Optimal Asynchronous Perpetual Grid Exploration*

Quentin Bramas[1], Stéphane Devismes[2], Anaïs Durand[3], Pascal Lafourcade[3], and Anissa Lamani[1]

[1] University of Strasbourg, ICUBE, CNRS
[2] Université de Picardie Jules Verne, MIS UR 4290, Amiens
[3] Université Clermont Auvergne, Clermont Auvergne INP, CNRS, LIMOS

Abstract. We address the perpetual grid exploration by a swarm of autonomous, asynchronous, myopic, and luminous robots. We first show that it is impossible for the robots to explore the grid regardless of their number and the number of colors they can take if their visibility range is one. We also show that PGE is impossible with three oblivious robots that have a visibility range of two hops. We then present three optimal algorithms solving the problem. The first algorithm uses four oblivious robots with a visibility range of two, but assumes they agree on a common chirality. For the two other algorithms, no common chirality is assumed. The former uses three robots that have a visibility range of two and a two-color light. The latter uses three oblivious robots under visibility range three.

Keywords: Asynchronous Myopic Robots · Grids · Perpetual Exploration

1 Introduction

Swarm robotics has drawn a lot of attention in the past decade. Inspired by natural systems, a lot of investigations focus on how to reproduce with artificial systems autonomous behaviors observed in nature. Given a collection of autonomous mobile entities called robots, the main goal is to determine the minimum hypotheses allowing the robots to solve a given task.

In this paper, we consider mobile robots that are autonomous (*i.e.*, there is no central authority to coordinate their move) and luminous (*i.e.*, they are endowed with lights that can take a finite number of different colors). Moreover, they are deaf-mute (*i.e.*, they have no direct means of communication) but they are endowed by visibility sensors allowing them to perceive their environment within a given distance called *visibility range*. Our robots are said to be myopic since can only sense at constant (typically small) distance. Robots operate in the well-known Look-Compute-Move (LCM) model. That is, they operate in cycles which comprise three phases: Look, Compute, and Move. During the first phase (Look), robots take a snapshot of their environment using their visibility sensors. In the second phase (Compute) and based on the previous snapshot, they decide a destination in their surrounding and update their color. Finally, in the last phase (Move), they move to the computed destination, if the destination is different from their current location. We consider the (fully) asynchronous model (ASYNC), meaning that the time between each Look, Compute, and Move phases is finite yet arbitrary long.

* This study has been partially supported by the French ANR project SkyData (ANR-22-CE25-0008).

We study the case in which the robots have to solve the perpetual exploration problem. In this problem, robots evolve in a discrete universe and have to ensure that each location (node) is visited by at least one robot infinitely often. We consider here the universe is a finite grid, hence the problem will be referred to as the perpetual grid exploration (PGE). We focus on optimal *exclusive* solutions with respect to both the visibility range and the number of robots. Exclusiveness adds a constraint on robots behavior: they can neither occupy the same node simultaneously nor traverse the same edge at the same time.

Related Work. The exploration problem is one of the benchmark tasks when it comes to robots evolving on graphs. Various topologies have been studied: lines [13], rings [1, 7, 11, 14, 17], tori [10], grids [2, 4, 5, 9], cuboids [3], and trees [12]. Two variants of the problem have been investigated: (i) the perpetual exploration problem [1–3, 6, 18], considered in this paper, which requires the robots to visit each node of the graph infinitely often and (ii) the terminating exploration problem [7, 9–14, 16] which requires the robots to visit each node of the graph at least once and then stop moving.

Most of the investigations consider robots with unlimited visibility range allowing them to observe every node of the system [1, 2, 9–14]. Robots are in this case oblivious (*i.e.*, they cannot remember past actions) and have to solve the terminating exploration problem. Myopic robots have also been considered in both variants of the problem [4, 6–8, 17]. When it comes to the perpetual exploration problem, an additional assumption has an impact on the feasibility of the task and the optimality of the proposed solutions. This assumption endows the robots with a common chirality. In fact, chirality is usually assumed when robots evolve in the continuous 2D Euclidean plan but some investigations have also recently considered it in the discrete universe [5, 6].

On finite and infinite grids, the exploration problem by myopic robots has been investigated almost exclusively assuming the robots are synchronous *i.e.*, they all wake up at the same time look at their surroundings and perform their actions simultaneously. On finite grids, it has been shown that two (resp. three) synchronous robots with three colors (resp. one color) are sufficient to solve the problem when robots have visibility one and share a common chirality [6]. The case in which robots have no common chirality was investigated in [18]. It was proven that the problem is not solvable with only two robots having a finite number of colors and a finite visibility range. An optimal solution is also presented using only three robots having visibility range one, using only three colors. The case in which robots are oblivious and visibility range 2 was solved using five robots. In the case of infinite grids, assuming robots with visibility range one and few colors (O(1)), five (resp. six) synchronous robots are necessary and sufficient to solve the problem with (resp. without) the common chirality assumption [4, 5]. Finally, in the case of cuboids, it has been shown in [3] that three synchronous robots with a common chirality endowed with five colors are necessary and sufficient to solve the perpetual exploration problem. Asynchronous myopic robots have been considered in the context of the terminating exploration problem on finite grids [16] assuming robots can start from distinct positions but allowing them to occupy the same location simultaneously during the execution (*i.e.*, they are not exclusive). Now, in such a setting, storing at a given instant several robots at the same location can be used to remember some past actions and so may help solve the problem using fewer colors, a smaller number of robots, and/or a smaller visibility range. In the same setting as ours (perpetual exploration with

Chirality	Visibility	# Robots	# Colors	Algorithm
Yes	1	finite	finite	Impossible (Theorem 2)
Yes	2	3	1	Impossible (Theorem 3)
Yes	2	3	2	[6]
Yes	2	4	1	A_1
No	2	3	2	A_2
No	3	3	1	A_3

Table 1: Summary of our results.

exclusive, myopic robots), the case of asynchronous robots has only been considered in [6] by showing that a modified version of a synchronous algorithm also works in the asynchronous setting. This algorithm is mentioned in our summary table for comparison.

Asynchronous robots with colors and limited visibility range have been considered to solve the maximum independent set problem on grids [15]. This model is slightly different from ours as robots can cross the same link and occupy the same location at the same time. They also enter the grid initially one by one from the same location.

Contribution. We first present two impossibility results: the first one shows that perpetually exploring any finite grid is not solvable with asynchronous robots having a visibility range of one regardless of the number of robots and the number of colors for their lights. Next, we show that the problem is also unsolvable with three asynchronous robots without colored lights (*i.e.*, with oblivious robots) and a visibility range of two.

We then present three optimal algorithms solving the problem. The first algorithm assumes a common chirality and requires four oblivious robots having visibility range two. The second algorithm uses three robots without common chirality but endowed with a light of two colors and having visibility range two. The third algorithm uses three oblivious robots without common chirality and having visibility range three. Table 1 below summarizes our contribution.

2 Model

We consider a set of $n > 0$ robots located on a *finite grid* made of $\mathcal{L} \geq n$ lines and $\mathcal{C} \geq n$ columns, *i.e.*, robots evolve in an undirected graph $G(V, E)$ where $V = \{(i,j) : i \in [0, \mathcal{C}-1], j \in [0, \mathcal{L}-1]\}$ and $E = \{\{(i,j),(k,l)\} : (i,j) \in V \wedge (k,l) \in V \wedge |i-k| + |j-l| = 1\}$. The size of the grid is then $\mathcal{L} \times \mathcal{C}$. Grid coordinates are used for the analysis only, *i.e.*, robots cannot access them.

Robots can move from a node to one of its neighbors in the grid but, in our algorithms, we prevent any two robots from being located at the same node simultaneously. In other words, any algorithm allowing such a behavior is considered as not well-defined. A node is *occupied* when a robot is located at this node, otherwise it is *empty*. Robots are *luminous*, *i.e.*, they have lights of different colors that can be seen by robots in their surrounding. We denote by Cl the set of all possible colors. The *state* of a node is then the color of the light of the robot located at this node if it is occupied, $\perp$ otherwise.

The robots operates by executing infinitely many *asynchronous Look-Compute-Move* cycles, *i.e.*, each robot performs its own cycles in sequence, however the time between each Look, Compute, and Move phases is finite yet unbounded and decided by an adversary. The only constraint is that both Move and Look are instantaneous (each compute phase may be arbitrarily long, yet finite). In the *Look* phase, a robot r gets a snapshot of the subgraph induced by the nodes within distance $\Phi \in \mathbb{N}^*$ from its position. Φ is called the *visibility range* of the robots. In the snapshot, nodes are labeled with their state. The snapshot is not oriented in any way as the robots do not agree on a common North. However, it is implicitly ego-centered since the robot that performs a Look phase is located at the center of the subgraph in the obtained snapshot. Notice that, since moves are instantaneous, robots are always located at nodes in a snapshot. Then, during the *Compute* phase, the robot selects a destination (either Up, Left, Down, Right, or Idle) and can change its color based only on the snapshot it received. Finally, it *moves* towards its computed destination. We say that a move is *pending* when a robot has decided to move (during its Compute phase) but has not moved yet. Note that light colors are both the only permanent memory of a robot and an indirect communication mean. The particular case where $|Cl| = 1$ corresponds to the *oblivious* assumption.

In all our algorithms, we also prevent any two robots from traversing the same edge simultaneously. Since we already forbid them to occupy the same position simultaneously, this means that we should additionally prevent robots from swapping their position. Algorithms verifying this property are said to be *exclusive*. However, to be as general as possible, we do not make this additional assumption in our impossibility results.

Configurations. A *configuration* C in a grid $G(V, E)$ is a set of pairs (p, c), where $p \in V$ is an occupied node and $c \in Cl$ is the color of the robot located at p. A node p is empty if and only if $\forall c,\ (p, c) \notin C$. We sometimes just write the set of occupied nodes when the colors are clear from the context.

Views. We denote by G_r the *globally oriented view* centered at the robot r, *i.e.*, the subset of the configuration containing the states of the nodes at distance at most Φ from r, translated so that the coordinates of r is $(0, 0)$. We use this globally oriented view in our analysis to describe the movements of the robots (see, for example, Fig. 1): when we say "the robot moves Up", it is according to the globally oriented view. However, since robots do not agree on a common North, they have no access to the globally oriented view. When a robot looks at its surroundings, it instead obtains a snapshot. To model this, we assume that the *local view* acquired by a robot r in the Look phase is the result of an arbitrary *indistinguishable transformation* on G_r. The set $\mathcal{IT}$ of indistinguishable transformations depends on the assumption we make on the chirality. $\mathcal{IT}$ always contains the rotations of angle 0 (to have the identity), $\pi/2$, π and $3\pi/2$, centered at r. When we do not assume robots agree on a common chirality, we add the mirroring (robots cannot distinguish between clockwise and counterclockwise orientations) and any combination of aforementioned rotations and mirroring. Finally, we assume *self-inconsistent* robots, meaning that different transformations may be applied at different rounds.

It is important to note that when a robot r computes a destination d, it is relative to its local view $f(G_r)$, which is the globally oriented view transformed by some $f \in \mathcal{IT}$. So, the actual movement of the robot in the *globally oriented view* is $f^{-1}(d)$. For example, if $d = \mathit{Up}$ but the robot sees the grid upside-down (f is the π-rotation), then the robot moves $\mathit{Down} = f^{-1}(\mathit{Up})$. In a configuration C, $V_C(i, j)$ denotes the globally oriented view of a robot located at (i, j).

Algorithm. An algorithm A is a tuple (Cl, Init, T) where Cl is the set of possible colors, Init is a mapping from any considered grid to a non-empty set of initial configurations in that grid, and T is the transition function $Views \to \{Idle, Up, Left, Down, Right\} \times Cl$, where $Views$ is the set of possible local views.

Scheduler and Execution. During an execution, the robots perform their Look-Compute-Move cycles independently and asynchronously. However, without the loss of generality, we assume that each phase of a cycle is atomic. Indeed, recall that Look and Move are already assumed to be instantaneous, moreover, we can assume each compute phase is atomic since it is only based on the previous snapshot. An adversarial scheduler selects when a robot performs the phases of its cycle. Again, without loss of generality, we assume that at most one robot performs a phase of its cycle at each time instant. Indeed, if two robots perform a phase of their cycle at the same time, we can assume that one of them performs its phase strictly before the other as the resulting configuration is the same if the order is carefully chosen.

Hence, we can consider that the time is discretized in time instants, w.l.o.g. with non-negative integers $0, 1, 2, \ldots$, that represents the instants at which one robot performs one phase of its cycle. It is easy to see that the exact values of the times are not important in our context, only the order of the events is. Hence, we define a *schedule* $S = (S_i)_{i \geq 0}$ as a sequence actions describing the order in which the robots perform their phases: $S_i = (r_i, Look|Compute|Move)$ means that at time i, the robot r_i performs the phase $Look$, $Compute$, or $Move$. A schedule is *well-defined* if, by considering the sequence of actions associated with a single robot, the order of the actions is periodic and alternates between Look, Compute, and Move starting from a Look. A schedule is *fair* if every robot is selected infinitely often. In the remainder of the paper, we assume that all the schedules are fair and well-defined.

An execution of A in the grid G is then an infinite sequence of configurations $(C_i)_{i \in \mathbb{N}}$ determined by its initial configuration C_0 and a schedule $S = (S_i)_{i \geq 0}$. Precisely, $C_0 \in Init(G)$ and $\forall i \in \mathbb{N}$, C_{i+1} is obtained by applying $S_i = (r_i, a_i)$ on C_i as follows:

- If $a_i =$ Look, then $C_{i+1} = C_i$.

Otherwise, let j the maximum integer satisfying $j < i$ and $S_j = (r_i, Look)$; let $f_j \in \mathcal{IT}$ be the transformation applied on the view of r_i in C_j; and let $p = (x, y)$ and c respectively be the node occupied by and the color of r_i in C_i.

- If $a_i =$ Compute, then $C_{i+1} = C_i \setminus \{(p, c)\} \cup \{(p, c')\}$, where $T(f_j(V_{C_j}(p))) = (\_, c')$, *i.e.*, c' is the new color of r_i (maybe $c = c'$) computed by r_i from the view it obtained in C_j.
- If $a_i =$ Move, then $C_{i+1} = C_i \setminus \{(p, c)\} \cup \{(p', c)\}$, where
  - $p' = p$ if $f_j^{-1}(T(f_j(V_{C_j}(p)))) = (Idle, \_)$;
  - $p' = (x+1, y)$ if $f_j^{-1}(T(f_j(V_{C_j}(p)))) = (Right, \_)$;
  - $p' = (x-1, y)$ if $f_j^{-1}(T(f_j(V_{C_j}(p)))) = (Left, \_)$;
  - $p' = (x, y+1)$ if $f_j^{-1}(T(f_j(V_{C_j}(p)))) = (Up, \_)$; and
  - $p' = (x, y-1)$ if $f_j^{-1}(T(f_j(V_{C_j}(p)))) = (Down, \_)$.

Given $C_0 \in Init(G)$ and a scheduler S, we obtain an execution $e = (C_i)_{i \in \mathbb{N}}$ of A. For $0 \leq i \leq j$, we write $C_i \overset{e}{\mapsto} C_j$ the fact that C_j is reached from C_i in e.

Let $r_0, \ldots, r_{n-1}$ be n robots and $Sync^{3n} = (Sync_i)_{i \in [0, 3n-1]}$ such that for all $i \in [0, n-1]$, $Sync_i = (r_i, Look)$, $Sync_{i+n} = (r_i, Compute)$, and $Sync_{i+2n} =$

$(r_i, Move)$. The synchronous scheduler is then $Sync = (Sync^{3n})^\omega$. We denote by $C \xmapsto{Sync^{3n}} C'$ the fact that C' is reached from C under the synchronous scheduler.

Perpetual Finite Grid Exploration. An execution $(C_i)_{i\in\mathbb{N}}$ of $\mathsf{A} = (Cl, Init, T)$ in a grid $G = (V, E)$ *achieves the Perpetual (Finite) Grid Exploration* (PGE) if for every node $u \in V$ and for every time t, there exists a time $t' \geq t$ such that u is occupied in $C_{t'}$. An algorithm A that uses n robots *solves* (resp., *synchronously solves*) the *Perpetual Finite Grid Exploration* (PGE) problem if for every finite grid $G = (V, E)$ with at least n lines and n columns and every initial configuration $C_0 \in Init(G)$, we have every execution (resp. every execution under the synchronous scheduler) of A in G starting from C_0 that achieves the PGE.

Well-defined Algorithms. Recall that robots are assumed to be self-inconsistent. In this context, we say that an algorithm $(Cl, Init, T)$ is *well-defined* if the global destination computed by a robot does not depend on the applied indistinguishable transformation f, *i.e.*, for every globally oriented view V, and every transformation $f \in \mathcal{IT}$, we have $T(V) = f^{-1}(T(f(V)))$. All our algorithms are well-defined. However, to be as general as possible, we do not make this assumption in our impossibility results.

An Algorithm as a Set of Rules. We write an algorithm as a set of rules, where *a rule* is a triplet $(V, d, c) \in Views \times \{Idle, Up, Left, Down, Right\} \times Cl$. We say that an algorithm $(Cl, Init, T)$ includes the rule (V, d, c), if $T(V) = (d, c)$. By extension, the same rule applies to indistinguishable views, *i.e.*, $\forall f \in \mathcal{IT}, T(f(V)) = (f(d), c)$. Consequently, we forbid an algorithm to contain two rules (V, d, c) and (V', d', c') such that $V' = f(V)$ for some $f \in \mathcal{IT}$. Hence, an algorithm corresponds to a set of rules if each destination is the result of applying one of its rules.

As an illustrative example, consider the rule R_1 given in Fig. 1. This rule is defined for robots having a visibility range of two. This rule means that, when a blue robot B sees three robots with color R, one on top at distance 2, one on the left, and one in diagonal, then the blue robot is dictated to move Up. By extension, the same rule applies if the view is rotated by π, but in that case, the destination would be Down.

In the same figure, R_2 is a rule where the three black nodes represent a part of the outer boundary of the grid, that we call *a wall* in the remaining of the paper. In our algorithms, we often define similar rule that apply regardless of the presence of a wall in some part of the view. Thus, to avoid defining several time rules with very similar views, we propose a notation to express several rules in just one picture. For example, R_3 in Fig. 1 has three nodes hatched with vertical lines, which means that the rule applies regardless of the presence of a wall located at those nodes. In practice, every rule that contains such vertical (resp. horizontal) hatched lines, represents a set of rules obtained by replacing each of those lines either by walls or by empty nodes. For example, R_3 in Fig. 1 is a concise representation of R_1 and R_2.

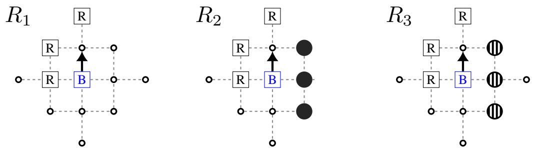

Fig. 1: Examples of rules.

Algorithms having locally-defined initial configurations. In a given grid, the set of possible initial configurations of an algorithm can be reduced to a singleton. In such a case, the scalability and flexibility of the algorithm is weak. To be more general, two of the algorithms we propose have *locally-defined* sets of initial configurations. Configurations in a locally-defined set of initial configurations are defined by colors and relative positions of the robots only. Hence, for a given grid, every two possible initial configurations are equal up to possible transformations applied on all robots positions. The set of possible transformations includes any combinations of translations and rotations of angle $\frac{\pi}{2}$ applied to all robot positions. Moreover, when no common chirality is assumed, the previous combinations can also be augmented with mirroring. The set of all possible initial configurations is then closed by the set of possible transformations.

Synchronous Sequential Algorithms. Let A be well-defined algorithm. Informally, A is sequential if robots move or change their color one at a time. More formally, we start by defining the notion of *seq-enabled* robots. In a configuration C, a robot r at a position p is said to be *move-enabled*, resp. *col-enabled* if we have $T(V_C(p)) = (move, \_)$, resp. $T(V_C(p)) = (\_, c)$, where $move \neq Idle$, resp. c is not the color of r in C. A robot that is move-enabled or col-enabled is said to be *enabled*. Finally, it is said to be *seq-enabled* if it is either move-enabled or col-enabled, but not both. An synchronous execution $(C_i)_{i \in \mathbb{N}}$ under is said to be *sequential* if $\forall j \in \mathbb{N}$, no robot except one is enabled in C_{3nj}, and this latter being actually seq-enabled in C_{3nj}. An algorithm is said to be *synchronous-sequential* if it is well-defined and all its synchronous executions are sequential.

We show the correctness of our algorithm by induction on the size of the grid in which they are deployed. We use Theorem 1 below to establish the base cases of these inductions using a simulator.

Theorem 1. *Let* A *be a synchronous-sequential algorithm using* n *robots and* G *be a grid. Every synchronous execution of* A *in* G *achieves the PGE if and only if every (asynchronous) execution of* A *in* G *achieves the PGE.*

Let $(C_i)_{i \in \mathbb{N}}$ a synchronous execution of A on some grid G. Our simulator actually computes the sequence of configurations $(C_{3ni})_{i \in \mathbb{N}}$. To apply Theorem 1, we then need to check if:

- A *is well-defined.* To that goal, we test every indistinguishable transformation (a finite number of transformations) on each of its rules (the number of rules of A is also finite).
- A *is synchronous-sequential and its synchronous execution achieves the PGE.* Since A is well-defined, all its synchronous executions are fully determined by their initial configuration. So, we should make one simulation per possible initial configuration (the set of all possible initial configurations can be computed by applying every authorized transformation on the local initial pattern). Each simulation stops as soon as a configuration appears again. This necessarily happens since the number of configurations is finite. Then, we have to check all nodes have been visited in the execution prefix (to show the correctness) and in all encountered configurations exactly one robot is enabled and this robot is actually seq-enabled (to show that A is synchronous-sequential).

3 Impossibility Results

In this section, we first establish in Theorem 2 a general impossibility result for the PGE problem with robots having visibility one. To do so, we show that a team of robots cannot cross a fence (defined below). We then give a specific impossibility result for the PGE problem with three oblivious robots having visibility range two. To show Theorem 2, we prove a variant of the test of the fence (introduced in [4]) for the case of finite grids. In our setting, a fence is the boundary of width two of a square.

More formally, Let $Square((x_1, y_1), (x_2, y_2))$ be the nodes in the square delimited by the given points, with $x_1 < x_2$ and $y_1 < y_2$: $Square((x_1, y_1), (x_2, y_2)) = \{(x, y) \in \mathbb{Z}^2 \mid x_1 \leq x \leq x_2 \text{ and } y_1 \leq y \leq y_2\}$. Then, a fence $F_{(x_1,y_1),(x_2,y_2)}$, with $x_1 + 4 < x_2$ and $y_1 + 4 < y_2$ is defined as: $F_{(x_1,y_1),(x_2,y_2)} = Square((x_1, y_1), (x_2, y_2)) \setminus Square((x_1 + 2, y_1 + 2), (x_2 - 2, y_2 - 2))$. We say a robot is outside (resp. inside) the fence if it is outside the outer square (resp. inside the inner square). We say that a set of robots has crossed a fence when they are all inside the fence at a given time. Notice that this does not mean that the robots always stay inside of the fence afterward. Formally, we say that a set of robots S *has crossed the fence* $F_{(x_1,y_1),(x_2,y_2)}$ *at Round* t if there exists $t' \leq t$ such that every robot $r \in S$ is at some coordinates $(x, y) \in Square((x_1 + 2, y_1 + 2), (x_2 - 2, y_2 - 2))$ at Round t'.

We say a set of robots S *single-handed crosses the fence* F *between* t *and* t' if for every robot $r \in S$, (1) r is located outside F at Round t; (2) r is located at inside F at Round t'; and (3) only robots of S are within distance one of r between Round t and Round t'. We say that a set of robots S *has single-handed crossed* the fence F at Round t if $\exists t' < t'' \leq t$ such that S single-handed crosses the fence F between t' and t''.

To be more general, we now consider any algorithm, *i.e.*, well-defined or not. We first prove that if robots explore any finite grid, then there is a fence that is *single-handed crossed* by a subset of robots; see Lemma 1. This latter result will be used to show that, if robots have visibility one the PGE problem is impossible, whatever the number of robots is, indeed we show that the test of the fence fails; see Theorem 2.

Lemma 1 (The test of the fence in finite grids). *Let* A *be any algorithm solving the PGE using* n *robots. There is an execution of* A*, a grid* G*, a fence* F *of* G*, and a subset of robots* S *such that* S *single-handed crosses* F *within a finite number of rounds.*

Theorem 2. *There exists no algorithm solving the PGE problem with robots having visibility range one, even assuming a common chirality.*

The proof of the next theorem follows the same principles as Theorem 3.5 in [6]. We had just to adapt these principles for our specific setting. In particular, we had to show that, whatever the algorithm, the adversary can impose a single way to translate the positions of three oblivious robots by one, when they do not see any wall.

Theorem 3. *There exists no algorithm solving the PGE problem with three oblivious robots having visibility range two, even assuming a common chirality.*

4 Algorithms

4.1 Algorithm with 4 Oblivious Robots and Common Chirality

We first present an algorithm, denoted by A_1, that uses four oblivious robots, assuming a common chirality and visibility range two[4]. The algorithm we present here is for when the number of lines and columns is at least 5. The particular cases where the number of lines or columns is 4 are handled with special rules shown in the interactive simulation.

The exploration of the grid is performed alternatively rows by rows and columns by columns in a given direction. Precisely, three robots explore a line (a row or a column), while leaving the fourth robot, called *sentinel*, adjacent to a wall. When the three robots have explored the line, they perform a U-turn and move back toward the sentinel. When they meet the sentinel, the four robots perform a sequence of moves in a way that allows them to explore the next line. The lines are explored one by one in a given direction called the *exploring direction* which is initially $\downarrow$ in the illustration. After all the lines are explored, the four robots perform a sequence of moves to change the exploring direction, from $\downarrow$ to $\leftarrow$. In more detail, the four robots initially form an L-shaped pattern with a **unique** robot being adjacent to a wall (see Fig. 2) and the robot alone in its row must not see any wall. Assume without loss of generality that the sentinel is placed on Row ℓ_i and let ℓ_{i+1} and ℓ_{i+2} be the next two rows from ℓ_i in the exploring direction. The exploration of the rows ℓ_i and ℓ_{i+1} is initiated by the robot adjacent to the sentinel by moving to its adjacent node in the exploration direction $\downarrow$ by executing Rule a) of Fig. 4. While the sentinel remains idle on its node, the three other robots, being in exploration formation (Fig. 3), move in a straight line to explore Rows ℓ_i and ℓ_{i+1}. This is done by moving in a sequential manner starting from the robot that is located on ℓ_i by executing the rules defined in Fig. 4, (b – f).

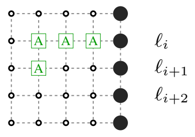

Fig. 2: Instance of an initial configuration.

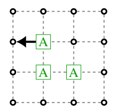

Fig. 3: Exploration formation.

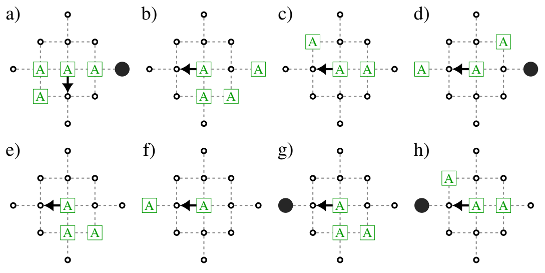

Fig. 4: Rules to move in a straight line.

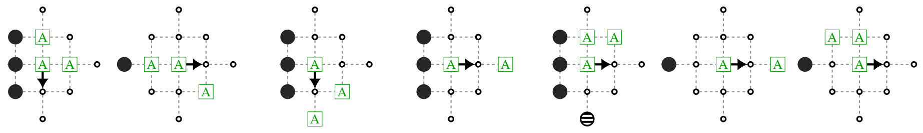

Fig. 5: Rules to perform the U-turn and initiate the exploration towards the sentinel.

[4] An interactive simulation of A_1 is given at https://robots.app.bramas.fr/?SSS2024/0

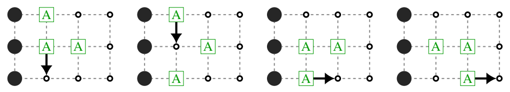
Fig. 6: Sequence of configurations during a change of row and a U-turn.

Once the three explorer robots reach the opposite wall, they not only move to the next adjacent row with respect to direction $\downarrow$ but also perform a U-turn so that they move back on Rows ℓ_{i+1} and ℓ_{i+2} towards the sentinel. The U-turn is performed by executing the rules of Fig. 5. The sequence of configurations during this process is illustrated in Fig. 6. When moving towards the sentinel, a robot eventually becomes adjacent to a wall and it can see the sentinel at distance two. Robots then move to re-create the L-shaped pattern but this time on row ℓ_{i+1}. This is done by executing the rules of Fig. 8. The sequence of configurations during this process is illustrated in Fig. 7.

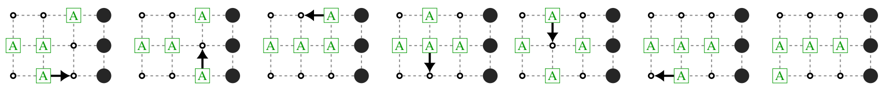
Fig. 7: Sequence of configurations during a change of row for the sentinel.

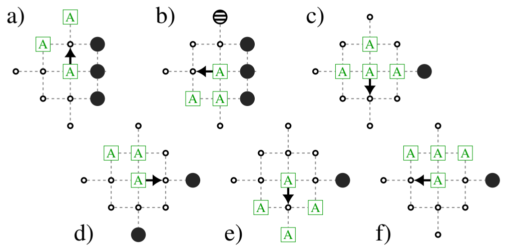

Fig. 8: Rules to move the sentinel to the next row to be explored.

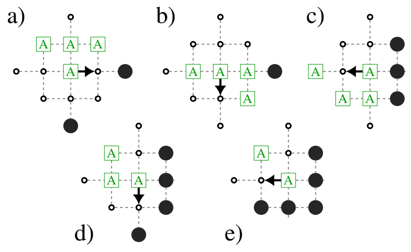

Fig. 9: Rules to switch to the columns exploration.

Let $\ell_1, \ell_2, \ldots, \ell_{\mathcal{L}}$ be the sequence of rows in direction $\downarrow$. By repeating the same process, the sentinel will eventually be located at $\ell_{\mathcal{L}-3}$ and the robots explore Rows $\ell_{\mathcal{L}-3}, \ell_{\mathcal{L}-2}$ and $\ell_{\mathcal{L}-1}$. When robots move back towards the sentinel and reach it, they move to initiate the exploration of the grid columns by columns in direction $\leftarrow$, labeled $m_1, m_2, \ldots, m_{\mathcal{C}}$. Note that the robots cannot differentiate between rows and columns, thus, they apply the same process to the columns. That is, robots will explore the columns as they did with the rows. To switch from rows to columns exploration, the robots perform a sequence of moves first by using the rules of Fig. 8 as they are changing rows except that this time, Rule (f) will not be enabled as the robot observes another wall below, at distance two. Instead, the robot executes Rule (a) of Fig. 9 and moves towards the wall on its current row to notify the other robots that they are switching to columns exploration. The robots then create the desired L-shape on m_2 by executing the rules of Fig. 9. The sequence of configurations during this process is presented in Fig. 10. Robots explore the columns as they did for the rows and then switch again to the rows exploration in direction $\uparrow$ by placing again the sentinel adjacent to a wall on rows and so on. That is, the sentinel moves along the wall of the grid in the clockwise direction.

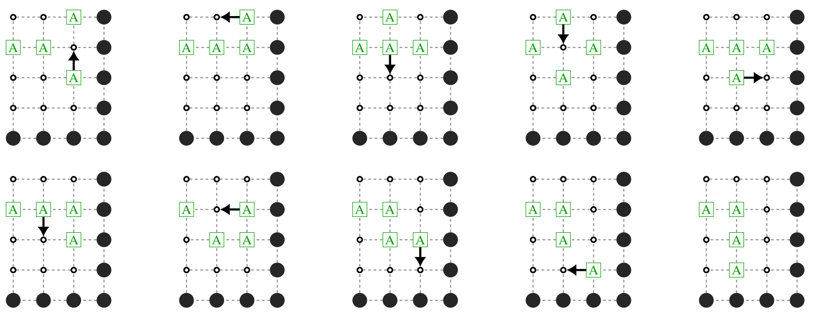

Fig. 10: Sequence of configurations during rows/columns exploration switch.

Theorem 4. *Algorithm* A_1 *solves the PGE problem using four oblivious asynchronous robots with visibility range two and a common chirality.*

Proof Outline. We first observe that A_1 is well-defined. Next, in the remaining of the proof we only consider a synchronous scheduler, first to prove that A_1 is synchronous-sequential and then to show that A_1 solves the problem under a synchronous scheduler. These two claims are shown by induction on $n \times m$, where n is the number of lines and m is the number of columns of the grid (*n.b.*, the base case is established using our simulator). Hence, by Theorem 1, this covers the case of an asynchronous scheduler as well. $\square$

4.2 Algorithm with 3 2-Color Robots without Common Chirality

We now present an algorithm, denoted by A_2, that uses three robots, each having lights of two colors, and assumes visibility range two. Note that, this time, we do not make any assumption on chirality.[5]. The exploration is similar to Algorithm A_1, except that here two robots are used to explore a line, and these robots move back on the same line. The two exploring robots are called the *leader*, with color A, and the *follower*, with color B. They explore the grid row by row, going from one wall to the other and coming back on the same row. The third robot (of color A) plays the role of a *sentinel*. It remains at one side of the row being explored and points to the next row. The rows are explored in one direction, *e.g.*, initially from top to bottom, then the robots do a quarter turn at a corner and continue their exploration column by column, *e.g.*, from left to right, and so on.

When exploring a line, the leader and the follower move in a straight line in two steps, using the rules of Fig. 11: the leader (robot A) first moves away from its follower, then robot B follows. When they reach the opposite wall, they switch their role and colors, using the rules of Fig. 12. The leader changes its color to B, then the follower gets Color A. When their roles are switched, they start moving along the row in the opposite direction.

[5] An interactive simulation of A_2 is given at https://robots.app.bramas.fr/?SSS2024/1

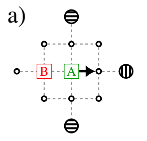

Fig. 11: Rules to move straight.

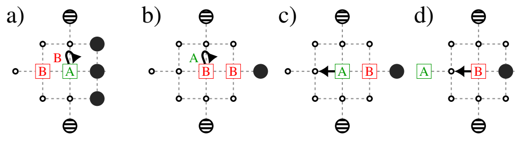

Fig. 12: Rules to turn around on the same row.

When they reach back the first wall, the robots, along with the sentinel, move to the next row in the direction pointed by the sentinel, using the rules of Fig. 13. The sequence of configurations during this process is illustrated in Fig. 14. After changing rows, the new leader and follower start to explore the new row as previously.

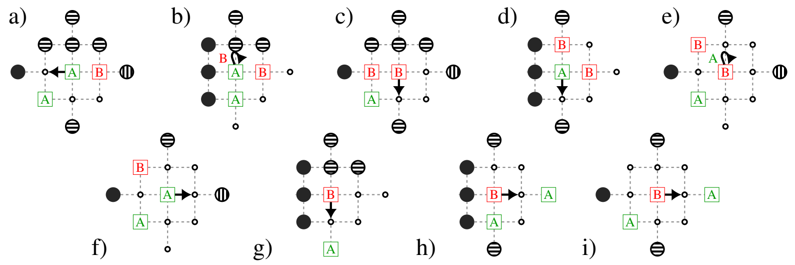

Fig. 13: Rules to perform a turn and a change of row.

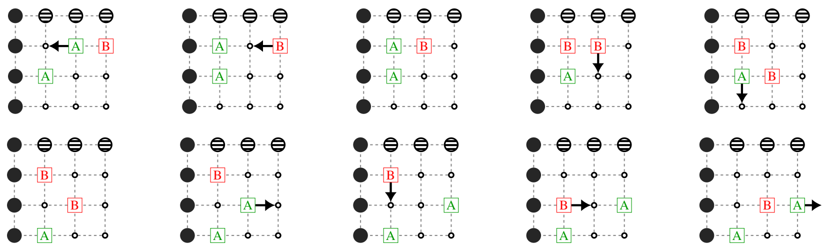

Fig. 14: Sequence of configurations during a change of row.

When the robots have explored the penultimate row, they cannot perform the same steps as before to move to the next row since the sentinel cannot move forward. Thus, the robots perform a quarter turn and change the direction of exploration: they will now explore the grid column by column, *e.g.*, from left to right if they were going from top to bottom previously. The rules for the quarter turn are shown in Fig. 15 and the sequence of configurations reached during this process is given in Fig. 18. After the turn, the new leader and follower start to explore the second column (ordered from right to left) as previously. Notice that for grids with one side of size 3 (*e.g*, 3×3 or 3×4), the algorithm requires an additional rule; see Fig. 16. Indeed, when the robots do the U-turn at the end of the row, the former follower sees the sentinel. This does not happen in larger grids. Nonetheless, the exploration follows the same principle.

Initial configurations. Initially, the three robots form an "L"-shape: one robot (the future sentinel) has color A and is adjacent to another robot with color A (the future leader),

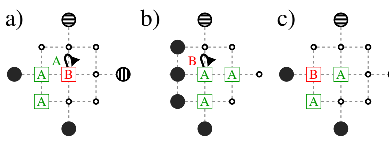

Fig. 15: Rules to perform a quarter turn and change the direction of exploration on the last row.

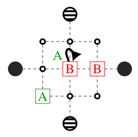

Fig. 16: Additional rule for 3×3 grids.

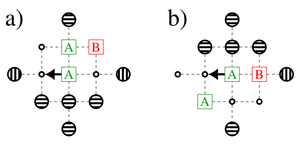

Fig. 17: Rules to reach a wall from the initial configuration.

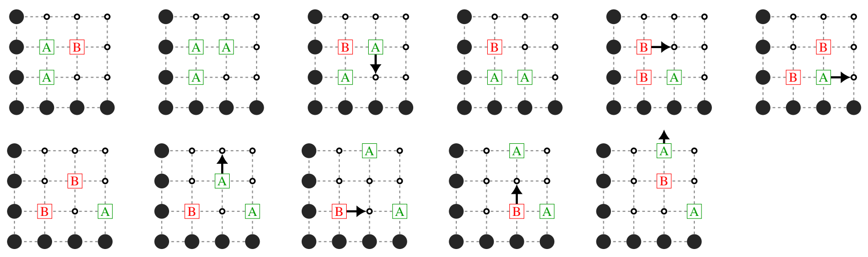

Fig. 18: Sequence of configurations during a quarter turn when robots are on the last row and change the direction of exploration.

which is adjacent to the third robot with color B (the future follower); as shown in Fig. 19. The three robots can be anywhere in the grid as long as they respect these relative positions. Hence, the set of initial configurations is locally-defined.

The robots will move in the direction of the wall pointed by the two robots of color A, using the rules in Fig. 17. The three following steps are repeated until reaching the wall: the sentinel moves on its row, away from the two other robots. Then, the leader moves in the same direction and the follower takes its place. When they reach the wall, they are in the same position as when they move to the next row in the exploration part. Thus, the exploration begins.

Theorem 5. *Algorithm* A_2 *solves the PGE problem using three asynchronous robots equipped with two colors and visibility range two.*

4.3 Algorithm with 3 oblivious Robots under Visibility Range 3 without Common Chirality

Finally, we present an algorithm, denoted by A_3, that uses three oblivious robots and assumes a visibility range three. Again, we make no assumption on the chirality.[6]. Contrary to the previous algorithms, the robots only explore the grid row by row, without switching to the column exploration. Moreover, no "sentinel" remains adjacent to a wall. The three robots explore the grid in a "snake" shape: initially the *leader* is alone on its row and column; the two *followers* are on the adjacent row, one of them being on a diagonal to the leader; as shown in Fig. 21. The three robots can be initially anywhere in the grid as long as they respect these relative positions. Hence the set of initial configurations is locally-defined. The rows are explored in one direction, *e.g.*, from top to bottom, then the robots switch directions, *e.g.*, from bottom to top.

[6] An interactive simulation of A_3 is given at https://robots.app.bramas.fr/?SSS2024/2

When exploring a row, the leader and its followers move straight from one wall to the other using the rules of Fig. 20. The leader moves away from its followers. Then, the followers follow, the closest one first, the other afterward. Upon reaching the opposite wall, they perform a turn to start exploring the next row, on the opposite side from the leader, using the rules of Fig. 22. The first follower moves to the row on the opposite side from its leader. The leader follows and then, the second follower (that did not move) becomes the new leader and conversely. They start exploring the row in the opposite direction (*e.g.*, from left to right if they were exploring from right to left previously).

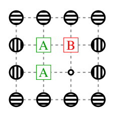

Fig. 19: Locally-defined initial configurations of A_2.

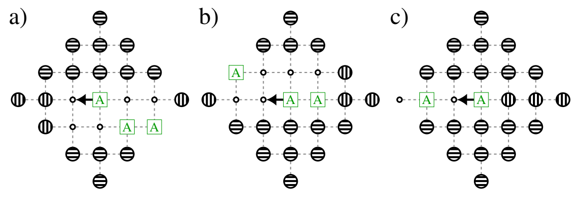

Fig. 20: Rules to move in a straight line.

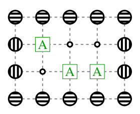

Fig. 21: Locally-defined initial configuration of A_3.

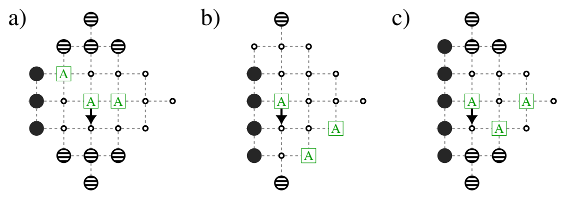

Fig. 22: Rules to move to the next row.

When the robots end the exploration of the last row and reach a corner they perform a turn in order to start exploring the rows in the opposite direction, using the rules of Fig. 23. After three moves, the second follower becomes the leader, the leader becomes the first follower and the first follower becomes the second follower. Then, the robots start the exploration of the grid in the opposite direction (*e.g.*, from bottom to top if they were exploring the rows from top to bottom previously). Notice that the robots have to do some special moves when turning after exploring the first row along the wall; see rules of Fig. 24.

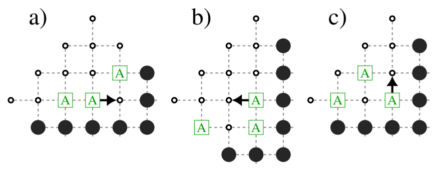

Fig. 23: Rules to turn in a corner.

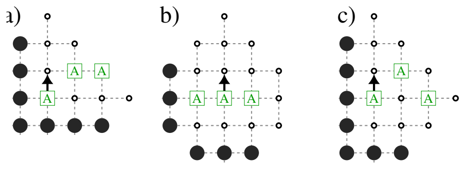

Fig. 24: Rules to turn after exploring the first row.

Theorem 6. *Algorithm* A_3 *solves the PGE problem using three asynchronous oblivious robots with visibility range three.*

5 Conclusion

We have presented three grid exploration algorithms for swarms of luminous robots. Depending on how many they are, how many colors they have, their visibility range, and whether they share a common chirality, one can choose the most appropriate algorithm. We have proven that our algorithms are optimal as the problem becomes not solvable by weakening any of the assumptions. However, some improvements can still be made in future work. For instance, our first algorithm is not locally-defined, precisely if the robots are deployed in the center of the grid, the algorithm does not work. Then, our second algorithm assumes that a robot can see another robot even if it is behind a third one (*i.e.*, we do not assume opacity). This might not be the case in practice and designing an optimal algorithm that works in this case is a challenging open problem.

References

1. L. Blin, A. Milani, M. Potop-Butucaru, and S. Tixeuil. Exclusive perpetual ring exploration without chirality. In *DISC 2010*.
2. F. Bonnet, A. Milani, M. Potop-Butucaru, and S. Tixeuil. Asynchronous exclusive perpetual grid exploration without sense of direction. In *OPODIS 2011*.
3. Q. Bramas, S. Devismes, A. Durand, P. Lafourcade, and A. Lamani. Beedroids: How luminous autonomous swam of UAVs can save the world? In *FUN 2022*.
4. Q. Bramas, S. Devismes, and P. Lafourcade. Finding water on poleless using melomaniac myopic chameleon robots. In *FUN 2020*.
5. Q. Bramas, S. Devismes, and P. Lafourcade. Infinite grid exploration by disoriented robots. In *NETYS 2020*.
6. Q. Bramas, P. Lafourcade, and S. Devismes. Optimal exclusive perpetual grid exploration by luminous myopic opaque robots with common chirality. *Theor. Comput. Sci.*, 2023.
7. A. K. Datta, A. Lamani, L. L. Larmore, and F. Petit. Enabling ring exploration with myopic oblivious robots. In *IPDPS 2015*.
8. A. K. Datta, A. Lamani, L. L. Larmore, and F. Petit. Ring exploration by oblivious agents with local vision. In *ICDCS 2013*.
9. S. Devismes, A. Lamani, F. Petit, P. Raymond, and S. Tixeuil. Terminating exploration of A grid by an optimal number of asynchronous oblivious robots. *Comput. J.*, 2021.
10. S. Devismes, A. Lamani, F. Petit, and S. Tixeuil. Optimal torus exploration by oblivious robots. *Computing*, 2019.
11. S. Devismes, F. Petit, and S. Tixeuil. Optimal probabilistic ring exploration by semi-synchronous oblivious robots. *Theor. Comput. Sci.*, 2013.
12. P. Flocchini, D. Ilcinkas, A. Pelc, and N. Santoro. Remembering without memory: Tree exploration by asynchronous oblivious robots. *Theor. Comput. Sci.*, 411, 2010.
13. P. Flocchini, D. Ilcinkas, A. Pelc, and N. Santoro. How many oblivious robots can explore a line. *Inf. Process. Lett.*, 111(20):1027–1031, 2011.
14. P. Flocchini, D. Ilcinkas, A. Pelc, and N. Santoro. Computing without communicating: Ring exploration by asynchronous oblivious robots. *Algorithmica*, 2013.
15. S. Kamei and S. Tixeuil. An asynchronous maximum independent set algorithm by myopic luminous robots on grids. *Comput. J.*, 67(1), 2024.
16. S. Nagahama, F. Ooshita, and M. Inoue. Terminating grid exploration with myopic luminous robots. *Int. J. Netw. Comput.*, 12(1), 2022.
17. F. Ooshita and S. Tixeuil. Ring exploration with myopic luminous robots. In *SSS 2018*.
18. A. Rauch, Q. Bramas, S. Devismes, P. Lafourcade, and A. Lamani. Optimal exclusive perpetual grid exploration by luminous myopic robots without common chirality. In *NETYS*.