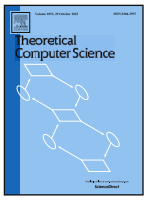
Theoretical Computer Science 1082 (2026) 116072

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs



Optimal asynchronous perpetual finite grid exploration ⋆

Quentin Bramas [a,*], Stéphane Devismes [b], Anaïs Durand [c], Pascal Lafourcade [c], Anissa Lamani [a]

[a] University of Strasbourg, ICUBE, CNRS, France
[b] Université de Picardie Jules Verne, MIS UR 4290, Amiens, France
[c] University Clermont Auvergne, Clermont Auvergne INP, CNRS, LIMOS, France

ARTICLE INFO

Section Editor: Seth Pettie
Handling Editor: Katie Harris

Keywords:
Asynchronous myopic luminous robots
Grids
Perpetual exploration

ABSTRACT

We address the perpetual grid exploration (PGE) 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.

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 moves) 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 introduced by Suzuki and Yamashita [1]. 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 unbounded.

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 is visited by at least one robot infinitely often. We consider here the universe is a *finite grid*, where nodes represent possible locations and edges indicates the possibility of moving from one node to another. In the

⋆ This study has been partially supported by the French ANR project SKYDATA (ANR-22-CE25-0008).
* Corresponding author.
E-mail address: bramas@unistra.fr (Q. Bramas).

https://doi.org/10.1016/j.tcs.2026.116072
Received 21 February 2025; Received in revised form 27 March 2026; Accepted 27 May 2026
Available online 4 June 2026

Table 1
Summary of our results.

Chirality	Visibility	# Robots	# Colors	Algorithm
Yes	1	finite	finite	Impossible (Theorem 2)
Yes	2	3	1	Impossible (Theorem 3)
Yes	2	3	2	[12]
Yes	2	4	1	A_1
No	2	3	2	A_2
No	3	3	1	A_3

following, the problem will be referred to as the *perpetual grid exploration (PGE)*. Notice also that we will focus on optimal *exclusive* solutions with respect to both the visibility range and the number of robots. *Exclusiveness* adds the following constraint on robots behavior: they can neither occupy the same node nor traverse the same edge simultaneously.

Related Work. The exploration problem is one of the benchmark tasks when it comes to robots evolving on finite graphs. Various topologies have been studied: lines [2], rings [3–5], tori [6], grids [7,8], cuboids [9], and trees [10]. Two main variants of the problem have been investigated: (*i*) the *perpetual exploration* [3,7,9,11,12] (considered in this paper) which requires the robots to visit each node of the graph infinitely often and (*ii*) the *terminating exploration* [2,4,6,8,10,13–15] which requires the robots to visit each node of the graph at least once and then stop moving.

Most of the literature consider robots with unlimited visibility range allowing them to observe every node of the system [2–4,6–8,10,13]. In this case, robots are oblivious (*i.e.*, they cannot remember past actions) and usually solve the terminating exploration.

Myopic robots have been considered in both variants of the problem [5,12,14,16]. 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, *i.e.*, , a common notion of clockwise orientation. In fact, chirality is usually assumed when robots evolve in the continuous 2D Euclidean plan [17] but, recently, it has been investigated in the discrete universe [12].

On 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 and operates each LCM cycle synchronously. 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 [12]. The case where robots have no common chirality is investigated in [11]. In this paper, it was proven that the problem is not solvable with only two robots whatever be the number of colors and whatever be the finite visibility range. An optimal solution *w.r.t.* the visibility is then proposed: it uses only three robots under visibility range one and three colors. The case where robots are oblivious and the visibility range is 2 is solved using five robots.

Asynchronous myopic robots have been considered in the context of the terminating exploration problem on grids [15] assuming robots can start from distinct positions but allowing them to occupy the same location simultaneously during the execution (*i.e.*, the exploration is 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 (*i.e.*, perpetual exclusive exploration and myopic robots), the case of asynchronous robots has only been considered in [12] 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.

In the case of cuboids, it has been shown in [9] that three synchronous robots with a common chirality endowed with five colors are necessary and sufficient to solve the perpetual exploration problem.

Asynchronous robots with colors and limited visibility range have been considered to solve the maximum independent set problem on grids [18]. 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 under a visibility range one, regardless of the number of robots and the number of colors for their lights. Next, we show that the problem is also unsolvable under visibility range two with three asynchronous oblivious robots (oblivious means that each robot is only endowed with a one-color light).

We then present three optimal algorithms solving the problem. The first algorithm assumes a common chirality and requires four oblivious robots under visibility range two. The second one uses three robots without common chirality under visibility range two but using two colors. The third algorithm uses three oblivious robots without common chirality under visibility range three.

Table 1 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 [1], *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 $E = \{\{(i, j), (k, l)\} \ : \ (i, j) \in V \wedge (k, l) \in V \wedge |i - k| + |j - l| = 1\}$.

[1] The requirement on the numbers of lines and columns is only made for the sake of simplicity.

Grid coordinates are used for the analysis only, *i.e.*, robots cannot access them. Without the loss of generality we assume that $\mathcal{L} \leq C$. The size of the grid is $\mathcal{L} \times C$.

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). It has been shown [19] that the assumption of instantaneous moves is equivalent to the assumption that robots can be seen on the edge during their movements. 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 $d = Up$ but the robot sees the grid upside-down (f is the π-rotation), then the robot moves $Down = f^{-1}(Up)$. 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 immediately 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 $(Sync^{3n})^\omega$. We denote by $C \overset{Sync^{3n}}{\longmapsto} C'$ 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 describe possible transitions of 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 $(Cl, Init, T)$ contains 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 only consider sets of rules that are *minimal* where a set of rules S is minimal if it does not contain two rules (V, d, c) and (V', d', c) such that $V' = f(V)$ and $d' = f(d)$ for some $f \in \mathcal{IT}$.

Hence, an algorithm $(Cl, Init, T)$ can be equivalently reformulated as a triplet $(Cl, Init, S)$ where S is a minimal set of rules satisfying $\forall V \in Views$, $T(V) = (d, c)$ if and only if $\exists f \in \mathcal{IT}, (f(V), f(d), c) \in S$.

Example 1. 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 (B) robot sees three robots r_1, r_2, and r_3 with color R such that

1. r_1 is on top at distance 2 from the blue robot,
2. r_2 is at distance 1 from the blue robot on its left, and
3. r_3 is on a diagonal of the blue robot, and respectively at distance two from r_1 and distance one from r_2,

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.

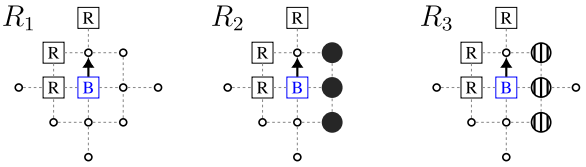


Fig. 1. Examples of rules.

Algorithms having locally-defined initial configurations. It is clear the problem is not solvable without any assumption on the set of initial configurations. However, it may seem too powerful to assume that the initial configuration is a particular one. One can see that no algorithm can solve the PGE problem in all solvable initial configuration because some of them can be incompatible (an algorithm solving the problem starting from one configuration would not solve the problem starting from another, see for instance in [12]). In one hand, assumging a given set of initial configurations makes our impossibility results stronger. On the other hand, we try to avoid making assumptions on the initial configuration that are too strong, as it would make our algorithms less general.

Hence, 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.*

Proof. The if part is trivial since a synchronous scheduler is a particular case of an asynchronous scheduler. So, let us focus on the only if part: let $(C_i)_{i\in\mathbb{N}}$ be an execution of A in G under the synchronous scheduler. $\forall i \in \mathbb{N}$, let $\mathfrak{C}_i = C_{3ni}$ and we denote by c_r^i (resp. p_r^i) the color (resp. the position) of robot r in $\mathfrak{C}_i$, moreover we let r_i be the seq-enabled robot in $\mathfrak{C}_i$.

Let $i \in \mathbb{N}$. When executing the synchronous scheduler $Sync^{3n}$ on $\mathfrak{C}_i$, the robot r_i is the only enabled robot. So, $\mathfrak{C}_{i+1}$ is obtained from $\mathfrak{C}_i$ by applying $T(V_{\mathfrak{C}_i}(r_i)))$. In other words, $\mathfrak{C}_{i+1} = \mathfrak{C}_i \setminus (p_{r_i}^i, c_{r_i}^i) \cup (p_{r_i}^{i+1}, c_{r_i}^{i+1})$.

Let $e = (C'_i)_{i\in\mathbb{N}}$ be any execution of A in G under an arbitrary scheduler S such that $C'_0 = \mathfrak{C}_0$ $(= C_0)$. Since S is fair, we can define recursively the sequence of time instants $(t_i)_{i\in\mathbb{N}}$ as follows. First, $t_0 = 0$. Then, for any $i \geq 0$, we define t_{i+1} as the first time $t_{i+1} > t_i$ such that r_i makes a Look phase at some time $t \in [t_i, t_{i+1} - 1)$ and either (a) its Compute phase at time $t_{i+1} - 1$ if it was col-enabled at time t, or (b) its Move phase at time $t_{i+1} - 1$ if it was move-enabled at time t.

We now show by induction on i that $\forall i \in \mathbb{N}$,

1. $C'_{t_i} = \mathfrak{C}_i$; and
2. for any robot r such that its next action after t_i in the schedule S is Compute or Move, $T(V_{C'_t}(r)) = (Idle, c_r^i)$ where $t < t_i$ is the time of its previous Look in e.

For $i = 0$, by definition we have $C'_{t_0} = C'_0 = \mathfrak{C}_0$ and the next action of every robot is Look, so the base case trivially holds. Assume now that the claim holds for some $i \geq 0$. We have to show that it holds for $i + 1$. Since $C'_{t_i} = \mathfrak{C}_i$ by induction hypothesis, r_i be the only enabled robot in C'_{t_i} and r_i is actually seq-enabled. Assume that r_i is move-enabled (similar arguments will hold if it is col-enabled). So, $T(C'_{t_i}(r_i)) = T(\mathfrak{C}_i(r_i)) = (move, c_{r_i}^i)$ and $move \neq Idle$.

To show the induction step, we first need to show that $\forall t \in [t_i, t_{i+1} - 1]$, $C'_t = C'_{t_i}$. We proceed by induction on t. The base case $t = t_i$ is trivial. Consider now the case $t > t_i$. By induction hypothesis, we have $C'_{t-1} = C'_{t_i}$ Let $S_{t-1} = (r_{t-1}, a_{t-1})$ be the action of the scheduler at time $t - 1$ and consider the following two cases:

$r_{t-1} \neq r_i$: We have the following three possible cases for Action a_{t-1}. a_{t-1} is (i) a Compute phase or a Move phase and the previous Look of r_{t-1} occurs before t_i in which case r_{t-1} stays $Idle$ and keeps its previous color in the step from $t - 1$ to t using the hypothesis of the induction on i meaning that $C'_t = C'_{t-1} = C'_{t_i}$, (ii) a Look phase and so $C'_t = C'_{t-1} = C'_{t_i}$, or (iii) a Compute phase or a Move phase and the previous Look of r_{t-1} occurs at some time $t' \in [t_i, t - 1)$ and, again, $T(V_{C'_{t'}}(r_{t-1})) = (Idle, c_{r_t}^{t_i})$

since $C'_{t'} = C'_{t_i} = \mathfrak{C}_i$ by induction hypothesis and $r_{t-1} \neq r_i$, so r_{t-1} stays $Idle$ and keeps its previous color in the step from $t-1$ to t meaning that $C'_t = C'_{t-1} = C'_{t_i}$.

$r_{t-1} = r_i$: In this case, by definition of t_{i+1}, we have the following two possible cases. (1) a_{t-1} is either a Compute phase or a Move phase and previous Look of r_{t-1} occurs before t_i in which case r_{t-1} stays $Idle$ and keeps its previous color in the step from $t-1$ to t using the hypothesis of the induction on i meaning that $C'_t = C'_{t-1} = C'_{t_i}$; (2) a_{t-1} is a Look phase and so $C'_t = C'_{t-1} = C'_{t_i}$.

Hence, the induction on t holds.

Now, r_i executes its move at time $t_{i+1} - 1$ based on a Look phase performed at time t such $t_i \leq t < t_{i+1} - 1$ (by definition of t_{i+1}). From the previous induction, $T(V_{C'_t}(r_i)) = T(V_{C'_{t_i}}(r_i)) = T(V_{\mathfrak{C}_i}(r_i)) = (move, c^i_{r_i})$. So, $C'_{t_{i+1}-1} = C'_{t_i} = \mathfrak{C}_i$ is modified in at time $t_{i+1} - 1$ according to $(move, c^i_{r_i})$ and consequently $C'_{t_{i+1}} = \mathfrak{C}_{i+1}$.

Consider now any robot r such that its next action after t_{i+1} in the schedule S is Compute or Move. Since r_i executes its Move phase at time t_{i+1}, its next action after t_{i+1} in the schedule S is necessarily a Look phase. So, $r \neq r_i$. Let t the maximum time such that $t < t_{i+1}$ and r executed a Look phase at time t. If $t < t_i$, $T(V_{C'_t}(r)) = (Idle, c^i_r)$ by induction hypothesis. Otherwise, we have shown that $C'_t = C'_{t_i}$, so $T(V_{C'_t}(r)) = T(V_{C'_{t_i}}(r)) = T(V_{\mathfrak{C}_i}(r)) = (Idle, c^i_r)$ since r is not enabled in $\mathfrak{C}_i$ (recall that $r \neq r_i$). This concludes the induction on i.

To summarize, we have the following diagram:

$$\begin{array}{ccc} C_{3ni} & \xmapsto{Sync^{3n}} & C_{3n(i+1)} \\ \| & & \| \\ C'_{t_i} & \xmapsto{e} & C'_{t_{i+1}} \end{array}$$

Moreover, notice that since A is synchronous-sequential, $\exists j \in [3ni, 3n(i+1)-1)$ such that $\forall \ell \in [3ni, j]$, $C_\ell = C_{3ni}$ and $\forall \ell' \in (j, C_{3n(i+1)}]$, $C_{\ell'} = C_{3n(i+1)}$. Then, since A solves the PGE problem under the synchronous scheduler, every node is visited infinitely often in $(C_i)_i$, hence in $(C'_i)_i$ as well, according to the previous property and the induction. This concludes the proof. □

Notice that the previous theorem can be trivially extended to topologies other than grids (just by generalizing the modeling to arbitrary graphs, in particular the transition function T).

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 (*e.g.*, when considering a locally-defined algorithm, 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 Section 3.1, we establish 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). In Section 3.2, we give a specific impossibility result for the PGE problem with three oblivious robots having visibility range two.

3.1. Impossibility for robots with visibility one

To show Theorem 2, we prove a variant of the test of the fence (introduced in [20]) for the case of finite grids. In our setting, a fence is the boundary of width two of a rectangle.

More formally, Let $Rectangle((x_1, y_1), (x_2, y_2))$ be the nodes in the rectangle delimited by the given points, with $x_1 < x_2$ and $y_1 < y_2$: $Rectangle((x_1, y_1), (x_2, y_2)) = \{(x, y) \in \mathbb{Z}^2 \mid x_1 \leq x \leq x_2 \text{ and } y_1 \leq y \leq y_2\}$. 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: $F_{(x_1,y_1),(x_2,y_2)} = Rectangle((x_1, y_1), (x_2, y_2)) \setminus Rectangle((x_1 + 2, y_1 + 2), (x_2 - 2, y_2 - 2))$. We say a robot is *outside (resp., inside) the fence* if it is outside the outer rectangle (resp., inside the inner rectangle). 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 $(x, y) \in Rectangle((x_1 + 2, y_1 + 2), (x_2 - 2, y_2 - 2))$ 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 of size at least $n \times n$, 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.*

Proof. Consider a grid G of size large enough so that regardless of the initial configuration, there exists a node u that is at distance at least $2n+1$ from any robot and any wall.[2] u should be visited within a finite number of rounds and to do so, a group of robots must cross at least n fences to reach u. If no subset of robots single-handed crosses at least one of them, then this means that each time a fence is crossed, some robots are not crossing and are left behind. Since n fences must be crossed to reach u, there is not enough robots to cross the n fences to reach u. Hence, a subset of robots single-handed crosses a fence in a finite number of rounds. □

Lemma 2. *If a robot is ordered to move, then its destination is empty.*

Proof. Assume by contradiction that a robot r is ordered to move to a neighboring node u that is not empty. Then, if the scheduler only activates r, a collision occurs, which is forbidden. □

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

Proof. Assume by contradiction that such an algorithm exists. We will show that a group of robots cannot travel an arbitrary distance. To see this, we will prove that the group does not pass the test of the fence (Lemma 1), which means that a group of robots that do not see any wall cannot move from outside of the fence to the inside of the fence, without leaving at least a robot behind.

Recall that if a rule is ambiguous, then the destination depends on the indistinguishable function chosen by the adversary. In the remaining, we assume that if a robot r executes an ambiguous rule, then the adversary always chooses an indistinguishable function such that the destination *is not* towards the inside of the fence.

Consider an execution where the scheduler selects robots one by one in a round-robin fashion and activates each robot three times so that it performs a full cycle before the next robot is activated. Assume, by the contradiction, that the test of the fence succeeds. So, consider a group of robots is initially outside of a fence F and every robot is at least at distance 2 from any wall. Let t be the first time at which all the robots are inside F.

By assumption on the scheduler, at $t-3$ the last robot r on F is activated and at time $t-1$ it moves towards the inside of F. Since it is the last robot on F, it is isolated. Indeed, there is no robot outside F, no other robot on F, and its destination inside F should be also empty, by Lemma 2. So, the rule executed by r is ambiguous and this contradicts the fact that its destination is inside F. □

3.2. Impossibility for three oblivious robots with visibility two

Lemma 3. *Consider any algorithm* A *solving the PGE using three oblivious robots having visibility range 2. Let $e = (C_i)_{i\in\mathbb{N}}$ an execution of* A *in a grid G under scheduler S.*

There does not exist $i \in \mathbb{N}$ such that (1) no robot have a pending move at time i and (2) every robot is isolated and does not see any wall in C_i.

Proof. Assume, by the contradiction, $\exists i \in \mathbb{N}$ such that (1) no robot has a pending move at time i and (2) every robot is isolated and does not see any wall in C_i. Notice then that G has size at least 7×8. Recall that when an isolated robot moves, the adversary can choose its destination using the appropriate indistinguishable transformation. Consider now the execution e' having the prefix $C_0 \dots C_i$ and where the synchronous scheduler is applied from C_i using indistinguishable transformations as follows: (1) the adversary chooses for each robot a transformation so that each robot is still isolated and still does not see any wall in the configuration C' reached from C_i after $Sync^9$, and (2) the adversary chooses for each robot a transformation so that C_i is reached again from C' after $Sync^9$. Consequently, at most 6 out of at least 56 are visited infinitely often in e', a contradiction. □

Lemma 4. *Consider any algorithm* A *solving the PGE using three oblivious robots having visibility range 2 say r_1, r_2, and r_3. Let $e = (C_i)_{i\in\mathbb{N}}$ an execution of* A *in a grid G of size at least 12×12 under scheduler S.*

There does not exist $i \in \mathbb{N}$ such that (1) no robot has a pending move at time i, (2) r_1 is isolated and does not see any wall in C_i, and (3) the two others are at distance at least 5 from any wall in C_i.

Proof. Assume, by the contradiction, $\exists i \in \mathbb{N}$ such that (1) no robot have a pending move at time i, (2) r_1 is isolated and does not see any wall in C_i, and (3) the two others are at distance at least 5 from any wall in C_i. First, by Lemma 3, r_2 and r_3 are at distance at most 2 from each other in C_i. Consider first the case where r_2 and r_3 are neighbors in C_i.

- If r_2 and r_3 are not enabled in C_i, then r_1 should be. Consider now the execution e' having the prefix $C_0 \dots C_i$ and where the synchronous scheduler is applied from C_i using indistinguishable transformations as follows: (1) the adversary chooses for r_1 a

[2] Take for instance a grid size $[(2n+1)^3 + 2n] \times [(2n+1)^3 + 2n]$ by observing that each robot is at distance at most n from at most $(2n+1)^2$ nodes, since there are n robots, at most $n(2n+1)^2$ are "covered" in this way, there must be a node that is not "covered". The addition of $2n$ is to make sure that there exists such a node that is also at distance n from any wall.

transformation so that r_1 is still isolated and still does not see any wall in the configuration C' reached from C_i after $Sync^9$, and (2) the adversary chooses for r_1 a transformation so that C_i is reached again from C' after $Sync^9$ (since their views have not changed, r_2 and r_3 are still disabled in C'). Consequently, at most 4 nodes over at least 144 are visited infinitely often in e', a contradiction.

- Otherwise, since r_1 is isolated, the views of r_2 and r_3 are identical up to a rotation (of angle π) in C_i. Among r_2 and r_3, the scheduler can then activate the farthest from r_1 for a full cycle. Its destination should be empty (Lemma 2) and the adversary can choose a destination at distance at least 3 from r_1. Consequently, a configuration where (1) no robot has a pending move, (2) r_1 is isolated and does not see any wall, (3) r_2 and r_3 are at distance 2 from each other, and (4) at distance at least 4 from any wall can be reached.

Overall, we can then assume w.l.o.g., that the execution reaches a configuration C_j where (1) no robot has a pending move, (2) r_1 is isolated and does not see any wall, (3) r_2 and r_3 are at distance 2 from each other, and (4) at distance at least 4 from any wall. Again remark that the views of r_2 and r_3 are identical up to a rotation (of angle π) in C_j. So, either r_2 and r_3 are both idle or are both enabled.

In the former case, r_1 should be enabled in C_j. So, consider the execution e' having the prefix $C_0 \dots C_j$ and where the synchronous scheduler is applied from C_j using indistinguishable transformations as follows: (1) the adversary chooses for r_1 a transformation so that r_1 is still isolated and still does not see any wall in the configuration C' reached from C_j after $Sync^9$, and (2) the adversary chooses for r_1 a transformation so that C_j is reached again from C' after $Sync^9$ (since their views have not changed, r_2 and r_3 are still disabled in C'). Consequently, at most 4 nodes over at least 144 are visited infinitely often in e', a contradiction.

In the latter case, the views of r_2 and r_3 are identical up to a rotation (of angle π) in C_j. From C_j the destination of r_2 and r_3, if activated, are opposed. Moreover, they cannot move toward each other, otherwise using a synchronous scheduler, we obtain a collision, a contradiction. So, if activated, they move away from each other. Then, among r_2 and r_3, the scheduler can activate one of the farthest robots from r_1 for a full cycle. The adversary can choose an indistinguishable transformation so that a destination at distance at least 3 from r_1. Consequently, a configuration where (1) no robot has a pending move, (2) the three robots are isolated, (3) do not see any wall can be reached, which contradicts Lemma 3. □

The proof of the next theorem follows the same principles as Theorem 3.5 in [12]. 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.*

Proof. We proceed by the contradiction: we assume an algorithm solves PGE problem with three oblivious robots having visibility range two and assuming a common chirality.

In the following proof and unless otherwise mentioned, we assume that when a robot is activated by the scheduler, it atomically executes a full LCM cycle.

We first prove that we can construct an adversarial scheduler that imposes a single way, for a group of three oblivious robots, to travel an arbitrary distance without ever seeing any wall. More formally, we can represent a configuration C by a pair $(\overline{C}, tr)$ where

- $\overline{C}$ is the configuration where the node states of C are translated in such way that the coordinates of the bottom-left corner of the smallest rectangle that encloses the three robots is $(0, 0)$; and
- tr is a translation such that $C = tr(\overline{C})$.

In other words, $\overline{C}$ represents the "local" configuration of C (the relative position of the robots without taking their absolute position on the grid into account). We say that a configuration $(\overline{C}, tr)$ is a (non-trivial) translation of a configuration $(\overline{C'}, tr')$ if $\overline{C} = \overline{C'}$ and $tr \neq tr'$.

We call *open-air shift* any execution factor $C_0, \dots, C_k$ such that

1. $\forall i \in [0.k]$, no robot sees a wall is C_i,
2. $\forall i \in [1.k-1]$, C_i is not a translation of C_0,
3. C_k is a translation of C_0.

In an arbitrary large grid, the robots should be able to achieve open-air shifts to continue the perpetual exploration, *e.g.*, once there are in the middle of the grid. However, we now show that, in large grids, there is a fair scheduler that can impose a unique cycle of local configurations generated by open-air shifts. That is, for every two open-air shifts $C_0, \dots, C_k$ and $C'_0, \dots, C'_{k'}$, we have $k = k'$ and $\exists i \in \{0, \dots, k\}$ such that $\forall j \in \{0, \dots k\}$, $\overline{C_j} = \overline{C'_{(j+i) \bmod (k+1)}}$.

To show this uniqueness result, we now assume the grid is large enough to allow the use of Lemmas 3 and 4. Consequently, no robot can be isolated in any configuration of an open-air shift.

Consider the graph of all possible local configurations of three oblivious robots (up to rotations) where no robot is isolated, such that a link exists between two local configurations if one is reachable from the other after a single robot move (see Fig. 3). Some particular rules are defined Fig. 2 and the transitions in Fig. 3 associated with rule R_0 are labeled. A periodic sequence of sequential movements corresponds to a cycle in this graph.

Since the grid is assumed to be arbitrary large and the robots should stay close together (Lemmas 3 and 4), the group of three robots has to be translated many times consecutively to traverse arbitrary long distance without seeing wall using open-air shift

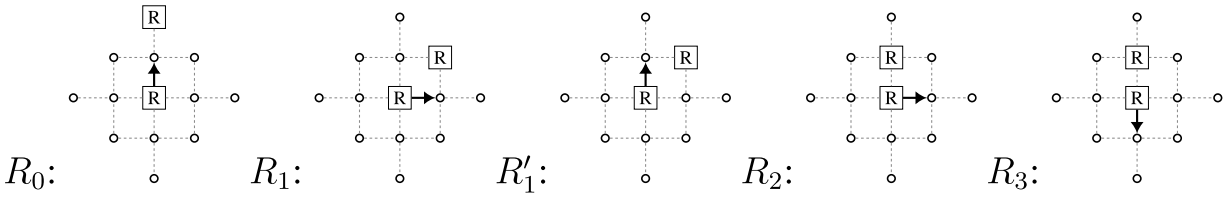


Fig. 2. Some particular rules that are useful in the proof. R is the only color used by the robots.

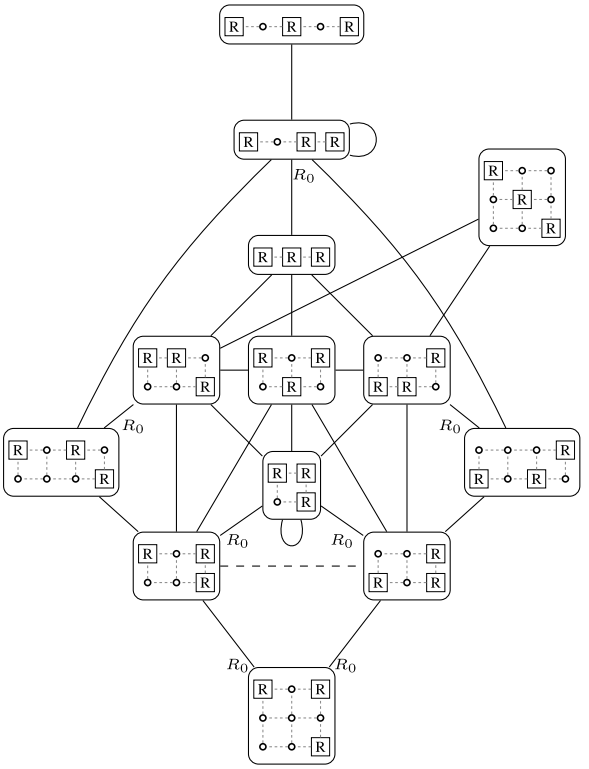


Fig. 3. The graph of all possible local configurations of three oblivious robots (up to rotations) where no robot is isolated, such that a link exists between two local configurations if one is reachable from the other after a single robot move. R is the only color used by the robots. The link with label R_0 highlights where this rule, defined in Fig. 2, is executed. The dashed link cannot be part of an algorithm.

cycles. Then, one can observe that if an open-air shift cycle exists, the adversary can impose that the open-air shift does not contain two times the same configuration. Indeed, if $C_i = C_j$, with $i < j$, then, in configuration C_i, the adversary can activate the robot such that the next configuration is C_{j+1} instead of C_{i+1}. This does not break the fairness of the scheduler since the cycle remains an open-air shift and, as we have just seen, executing an open-air shift cycle several times (at least three times) consecutively ensure that all the robots have move at least once (since they are all translated). This is why in the remaining we sometimes consider that if C_0 is the configuration previous to C_1, then it cannot be the next configuration as well.

Step 1: Pruning the local configuration graph We first prove that the graph can be pruned by removing some local configurations.

Claim 1: *The scheduler can enforce*  *not to be a part of a cycle.*

Let $\overline{C} =$  and $\overline{C'} =$  . If local configuration $\overline{C''} =$  is part of a cycle, then the previous local configuration in the cycle is either $\overline{C}$ or $\overline{C'}$. Also, to obtain the next local configuration, Rule R_0 is executed (this is the only rule that can be executed in $\overline{C}''$ to avoid isolating a robot), hence Rule R_0 is part of the algorithm. But since R_0 can be executed directly on $\overline{C}$ and $\overline{C'}$, the scheduler can execute the rule to obtain  instead of $\overline{C}''$. From there, any robot can be executed to maintain fairness if necessary. Thus, the scheduler can ensure that  is not part of a cycle.

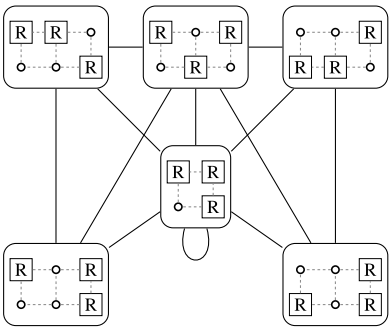
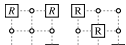
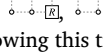


Fig. 4. Pruned local configuration graph. R is the only color used by the robots.

Claim 2: *The adversary can impose that a cycle containing a rotationally symmetric local configuration does not result in a translation after two periods.*

Either the cycle only consists of these rotationally symmetric local configurations and the periodic movements do not result in a translation, or it is part of a cycle that contains another local configuration. In the latter case, this results in the execution of an ambiguous rule so the adversary can alternate between using the π-rotation and the identity for the indistinguishable function to ensure that the robots come back to their position every two periods while ensuring fairness.

Using the previous claims, we can remove  ,  , , , and  , from the local configuration graph. We can also remove the dashed link because if the rule allowing this transition exists in the algorithm, it can be applied again and disconnect the robots.

Claim 3: *The adversary can make sure that a cycle containing* (*a*) *or* (*b*) *does not result in a translation.*

We only study Case (*a*), since Case (*b*) is identical up to a mirroring. So, let us study a cycle containing local configuration $\overline{C_1} =$. Assume first that the previous local configuration is $\overline{C_0} =$. By the above observation, the next local configuration cannot be C_0 because this implies that the previous and next (global) configurations are the same and we assumed this is not the case. Then, the next local configuration $\overline{C_2}$ is either or and is obtained after executing rule R_1 or R_1'. Now, R_1 or R_1' can be executed directly on $\overline{C_0}$ together with the already scheduled rule to obtain $\overline{C_2}$ without reaching $\overline{C_1}$ and without compromising fairness. So, the scheduler can ensure that $\overline{C_1}$ is not part of such a cycle. The reasoning is similar if $\overline{C_0}$ is or and $\overline{C_2}$ is .

Consider the case where the previous local configuration is $\overline{C_0} =$ and the next local configuration is $\overline{C_2} =$ (the same reasoning will apply for $\overline{C_0} =$ and $\overline{C_2} =$). $\overline{C_2} =$ is then obtain from $\overline{C_1}$ by executing R_1. Moreover, to move from $\overline{C_0}$ to $\overline{C_1}$, rule R_2 should be executed. Since rule R_2 is part of the algorithm, the scheduler can ensure that the next local configuration is $\overline{C_3} =$. Then, rule R_1, which is also part of the algorithm, can be executed by the scheduler and the next local configuration is $\overline{C_4} =$. From there, rule R_2 can be executed by the scheduler to reach local configuration $\overline{C_5} =$, which is the rotated local configuration of shown in Fig. 3. With the given rules, if the next local configuration is , its mirror , or , then from there, executing rule R_2 would result in reaching $\overline{C_5}$ again (*n.b.*, has been removed). Thus, the cycle must terminate with either (*a*) $\overline{C_6'} =$, which is local configuration $\overline{C_0}$ rotated by π, or (*b*) with $\overline{C_6} =$ and a local configuration $\overline{C_7} =$, which is local configuration $\overline{C_0}$ rotated by $-\pi/2$. In the former case, the sequence does not induce a translation. In the latter case, one can see that the configuration obtained after 4 periods is exactly the same as $\overline{C_0}$ (without any translation). The cycle from $\overline{C_0}$ to $\overline{C_7}$ is illustrated in Fig. 5.

So if a cycle exists, it exists in the pruned subgraph illustrated in Fig. 4.

Step 2: Proving that a single cycle exists in the pruned graph. In the pruned local configuration graph, we can see that a cycle must include either local configuration or . One can easily check that if a cycle does not include , then the cycle does not result in a translation. Thus, any algorithm such that 3 oblivious robots travel an arbitrary distance, must include a cycle that includes $V_0 =$.

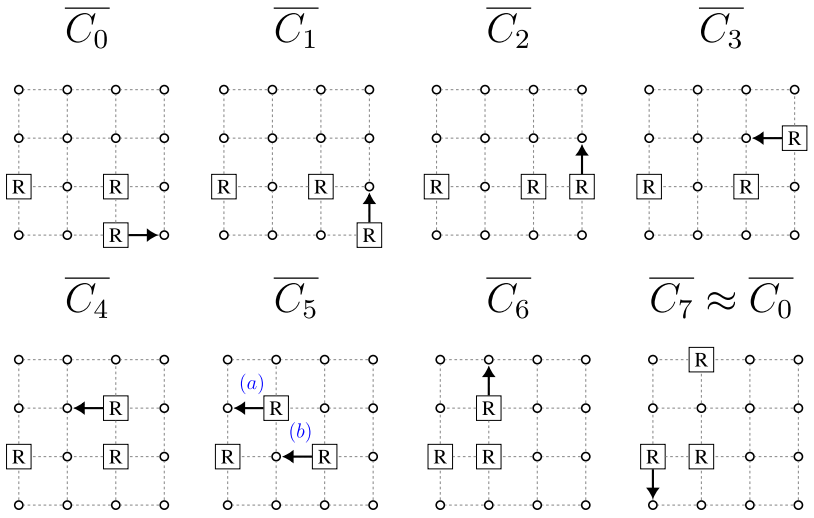


Fig. 5. The sequence of 7 configurations, as described in the proof, that is invariant after 4 periods. Case (*a*) and (*b*) show two possible moves from configuration $\overline{C_5}$. If case (*a*) occurs, the obtained configuration is similar to $\overline{C_0}$ (rotated by π). If case (*b*) occurs, configuration $\overline{C_6}$ is obtained and then $\overline{C_7}$, similar to $\overline{C_0}$ (rotated by $-\pi/2$).

An adversary can ensure that the same cycle is always used *i.e.*, once the local configuration is V_0 the adversary always activates the robot so that the local configuration ends up in the next local configuration of the cycle, and so on until V_0 is obtained again. When the robots move away from a wall and start exploring a row, they perform this cycle until reaching the opposite wall. Every time a cycle is performed, the robots are translated by one. So the way the robots reach the opposite wall does not depend on how they left the first wall.[3]

The end of the proof is similar to the impossibility proof of Theorem 3.5 of [12], which we adapt for our specific settings.

Step 3: proving that a single way to perform an open-air shift is not enough. First, remark that a group of two oblivious robots cannot travel an arbitrary distance without ever seeing any wall or the third robot. Indeed, in such a case, the view of each robot in the group is the one of the other rotated by the angle π. So, the adversary can decide to apply an indistinguishable function such that they move in the opposite direction. So, to perpetually explore the whole grid, the three robots have to be grouped to travel an arbitrary distance without seeing any wall (*e.g.*, to visit a center of the grid).

Consider now the first time t the three robots reach a wall (this necessarily happens, *e.g.*, after visiting the middle of the grid). There exists a constant S (independent of the size of the grid) such that the three robots can either (1) remain indefinitely in a subgrid of size S, or (2) remain in a subgrid of size S for a finite number of rounds, then leave the wall and move, the three together, straight toward the opposite wall, or (3) remain at distance at most S from the wall until being close to a corner. Fig. 6 illustrates the last two cases.

In Case (1), some nodes will not be visited if the grid is large enough, a contradiction.

In Case (2), the scheduler can impose the robots executes the same movements as they made to reach the wall at time t, rotated by angle π (to move in the opposite direction). Indeed, we have just seen that there is the unique way for the three robots to travel without seeing walls. Hence, when reaching the opposite wall, since they cannot distinguish between the two walls, robots can perform the same turn, remain in a subgrid of size S for the same finite number of rounds, and then leave the wall to move straight towards the first wall. Hence, they will end up in the exact same position as at time t. The three robots may continue this periodic movement infinitely, leaving nodes unvisited if the grid is large enough, a contradiction.

Consider now Case (3). After following the wall (staying at distance at most S from it until reaching another wall), the robots become close to a corner and again we have three possibilities: either (*a*) they remain in a subgrid of size at most S' where S' is a constant (independent of the size of the grid), (*b*) they follow a wall (either the same one on the opposite direction, or the new encountered one) for a finite number of rounds, then leave the wall to move in straight line until reaching the opposite wall, or (*c*) they follow a wall until being close to another corner. In Case (*a*), some nodes are never visited if G is large enough, a contradiction. In Cases (*b*) and (*c*), there exists a constant S'' (independent of the size of the grid) such that the robots always stay at distance at most S'' from a wall, and when they reach a new wall, the same thing occurs again, so if G is large enough, some nodes that are far from all walls are never visited, leading to a contradiction. □

[3] Observe that the fact that the translation is by one is important because it means that there is only one local configuration when the robots reaches the wall and so the sequence of moves when the robots reaches the wall is always the same.

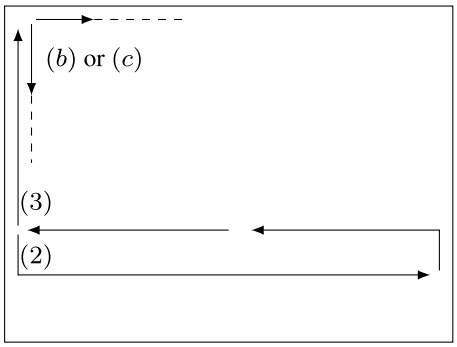


Fig. 6. The movement leading to a contradiction.

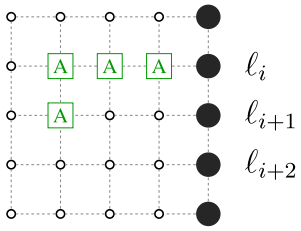


Fig. 7. Instance of an initial configuration.

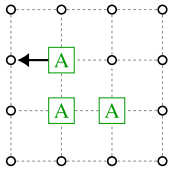


Fig. 8. Exploration formation.

4. Algorithms

4.1. Algorithm with four 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 requires the number of lines and columns to be 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 row by row and column by column 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 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 ↓ in the illustration. After all the lines are explored, the four robots perform a sequence of moves to change the exploring direction, from ↓ to ←. In more detail, the four robots initially form an L-shaped pattern with a **unique** robot being adjacent to a wall (see Fig. 7) 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 ↓ by executing Rule a) of Fig. 9. While the sentinel remains idle, the three other robots, being in exploration formation (Fig. 8), 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. 9, (b – f).

Once the three explorer robots reach the opposite wall, they not only move to the next adjacent row with respect to direction ↓ 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. 10. The sequence of configurations during this process is illustrated in Fig. 11. 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

[4] An interactive simulation of A_1 is given at https://robots.app.bramas.fr/?SSS2024/A1.

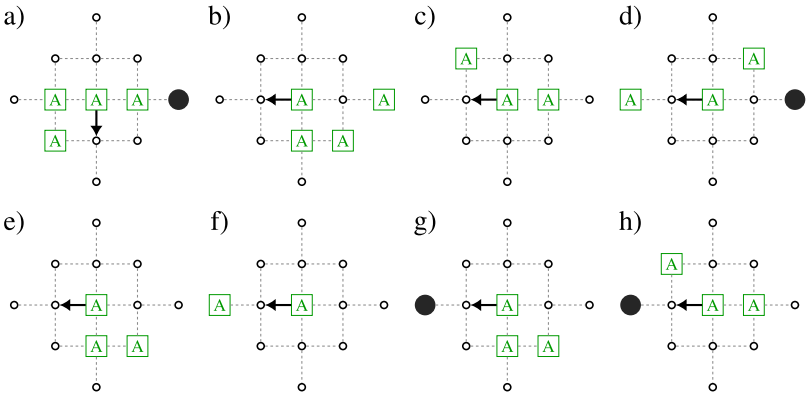


Fig. 9. Rules to move in a straight line.

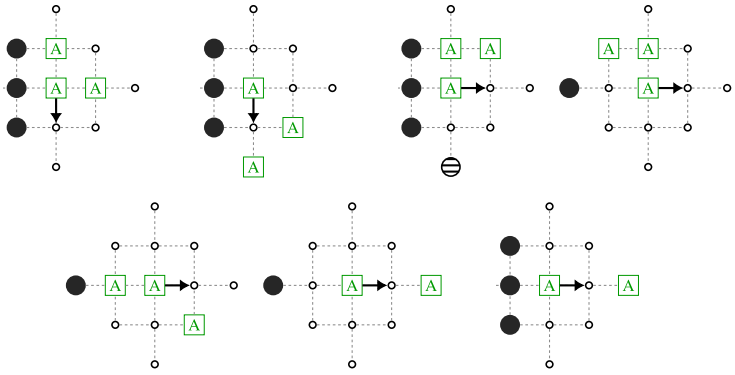


Fig. 10. Rules to perform the U-turn and initiate the exploration towards the sentinel.

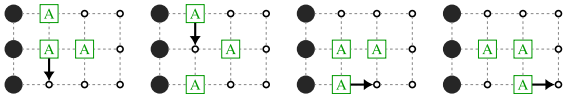


Fig. 11. Sequence of configurations during a change of row and a U-turn.

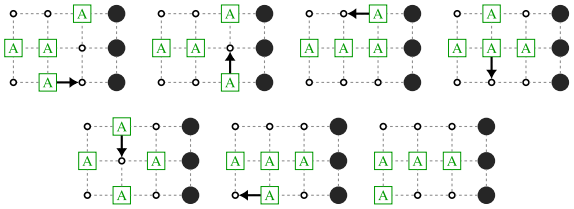


Fig. 12. Sequence of configurations during a change of row for the sentinel.

pattern but this time on row ℓ_{i+1}. This is done by executing the rules of Fig. 13. The sequence of configurations during this process is illustrated in Fig. 12.

Let $\ell_1, \ell_2, \ldots, \ell_{\mathcal{L}}$ be the sequence of rows in direction ↓. 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 ←, 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. 13 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. 14 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. 14. The sequence of configurations during this process is presented in Fig. 15. Robots explore the columns as they did for the rows and then

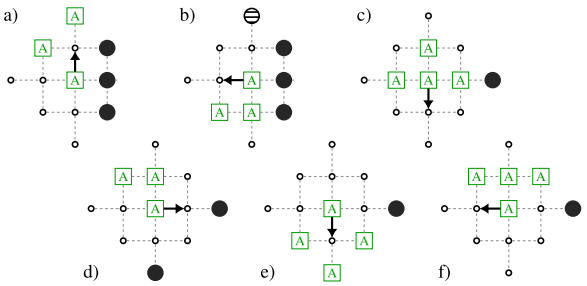


Fig. 13. Rules to move the sentinel to the next row to be explored.

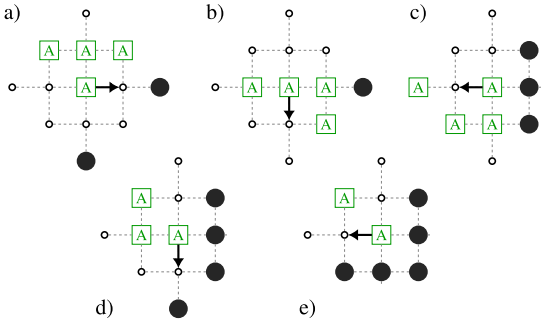


Fig. 14. Rules to switch to the columns exploration.

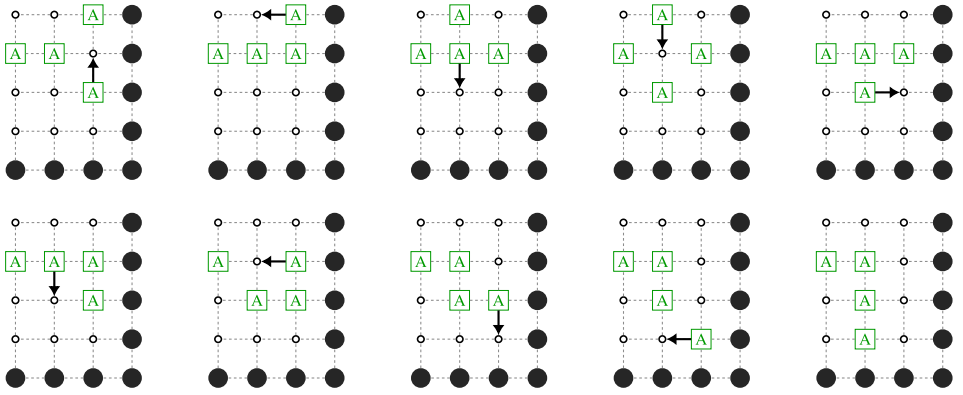


Fig. 15. Sequence of configurations during rows/columns exploration switch.

switch again to the rows exploration in direction ↑ 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.

Special cases in $4\times *$ *grids.* For the cases where the number of lines or the number of columns is 4, we add the rules presented in Fig. 16 to the rules of Algorithm A_1. It is important to notice that these rules do not interfere with the previous rules and the execution of the algorithm in larger grids. An example of execution can be found in Fig. 17. A similar execution can be done for any grid of size $4 \times m$ or $n \times 4$, as soon as the robots are placed in a similar initial configuration.

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

Proof. First observe that our algorithm is well-defined. In the remaining of the proof we only consider a synchronous scheduler, first to prove that it is synchronous-sequential and then that it solves the problem under a synchronous scheduler. Hence, by Theorem 1, this covers the case of an asynchronous scheduler as well.

The proof is by induction on $n \times m$ where n is the number of lines and m is the number of columns of the grid. The base case 5×5 has been validated thanks to our simulator.[4] Indeed, A_1 being well-defined, there is a single synchronous execution starting from a

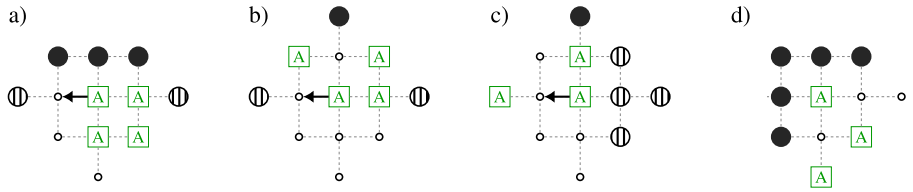


Fig. 16. Rules to explore $4\times *$ and $* \times 4$ grids.

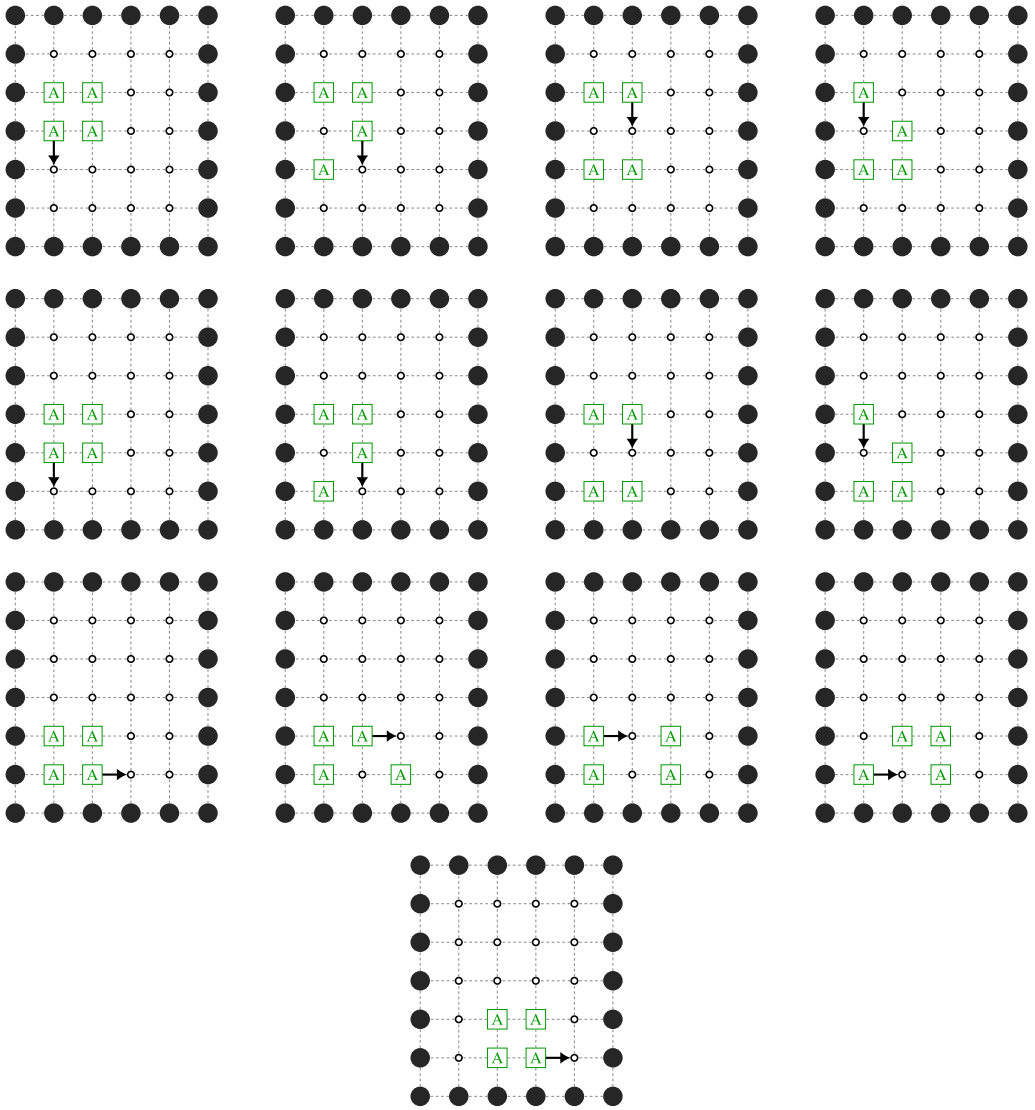


Fig. 17. Sequence of configurations during the exploration of a 6×4 grid.

given initial configuration. So the solvability can be checked by simply applying the rules by hand (or with the help of a computer). Moreover, one can easily check that at each round, only a single robot is enabled (and it is only move-enabled).

Let us assume that A_1 solves the PGE problem in all grids of size $x \times y$ with $5 \leq x \leq n$ and $5 \leq y \leq m$ for some values $n, m \geq 5$. We prove in the following that A_1 solves the PGE problem in grids of size $(n+1) \times m$ and $n \times (m+1)$. First focus on the case in which the size of the grid is $(n+1) \times m$. Given the size of the grid and the visibility range of the robots, a robot cannot detect the size of the grid and apply the algorithm as if they were in an $n \times m$ grid. As in smaller grids, the robots explore the lines one by one. The fact that there is one more line just means that the exploration of a line is repeated one more time until the robots execute the sequence

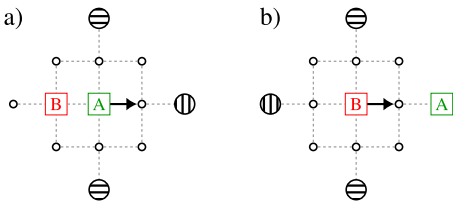


Fig. 18. Rules to move straight.

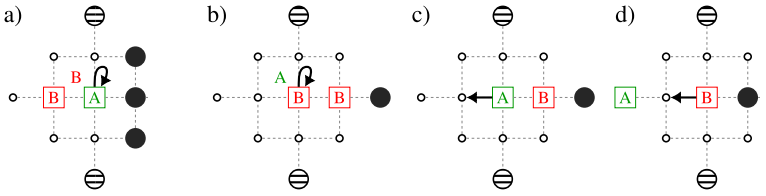


Fig. 19. Rules to turn around on the same row.

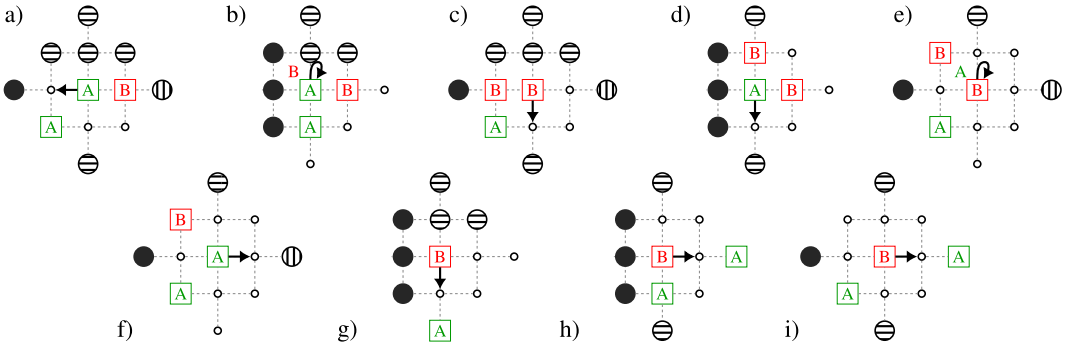


Fig. 20. Rules to perform a turn and a change of row.

illustrated in Fig. 15 to switch from exploring line to exploring columns. When exploring a column, again, the group of explorers cannot detect the length of the column and hence, explores the column and comes back to the sentinel as if they were in a $n \times m$ grid. The process repeats until the robots have explored all the columns and so on until the grid is fully explored and they have returned to the initial configuration. The case in which the size of the grid is $n \times (m+1)$ is similar as robots do not differentiate between lines and columns and hence, the same arguments apply. □

4.2. Algorithm with three 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 is located on the next row to be visited. 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. 18: 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. 19. The leader changes its color to B, then the follower gets Color A. Once their roles are switched, they start moving along the row in the opposite direction.

When they return to the first wall, the robots, along with the sentinel, execute the rules of Fig. 20 to (1) shift one row in the direction pointed by the sentinel and (2) switch the roles of the leader and the follower. The sequence of configurations during this process is illustrated in Fig. 21. After changing rows, the new leader and follower start to explore the new row as previously.

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

[5] An interactive simulation of A_2 is given at https://robots.app.bramas.fr/?SSS2024/A2.

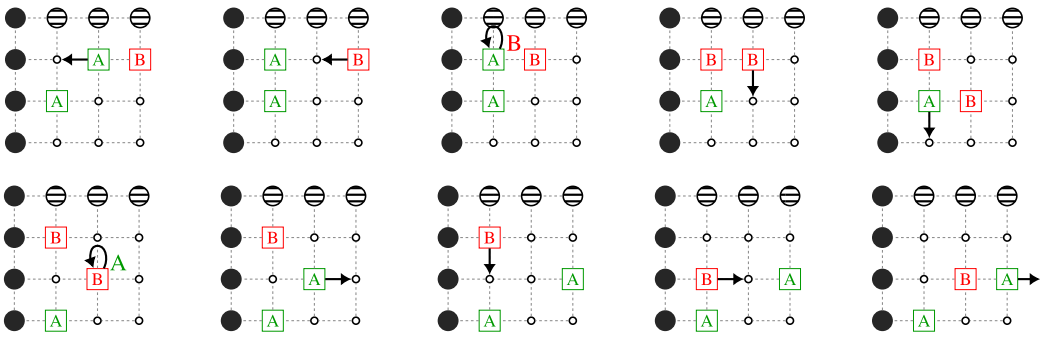


Fig. 21. Sequence of configurations during a change of row.

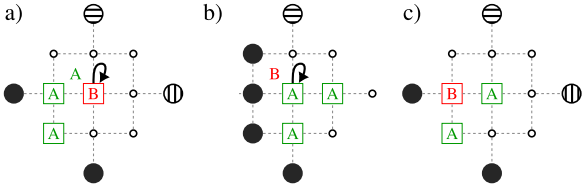


Fig. 22. Rules to perform a quarter turn and change the direction of exploration on the last row.

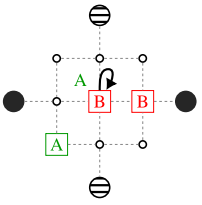


Fig. 23. Additional rule for 3×3 grids.

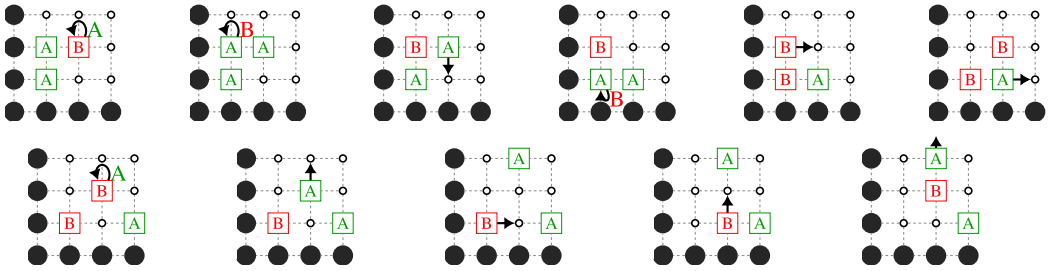


Fig. 24. Sequence of configurations during a quarter turn when robots are on the last row and change the direction of exploration.

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. 22 and the sequence of configurations reached during this process is given in Fig. 24. 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. 23. 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), which is adjacent to the third robot with color B (the future follower); as shown in Fig. 25. 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. 26. 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

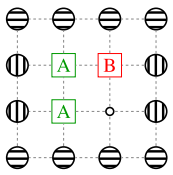


Fig. 25. Locally-defined initial configurations of A_2.

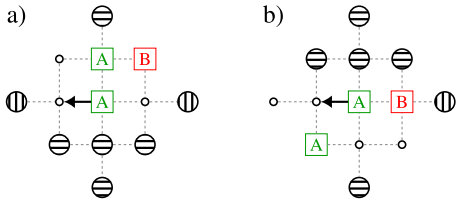


Fig. 26. Rules to reach a wall from the initial configuration.

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 actual exploration begins.

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

Proof. The proof is similar to the one of Algorithm A_1.

First observe that our algorithm is well-defined. In the remaining of the proof we only consider a synchronous scheduler, first to prove that it is synchronous-sequential and then that it solves the problem under a synchronous scheduler. Hence, by Theorem 1, this covers the case of an asynchronous scheduler as well.

The proof is by induction on $n \times m$ where n is the number of lines and m is the number of columns of the grid. Again, the base cases 3×3, 3×4, 4×3, and 4×4 have been validated thanks to our simulator.[5]

First, let us consider the special case of grids with one side of size 3, *i.e.*, $x \times 3$ or $y \times 3$ with $x, y \geq 5$. Let us assume that A_2 solves the PGE problem in all grids of size $x \times 3$ and $3 \times y$ with $4 \leq x \leq n$ and $4 \leq y \leq m$ for some values of $n, m \geq 4$. We prove that A_2 solves the PGE problem in grids of size $(n+1) \times 3$ and $3 \times (m+1)$. We focus first on the grid of size $(n+1) \times 3$. Without loss of generality, let assume that the robots start their exploration going on a line of size 3. Given the size of the grid and the visibility range of the robots, a robot cannot detect the size of the wider side. Thus, the robots apply the algorithm as if they were in a grid of size $n \times 3$ and explore the lines one by one. The fact that there is one more line just means that the exploration of a line is repeated one more time until the robots execute the sequence illustrated in Fig. 24 to switch from exploring line to exploring columns. When exploring a column, again, the group of explorers cannot detect the length so it explores the column and comes back to the sentinel as if they were in a grid of size $n \times 3$. The process repeats until the robots have explored all the columns and so on until the grid is fully explored and they have returned to the initial configuration. The case in which the size of the grid is $3 \times (m+1)$ is similar as robots do not differentiate between lines and columns and hence, the same arguments apply.

Now lets consider the more general case of grids of size $x \times y$ with $4 \leq x \leq n$ and $4 \leq y \leq m$ for some values $n, m \geq 4$. Let us assume that A_2 solves the PGE problem in all grids of size $x \times y$ with $4 \leq x \leq n$ and $4 \leq y \leq m$ for some values of $n, m \geq 4$. We prove that A_2 solves the PGE problem in grids of size $(n+1) \times m$ and $n \times (m+1)$ using similar arguments than in the previous case. □

4.3. Algorithm with three oblivious robots under visibility range three 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. 27. 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.

When exploring a row, the leader and its followers move straight from one wall to the other using the rules of Fig. 28. 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. 29. 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).

[6] An interactive simulation of A_3 is given at https://robots.app.bramas.fr/?SSS2024/A3.

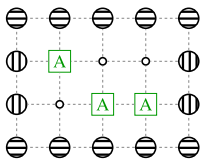


Fig. 27. Locally-defined initial configuration of A_3.

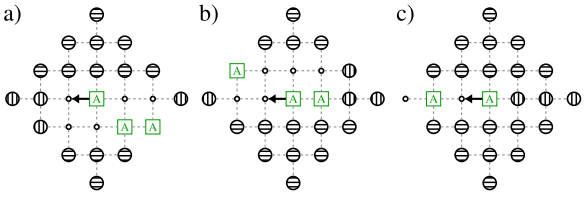


Fig. 28. Rules to move in a straight line.

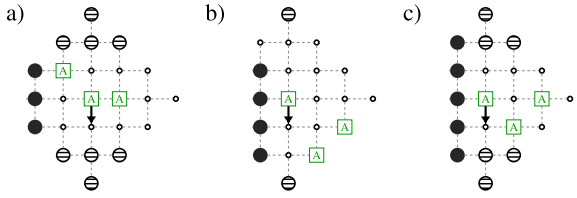


Fig. 29. Rules to move to the next row.

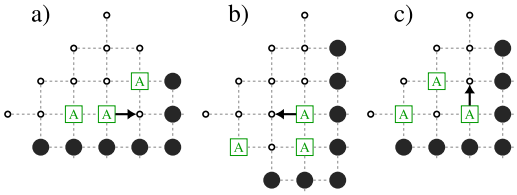


Fig. 30. Rules to turn in a corner.

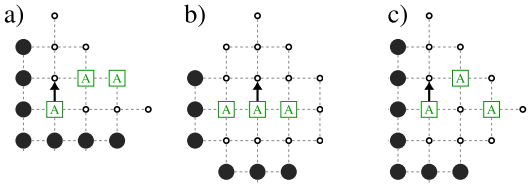


Fig. 31. Rules to turn after exploring the first row.

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. 30. 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. 31.

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

Proof. The proof is similar to the one of Algorithm A_1.

Again, the algorithm is well-defined. In the remaining of the proof we only consider a synchronous scheduler, first to prove that it is synchronous-sequential and then that it solves the problem under a synchronous scheduler. Hence, by Theorem 1, this covers the case of an asynchronous scheduler as well.

The proof is by induction on $n \times m$ where n is the number of lines and m is the number of columns of the grid. By simulation,[6] the base cases 6×6, 6×7, 7×6, and 7×7 have been checked.

Let us assume that A_3 solves the PGE problem in all grids of size $x \times y$ with $6 \leq x \leq n$ and $6 \leq y \leq m$ for some values $n, m \geq 7$. We prove in the following that A_3 solves the PGE problem in grids of size $(n+1) \times m$ and $n \times (m+1)$. Consider first the case in which the size of the grid is $(n+1) \times m$. Since the robots cannot detect the size of the grid (given its size and the visibility range of the robots), they apply the algorithm as if they were in an $n \times m$ grid: they explore lines on by one moving until they reach a wall where they switch to the next line that is explored in the opposite direction. Adding one column to grid just increases the number of periodic movement they do to reach the wall.

Consider now the second case in which the size of the grid is $n \times (m+1)$. Since $m - 1 \geq 6$, by induction hypothesis, A_3 solves the PGE problem in a $n \times (m-1)$ grid. Now, notice that between two explorations of a line in the same direction, the robots are two lines above. For example, if the robots move from the left wall to the right wall with the leader on line $1 \leq \ell \leq m-4$ and the two followers on line $\ell + 1$, then, the next time they explore from the left wall to the right wall, they are on line $\ell + 2$ and $\ell + 3$ respectively, and they repeat the exploration. So, compared to the execution in the $n \times (m-1)$ grid, adding two rows only implies that the robots repeat this periodic movement one more time. The remaining of the execution is indistinguishable for the robots. □

5. Conclusion

We have presented three grid exploration algorithms for swarms of luminous robots. Depending on the number of robots, 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, *e.g.*, 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.

CRediT authorship contribution statement

Quentin Bramas: Writing – review & editing, Writing – original draft, Validation, Investigation, Formal analysis, Conceptualization; **Stéphane Devismes:** Writing – review & editing, Writing – original draft, Validation, Methodology, Formal analysis, Conceptualization; **Anaïs Durand:** Writing – review & editing, Writing – original draft, Validation, Methodology, Formal analysis, Conceptualization; **Pascal Lafourcade:** Writing – review & editing, Writing – original draft, Validation, Investigation, Formal analysis, Conceptualization; **Anissa Lamani:** Writing – review & editing, Writing – original draft, Validation, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] I. Suzuki, M. Yamashita, Distributed anonymous mobile robots, in: SIROCCO, 1996, pp. 313–330.
[2] P. Flocchini, D. Ilcinkas, A. Pelc, N. Santoro, How many oblivious robots can explore a line, Inf. Process. Lett. 111 (20) (2011) 1027–1031.
[3] L. Blin, A. Milani, M. Potop-Butucaru, S. Tixeuil, Exclusive perpetual ring exploration without chirality, in: DISC 2010, Vol. 6343, 2010, pp. 312–327.
[4] P. Flocchini, D. Ilcinkas, A. Pelc, N. Santoro, Computing without communicating: ring exploration by asynchronous oblivious robots, Algorithmica 65 (3) (2013) 562–583.
[5] F. Ooshita, S. Tixeuil, Ring exploration with myopic luminous robots, in: SSS 2018, 2018, pp. 301–316.
[6] S. Devismes, A. Lamani, F. Petit, S. Tixeuil, Optimal torus exploration by oblivious robots, Computing 101 (9) (2019) 1241–1264.
[7] F. Bonnet, A. Milani, M. Potop-Butucaru, S. Tixeuil, Asynchronous exclusive perpetual grid exploration without sense of direction, in: OPODIS 2011, 2011, pp. 251–265.
[8] S. Devismes, A. Lamani, F. Petit, P. Raymond, S. Tixeuil, Terminating exploration of a grid by an optimal number of asynchronous oblivious robots, Comput. J. 64 (1) (2021) 132–154. https://doi.org/10.1093/comjnl/bxz166
[9] Q. Bramas, S. Devismes, A. Durand, P. Lafourcade, A. Lamani, Beedroids: how luminous autonomous swam of UAVs can save the world?, in: FUN 2022, 2022, pp. 7:1–7:21.
[10] P. Flocchini, D. Ilcinkas, A. Pelc, N. Santoro, Remembering without memory: tree exploration by asynchronous oblivious robots, Theor. Comput. Sci. 411 (14-15) (2010) 1583–1598.
[11] A. Rauch, Q. Bramas, S. Devismes, P. Lafourcade, A. Lamani, Optimal exclusive perpetual grid exploration by luminous myopic robots without common chirality, in: NETYS, 2021, pp. 95–110.
[12] Q. Bramas, P. Lafourcade, S. Devismes, Optimal exclusive perpetual grid exploration by luminous myopic opaque robots with common chirality, Theor. Comput. Sci. (2023) 114162. https://doi.org/10.1016/j.tcs.2023.114162
[13] S. Devismes, F. Petit, S. Tixeuil, Optimal probabilistic ring exploration by semi-synchronous oblivious robots, Theor. Comput. Sci. 498 (2013) 10–27.
[14] A.K. Datta, A. Lamani, L.L. Larmore, F. Petit, Enabling ring exploration with myopic oblivious robots, in: IPDPS 2015, 2015, pp. 490–499.
[15] S. Nagahama, F. Ooshita, M. Inoue, Terminating grid exploration with myopic luminous robots, Int. J. Netw. Comput. 12 (1) (2022) 73–102.
[16] A.K. Datta, A. Lamani, L.L. Larmore, F. Petit, Ring exploration by oblivious agents with local vision, in: ICDCS 2013, 2013, pp. 347–356.
[17] S. Cicerone, G.D. Stefano, A. Navarra, Solving the pattern formation by mobile robots with chirality, IEEE Access 9 (2021) 88177–88204.
[18] S. Kamei, S. Tixeuil, An asynchronous maximum independent set algorithm by myopic luminous robots on grids, Comput. J. 67 (1) (2024) 57–77. https://doi.org/10.1093/COMJNL/BXAC158

[19] T. Balabonski, P. Courtieu, R. Pelle, L. Rieg, S. Tixeuil, X. Urbain, Continuous vs. discrete asynchronous moves: a certified approach for mobile robots, in: M.F. Atig, A.A. Schwarzmann (Eds.), Networked Systems, Springer International Publishing, Cham, 2019, pp. 93–109.
[20] Q. Bramas, S. Devismes, P. Lafourcade, Finding water on poleless using melomaniac myopic chameleon robots, in: FUN 2020, 2020, pp. 6:1–6:19.