Oblivious Robots on Graphs: Exploration

David Ilcinkas

CHAPTER 1

Oblivious Robots on Graphs: Exploration

David Ilcinkas

CNRS, LaBRI, Univ. Bordeaux, Talence, France
david.ilcinkas@labri.fr
Homepage: https://www.labri.fr/perso/ilcinkas/

Abstract. This chapter focuses on the problem of exploring a graph by a team of mobile robots endowed with vision. More precisely, we consider here mobile robots operating under the `Look-Compute-Move` paradigm in discrete environments modeled as graphs. The goal for these robots is to explore the graph in which they are, that is to visit all vertices of the graph.

Keywords: Oblivious robots · Graph exploration · Terminating exploration · Exclusive perpetual exploration

1 Introduction

In this chapter, we consider mobile entities, called robots, operating under the `Look-Compute-Move` paradigm. An activated robot starts first by taking an instantaneous snapshot of its environment (the `Look` phase), then it computes whether and where it wants to move (the `Compute` phase), and finally it moves to the decided new position (the `Move` phase). Robots operating under the `Look-Compute-Move` paradigm are classically considered in continuous environments, usually the plane. The studies on these robots were however recently extended to the case of discrete environments, modeled as graphs (see [6] and [24] for short surveys on the subject). This chapter focuses on these discrete environments. One motivation to consider discrete environments is to get rid of possibly annoying geometric considerations in order to focus on issues directly related to the weaknesses of the robots (anonymity, obliviousness, etc.), to the symmetries of the environment, and to the asynchrony. Another motivation is more practical and comes from the fact that vision sensors do not have an infinite precision. Considering discrete environments thus acknowledges the fact that many vision sensors output digital and thus discrete snapshots of the environment.

We consider in this chapter the graph exploration problem, in which the robots have to visit every vertex. More precisely, two variants of the problem were studied so far in the literature (in the considered model). The first variant is called *terminating exploration* and requires that, first, each vertex is visited by at least one robot, and second, that eventually all robots stop moving. The second variant is called *exclusive perpetual exploration* and requires that, first,

each robot visits every vertex of the graph infinitely often, and second, that no two robots traverse the same edge at the same time nor visit the same vertex at the same time. Exploring a graph is a fundamental task in mobile robot computing and can be used, for example, to search for a specific information, or to discover and list all the services provided by the vertices. Exploration (perpetual in particular) is also interesting for maintenance purposes, where the robots can check forever whether the vertices are properly functioning. Finally, the exclusivity property models the physical constraints that the robots may have if, for example, they operate in environments with limited available space that prevent them from crossing each other or being at the same place.

The chapter is structured as follows. Section 2 defines more precisely the model and the problems, and gives a first simple preliminary result. Sections 3 and 4 respectively consider the terminating exploration and the exclusive perpetual exploration problems. In both cases, known results from the literature are presented, and then the usual tools and techniques used in these works are summarized. Finally Section 5 discusses what concerns the correctness of the results, while Section 6 concludes the chapter.

2 Model and preliminaries

2.1 Model

The environment. We model the environment as a simple undirected connected graph $G = (V, E)$. The number $|V|$ of vertices is usually denoted n, while the number $|E|$ of edges is usually denoted m. The graph is assumed to be anonymous: neither vertices nor edges are labeled (or, equivalently, such labels cannot be seen by the robots).

The robots. On this graph operate mobile entities called *robots*. They can move from vertex to vertex via the edges of the graph. The robots are all anonymous and identical, i.e. they all execute the same algorithm. They have no direct means of communication. Unless otherwise specified, the robots will be assumed to be oblivious: they do not have persistent memory. When several robots occupy the same vertex, we say that there is a *tower* on this vertex.

The `Look-Compute-Move` *cycle.* The robots operate by repeatedly executing `Look-Compute-Move` cycles. In the `Look` phase, a robot takes an instantaneous egocentric snapshot of its environment. This includes the structure of the graph around it, and the presence of robots on the seen vertices. The structure of the snapshot will be detailed later. Note however that all robots are perceived on vertices, not on edges. In the `Compute` phase, the robot decides whether to move or not, and in the first case to which neighboring vertex. For oblivious robots, this computation is solely based on the last snapshot. Finally, in the `Move` phase, the robot moves to the chosen neighbor, or stays idle if it decided to do so. Moves are considered instantaneous, which is consistent with the fact that robots are seen on vertices in the snapshots. Note nevertheless that this model has been

shown equivalent to the model with continuous moves (but still considering that robots are always seen on vertices), and this equivalence is certified using the Coq proof assistant [1].

Timing assumptions. Different levels of asynchrony are classically considered in the literature. In the fully synchronous model $\mathcal{F}$SYNC, all robots execute their `Look-Compute-Move` cycles simultaneously. Differently speaking, at each round, every robot executes its full `Look-Compute-Move` cycle. In the semi-synchronous model $\mathcal{S}$SYNC, at each round, a non-empty subset of the robots, chosen by an adversary, execute a full `Look-Compute-Move` cycle. Finally, in the asynchronous model $\mathcal{A}$SYNC, the timing between the different phases of the `Look-Compute-Move` cycles performed by the different robots is arbitrary, with the only constraint that, for each robot, the time between two consecutive phases is finite (but possibly unbounded). As a consequence, a move can be performed based on an outdated snapshot in this model.

The snapshot. During the `Look` phase, a robot perceives the structure of the graph and the presence of robots around it within a *visibility radius* ρ given by the model. More precisely, the snapshot taken by a robot consists of the rooted subgraph induced by the vertices at distance at most ρ from the vertex occupied by the robot and, for each seen vertex, of the perceived number of robots on it. The accuracy of the perceived number of robots is given by another model parameter called the *multiplicity detection*. If weak multiplicity detection is assumed, a robot is only able to distinguish between the presence of "zero", "one", or "more than one" robots on a seen vertex. On the contrary, strong multiplicity detection assumes that a robot knows the exact number of robots that are present on a seen vertex. Orthogonally, multiplicity detection can be either local or global. In the case of local multiplicity detection, a robot only knows the multiplicity of the vertex it occupies (whether in the weak or the strong sense), while in the case of global multiplicity detection, a robot knows the multiplicity of all vertices.

Configurations, Views, and Symmetries. The description of the graph, together with the indication of the exact number of robots located on each vertex, constitute a *configuration*. The *view from vertex* u is any rooted graph isomorphic to the subgraph induced by the vertices at distance at most ρ (the visibility radius) from u, and the corresponding perceived number of robots on these vertices (depending on the multiplicity detection assumption). In the `Look` phase, a robot is given a view from the vertex it is located on. Therefore, symmetries of the graph are somehow still present in the snapshot. Indeed, for example in a ring, if a robot lies on an axis of symmetry of the configuration, then it will not be able to differentiate one direction from the other. Therefore, if it decides to move to a neighboring vertex, the actual move will be to a neighbor chosen by the adversary. More generally and more formally, we say that two vertices v and v' are *similar* with respect to u if there exists a view from vertex u such that there exist two vertices w and w' of the view and two isomorphisms ϕ and

ϕ' such that $\phi(w) = \phi'(w) = u$, $\phi(w') = v$, and $\phi'(w') = v'$. Intuitively, v and v' are similar with respect to u if v and v' are indistinguishable for a robot located in u (taking into account the visibility radius and the multiplicity detection). If a robot decides in the `Compute` phase to move to a vertex v, then in the `Move` phase it will actually move to any vertex v' similar to v with respect to the robot current position, and the choice of v' is made by the adversary.

Terminating exploration. A team of robots solves the problem of *terminating exploration* in a graph family $\mathcal{G}$ if, for any graph $G \in \mathcal{G}$, for any behavior of the adversary controlling the asynchrony and the choices between similar neighbors, and starting from any initial configuration without towers, each vertex of the graph is visited by *at least one* robot and the robots eventually reach a configuration in which no robots ever move.

Exclusive perpetual exploration. A team of robots solves the problem of *exclusive perpetual exploration* in a graph family $\mathcal{G}$ if, for any graph $G \in \mathcal{G}$, for any behavior of the adversary controlling the asynchrony and the choices between similar neighbors, and starting from any initial configuration without towers, each vertex of the graph is visited by *every* robot *infinitely often* and the exclusivity property is satisfied. This property is satisfied if no two robots are ever located at the same vertex at the same time, and no two robots ever cross the same edge at the same time in opposite directions.

2.2 Preliminaries

Let us first start by giving some justifications about the choice of the set of initial configurations, which is defined as the set of configurations without towers for both problem variants. First, note that the exclusivity property of the exclusive perpetual exploration problem is violated in any configuration with at least one tower. The set of configurations without towers is thus the maximal set of initial configurations that is meaningful for this problem. Concerning terminating exploration, note that its definition implies the existence of a configuration in which all robots decide not to move. If not all vertices are occupied by a robot, then this configuration must not be an initial configuration for the problem to be solvable, if the robots are oblivious. Differently speaking, for terminating exploration, the set of initial configurations must not contain all possible configurations when the robots are oblivious and are less than the number of vertices (which is the case in all the papers of the literature considered here). Taking as initial configurations the configurations without towers is thus also rather natural for this problem. Finally, one may also look for *universal* algorithms [23]. Given a number of robots in a specified model, a universal algorithm is an algorithm solving the problem from any initial configuration which is solvable in the considered setting.

Most papers in the literature on the subject concern specific families of graphs. The most commonly studied family is the family of all rings. Other studied graphs are the trees, the grids, the tori and some variations of these

graphs. Given such a family, the usual focus is then on the *smallest and/or largest exploring teams*, that is on the numbers $\kappa^-(n)$ and $\kappa^+(n)$ defined as the respectively smallest and largest numbers of robots that can explore any n-vertex graph of the given family. In the following, the considered family and exploration type will be clear from the context.

We now present a first simple technical result, inspired from Lemma 2.1 in [20], which allows nevertheless to already draw some conclusions on the value $\kappa^-(n)$ in the case of the rings.

Lemma 1. *Let $n \geq 3$ and $k < n/2$ be two positive integers such that k divides n. Then both terminating exploration and perpetual exclusive exploration of an n-vertex ring are not deterministically solvable by a team of k (possibly non-oblivious) robots, even with full visibility, global strong multiplicity detection, and in the $\mathcal{F}$SYNC model.*

Proof. Let us fix any algorithm for a team of k robots. Consider as initial configuration a configuration in which the k robots are regularly scattered around the ring. The configuration being perfectly periodic, all robots have the same view. If a robot decides to move to a neighbor (both neighbors are similar with respect to the robot's current position), then all robots decide to move (they have the same initial state), and the adversary makes them move in the same direction. Therefore, the configuration stays periodic and all robots still have the same state (which may be different from the initial state if the robots are non-oblivious). We make the adversary continue to act that way. More specifically, the adversary makes them move in some fixed direction for each odd round, and in the other direction for each even round (forgetting about rounds in which the robots decide to stay idle). In such an execution, and until they decide to stay idle, each robot keeps going back and forth between its initial location and one specific neighbor. Since $n/k \geq 3$, there exist k vertices which are never visited by the robots. Thus exploration (whatever its type) is not solved by this algorithm. □

As a corollary, a ring of size n equal to three times the least common multiple of $1, 2, \cdots, k-1$ cannot be deterministically explored by less than k robots. Some calculations using the Prime Number Theorem show that $k = \Theta(\log n)$ in that case, see [20]. Therefore, there exists a positive constant c such that, for infinitely many n, we have $\kappa^-(n) \geq c \log n$ for deterministic algorithms.

3 Exploration with stop

In this section, we will almost only consider oblivious robots, that is robots using in the `Compute` phase only the snapshot taken in the preceding `Look` phase. In particular, the robots do not have access to time, and thus they do not know whether this is the beginning of the execution or not when they see a configuration without towers.

3.1 Known results

In the rings.

As already noted in Section 2.2, and already proven by Flocchini et al. in [20], there exists a positive constant c such that, for infinitely many n, we have $\kappa^-(n) \geq c \log n$ for deterministic algorithms. This in fact remains true for probabilistic algorithms [15], but only for the asynchronous model $\mathcal{A}$SYNC. Indeed, in the semi-synchronous model $\mathcal{S}$SYNC, a constant number of robots, namely 4 probabilistic robots, are necessary and sufficient to solve the terminating exploration problem in any n-vertex ring with $n > 4$, see [15].

Let us now focus on deterministic algorithms. The lower bound 4 on $\kappa^-(n)$ still holds in $\mathcal{S}$SYNC (in $\mathcal{F}$SYNC, only 3 is a clear lower bound for every n, in particular when n is odd). When n is even, $\kappa^-(n) \geq 5$ in the $\mathcal{S}$SYNC model [22]. These values are somehow optimal. Indeed, 4 robots can explore the rings of odd size in $\mathcal{S}$SYNC [21], and provided that n is not a multiple of 5, a team of 5 robots can explore the n-vertex ring even in the $\mathcal{A}$SYNC model [22]. As pointed out, $\kappa^-(n)$ may be logarithmic in n for infinitely many values of n, but this cannot go worse in the sense that $\kappa^-(n)$ is always in $O(\log n)$. Indeed, for any $k \geq 17$ that is co-prime with n, a team of k robots can explore the n-vertex ring [20].

Note that all the results presented here so far for the case of the rings are strong with respect to multiplicity detection in the following sense. All lower bounds (impossibility results) are valid even with global strong multiplicity detection, while all upper bounds (algorithms) are assuming global weak multiplicity detection. Moreover, note that the results are valid for sufficiently large values of n, and may vary for small values of n.

Limited visibility has also been considered, for deterministic algorithms, and in the case of global (up to the visibility radius) strong multiplicity detection. When the visibility radius ρ is 1, even a limited amount of asynchrony renders the problem impossible to solve: there are no deterministic algorithms working in the $\mathcal{S}$SYNC model, for any number $k < n$ of robots [11]. In the same paper, the authors show that 5 robots are necessary in the $\mathcal{F}$SYNC model, and they present an algorithm for 5 robots working when all robots are on consecutive vertices in the initial configuration. If robots however have 1 bit of persistent memory which is visible/accessible by any robot within their visibility radius (the $\mathcal{LUMINOUS}$ model), then three (in $\mathcal{F}$SYNC) or four (in $\mathcal{S}$SYNC and $\mathcal{A}$SYNC) robots are necessary and sufficient to solve terminating exploration [23] (for specific initial configurations). These numbers are reduced by one for (non-exclusive) perpetual exploration.

The cases $\rho = 2$ and $\rho = 3$ are considered by Datta et al. in [12]. For $\rho = 2$, there exists an algorithm in the $\mathcal{A}$SYNC model for 7 robots when all robots are on consecutive vertices in the initial configuration. The number of robots can be reduced to 5 when $\rho = 3$. Finally, for $\rho = 3$, there exists an $\mathcal{A}$SYNC algorithm for 7 robots that can handle more general initial configurations: the robots start in a position such that they are "connected by vision" but need not to be on consecutive vertices.

In the trees.

The trees [18], and the sub-case of the lines [19], have only been considered in the deterministic setting and assuming global weak multiplicity detection.

In trees, the absence of port numbers (the anonymous graph assumption) makes empty leaves having the same parent indistinguishable (they are similar with respect to the parent). Therefore, in order to explore sibling leaves despite any choice of the adversary concerning similar vertices, at least one robot must be sent to each leaf attached to a given parent. If a vertex has more than two leaves attached to it and all of them are occupied by robots, then at least two of them are similar, having both either a single robot or a tower. The adversary can thus make these robots merge if the algorithm decides to move them. Therefore, one can prove that $\Omega(n)$ robots are necessary in some trees (at least in complete ternary trees) in the $\mathcal{S}$SYNC model. Note that this lower bound heavily relies on the weak multiplicity assumption. For trees of maximum degree 3, less robots may be used: $O(\log n / \log \log n)$ robots are sufficient in such trees, even in the $\mathcal{A}$SYNC model. This number is actually necessary for some trees because $\Omega(\log n / \log \log n)$ robots are necessary to explore complete binary trees, even with global strong multiplicity detection and in the $\mathcal{F}$SYNC model.

In lines, symmetries are much more limited, and the solvable cases are fully characterized. A team of $k < n$ robots can solve terminating exploration in the n-vertex line if and only if $k = 3$, or $k \geq 5$, or $k = 4$ and n is odd. The lower bounds are proved in the $\mathcal{S}$SYNC model while the upper bounds hold even in the $\mathcal{A}$SYNC model.

In the grids and tori.

The situation for grids [13] and tori [14] resembles the situation for lines and rings.

In grids, where symmetries are limited, we have $\kappa^-(i,j) = 3$ for all (i,j)-grids (except the $(2,2)$-grid and the $(3,3)$-grid for which we have $\kappa^-(2,2) = 4$ and $\kappa^-(3,3) = 5$). The lower bound holds even for probabilistic algorithms, in the $\mathcal{S}$SYNC model, and assuming global strong multiplicity detection, while the algorithm proving the upper bound is deterministic, works in the $\mathcal{A}$SYNC model, and assumes global weak multiplicity detection.

Tori have much more symmetries and thus require more robots. Indeed, $\kappa^-(i,j) \geq 5$ for deterministic algorithms solving the terminating exploration problem in (i,j)-tori. Allowing probabilistic algorithms, we have $\kappa^-(i,j) = 4$ (for sufficiently large tori). The lower bounds assume global strong multiplicity detection while the upper bound assumes global weak multiplicity detection. All results for tori are proved in the $\mathcal{S}$SYNC model.

In the general graphs.

The case of arbitrary graphs has been considered by Chalopin et al. in [10]. More precisely, the paper considers arbitrary graphs with port numbers, that is graphs for which, at each vertex, the incident edges are distinguished by local port numbers from 1 to the degree of the vertex. The class $\mathcal{H}_k$ is then defined

as the class of rigid configurations of k robots, i.e. the class of configurations of k robots (the graphs with port numbers and the positions of the k robots) such that there is no non-trivial automorphism preserving the port numbers and the robots locations. Chalopin, Flocchini, Mans, and Santoro studied the terminating exploration problem in the $\mathcal{A}$SYNC model in these classes, assuming global weak multiplicity detection. They proved that exploration is impossible for $k < 3$ robots, they characterized the graphs that are explorable in $\mathcal{H}_3$, and they show that all graphs are explorable in $\mathcal{H}_4$ and in every $\mathcal{H}_k$ with an odd $k > 3$. The case of even $k > 4$ is left open but can be reduced to the existence of a gathering algorithm for $\mathcal{H}_k$.

3.2 Usual tools and techniques

Impossibility results.

The lower bounds on the number of robots that is necessary to solve terminating exploration are of similar flavor for the different considered topologies, and use the following arguments.

First of all, since the specification of the problem requires termination, there must exist a configuration in which no robot decides to move. When $k < n$, and because of obliviousness, this configuration cannot be an initial configuration, and thus at least one tower must be formed. This already proves that a single robot is never sufficient (if $n > 1$).

Then note that any suffix of a valid execution must remain valid as long as the first configuration of the truncated execution has no towers. Indeed, such a configuration can be an initial configuration, and, because of obliviousness, these two executions (the initial one and the truncated one) both respect the algorithm. Exploration must thus be performed after a tower is formed and keeping at least one tower in each configuration.

The next observation is that towers are difficult to move in asynchronous environments. Indeed, even in the semi-synchronous $\mathcal{S}$SYNC model, if the robots in a tower decide to move, then only one may be activated and the tower may be destroyed. As this is often the case (in particular for a small number of robots), let us assume that it is impossible to move towers. Therefore exploration must be performed by at least another robot and thus 2 robots are not sufficient either.

In order to obtain a larger lower bound, one generally needs a further observation about the suffix of the execution in which all configurations contain at least one tower. This observation is the fact that any two configurations in this suffix must be distinguishable from the point of view of the robots. Indeed, if this is not the case, the adversary can make the execution periodic by repeating the same choices, and the termination requirement is not fulfilled. Besides, at each step of the execution, at most k new vertices are explored, and possibly even less if some robots are blocked in a stationary tower. Therefore, there must exist sufficiently many distinguishable configurations with a tower for the problem to be solvable with k robots. Such a counting argument is generally sufficient to obtain rather good lower bounds.

Algorithmic techniques.

The algorithms presented in the different papers also have some similarities. Indeed, as previously seen, exploration must be performed after a tower is formed, and at least a tower must be kept until termination. Therefore, the algorithms for terminating exploration by oblivious robots generally consist of three phases: a set-up phase in which no tower is created but a special configuration is reached, a tower-creation phase in which a tower is created, and finally the exploration phase in which some of the robots explore the graph.

The first phase is usually the most complex one. Indeed, this phase starts from an arbitrary initial configuration while the two other phases start from specific configurations (or classes of configurations). The special configurations that the robots try to reach in the set-up phase are generally configurations without towers where robots are gathered next to each other. In the rings, a special configuration is typically a configuration in which all robots form a block by positioning themselves on consecutive vertices. In the trees, all robots go toward the leaves. In the grids, the robots go towards one of the corners.

Reaching such a special configuration is generally highly non trivial, in particular because robots are oblivious. This constraint prevents the robots from remembering what their plan was at the beginning of the execution. Intuitively, if one constructs a directed graph whose vertices are the possible configurations and in which there is an arc between two configurations if one can reach the second configuration from the first one by applying one step of the algorithm, then this directed graph must be acyclic. This may be not too hard to achieve in graphs for which there are no non-trivial automorphisms, but it can be very tricky in graphs with a lot of symmetries.

The issue with symmetries is that it may be hard, or even impossible, to break them. Therefore, an algorithm may be forced to schedule several robots having the same view of the environment to move in the current configuration. Combined with the asynchrony, several different configurations can be obtained depending on the choices made by the adversary. In the ring for example, it may happen that from a symmetric configuration only one robot is scheduled to move by the adversary and the obtained configuration is again symmetric but with a different axis of symmetry. Even worse, in the asynchronous $\mathcal{A}$SYNC model, one robot may decide to move but the adversary decides to delay this move while the other robots are progressing. The delayed move is said to be *pending* in this case. When this move is finally allowed by the adversary, it does not necessarily correspond to the current situation and may create issues. It is thus often very difficult to design an algorithm that avoids cycling among the configurations.

One can nevertheless express two guidelines that algorithm designers should try to follow. The first one is to minimize the numbers of robots that decide to move in a given configuration. Typically, in an asymmetric configuration, the algorithm should designate a single robot to move. The second guideline could be expressed as follows. In a given configuration, if a robot may have a pending move (for example because the current configuration may come from a symmetric configuration where several robots with the same view were designated to move),

then the algorithm should choose this robot to move. Indeed, this would force the adversary to execute the pending move.

Once a special configuration is reached, it is however rather easy to form a tower, since robots are located on contiguous vertices. In rings for example, the robot in the middle of an odd block of robots can simply move in an arbitrary direction to form a tower with its neighbor.

In the exploration phase, usually just a few robots, 1 or 3, are actually exploring the graph. The other robots are used to keep track of the process and to break symmetries. In rings, grids and tori, a tower and a few stationary robots are sufficient to break symmetries. The logarithmic number of robots that may be needed in some rings, see Section 2.2, is only due to the fact that smallest numbers of robots allow periodic configurations. On the contrary, the $\Theta(\log n / \log\log n)$ robots used in the trees are really used by the algorithm. Indeed, in very regular trees (typically the complete binary trees), there are many a priori indistinguishable leaves and the robots need a way to distinguish them. For this purpose, the robots maintain a counter that stores the number of explored leaves. This allows the exploration team (two robots forming a tower, and an isolated robot) to know which leaf is the next one to explore. This counter is visually implemented in the tree by locating the $\Theta(\log n / \log\log n)$ robots at carefully chosen positions.

4 Exclusive perpetual exploration

In this section, we will only consider deterministic algorithms. Recall that contrary to terminating exploration, exclusive perpetual exploration requires that each vertex is visited by *every* robot, and so infinitely often. Moreover, the exclusivity property forbids the robots to cross each other on an edge or to be on the same vertex at the same time.

4.1 Known results

With memory.

The exclusive perpetual exploration problem has first been investigated in the case of robots having memory, in the fully synchronous $\mathcal{F}$SYNC model. In [3], given any graph, a labeled mobility tree is defined and a parameter q is associated to it. The authors then prove that $\kappa^+(n) \leq n - q$ for any n-vertex graph of associated parameter q, even with infinite visibility radius.

It turns out that this bound is tight for the partial grids with sense of direction. These are subgraphs of the grids such that edges are locally labeled by the cardinal points N, S, E, W. More precisely, a graph can be explored if and only if the number k of robots is less than or equal to this bound $n-q$, in the case of an infinite visibility radius. For a null visibility radius, the problem is impossible to solve. For visibility radius 1, the problem is solvable if and only if $k \leq n - q$ except when $q = 0$, in which case the condition is $k \leq n - 1$, see [4].

In the rings.

As already noted in Section 2.2, exclusive perpetual exploration cannot be solved when the number k of robots divides the number n of vertices. Since towers are not permitted, the same reasoning can be applied to configurations in which vertices without robots are regularly spaced. Therefore, the problem for $k = n - k'$ robots cannot be solved as well when k' divides n, see [5]. Because of symmetries and of the exclusivity property, the problem cannot be solved either when the number of robots is even. All these results hold even in the $\mathcal{F}$SYNC model. Finally, in the same paper, the authors claim that for n and k coprime, in the $\mathcal{A}$SYNC model, $\kappa^-(n) = 3$ if $n \geq 10$ (and is larger otherwise), and $\kappa^+(n) = n - 5$ (for k odd). The algorithm in [5] justifying $\kappa^-(n) = 3$ is actually not correct for $n = 10$, but a corrected version is given in [9], see Section 5.2 for details.

Focusing on rigid initial configurations, i.e. on initial configurations such that there is no non-trivial automorphism preserving the robots locations, the situation is a bit different. In [17], the authors present an algorithm for k robots solving the exclusive perpetual exploration problem in n-vertex rings in the $\mathcal{A}$SYNC model when $n \geq 10$ and $5 \leq k \leq n - 3$, except for $k = 5$ and $n = 10$.

This latter case has been investigated by Bonnet et al. in [7], where a generic method is proposed, and implemented, to list all possible protocols using only rigid configurations of a given number k of robots in a given graph, in the semi-synchronous $\mathcal{S}$SYNC model. The authors used this method to prove that exclusive perpetual exploration is impossible for $k = 5$ robots in a ring of $n = 10$ robots. The proof uses the full specification of the problem in the sense that, if one relaxes the definition of the problem by allowing vertices to be visited by only some robots and not all of them, then the problem becomes solvable for this case (still in the $\mathcal{S}$SYNC model).

In the grids.

Similarly as for terminating exploration, exclusive perpetual exploration has also been considered in grids [8]. Contrary to the papers considering the partial grids with sense of direction, this paper considers grids without holes and without any edge labels or port numbers (and thus without sense of direction), focuses on oblivious algorithms, and assumes the $\mathcal{A}$SYNC model. The main result of the paper is a proof that $\kappa^-(n) = 3$ in all n-vertex grids having at least two rows and two columns, except for the $(2, 2)$- and $(2, 3)$-grids in which exclusive perpetual exploration is impossible for any k.

4.2 Usual tools and techniques

The labeled mobility tree and its associated parameter q.

We now describe in more details how the labeled mobility tree and its associated parameter q are defined from any graph. Recall that this parameter q is used to bound the maximal number of robots that can solve exclusive perpetual exploration in a graph.

Consider any graph G. The first step is to label the vertices with labels from the set $\{0, 1, 2\}$ as follows. Label a vertex with 0 when it is of degree 2 and it does not belong to any non-singleton bridge-less subgraph of G (i.e. it does not belong to any 2-edge-connected component of G of size at least 2). Label a vertex with 1 if it is a leaf of G or if it belongs to a non-singleton bridge-less subgraph of G. Label a vertex with 2 in any remaining case.

The second step consists in compressing each maximal non-singleton bridge-less subgraph of G (i.e. each non-singleton 2-edge-connected component of G) into a single vertex with label 1. The obtained tree is called the labeled mobility tree associated to G.

A mutual exclusion path in this labeled mobility tree is then a path whose extremities have a label different from 0 but whose internal vertices (if they exist) have label 0. The length of such a path is defined as the number of edges of the path plus the number of extremities with label 2. The parameter q associated to the labeled mobility tree, and thus to the graph G, is then simply the maximal length of a mutual exclusion path.

Intuitively, a mutual exclusion path is a sequence of bridges of G acting as a long bridge of the graph separating two parts of it. If a robot wants to move from one part to the other (which it has to do infinitely often to solve the problem), then it must traverse this long bridge without crossing or meeting any other robot. This more or less explains why exclusive perpetual exploration cannot be solved by more than $n - q$ robots.

Algorithmic techniques for oblivious robots in rings and grids.

Similarly as for terminating exploration, an algorithm roughly defines a directed graph of configurations. In the case of exclusive perpetual exploration, instead of having a DAG (directed acyclic graph) pointing to a set of terminal configurations (configurations without outgoing edges), the DAG points toward some (generally just one) cycles of configurations achieving exploration. Differently speaking, the robots try to enter a cycle of configurations such that every vertex is explored by every robot when this cycle of configurations is performed forever.

In general grids with three robots for example, the robots gather towards a corner and then two robots become more or less stationary, mainly acting as symmetry breakers, while the third robot explores a large part of the grid (at least half of it). When the explorer terminates its part, the robots are in the same position as before but near the opposite corner and with permuted roles. After six such phases, every robot has explored the whole grid.

In rings with three robots, the exploration is performed simultaneously by all the robots. The purpose of the algorithm is to make the robots form a small asymmetric pattern that moves along the cycle forever like a worm. More precisely, the basic pattern consists of two robots on consecutive vertices and the third robot, marking the tail of the worm, a little bit further (two vertices are left empty between the head and the tail). Let us denote such a pattern by 11001 (1 denotes an occupied vertex while 0 denotes an empty vertex). The worm is

then moved by moving repeatedly each robot at a time in one direction from the head to the tail. Hence, from a pattern 11001, the pattern 101001 is formed, then the pattern 110001, and finally the tail moves forming back the pattern 11001, but all robots are now shifted by one vertex in the same direction.

Obviously, as for terminating exploration, the difficult part is the initial phase in which the robots have to deal with asynchrony and symmetries while forming a predefined pattern of the exploration phase.

5 Correctness and related questions

As it is often the case in distributed computing, proving impossibility results and the correctness of algorithms may be challenging. This section discusses the related techniques used in the literature.

5.1 Hand-written proofs

Because of asynchrony and of the symmetries, many executions are possible given an initial configuration and an algorithm. Proofs of correctness are thus often based on a case-by-case analysis, usually rather tedious. This can also be the case for impossibility proofs, where a wide variety of possible algorithms may be explored. In both cases, this is especially true for small graphs, for which general arguments may be more difficult to find.

The situation can sometimes be summarized by exhibiting a DAG (directed acyclic graph) of configurations allowing to visualize the different cases and justifying the convergence to a specific configuration (or a class of configurations). This may be convenient for small graphs but, again, may be intractable for arbitrarily large graphs.

As a consequence, potential functions may also be used to prove convergence. It is however not always simple to find such a potential function and one solution consists in combining all these approaches. Typically, one can use a case-by-case analysis to distinguish several classes of configurations, construct a directed graph of accessibility among these classes and prove that this directed graph is almost a DAG, except for few cycles for which a potential function allows to prove that such a cycle is used a finite number of times.

Such a combined approach gives a proof in which one can have some confidence, but even such proofs are prone to human errors. In particular, it is usually very difficult to convince oneself that no tricky sub-cases have been forgotten. This lack of confidence in human-written (and human-checked) proofs gave rise in the recent years to papers using formal verification tools.

5.2 Formal verification

The first use of automated tools for studying the graph exploration problem in the `Look-Compute-Move` paradigm concerns the exclusive perpetual exploration of small rings [7]. As already mentioned in Section 4.1, a generic method is

proposed, and implemented, to list all possible protocols of a given number k of robots in a given graph, in the semi-synchronous $\mathcal{S}$SYNC model. The authors use this method to prove that exclusive perpetual exploration is impossible for $k = 5$ robots in a ring of $n = 10$ robots, and to list all algorithms solving the weaker version of the problem for the same setting ($k = 5$ and $n = 10$).

A more general formal verification tool was introduced a few years later by Bérard et al [9]. This one is compatible with all three models of (a)synchrony, and can handle both variants of the exploration problem. First, the terminating exploration algorithm for rings from [20] has been studied. The first and most difficult phase of the algorithm, the set-up phase, is proved correct for small values of k (at most 21), and small values of n (at most 22), and even for some settings not covered by the hand-written proof in [20]. Second, the exclusive perpetual exploration algorithm for rings from [5] has been studied. This time, a counter-example is found, for the case of $k = 3$ robots in a ring of size $n = 10$. This counter-example is a particular execution starting from a symmetric configuration and using the asynchronous model $\mathcal{A}$SYNC to maintain pending moves (i.e. moves that are already computed but not yet executed) even when the symmetry has already been broken. Basically, the worm that we described in Section 4.2 has been stretched in such a way that the two robots forming its head do not agree on which direction the worm is moving. This eventually leads to a collision between these two robots. Note that the existence of an ambiguity on the moving direction of the worm heavily relies on the small size of the ring. Besides, the same paper presents a corrected version of the algorithm, which is formally proved correct for n up to 16 and manually proved correct for larger values of n.

Finally, Doan et al. [16] also study the exclusive perpetual exploration algorithm for 3 robots in rings from [5], and the same counter-example is found. However, the model checker is different as well as the modeling details, allowing for potentially different performance of the verification.

6 Conclusion and perspectives

The two variants of the graph exploration problem have been well studied in rings, and in some other topologies. Not surprisingly, the amount of asynchrony and of symmetries influences the amount of robots needed to solve the problem, especially for the terminating exploration problem. Probabilistic approaches may also help, but further studies are needed on this aspect. Also, it would be interesting to extend the investigation to larger families of graphs, typically to planar graphs.

An interesting research axis concerns the limited visibility. Indeed, having a limited vision sounds much more realistic, and the first results on this subject seem to indicate that interesting performance can still be achieved despite this limitation. Another direction of research concerning vision could be to consider non-egocentric views, but with sense of direction. This corresponds to the sit-

uations in which a camera can see the whole theater of operation, for example when it is attached to the ceiling in a room or embedded in a satellite.

Finally, some effort is required to further formalize and verify the results in the domain. Recent papers showed that hand-written proofs could be flawed, because of the many cases to be considered due to asynchrony and symmetries. Formal verification through model checking seems to also bear some limitations, due to the combinatorial explosion of the problems. An interesting but challenging alternative would consist in using proof assistants like Coq in order to formally prove the results for arbitrary values of the parameters, and not just for small values like in model checking so far. Simple impossibility results in the flavor of Lemma 1 have recently been certified using the Pactole Coq framework [2].

References

1. Balabonski, T., Courtieu, P., Pelle, R., Rieg, L., Tixeuil, S., Urbain, X.: Continuous vs. Discrete Asynchronous Moves: a Certified Approach for Mobile Robots. Research Report, Sorbonne Université, CNRS, Laboratoire d'Informatique de Paris 6, LIP6, F-75005 Paris, France (2018), https://hal.sorbonne-universite.fr/hal-01762962, see also SSS'18 proceedings.
2. Balabonski, T., Pelle, R., Rieg, L., Tixeuil, S.: A Foundational Framework for Certified Impossibility Results with Mobile Robots on Graphs. In: Proceedings of the 19th International Conference on Distributed Computing and Networking. pp. 5:1–5:10. ICDCN '18, ACM, New York, NY, USA (2018). https://doi.org/10.1145/3154273.3154321, http://doi.acm.org/10.1145/3154273.3154321
3. Baldoni, R., Bonnet, F., Milani, A., Raynal, M.: Anonymous graph exploration without collision by mobile robots. Information Processing Letters **109**(2), 98–103 (Dec 2008). https://doi.org/10.1016/j.ipl.2008.08.011
4. Baldoni, R., Bonnet, F., Milani, A., Raynal, M.: On the Solvability of Anonymous Partial Grids Exploration by Mobile Robots. In: Principles of Distributed Systems. pp. 428–445. Springer, Berlin, Heidelberg (Dec 2008). https://doi.org/10.1007/978-3-540-92221-6_27
5. Blin, L., Milani, A., Potop-Butucaru, M., Tixeuil, S.: Exclusive Perpetual Ring Exploration without Chirality. In: Distributed Computing. pp. 312–327. Springer, Berlin, Heidelberg (Sep 2010). https://doi.org/10.1007/978-3-642-15763-9_29
6. Bonnet, F., Defago, X.: Exploration and Surveillance in Multi-robots Networks. In: 2011 Second International Conference on Networking and Computing. pp. 342–344 (Nov 2011). https://doi.org/10.1109/ICNC.2011.66
7. Bonnet, F., Défago, X., Petit, F., Potop-Butucaru, M., Tixeuil, S.: Discovering and Assessing Fine-Grained Metrics in Robot Networks Protocols. In: 2014 IEEE 33rd International Symposium on Reliable Distributed Systems Workshops. pp. 50–59 (Oct 2014). https://doi.org/10.1109/SRDSW.2014.34
8. Bonnet, F., Milani, A., Potop-Butucaru, M., Tixeuil, S.: Asynchronous Exclusive Perpetual Grid Exploration without Sense of Direction. In: Principles of Distributed Systems. pp. 251–265. Springer, Berlin, Heidelberg (Dec 2011). https://doi.org/10.1007/978-3-642-25873-2_18

9. Bérard, B., Lafourcade, P., Millet, L., Potop-Butucaru, M., Thierry-Mieg, Y., Tixeuil, S.: Formal verification of mobile robot protocols. Distributed Computing **29**(6), 459–487 (Nov 2016). https://doi.org/10.1007/s00446-016-0271-1
10. Chalopin, J., Flocchini, P., Mans, B., Santoro, N.: Network Exploration by Silent and Oblivious Robots. In: Graph Theoretic Concepts in Computer Science. pp. 208–219. Springer, Berlin, Heidelberg (Jun 2010). https://doi.org/10.1007/978-3-642-16926-7_20
11. Datta, A.K., Lamani, A., Larmore, L.L., Petit, F.: Ring Exploration by Oblivious Agents with Local Vision. In: 2013 IEEE 33rd International Conference on Distributed Computing Systems. pp. 347–356 (Jul 2013). https://doi.org/10.1109/ICDCS.2013.55
12. Datta, A.K., Lamani, A., Larmore, L.L., Petit, F.: Enabling Ring Exploration with Myopic Oblivious Robots. In: 2015 IEEE International Parallel and Distributed Processing Symposium Workshop. pp. 490–499 (May 2015). https://doi.org/10.1109/IPDPSW.2015.137
13. Devismes, S., Lamani, A., Petit, F., Raymond, P., Tixeuil, S.: Optimal Grid Exploration by Asynchronous Oblivious Robots. In: Stabilization, Safety, and Security of Distributed Systems. pp. 64–76. Springer, Berlin, Heidelberg (Oct 2012). https://doi.org/10.1007/978-3-642-33536-5_7
14. Devismes, S., Lamani, A., Petit, F., Tixeuil, S.: Optimal Torus Exploration by Oblivious Robots. In: Networked Systems. pp. 183–199. Springer, Cham (May 2015). https://doi.org/10.1007/978-3-319-26850-7_13
15. Devismes, S., Petit, F., Tixeuil, S.: Optimal probabilistic ring exploration by semi-synchronous oblivious robots. Theoretical Computer Science **498**, 10–27 (Aug 2013). https://doi.org/10.1016/j.tcs.2013.05.031
16. Doan, H.T.T., Bonnet, F., Ogata, K.: Model Checking of a Mobile Robots Perpetual Exploration Algorithm. In: Structured Object-Oriented Formal Language and Method. pp. 201–219. Springer, Cham (Nov 2016). https://doi.org/10.1007/978-3-319-57708-1_12
17. D'Angelo, G., Di Stefano, G., Navarra, A., Nisse, N., Suchan, K.: Computing on Rings by Oblivious Robots: A Unified Approach for Different Tasks. Algorithmica **72**(4), 1055–1096 (Aug 2015). https://doi.org/10.1007/s00453-014-9892-6
18. Flocchini, P., Ilcinkas, D., Pelc, A., Santoro, N.: Remembering without memory: Tree exploration by asynchronous oblivious robots. Theoretical Computer Science **411**(14), 1583–1598 (Mar 2010). https://doi.org/10.1016/j.tcs.2010.01.007
19. Flocchini, P., Ilcinkas, D., Pelc, A., Santoro, N.: How many oblivious robots can explore a line. Information Processing Letters **111**(20), 1027–1031 (Oct 2011). https://doi.org/10.1016/j.ipl.2011.07.018
20. Flocchini, P., Ilcinkas, D., Pelc, A., Santoro, N.: Computing Without Communicating: Ring Exploration by Asynchronous Oblivious Robots. Algorithmica **65**(3), 562–583 (Mar 2013). https://doi.org/10.1007/s00453-011-9611-5
21. Lamani, A., Potop-Butucaru, M., Tixeuil, S.: Optimal deterministic ring exploration with oblivious asynchronous robots. arXiv:0910.0832 [cs] **6058**, 183–196 (2010), arXiv: 0910.0832
22. Lamani, A., Potop-Butucaru, M.G., Tixeuil, S.: Optimal Deterministic Ring Exploration with Oblivious Asynchronous Robots. In: Structural Information and Communication Complexity. pp. 183–196. Springer, Berlin, Heidelberg (Jun 2010). https://doi.org/10.1007/978-3-642-13284-1_15
23. Ooshita, F., Tixeuil, S.: Ring Exploration with Myopic Luminous Robots. arXiv:1805.03965 [cs] (May 2018), http://arxiv.org/abs/1805.03965, see also SSS'18 proceedings.

24. Potop-Butucaru, M., Raynal, M., Tixeuil, S.: Distributed Computing with Mobile Robots: An Introductory Survey. In: 2011 14th International Conference on Network-Based Information Systems. pp. 318–324 (Sep 2011). https://doi.org/10.1109/NBiS.2011.55