2$ is not on the longest borderline. In this case $R2$ is the one allowed to move, its destination is its adjacent empty node outside the borderline. (Note that in the case there is no such an empty node, $R2$ moves first to an empty node on its own borderline and then it moves outside the borderline.)

   - The configuration is of type `All-Leader`: All the robots are on the same borderline as $R1$. Let refer to the other robots as $R2$ and $R3$, respectively. Note that $R2$ and $R3$ are not necessary on the same borderline. Thus, we have the two subcases as follow:

     - The configuration is of type `Fully-Leader`: In such a configuration all the robots are on the same borderline, $D1$. The two following subcases are then possible:
       - The configuration is of type `Fully-Leader1`: in the case where $D1$ is the longest borderline and the robots do not form a line, then let $R2$ be the closest robot from $R1$. If $R1$ and $R2$ are not neighbor, then $R2$ is the only robot allowed to move and its destination is its adjacent empty node towards $R1$. In the case where $R1$ and $R2$ are neighbor, the remaining robot, $R3$, will be the only robot allowed to move, its destination will be its adjacent empty node towards $R2$.
       - The configuration is of type `Fully-Leader2`: In the case $D1$ is not the longest borderline. Then, the robot that is the closest to $R1$ leaves the borderline by moving to its neighboring empty node outside the borderline it belongs to.
     - The configuration is of type `Semi-Leader`: $R2$ and $R3$ are not on the same borderline. Two subcases are possible:
       - The configuration is of type `Semi-Leader1`: in this case $i \neq j$. Either $R2$ or $R3$ is located on the smallest borderline. This latter moves to its adjacent empty node outside the borderline.
       - The configuration is of type `Semi-Leader2`: in the case $i = j$. Let denote by $Distance(R, R')$ the *distance* (that is, the shortest path) in the grid between the two nodes where R and R' are located. If $Distance(R1, R2) \neq Distance(R1, R3)$ then the robot that is the closest to $R1$ is the only one allowed to move, its destination is its adjacent empty node outside the borderline. Otherwise ($Distance(R1, R2) = Distance(R1, R3)$), either (a) there is an empty node between $R1$ and $R2$, or (b) $R1$

is both neighbor to $R2$ and $R3$. In case (a), $R1$ is the only robot allowed to move and its destination is its adjacent empty node towards one of its two borderlines (the adversary will make the choice). In case (b), $R2$ and $R3$ moves and their destination is their adjacent empty node on their borderline.

2. The configuration is of type `Choice`: In this configuration there is at least two robots located at a corner of the grid. We split our study into the following cases:

   - The configuration is of type `Choice1`: In this configuration there are exactly two robots that are located at a corner of the grid. Let $R1$ and $R2$ be these robots. (i) In the case where $R3$ is on the same borderline as either $R1$ or $R2$ but not both (suppose that it is $R1$) then $R2$ is the one allowed to move, its destination is its adjacent empty node towards the borderline that contains both $R1$ and $R3$. (ii) In the case where the three robots are on the same borderline, then if $Distance(R1, R3) < Distance(R2, R3)$ then $R2$ moves to its adjacent empty node on the border lone towards $R3$. Otherwise ($Distance(R1, R3) = Distance(R2, R3)$), $R3$ will move to its adjacent empty node on the same borderline towards either $R1$ or $R2$ (the scheduler will choose the destination in that case). (iii) If $R3$ is not on a borderline it moves to the closest longest borderline that contains either $R1$ or $R2$.

   - The configuration is of type `Choice2`: in this configuration all the robots are located at a corner of the grid. The robot allowed to move is the one that is located at a node that is common to the two borderlines of the other robots. Let $R1$ be this robot. The destination of $R1$ is its adjacent empty node on the longest borderline (Note that in the case of symmetry, the scheduler will choose the direction to take).

3. The configuration is of type `Undefined`: in this configuration, there is no robot that located at a corner of the grid. The cases below are then possible:

   - The configuration is of type `Undefined1`: In such a configuration, there is exactly one robot that is the closest to one corner of the grid. In this case, this robot is the only one allowed to move, its destination is its adjacent empty node on a shortest path towards the closest empty node that is at the corner (if there are several shortest paths, the scheduler makes the choice).

   - The configuration is of type `Undefined2`: In such a configuration $i = j$ and there is one borderline that contains two robots. Let $D1$ be this borderline. The robot that is not located on $D1$ is the only one allowed to move and its destination is its adjacent empty node on the shortest path towards $D1$.

   - The configuration is of type `Undefined3`: There are exactly two robots that are the closest to one corner of the grid. Let $R1$ and $R2$ be these two robots. If $Distance(R1, R3) = Distance(R2, R3)$ then the third robot ($R3$) is the only one allowed to move, its destination is its adjacent empty node that is on a shortest path towards one of the two robots $R1$ or $R2$. (Note that the scheduler will choose the direction to take.) If $Distance(R1, R3) \neq Distance(R2, R3)$ then the robot that is the closest to $R3$ will be the only one allowed to move. Its destination is its adjacent empty node that is on a shortest path to the closest corner of the grid (if there are several shortest paths, the scheduler makes the choice).

   - The configuration is of type `Undefined4`: There are three robots that are closest to a corner of the grid. The cases below are then possible:

     - The configuration is of type `Undefined4-1`: There is exactly one robot that is on a borderline of the grid. In this case this robot is the only one allowed to move. Its destination is its adjacent empty node that is on a shortest path to a closest corner of the grid. (In case of two shortest paths, the adversary breaks the symmetry in the first step.)
     - The configuration is of type `Undefined4-2`: In such a configuration there are exactly two robots on the borderline of the grid. Let $R1$ and $R2$ be these two robots. The

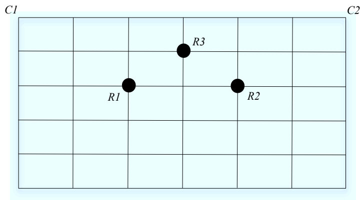

Figure 4: Sample of a configuration of type `Undefined4-4`

robot allowed to move is $R3$. Its destination is its adjacent empty node towards a closest corner. (the adversary may have to break the symmetry.)

- The configuration is of type `Undefined4-3`: The three robots in this configuration are on borderlines of the grid. (i) If there are more than one robot on one borderline. Note that in this case there are exactly two robots on the same borderline, and let $R1$ and $R2$ be these robots. Then $R3$ will be the only one allowed to move and its destination is its adjacent empty node towards the corner. (ii) If there is at most one robot on each borderline: Exactly one borderline is perperdicular to the two others. The robot on that borderline is the only one allowed to move and its destination is its adjacent node towards the closest corner.
- The configuration is of type `Undefined4-4`: In this case there is no robot on any borderline. (i) In the case where there are two robots, $R1$ and $R2$, that are closest to the same corner, and $R3$ is the only robot that is closest to another corner, then $R3$ is the only one allowed to move and its destination is its adjacent empty node on a shortest path towards the closest corner (if there are several shortest paths, the scheduler makes the choice). (ii) In the case where there are two robots, $R1$ and $R2$, that are closest to $C1$ and $C2$, respectively, where $C1 \neq C2$, and $R3$ is closest to both $C1$ and $C2$, then $R3$ is the only one allowed to move (refer to Figure 4), and it moves toward $C1$ or $C2$ according to a choice of the adversary. (iii) In the case where all the robots are closest to different corners, there is one robot $R1$ whom corner is between the two other target corners of $R2$ and $R3$. The robot allowed to move is $R1$, its destination is its adjacent empty node on the shortest path towards the closest corner (if there are several shortest paths, the scheduler makes the choice).

The correctness of the `Set-Up` phase is established by the two following lemmas.

Lemma 1 *Starting from any arbitrary towerless configuration,* `Set-Up` *phase does not create any tower.*

Proof. It is clear that in the case where one robot is allowed to move, no tower is created because the robot always moves to an empty adjacent node. Thus lets consider the cases in which there are at least two robots that are allowed to move:

- The configuration is of type `Strict-Leader`: Suppose that the robot that is at the corner is $R1$, and the two other ones (that are neither at a corner nor at the same borderline as $R1$) are $R2$ and $R3$, respectively. $R2$ and $R3$ are allowed to move at the same time only in the case they are at the same distance from $R1$. Since their destination is their adjacent empty node on the shortest path towards the longest borderline that contains $R1$, we are sure that the both will move to different empty nodes. Thus no tower is created in this case.

- The configuration is of type `Semi-Leader2`: we consider the case in which $Distance(R1, R2) = Distance(R1, R3)$ such as there is no empty node between $R1$ and both $R2$ and $R3$ respectively. It is clear that if the scheduler activates them at the same time no tower is created since they move to their adjacent empty node on the borderline they belong to, in the opposite direction of $R1$ (recall that they are in two different borderlines). In the case the scheduler activates only one robot ($R2$), no tower is created as well since it moves to its adjacent empty node on the borderline it belongs to (note that is this case $i = j$). Note that the configuration reached remains of type Semi-leader2, however, $Distance(R1, R2) \neq Distance(R1, R3)$. Thus the robot that is allowed to move now is $R3$, which is the one that was supposed to move at the first place. Thus either we retrieve the configuration in which both robots moved (this will happen in the case $R3$ has an outdated view). Or the configuration reached is of type Half leader1 and all the robots have a correct view.

From the cases above we can deduce that starting from any configuration that is towerless, `Set-Up` phase does not create any tower and the lemma holds. □

Lemma 2 *Starting from a configuration of type* `Leader`*, a configuration of type* `Set-Up` *is reached in a finite time.*

Proof. In a configuration of type `Leader`, there is only one robot that is at the corner of the grid (suppose that this robot is $R1$). It is easy to see that in the case $i \neq j$ all the robots will be on the longest borderline of the grid that contains $R1$ (refer to Strict Leader, HalfLeader1 configurations). Once the robots on the same longest borderline, it is also easy to create a line of robots keeping one robot at the corner. (The robot ($R2$) that is the closest to $R1$ moves first until it becomes neighbor of $R1$. Once it is done, the remaining robot ($R3$) moves to become neighbor of $R2$.) Hence we are sure that a configuration of type `Set-Up` is reached in a finite time. In the case $i = j$ when the robots move to the closest borderline that contains $R1$ either we have the same result as when $i \neq j$ (all the robots will be on the same borderline) and hence we are sure to reach a configuration of type `Set-Up`. Or, each robot $R2$ and $R3$ is on the same borderline as $R1$, however both of them are on different borderlines. The sub-cases are then possible as follow:

1. $Distance(R1, R2) \neq Distance(R1, R3)$. In this case the robot that is the closest to $R1$ moves to its adjacent node outside its own borderline (Let this robot be $R2$). Note that when it moves, its new destination is the closest empty node on the same borderline as both $R1$ and $R3$ (see `Semi-Leader2` configuration). Thus we are sure that $R2$ will be on the same borderline of $R1$ and $R3$ in a finite time, thus we are sure that the `Set-Up` configuration is reached in a finite time.

2. $Distance(R1, R2) = Distance(R1, R3)$. The two sub-case below are possible:

   (a) There is an empty node between $R1$ and the other robots. $R1$ is the one that will move, its destination is its adjacent empty node on one of its two adjacent borderlines. Note that once it has move, all the robots are a borderline such as there is one borderline that contains two robots, let $D1$ be this borderline (the configuration is of type `Undefined2`). The robot allowed to move is the one that is not part of $D1$, its destination is its adjacent empty node outside the borderline it belongs to. Once it moves, its new destination will be the borderline that contains two robots Thus we are sure that all the robots will be part of the same borderline in a finite time. It is clear that from this configuration is easy to build a configuration of type `Set-Up`. (Note that it is easy to break the symmetry,if any, since we have three robots.)

   (b) There is no empty node between $R1$ and the other robots $R2$ and $R3$. In this case $R2$ and $R3$ will be the ones allowed to move. Their destination is their adjacent empty node on their borderline. In the case the scheduler activates them at the same time, we retrieve case 2a. If the scheduler activates only one of the two robots, the configuration reached will be of type `Semi-Leader2` such as $Distance(R1, R2) \neq Distance(R1, R3)$, thus, The robot that is the closest to $R1$ is the one that is allowed to move. (Note that this robot is the one that was supposed to move at the first place.) If it has an outdated view it will move to its adjacent empty node and we retrieve case 2a. If not, it will move to its adjacent empty node outside its borderline. When it does, its new destination is the closest empty node on the same borderline of the two other robots. Note that when such a robot joins the new borderline, the configuration is of type `Set-Up`.

From the cases above, we can deduce that starting from a configuration of type `Leader`, a configuration of type `Set-Up` is reached in a finite time and the lemma holds.

□

Lemma 3 *Starting from a configuration of type* `Choice`*, a configuration of type* `Leader` *in reached in a finite time.*

Proof. It is clear that in the case where all the robots are on one corner of the grid, the next configuration reached is of type `Choice1` since there will be a single robot that will move (refer to Configuration of type `Choice2`). Note that when the configuration is of type `Choice1` the cases below are possible (Let the robots that are at the corner of the grid be $R1$ and $R2$ respectively and the third robot be $R3$):

1. $R3$ is on the same borderline $D1$ as $R1$ (Note that in this case $R2$ is not on $D1$). In this case $R2$ is the one allowed to move. Note that once it moves, it leaves the corner and the configuration will be of type `Leader` (refer to `Choice1`, case (i)).

2. All the robots are on the same borderline $D1$. In this case the robots $R3$ will be used to elect one of the two robots at the corner of the grid (refer to `Choice1` configuration case (ii)). If $Distance(R1, R3) \neq Distance(R2, R3)$ then the robot that is the farthest from $R3$ leaves the corner, thus, the configuration will contain a single robot that is at one corner of the grid. Hence the configuration will be of type `Leader` in a finite time. In the case $Distance(R1, R3) = Distance(R2, R3)$, $R3$ will move first on the borderline towards either $R1$ or $R2$ breaking the symmetry, and we retrieve the case in which $Distance(R1, R3) \neq Distance(R2, R3)$. Thus we are sure that a configuration of type `Leader` is reached in a finite time.

3. $R3$ is not on a borderline. In this case $R3$ is the one allowed to move. Its destination is its adjacent empty node on a shortest path towards the closest longest borderline that contains either $R1$ or $R2$. Thus we are sure that one of the two cases described above will be reached (refer to `Choice1` configuration, case (iii)).

From the cases above we can deduce that a configuration of type `Leader` is reached in a finite time and the lemma holds. □

Lemma 4 *Starting from a configuration of type* `Undefined`*, a configuration of either type* `SetUp` *or type* `Leader` *is reached in a finite time.*

Proof. It is clear that in the case where there is only one robot that is the closest to one corner of the grid, we are sure to reach a configuration of type `Leader` in a finite time since this robot will move until it reaches the corner. It is also clear that for the special configuration (`Undefined2`), a configuration where all the robots are on the same borderline is reached (note that this borderline is the longest one since $i = j$). Thus either there will be one robot that is the closest to one corner and hence we are sure that a configuration of type `Leader` is reached. Or their will be two robots that are the closest, this case is part of the cases below:

1. There are exactly two robots that are the closest to one corner of the grid (let these two robots be $R1$ and $R2$ respectively). In this case $R3$ will be used to break the symmetry: In the case $Distance(R1, R3) = Distance(R2, R3)$, $R3$ will be the one that will move first, it destination is its adjacent node towards either $R1$ or $R2$. Note that once it has moved,

$Distance(R1, R3) \neq Distance(R2, R3)$. In the case $Distance(R1, R3) \neq Distance(R2, R3)$, the robot that is the closest to $R3$ will be the one allowed to move, its destination is its adjacent empty node on a shortest path towards the corner. Note that once it has moved, either it reaches the corner or it becomes the closest one. Thus we are sure that a configuration of type `Leader` is reached in a finite time.

2. All the robots are the closest to a corner of the grid. If the configuration is of type `Undefined4-1`, then there will be one robot that will be allowed to move (the one that is on a borderline), once it has moved, it becomes the closest to one corner of the grid, thus we are sure to reach a configuration of type `Leader` in a finite time. In the case there are two robots at a borderline, The third robot (which is not on a borderline) is the one that will move becoming the closest robot to one corner of the grid. Thus in this case too, we are sure to reach a configuration of type `Leader`. In the case all the robots are on a borderline then, i) if there is more than one robot on the same borderline (note that in this case the borderline contains two robots), the robot that is not part of the borderline moves towards the closest corner becoming the closest one, thus we are sure that a configuration of type `Leader` is reached in a finite time. In the case there is one robot at each borderline, then one robot is easily elected to move becoming the closest to one corner of the grid. Thus, in this case too we are sure to reach a configuration of type `Leader` in a finite time. In the case there is no robot on the borderline. If there are two robots that are the closest to the same corner such as the third robot is the only closest robot to another corner then this robot is the one allowed to move, when it does it becomes the only one that is the closest to one corner of the grid. Thus we are sure to reach a configuration of type `Leader`. In the case there is one robot ($R3$) that is the closest to both corners $C1$ and $C2$ such as $R1$ and $R2$ are also the closest to $C1$ $C2$ respectively, then $R3$ is the one allowed to move towards one of the closest corner. Note that once it has moved, it becomes the closest one and hence we are sure that a configuration of type `Leader` is reached in a finite time. In the case all the robots are the closest to different corner, we are sure that one of them is the closest one to one corner that is between the two other target corners (the closest to the other robots). This robot is the one allowed to move, its destination is its adjacent empty node towards the closest corner. Note that one it moves it becomes either even closer (and hence it will be the only one that can move) or it will reach the corner. In both cases we are sure that a configuration of type `Leader` is reached.

From the cases above we can deduce that starting from a configuration of type `Undefined`, a configuration of type `Leader` is reached in a finite time and the lemma holds.

□

Lemma 5 Starting from any towerless configuration, a configuration of type `Set-Up` is reached in a finite time.

Proof. From Lemma 2, 3 and 4 we can deduce that starting from any arbitrary towerless configuration that does not contain a line of robots on the longest line of the grid, a configuration of type `Set-Up` is reached in a finite time and the lemma holds. □

Orientation phase. In this phase, an orientation of the grid is determined in the following manner: Note that the starting configuration of this phase contains a line of robots on the corner of the longest borderline of the grid, and its length is greater than 3. The robot that is at the corner is the one allowed to move, its destination is its adjacent occupied node. Once it has moved, a tower is created. Then, we can determine a coordination system where each node has unique coordinates, see Figure 3. 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 the node that does not contain the tower.

Lemma 6 *Starting from a configuration of type* `Set-Up`*, a configuration of type* `Oriented` *is reached in one step.*

Exploration phase From Lemmas 1-6, we know that, starting from any initial configuration, the system reaches an `Oriented` configuration in finite time and without creating any tower except in the last step.

In the first oriented configuration, we already know that nodes of coordinates $(0, 0)$ and $(0, 1)$ are already visited. Then, to ensure that the exploration phase remains distinct from the previous phases and to keep the coordinate system, we should only authorize the robot that is single in a node to moved. Let call this robot the *explorer*.

To explore all nodes except nodes of coordinates $(0, 0)$ and $(0, 1)$, the explorer should order all coordinates in the such way that (i) $(0, 0)$ and $(0, 1)$ are before its initial position (that is $(0, 2)$ and all other coordinates are after; and (ii) for all non-maximum coordinates (x, y), if (x', y') is the successor of (x, y) in the order, then the nodes of coordinates (x, y) and (x', y') are neighbors. Such an order can be defined as follows:

$$(a, b) \preceq (c, d) \equiv b < d \vee [b = d \wedge ((a = c) \vee (b \bmod 2 = 0 \wedge a < c) \vee (b \bmod 2 = 1 \wedge a > c)]$$

Using the order $\preceq$, the explorer moves as follows: while the explorer is not located at the node having the maximum coordinates according to $\preceq$, the explorer move to the neighbor whose coordinates are the successor of the coordinates of its current position.

Lemma 7 *The* `Exploration` *phase terminates in finite time and once terminated all nodes have been visited.*

By Lemmas 1-7, follows:

Theorem 5 *The deterministic exploration of any (i, j)-Grid with $j > 3$ can be solved in the CORDA model using 3 oblivious robots and the three phases* `Set-Up`*,* `Orientation`*, and* `Exploration`*.*

4.2 Exploring (2,3)-Grids

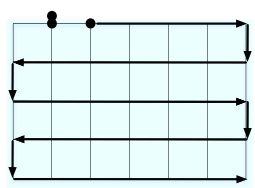

Figure 5: Exploration phase

The idea of the solution for the $(2, 3)$-Grid is rather simple. Consider the two longest lines of the grid. Since there are 3 robots on the grid, then there exists one of the two longest lines, D, that contains either all the robots or exactly two robots. In the second case, the robot that is not part of D moves to its adjacent empty node on the shortest path towards the empty node on D. Thus, the three robots are eventually located on D. Next, the robot not located on the corners of the grid moves to one of its two neighboring occupied nodes (the destination is chosen by the adversary). Thus, a tower is created. Once the tower is created, the grid is oriented. Then, the single robot moves to its adjacent empty node in the longest line that does not contain a tower. Next, it explores the nodes of this line by moving in the direction of the tower. When it becomes neighbor of the tower, all the nodes of the $(3, 2)$-Grid have been explored.

Theorem 6 *The deterministic exploration of a $(2, 3)$-Grid can be solved in the CORDA model using 3 oblivious robots.*

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

In this section, we propose an algorithm that solves the exploration with 5 robots on the $(3, 3)$-Grid. The algorithm works in two phases, the `Exploration` phase and the `Preparation` phase. Figures 6 and 13 depict the `Exploration` phase.

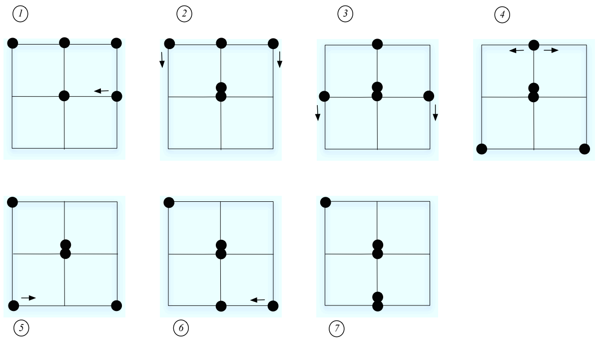

Figure 6: Exploration task on grids $(3, 3)$

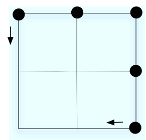

Figure 7: Configuration $(3, 1, 1)$

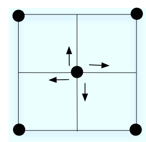

Figure 8: Instance of a configuration $(2, 1, 2)$

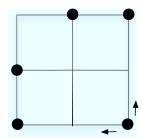

Figure 9: Instance of a configuration $(2, 1, 2)$

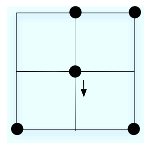

Figure 10: Instance of a configuration $(2, 1, 2)$

Preparation phase. The `Preparation` phase starts from any towerless configuration that is not one the three initial configurations of the exploration phase. The `Preparation` phase aims at reaching one of the three initial special configurations shown in either Figure 6-Case (1), Figure 13-Case($1a$), or Figure 13-Case($1b$).

Let us define some terms that will be used later: let the interdistance d be the minimal distance among distances between each pair of robots. We call a d.block a sequence of consecutive robots that are at distance d. The size of an 1.block is the number of robots it contains. We refer to a configuration by a set of three values $(X1, X2, X3)$ such as Xi represents the number of robots on the line i. Note that X1 and X3 are borderlines. Since the grid is of size $(3, 3)$, we do not know which borderlines correspond to $X1$ and $X3$. Some ambiguities can appear and thus for the same configuration there will be many possible sequences $(X1, X2, X3)$. The robots could be confused not knowing which action to take. To avoid this situation, we will use the following method: First we will choose one or two guide lines in the following manner: the line that contains the d.biggest d.block of robots is elected as a guide line. Note that the guide line can only contain two or three robots. In the case there are two possible guide lines that are perpendicular to each other, then i) in the case only one of this two guide lines is at the borderline of the grid, then this line is the guide line. ii) In the other case, the guide line is elected as follow: Let $D1$ be one possible guide line and $D2$, $D3$ be the lines that are horizontal to $D1$. In the same manner let $D'1$ be the other possible guide line and $D'2$, $D'3$ be the lines that are horizontal to $D'1$. Let B be the number of the biggest d.blocks on the lines Di and B' be the number of the biggest d.blocks on the lines $D'i$. The guide line is the one corresponding to the biggest value among B and B'. For Instance in Figure 11, the configuration can be $(2, 1, 2)$ or $(2, 2, 1)$. We can see that $d = 1$, and the size of the biggest 1.block is equal to 2. Note that there is an 1.block of size 2 on two borderlines that are perpendicular to each other (on $D3$ and $D'1$ —refer to Figure 11). Let B be the number of 1.blocks on the lines that are horizontal to $D3$, clearly $B = 2$. In the same manner, let B' be the number of 1.blocks of size 2 on the lines that are horizontal to $D'1$ (clearly $B' = 1$). We can see that $B > B'$, thus the guide lines are both $D3$ and $D1$ (The lines that are considered are the ones that are horizontal to $D3$ and $D1$). Thus the configuration is of type $(2, 1, 2)$.

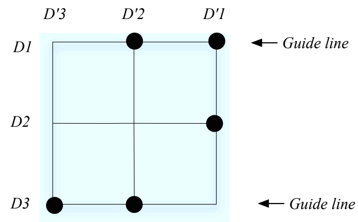

Figure 11: Guide-lines, configuration of type $(2, 1, 2)$

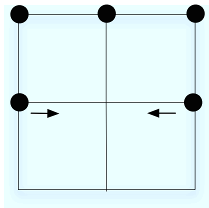

Figure 12: Instance of a configuration $(3, 2, 0)$

The triple set $(X1, X2, X3)$ refer then to the number of robots that are horizontal to the guide lines. The following cases are then possible:

- The configuration is of type $(1, 1, 3)$. Two sub-cases are possible: i) The configuration is similar to the one shown in Figure 7. It is clear that in this case no guide line can be determined. The robots allowed to move are the ones that are at the corner having one empty node as a neighbor, their destination is their adjacent empty node on the borderline they belong to. ii) The remaining cases: One line can be elected as the guide line, this line is the one that contains an 1.block of size 3 ($X3$). The robot that is alone on the borderline ($X1$) is the one allowed to move, its destination is its adjacent empty node on the shortest path towards the middle line (the one that contains $X2$). Note that in a case of symmetry, the scheduler will break the symmetry by choosing one of the two possible neighboring nodes.
- The configuration is of type $(1, 2, 2)$. The robot that is alone on the borderline ($X1$) is the one allowed to move, its destination is its adjacent empty node on the shortest path towards the empty node on the line that contains $X2$.
- The configuration is of type $(1, 3, 1)$. Two sub-cases are possible: i) The configuration is similar to the one shown in Figure 13, Step 1. Note that for this configuration, there is a dedicated algorithm that solves the exploration problem. The algorithm is detailed in Figure 13. Note that since the system is asynchronous, the scheduler in some steps of the algorithm can activates one of the two robots that are allowed to move. In this case, the robot that was supposed to move in the first place is the only one that can move, thus by moving the configuration reached when both robots were activated is reached again ii) The remaining cases: we are sure that there is one robot that is part of an 1.block of size 3 (in the middle line) that has two neighboring empty nodes (Note that there is only five robots and a single 1.block of size 3), let this robot be $R1$. $R1$ is the only one allowed to move, its destination is its adjacent empty node towards the closest robot that is in one of the two borderlines that are horizontal to the 1.block of size 3.
- The configuration is of type $(2, 1, 2)$. Note that the configuration does not contain an 1.block of size 3. Let $D1$ and $D3$ be the two borderlines corresponding to $X1$, $X2$ respectively. The sub-cases below are possible:
  - Both $D1$ and $D2$ contains robots at distance 2 ($d = 2$). In this case, we are sure that there is one robot on the center of the grid (on the middle of the middle line, otherwise

the configuration will contains an 1.block of size 3). This robot is the one allowed to move, its destination is one of adjacent empty node towards the borderline (refer to Figure 8).

  - The robots on $D1$ are at distance 1 and the robots on $D2$ are at distance 2. If the robot that is in the middle line (according to the guide line) is also on a borderline (see Figure 9), we are sure that there is one robot at the corner of the grid not having any neighboring robot. This robot is the one allowed to move, its destination is one of its adjacent empty node. If the robot is in the center of the grid (see Figure 10), then this robot is the one allowed to move its destination is its adjacent empty node towards $D2$.
  - Both $D1$ and $D2$ contains robots at distance 1 ($d = 1$). Let $D3$ be the middle line that is horizontal to both $D1$ and $D2$. The robot allowed to move is the one that is on $D3$, its destination is its adjacent node towards $D1$ or $D2$ (The scheduer will make the choice in the case of symmetry).

- The configuration is of type $(2, 3, 0)$. In this case the robot that in the middle line that contains three robots having an empty node as a neighbor on the line that contains two robots is the one allowed to move, its destination is this adjacent empty node.

- The configuration is of type $(3, 0, 2)$. In this case the robots that are in $X3$ (the line that contains two robots) are the one allowed to move, their destination is their adjacent empty node on the shortest path towards $X2$.

- The configuration is of type $(3, 2, 0)$ but is different from the special configuration (refer to Figure 12). The robots allowed to move are the two robots that are on the line corresponding to $X2$. Their destination is their adjacent empty node on the line that contains $X2$. Its is clear that in the case the scheduler activates only one of these two robots the configuration reached will be the Special configuration (see Figure 6, step 1), Thus the exploration task can be performed as shown in 6. In the case the scheduler activates both robots at the same time, then a tower is created and the configuration reached is like the one shown in Figure 6, step 2. In this case too the exploration can be performed.

Note that once one of the two special configurations is built, one tower is created and the exploration task can be performed. refer to Figures 6 and 13.

Correctness Proof.

Lemma 8 *Starting from a configuration of type* $(1, 2, 2)$*, a configuration of type* $(2, 3, 0)$ *is reached in a finite time.*

Proof. In a configuration of type $(1, 2, 2)$ the robot that is allowed to move is the one that is alone on the borderline containing $X1$, let $R1$ be this robot, its destination is its adjacent empty node towards $X2$, Since line $X2$ contains two robots, when $R1$ joins $X2$, $X2$ will contain an 1.block of size 3 and $X1$ will contain no robot. Thus the configuration reached is of type $(2, 3, 0)$ and the lemma holds. □

Lemma 9 *Starting from a configuration of type* $(1, 3, 1)$*, either a configuration of type* $(2, 2, 1)$ *or of type* $(2, 1, 2)$ *is reached in a finite time.*

Proof. When the configuration is of type $(1, 3, 1)$, we are sure that there is one robot that is part of the 1.block of size 3 on $X2$ that has two neighboring empty nodes. This robot is the one allowed to move its destination is its adjacent empty node towards the closest robot on either $X1$ or $X2$. Suppose that such a robot is the one that is in the middle of the 1.block of size 3. Once the robot has moved, the configuration becomes of type $(2, 1, 2)$ and the lemma holds. If such a robot is at the extremity of the 1.block of size 3, then by moving, the configuration reached is of type $(2, 2, 1)$ and the lemma holds. □

Lemma 10 *Starting from a configuration of type* $(2, 1, 2)$*, a configuration of type* $(3, 0, 2)$ *is reached in a finite time.*

Proof. The cases below are possible:

1. Both $D1$ and $D2$ contains robots at distance 2 ($d = 2$). It is clear that in this case there is one robot that is in the center of the grid. This robot is the one allowed to move, its destination is one of its adjacent empty node. By moving, the robot join a borderline. Note that this borderline contains an 1.block of size 3. Thus the configuration reached will be $(3, 0, 2)$.

2. The robots on $D1$ are at distance 1 and the robots on $D2$ are at distance 2. In this case the robot that is on the borderline on $D2$, being at the corner of the grid and not having any neighboring robot is the one that moves towards one of its adjacent empty node. Note that once the robot has moved, the configuration reached remains of type $(2, 1, 2)$, however, both $D1$ and $D2$ contains robots at distance 1.

3. Both $D1$ and $D2$ contains robots at distance 1 ($d = 1$). Let $D3$ be the middle line that is horizontal to both $D1$ and $D2$. In this case the robot that is on $D3$ is the one allowed to move, its destination is its adjacent empty node towards one of the two neighboring borderlines that contain an 1.block of size 2. Note that we are sure that this robot has at least one empty node as a neighbor otherwise the configuration contains a single 1.block of size 3 and the configuration will not be of type $(2, 1, 2)$. Once the robot has moved, a new 1.block of size 3 is created at one borderline and the configuration will be of type $(3, 0, 2)$.

From the cases above, we can deduce that starting from a configuration of type $(2, 1, 2)$, a configuration of type $(3, 0, 2)$ is reached in a finite time and the lemma holds. □

Lemma 11 *Starting from a configuration of type* $(2, 3, 0)$*, a configuration of type* $(3, 2, 0)$ *is reached in a finite time.*

Proof. When the configuration is of type $(2, 3, 0)$, the robot allowed to move is the one that is on the line that contains $X2$ having an empty node as a neighbor on the line that contains two robots. Note that once the robot has moved, a new 1.block of size 3 is created one borderline of the grid. Thus the configuration reached will be of type $(3, 2, 0)$ and the lemma holds. □

Lemma 12 *Starting from a configuration of type* $(3, 0, 2)$*, either a configuration of type* $(3, 2, 0)$ *or of type* $(3, 1, 1)$ *is reached in a finite time.*

Proof. When the configuration is of type $(3, 0, 2)$, the robots that are on the line that $X3$ are the one allowed to move. When they do, they move to their adjacent empty node towards the line that is horizontal to the the one containing an 1.block of size 3. Note that in the case the scheduler activates both robots allowed to move at the same time, then the configuration reached is of type $(3, 2, 0)$ and the lemma holds. If it is not the case, the configuration reached is of type $(3, 1, 1)$ and the lemma holds. □

Lemma 13 *Starting from a configuration of type* $(3, 1, 1)$*, either a configuration of type* $(3, 2, 0)$ *or of type* $(2, 2, 1)$ *is reached in a finite time.*

Proof. In the case the configuration is similar to the one shown in Figure 7. The robots that are at the corner having an empty node as a neighbor are the one allowed to move. Their destination is their adjacent empty node. Note that in the case the scheduler activates both robots at the same time, the configuration reached is of type $(2, 2, 1)$ and the lemma holds. In the case the scheduler activates only one robot, then the configuration reached remains of type $(3, 1, 1)$ but it is different from the Figure 7. For the other configurations of type $(3, 1, 1)$ (all the configurations that are different from the one shown in Figure 7). The robot that is allowed to move is the one that is single on the borderline that contains $X3$. Its destination is its adjacent empty node on the shortest path towards the line that contains $X2$. Note that once it has moved, the configuration reached is of type $(3, 2, 0)$ and the lemma holds. □

Lemma 14 *Starting from a configuration of type* $(1, 2, 2)$*, a configuration of type* $(3, 2, 0)$ *is reached in a finite time.*

Proof. From Lemma 8, we are sure that starting from a configuration of type $(1, 2, 2)$, a configuration of type $(2, 3, 0)$ is reached in a finite time. From Lemma 11 we are sure that starting from a configuration of type $(2, 3, 0)$, a configuration of type $(3, 2, 0)$ is reached in a finite time. Thus we can deduce that starting from a configuration of type $(1, 2, 2)$, a configuration of type $(3, 2, 0)$ is reached in a finite time and the lemma holds. □

Lemma 15 *Starting from a configuration of type* $(1, 3, 1)$*, a configuration of type* $(3, 2, 0)$ *is reached in a finite time.*

Proof. From Lemma 9, we are sure that starting from a configuration of type $(1, 3, 1)$, a configuration of type $(2, 2, 1)$ is reached in a finite time. From Lemma 14 we are sure that starting from a configuration of type $(1, 2, 2)$, a configuration of type $(3, 2, 0)$ is reached in a finite time. Thus we can deduce that starting from a configuration of type $(1, 3, 1)$, a configuration of type $(3, 2, 0)$ is reached in a finite time and the lemma holds. □

Lemma 16 *Starting from a configuration of type* $(2, 1, 2)$*, a configuration of type* $(3, 2, 0)$ *is reached in a finite time.*

Proof. From Lemma 10, we are sure that starting from a configuration of type $(2, 1, 2)$, a configuration of type $(3, 0, 2)$ is reached in a finite time. From Lemma 12, we are sure that starting from a configuration of type $(3, 0, 2)$, a configuration of type $(3, 2, 0)$ is reached in a finite time. Thus we can deduce that starting from a configuration of type $(2, 1, 2)$, a configuration of type $(3, 2, 0)$ is reached in a finite time and the lemma holds. □

Lemma 17 *Starting from any configuration that is towerless, a configuration of type* $(3, 2, 0)$ *is reached in a finite time.*

Proof. From Lemmas 11-16, we can deduce that starting from any configuration that is towerless, a configuration of type $(3, 2, 0)$ is reached in a finite time and the lemma holds. □

Exploration. `Exploration` starts from three special configurations shown in either Figure 6-Case (1), Figure 13-Case(1a), or Figure 13-Case(1b). In the former case, the unique robot that is (1) in a borderline, (2) not at a corner, and (3) not in the borderline linking the two occupied corners, moves toward the center. In Case (1a) in Figure 13, the unique robot located at a corner moves toward one of its neighbors (chosen by the scheduler). Similarly, in Case (1b) in Figure 13, the robot located at the center moves toward one of its neighbors. In the three cases, one tower is created and the system reaches Case 2 of either Figure 6 or Figure 13, depending on the initial configuration. Next, the exploration is made following the movements depicted in either Figure 6 or Figure 13, respectively.

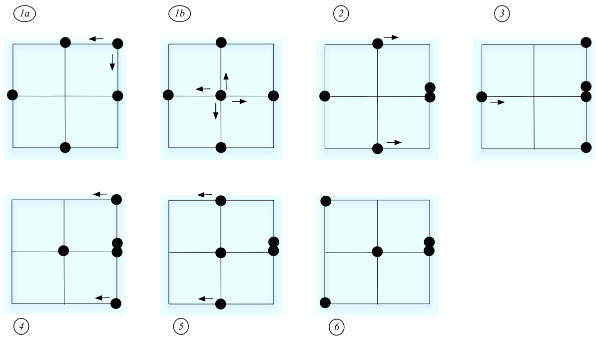

Figure 13: Special Exploration of grids $(3, 3)$

Lemma 18 *Starting from one of the three special configurations shown in Figure 6-Case* (1)*, Figure 13-Case*$(1a)$*, or Figure 13-Case*$(1b)$*, all the nodes of the grid are explored and the algorithm stops.*

Lemma 19 *Starting from any configuration of type* $(3, 2, 0)$*, the exploration can be performed.*

Proof. If the configuration is the special configuration (refer to Figure 6 (step 1)), then according to Lemma 18, the exploration task is performed and all the nodes of the ring are explored. If the configuration is as the one show in Figure 12, then the two robots that are not part of the 1.block of size 3 are the one allowed to move, their destination is their adjacent node in the center of the grid. In the case where the scheduler activates only one of the two robots allowed to move, the special configuration is reached and the lemma holds. If both robots are activated then a tower is created in the center of the grid and the configuration reached will be as the one shown in Figure 6 (Step2) and in this case too the exploration is performed and the lemma holds. □

From the lemmas above we can deduce that:

Theorem 7 *The deterministic exploration of a* $(3, 3)$*-Grid can be solved in the CORDA model using 5 oblivious robots.*

6 Conclusion

We presented necessary and sufficient conditions to explore a grid shaped network with a team of k asynchronous oblivious robots. Our results show that, perhaps surprisingly, exploring a grid is easier than exploring a ring, even when robots do not have any common orientation. In the ring, deterministic (respectively probabilistic) solutions essentially require five (respectively four) robots. In the grid, three robots are necessary (even in the probabilistic case) and sufficient (even in the deterministic case) in the general case, while particular instances of the grid do require four or five robots. The asymmetry induced by the three classes of nodes (nodes of degree two, three, and four) is the main reason for the problem to be solvable by a smaller number of robots. Expanding our results to torus shaped networks is a challenging ongoing work, as all nodes have the same degree in such networks.

References

[1] Roberto Baldoni, François Bonnet, Alessia Milani, and Michel Raynal. On the solvability of anonymous partial grids exploration by mobile robots. In *12th International Conference on Principles of Distributed Systems (OPODIS)*, pages 428–445, 2008.

[2] Shantanu Das, Paola Flocchini, Nicola Santoro, and Masafumi Yamashita. On the computational power of oblivious robots: forming a series of geometric patterns. In *29th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 267–276, 2010.

[3] Stéphane Devismes. Optimal exploration of small rings. In *Proceedings of the Third International Workshop on Reliability, Availability, and Security*, WRAS '10, pages 9:1–9:6, New York, NY, USA, 2010. ACM.

[4] Stéphane Devismes, Franck Petit, and Sébastien Tixeuil. Optimal probabilistic ring exploration by semi-synchronous oblivious robots. In *16th International Colloquium on Structural Information and Communication Complexity (SIROCCO 2009)*, LNCS, Springer, pages 203–217, 2009.

[5] Yoann Dieudonné, Ouiddad Labbani-Igbida, and Franck Petit. Circle formation of weak mobile robots. *ACM Transactions on Adaptive and Autonomous Systems (TAAS)*, 3(4), 2008.

[6] P. Flocchini, D. Ilcinkas, A. Pelc, and N. Santoro. Computing without communicating: Ring exploration by asynchronous oblivious robots. In *14th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 105–118, 2007.

[7] Paola Flocchini, David Ilcinkas, Andrzej Pelc, and Nicola Santoro. Remembering without memory: Tree exploration by asynchronous oblivious robots. In *15th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 33–47, 2008.

[8] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Peter Widmayer. Arbitrary pattern formation by asynchronous, anonymous, oblivious robots. *Theor. Comput. Sci.*, 407(1-3):412–447, 2008.

[9] Tomoko Izumi, Taisuke Izumi, Sayaka Kamei, and Fukuhito Ooshita. Mobile robots gathering algorithm with local weak multiplicity in rings. In *17th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 101–113, 2010.

[10] Ralf Klasing, Adrian Kosowski, and Alfredo Navarra. Taking advantage of symmetries: Gathering of asynchronous oblivious robots on a ring. In *12th International Conference on Principles of Distributed Systems (OPODIS)*, pages 446–462, 2008.

[11] Ralf Klasing, Euripides Markou, and Andrzej Pelc. Gathering asynchronous oblivious mobile robots in a ring. *Theor. Comput. Sci.*, 390(1):27–39, 2008.

[12] 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.

[13] Giuseppe Prencipe. Instantaneous actions vs. full asynchronicity : Controlling and coordinating a set of autonomous mobile robots. In Antonio Restivo, Simona Ronchi Della Rocca, and Luca Roversi, editors, *ICTCS*, volume 2202 of *Lecture Notes in Computer Science*, pages 154–171. Springer, 2001.

[14] Samia Souissi, Xavier Défago, and Masafumi Yamashita. Using eventually consistent compasses to gather memory-less mobile robots with limited visibility. *ACM Transactions on Adaptive and Autonomous Systems (TAAS)*, 4(1), 2009.

[15] Ichiro Suzuki and Masafumi Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM J. Comput.*, 28(4):1347–1363, 1999.