Optimal Exclusive Perpetual Grid Exploration by Luminous Myopic Opaque Robots with Common Chirality

Quentin Bramas
Bramas@unistra.fr
ICUBE, University of Strasbourg
CNRS, France

Pascal Lafourcade
Pascal.Lafourcade@uca.fr
LIMOS, Univ. Clermont Auvergne
Aubière, France

Stéphane Devismes
Stephane.Devismes@univ-grenoble-alpes.fr
Université Grenoble Alpes, VERIMAG

ABSTRACT

We consider swarms of luminous myopic opaque robots that run in synchronous Look-Compute-Move cycles. These robots have no global compass, but agree on a common chirality. In this context, we propose optimal solutions to the perpetual exploration of a finite grid. Precisely, we investigate optimality in terms of the visibility range, number of robots, and number of colors. In more detail, under the optimal visibility range one, we give an algorithm which is optimal w.r.t. both the number of robots and colors: it uses two robots and three colors. Under visibility two, we design two algorithms: the first one uses three robots with an optimal number of colors (*i.e.*, one), the second one achieves the best trade-off between the number of robots and colors, *i.e.*, it uses two robots and two colors.

KEYWORDS

Luminous myopic robots, perpetual exploration, finite grid, chirality, opacity, exclusiveness.

ACM Reference Format:
Quentin Bramas, Pascal Lafourcade, and Stéphane Devismes. 2018. Optimal Exclusive Perpetual Grid Exploration by Luminous Myopic Opaque Robots with Common Chirality. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

We consider swarms of autonomous robots endowed with visibility sensors, motion actuators, and lights of different colors. These so-called *luminous* robots [16] run in *synchronous* Look-Compute-Move cycles, where they first sense the environment (Look), then choose a destination and update their light color (Compute), and finally move to the chosen destination (Move).

We deal with luminous robots having very low capabilities. First, they are *myopic*, *i.e.*, they are only able to sense their surroundings within a limited visibility range. Then, they are *opaque*. Opacity means that a robot is able to see another robot if and only if no other robot lies in the line segment joining them. Furthermore, they have *no global compass*, *i.e.*, they agree neither on a North-South, nor a East-West direction. Finally, except from their lights, robots have neither persistent memories nor communication capabilities.

This study was partially supported by the French ANR projects ANR-16-CE40-0023 (DESCARTES) and ANR-16 CE25-0009-03 (ESTATE).

Woodstock '18, June 03–05, 2018, Woodstock, NY
2018. ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

However, they all agree on a *common chirality*, *i.e.*, when a robot is located on an axis of symmetry in its surroundings, it is able to distinguish its two sides, one from another.

We are interested in coordinating such weak robots, endowed with both typically small visibility range and few light colors (only a constant number of them), to solve an infinite global task called the *perpetual exploration*. In this problem, space is partitioned into locations, *e.g.*, rooms in a building, that should be visited by robots. Here, we consider a finite discrete space which is conveniently represented as a graph, where nodes represent locations and edges represent the possibility for a robot to move from one location to another, *e.g.*, through a door or a corridor. In this context, the *perpetual exploration* requires each node to be visited infinitely often by at least one robot. The direct application of such an exploration is, of course, patrolling in a zoned (maybe hazardous) area.

In this paper, we investigate optimal *exclusive* solutions to the perpetual exploration of a *finite grid*, where exclusiveness [1] requires any two robots to never simultaneously occupy the same position nor traverse the same edge.

Related Work. The robots we consider are known as *luminous robots* in the literature. They have been introduced by Peleg [16].

Up to now, exploration tasks have been considered in various topologies, *e.g.*, lines [13], rings [2, 7, 10, 14, 15], trees [12], torus [9], finite [3, 8] and infinite grids [4, 5].

In the context of finite graphs, two main variants, respectively called the *terminating* and *perpetual* exploration, have been considered. The terminating exploration requires every possible location to be eventually visited by at least one robot, with the additional constraint that all robots stop moving after task completion. In contrast, the perpetual exploration requires each location to be visited infinitely often by all or a part of robots. Terminating exploration has been tackled in [7–10, 12–14] while [2, 3] deal with the perpetual exploration problem. Notice that Ooshita and Tixeuil consider the two variants of the problem in [15].

In contrast with the present paper, a large part of the literature is devoted to "non-myopic" robots, *i.e.*, robots with an unbounded visibility range, meaning that the snapshot of each robot captures in the whole system configuration; see [2, 3, 8–10, 12–14]. In such a context, robots are always assumed to be anonymous and oblivious, *i.e.*, they have no state and cannot remember the past. Furthermore, opacity and chirality have never been considered in such settings.

Exploration algorithms satisfying exclusiveness are proposed in both finite [2, 3] and infinite graphs [4, 5].

However, only few works dedicated to discrete environment, *e.g.*, in (infinite) graphs [4], assume robots have a common chirality. Actually, up to now this property was rather considered in the 2D Euclidean plan; see *e.g.,* [11]. Now, the common chirality has an

Visibility	# Robots	# Colors	Algorithm
finite	1	finite	Impossible (Thm. 3.2)
finite	2	1	Impossible (Thm. 3.4)
1	2	2	Impossible (Thm. 3.5)
1	3	1	Impossible (Thm. 3.6)
1	**2**	**3**	Vone_3^2
2	**2**	**2**	Vtwo_2^2
2	**3**	**1**	Vtwo_1^3

Table 1: Summary of our results.

impact on the number of robots necessary to solve exploration: for example, with visibility range one and few colors ($O(1)$), five (resp. six) synchronous robots are necessary and sufficient to explore an infinite grid with (resp. without) the common chirality assumption [4, 5].

To the best of our knowledge, until now exploration with myopic robots was only addressed in finite rings [7, 15] or infinite grid [4, 5]. Hence, the present work is the first study of (perpetual) exploration with myopic (luminous) robots in finite grids.

Contribution. We investigate optimal solutions to the exclusive Perpetual Finite Grid Exploration (PFGE) in synchronous settings using luminous myopic opaque robots. Our contributions are summarized in Table 1 and can be split into tight lower bounds and algorithms matching those bounds.

Lower bounds: Under our settings, we establish several lower bounds. First, we show in Theorem 3.2 that one robot is not sufficient to solve the PFGE problem, whatever the visibility range is. Then, we prove in Theorem 3.4 that two robots with only one color are not able to solve the PFGE problem, whatever the visibility range is. When the visibility range is one, we demonstrate in Theorem 3.5 that it is impossible to solve the PFGE problem with only two robots and two colors, and in Theorem 3.5 that it is impossible to solve the PFGE problem with only three robots and one color. Notice that all these bounds hold even if exclusiveness is not required.

Optimal exclusive algorithms: According to the previous lower bounds, we propose three optimal algorithms denoted by Vone_2^3, Vtwo_2^2, and Vtwo_3^1, respectively. All these algorithms achieve exclusiveness. Under visibility range one, Vone_2^3 solves the PFGE problem using only two robots and three colors. Under visibility range two, Vtwo_2^2 solves the PFGE problem using two robots and two colors, while Vtwo_3^1 solves the problem with three oblivious robots (equivalently, three robots with only one color).

Notice also that we explain how to modify our non-oblivious solutions to obtain terminating exploration algorithms, with a constant overhead in terms of colors. Finally, we describe a simple transformation to make Vone_2^3 work in fully asynchronous setting, yet under visibility range two.

Roadmap. Section 2 is devoted to the computational model and definitions. In Section 3, we propose four impossibility results, giving our lower bounds. In Section 4, we describe an algorithm which is optimal not only w.r.t. the visibility range but the number of robots and colors. In Section 5, we give two solutions for visibility range two, the former uses three robots with an optimal number of colors (one), and the latter achieves the best trade-off between the number of robots (*i.e.*, two) and the number of colors (*i.e.*, two). We conclude with extensions and perspectives in Section 6.

2 MODEL

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

We assume discrete time and at each *round*, the robots synchronously perform a *Look-Compute-Move* cycle. In the *Look* phase, a robot 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. 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. Then, each robot *computes* a destination (either Up, Left, Down, Right or Idle) based only on the snapshot it received. Finally, it *moves* towards its computed destination. We also assume that robots are *opaque*, *i.e.*, they obstruct the visibility in such way that if three robots are aligned, the two extremities cannot see each other.

We forbid any two robots to occupy the same node simultaneously. A node is *occupied* when a robot is located at this node, otherwise it is *empty*. Robots may have *lights* with different colors that can be seen by robots within distance Φ from them. We denote by Cl the set of all possible colors.

The *state* of a node is either the color of the light of the robot located at this node, if it is occupied, or $\perp$ otherwise. In the Look phase, the snapshot includes the state of the nodes (within distance Φ). During the compute phase, a robot may decide to change the color of its light.

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 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 contains the rotations of angle 0 (to have the identity), $\pi/2$, π and $3\pi/2$, centered at r. Moreover, since robots may obstruct visibility, the function that removes the state of a node u if there is another robot between u and r is *systematically* applied to obtain the local view. Finally, we assume that robots are *self-inconsistent*, 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 local views. When the robots are in Configuration C, a configuration C' obtained after one round satisfies: for all $((i, j), c) \in C'$, there exists a robot in C with color $c' \in Cl$ and a transformation $f \in \mathcal{IT}$ such that one of the following conditions holds:

- $((i, j), c') \in C$ and $f^{-1}(T(f(V_C(i, j)))) = (Idle, c)$,
- $((i-1, j), c') \in C$ and $f^{-1}(T(f(V_C(i-1, j)))) = (Right, c)$,
- $((i+1, j), c') \in C$ and $f^{-1}(T(f(V_C(i+1, j)))) = (Left, c)$,
- $((i, j-1), c') \in C$ and $f^{-1}(T(f(V_C(i, j-1)))) = (Up, c)$, or
- $((i, j+1), c') \in C$ and $f^{-1}(T(f(V_C(i, j+1)))) = (Down, c)$.

We denote by $C \mapsto C'$ the fact that C' can be reached in one round from C (*n.b.*, $\mapsto$ is then a binary relation over configurations). An execution of Algorithm A in a grid G is then a sequence $(C_i)_{i\in\mathbb{N}}$ of configurations such that $C_0 \in Init(G)$ and $\forall i \geq 0, C_i \mapsto C_{i+1}$.

Perpetual Finite Grid Exploration. An execution $(C_i)_{i\in\mathbb{N}}$ in a grid $G = (V, E)$ *achieves the Perpetual Finite Grid Exploration* (PFGE) 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 the *Perpetual Finite Grid Exploration* (PFGE) 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 of A in G starting from C_0 that achieves the PFGE.

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. We notice that an execution of a well-defined algorithm is uniquely determined by its initial configuration.

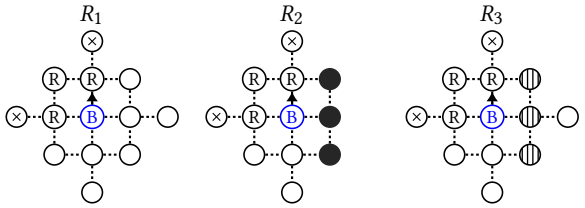

Figure 1: Examples of rules.

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, one on the left, and one in diagonal, then the blue robot is dictated to move Up. Notice that, due to the opacity of the robots, the state of the topmost node, and the leftmost node, are not accessible. This is indicated by a cross inside the node (to help the reader only). By extension the same rule applied if the view is rotated by π, but in that case, the destination would be Down.

In the same figure, rule 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. Notice that a wall is never hidden behind a robot (due to the opacity) as it simply represents the absence of a node. In our algorithms, we often define similar rules 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 represent several rules in just one picture. For example, rule 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, rule R_3 in Fig. 1 is a concise representation of rules R_1 and R_2.

For a more complex example, refer to the first rule in Fig. 13, which represents a set of 3^3 rules, as there are 3 potential walls (top, right, bottom), and the rule applies whether those walls are at distance 1, 2 or more (*i.e.*, not visible).

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, we propose algorithms with *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, every two possible initial configurations are equal up to a translation applied to all robot positions and for a given grid, the set of all possible initial configurations is closed by such translations.

3 IMPOSSIBILITY RESULTS

In this section we prove our four impossibility results; see Table 1 for a summary. Due to the lack of space, several proofs have been moved in the appendix. The technical lemma below is trivial yet very useful since it allows us to consider initial configurations where one robot has an arbitrary position, without the loss of generality.

Lemma 3.1. *Let $\mathsf{A} = (Cl, Init, T)$ be an algorithm that solves the PFGE problem using n robots. For every grid G with at least n lines and n columns, for every node u in G, there exist a configuration C where a robot is located at u and an algorithm $\mathsf{A}' = (Cl, Init', T)$ where $Init'$ is identical to Init unless $C \in Init'(G)$ such that A' solves the PFGE problem using n robots.*

When there is a single robot that is far enough not to see any wall (using the previous lemma, one can assume that the robot is initially in a center of a large enough grid), then, due to to the self-inconsistency assumption, there exists a possible execution where the robot visits at most two adjacent nodes, hence follows.

Theorem 3.2. *The PFGE problem is not solvable with only one robot, for any finite visibility range.*

Proof. Assume, by the contradiction, that A solves the PFGE problem with one robot. Let $\Phi > 0$ be the visibility range of the robot. Consider a grid G of size greater than $(2\Phi + 3) \times (2\Phi + 3)$. By Lemma 3.1, we can assume, without the loss of generality, that A solves the problem starting from the initial configuration where the unique robot is at a center of G. Thus, the robot sees no wall. If the algorithm dictates the robot to stay idle, then the robot will stay idle forever, a contradiction. So, the robot moves toward one direction in the globally oriented view, say $d \in \{Up, Down, Left, Right\}$. Now, after the move, the robot is in the same situation: it sees no wall. So, it will again move and the chosen destination in its local view will be the same. However, since the robot is self-inconsistent and the four possible destinations look identical, there is a transformation $f \in \mathcal{IT}$ such that the destination of the robot in the globally oriented view is actually the opposite of d. Hence, there is a possible execution where the robot starts at a center of the grid and forever alternates between two positions. The perpetual exploration is not achieved in this execution, a contradiction. □

Remark 1. *Consider any algorithm solving the PFGE problem with two robots and any configuration where the two robots have the same color and do not see the border of the grid. In this situation, there exist two transformations (*e.g.*, the identity transformation and the π-rotation), one for each robot, that make their local views identical. In this case and even if the algorithm is not well-defined, whenever one decides to move, the other moves too and the two robots move in opposite direction in the global view,* e.g.*, if one moves Up in the global view, the other moves Down. Conversely, if one stays idle, the other stays idle too. Moreover, if one robot decides to change its color, the other also switches to the same color.*

When there are only two robots, the following lemma states that, in many large grids, there exists an execution where the two robots eventually see each other, while one of them is located at a center. This is useful to prove the next two theorems, involving only two robots.

Lemma 3.3. *Consider any algorithm* A *that solves the PFGE problem with two robots and a grid G of size at least $(4\Phi + 6) \times (4\Phi + 6)$ whose number of lines or columns is even.*

Then, there exists an execution of A *in G that contains a configuration where the two robots see each other and one of them is located at a center of G.*

Proof. We proceed by the contradiction. Remark first that G contains at least two centers, c_1 and c_2, which are neighbors, by definition. Then, by Lemma 3.1, we can consider, without loss of generality, that A solves the PFGE problem starting from an initial configuration C where a robot r is located at one of the two centers of G. Let r' be the other robot. Now, each time r is located at c_1 or c_2, r does not see any wall by construction of G and does not see r' by the contradiction. So, in such a case if r decides to move, all destinations look identical and there is a transformation that makes it move from c_1 to c_2 if located at c_1, and from c_2 to c_1 otherwise. Hence, there is a possible execution where r visits at most c_1 and c_2 and never sees r'. Consider now such an execution.

Let u_1 be a node at distance $\Phi + 1$ from c_1 and distance $\Phi + 2$ from c_2. By construction of G, u_1 is at distance at least $\Phi + 2$ from a wall. Moreover, u_1 is eventually visited because A solves the PFGE problem. By construction, u_1 is necessarily visited first by r'. Moreover, r stays at c_1 or c_2 and r' is never in the visibility range of r meanwhile. When the first visit of u_1 by r' occurs, robots neither see each other nor any wall. Moreover, u_1 as a neighboring node, say u_2, at distance $\Phi + 1$ from a wall, distance $\Phi + 2$ from c_1, and distance $\Phi + 3$ from c_2. From that point, there exist transformations such that every time r' decides to move, the applied transformation makes r' alternatively moves from u_1 to u_2, and vice versa, while r still alternates between c_1 and c_2. Hence, from that point, there exists a possible execution suffix where all nodes, except c_1, c_2, u_1, and u_2, are never more visited, a contradiction. □

Theorem 3.4. *The PFGE problem is not solvable with two robots and only one color, for any finite visibility range.*

Proof. Assume, by the contradiction, that an algorithm A solves the PFGE problem with two robots and one color. Consider a grid of size at least $(4\Phi+6)\times(4\Phi+6)$ whose number of lines or columns is even. By Lemmas 3.1 and 3.3, we can assume, without the loss of generality, that A solves the PFGE problem starting from an initial configuration C where a robot r is located at a center of the grid and the other is located within distance Φ from r. Then, due to the size of the grid (at least $(4\Phi + 6) \times (4\Phi + 6)$), no robot sees a wall in this initial configuration.

Consider now an execution, starting from C, where every time the robots see each other without seeing a wall, the applied transformations are those that make the local views of the two robots identical. Since initially both robots see each other and one of them is in a center of the grid (*i.e.*, at distance at least $2\Phi + 3$ from any wall), the other robot is at distance at least $\Phi + 3$ from any wall.

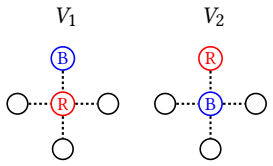

Figure 2: The two views of two robots with colors R and B.

Hence, initially their local views are the same and their destinations in the globally oriented view are opposite; see Remark 1 (of course, they cannot stay idle). Thus, in that execution, the middle of the two robots remains constant at least until one robots sees a wall. More formally, if robots respectively have coordinates (x, y) and (x', y'), then the two real numbers $M_x = \frac{x+x'}{2}$ and $M_y = \frac{y+y'}{2}$ remain constant until at least one robot sees a wall.

Initially, the robot r located at a center, say at coordinates (x, y), is at distance at least $2\Phi + 3$ from any wall, *i.e.*, $x \geq 2\Phi + 3$ and $y \geq 2\Phi + 3$. The other robot being within distance Φ from r, so $|x - x'| \leq \Phi$ and $|y - y'| \leq \Phi$. Thus, we have $x' \geq \Phi + 3$, $y' \geq \Phi + 3$, $M_x > \frac{3}{2}\Phi + 3$, and $M_y > \frac{3}{2}\Phi + 3$.

Consider now the first visited node which is at distance Φ+2 from a wall. Without loss of generality, assume that one of the closest wall is the left one, *i.e.*, its abscissa is $x = \Phi$+2. Since no robot has yet see a wall, we still have $M_x > \frac{3}{2}\Phi + 3$. By replacing x by its current value, we obtain: $\frac{\Phi+2+x'}{2} > \frac{3}{2}\Phi + 3 \Leftrightarrow x' > 2\Phi + 4 \Leftrightarrow x' - x > \Phi + 2$. Hence, the two robots neither see each other, nor a wall. From that point, using the same argument as in Theorem 3.2, we can construct a possible execution suffix where each robot either stays idle or alternates between two nodes. Now, in that suffix, there are many nodes that are never visited, a contradiction. □

In the following, we say that a robot is *isolated* whenever no other robot is within its visibility range.

Theorem 3.5. *The PFGE problem is not solvable with two robots, two colors and visibility range one.*

Proof. Assume by the contradiction that an algorithm A solves the PFGE problem in this setting. Consider a grid G of size at least 10×10 whose number of lines or columns is even. By Lemmas 3.1 and 3.3 (recall that $\Phi = 1$), we can assume, without the loss of generality, that A solves the PFGE problem starting from an initial configuration C where a robot r is located at a center of the grid and the other is located at the neighboring node. Moreover they have different colors, since otherwise we have a contradiction using the same argument as in the proof of Theorem 3.4. Indeed, the argument remains valid even if robot can change their color because robots with same views modify their color in the same way.

Let's say that in C, a robot is blue (B) and the other is red (R). We can choose an execution and the transformations such that, every time a red robot sees a blue robot, its local view is V_1 in Fig. 2 and every time a blue robot sees a red robot, its local view is V_2, in Fig. 2.

A outputs a single destination d_1, resp. d_2, for the local view V_1, resp. local view V_2. Moreover, in C, robots initially have local views V_1 and V_2. After one round, robots are either isolated, they have the same color, or their views are V_1 and V_2 respectively (*n.b.*, the two robots still do not see the border of G in this latter case). If robots are isolated, we can construct a possible execution where robots remain isolated forever and, depending on the algorithm, they either stay idle or alternate between two positions forever. Hence, in that case, the perpetual exploration is not achieved, a contradiction. If now the robots have the same color after one round, then depending on the algorithm, we can construct an execution where they either stay idle or swap their position forever while maintaining their colors identical, or they become isolated after the second round. In the latter case, they still do not see the border of G, so we can make the perpetual exploration fails, as previously explained.

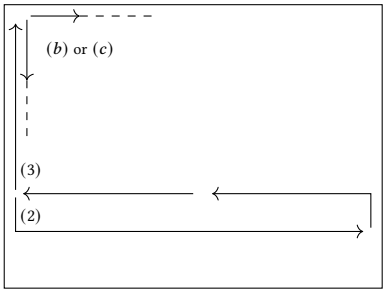

Figure 3: The movement leading to a contradiction.

Hence, the only remaining case is the one where their views of the two robots are V_1 and V_2 after the first round. In this case, the movements of robots are periodic while they do not reach a wall. So, either robots alternate between two configurations, or the two robots move in a straight line until at least one robot reaches a wall.

In the former case we immediately obtain a contradiction. So, we focus now on the latter case: when a robots reaches a wall, there exists a constant S (independent of the size of the grid) such that the two 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 straight toward the opposite wall, or (3) remain at distance at most S from the wall until they reach a corner. Fig. 3 illustrates the last two cases. In Case (1), some nodes will not be visited if G is large enough, a contradiction. In Case (2), when the robots leave the wall they must be in the same relative positions as initially, rotated by angle π (to move to the opposite direction). Indeed, we have just seen that this is the only way the two robots can travel without seeing walls. Hence, when reaching the opposite wall, since they cannot distinguish between the two walls, the robots will perform the same turn, remains in a subgrid of size S for the same finite number of rounds, then leave the wall to move straight towards the first wall. All the movements are the same as the movement performed when reaching the first wall, but rotated by π. Hence, they will end up in the exact same initial position. The two robots will continue this periodic movement infinitely, leaving nodes unvisited if G 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 reach the 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 reaching 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. □

Theorem 3.6. *The PFGE problem is not solvable with three robots, one color, and visibility range one.*

Sketch of Proof. This theorem is proven by contradiction. Assuming there exists an algorithm solving the PFGE problem in these settings, we first show that, in large enough grids, there is a reachable configuration where the three robots are far from all walls. We then show that it is possible, within one or two rounds, to make one robot isolated while keeping all robots not seeing any wall (to prove this, there are only few cases to consider, up to rotations, with three robots, one colors, and visibility one). From this latter situation, we show that it is possible to make all robots isolated still while maintaining the walls out of the visibility range of all robots. Being isolated and seeing no wall, robots either stay idle, or there is an execution where each robot alternates between two adjacent nodes while maintaining them isolated and seeing no wall, a contradiction. □

4 VISIBILITY RANGE ONE: Vone_2^3

We present an algorithm, denoted by Vone_2^3, which assumes a visibility range one and uses two robots and three colors. By Theorem 3.2, Vone_2^3 is optimal w.r.t. the number of robots, and by Theorem 3.5, Vone_2^3 is also optimal w.r.t. the number of colors. An animation illustrating the behavior of Vone_2^3 is available online; see [6].

Initially, one robot, called the *follower*, has color F and the other robot, called the *leader*, has any of the two colors L and R. The initial positions of the two robots are arbitrary in the grid, except that they should be neighbors. Hence, the set of all possible initial configurations is locally-defined.

The follower is colored F all along the execution, while the light of the leader alternates between color R and color L. Moreover, the two robots always stay neighbors during the execution: we say that they form a *group* all along the execution. For the purpose of explanation only, we now consider an arbitrary orientation of the grid (*n.b.*, the robots do not know this orientation since they do not agree on a common North): the four walls are respectively at the top, bottom, left, and right in the grid.

For example, consider the initial configuration C where (1) the leader has color R, (2) the group of robots is placed along a line linking the left wall to the right one, and (3) the leader is closer from the left wall than the follower.

From C, the group first performs an *ascending phase*: the group moves straight from a wall to another (*n.b.*, from C, the group first moves toward the left wall), moreover the group moves Up and turns around every time it reaches a wall. Once the group reaches the top wall, it does the same but now moves Down every time it reaches a wall: it switches to a *descending phase*. So, the group visits all the nodes following a serpentine belt alternating forever between ascending and descending phases.

To achieve these principles, we use the relative positions of the two robots as well as the two colors of the leader to give a kind of direction. Actually, the follower always moves toward the leader without changing its color. In contrast, the leader always moves from its current position to one of its three free neighboring nodes. In some particular cases, the leader uses its color to make the choice. Overall, the leader takes all decisions on the behalf of the group.

For instance, if the robots do not see any wall, the leader moves away from the follower and the follower follows the leader; see the rules in Fig. 4. This makes the group move straight.

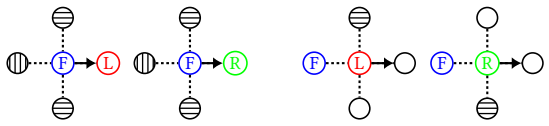

Figure 4: Moving straight.

Consider now the case where the leader reaches a wall during an ascending (resp. descending) phase and the leader is not on the line along the top (resp. bottom) wall. Then, the group should *turn* to reach the immediately upper (resp. lower) line. In this case, the color of the leader is used to decide the rotation sense of the turn. Precisely, a turn is performed in two rounds. Upon reaching the wall, the leader turns Right if it has color R, Left otherwise (*n.b.*, if the destination line is Up, the phase is ascending, otherwise it is descending). Moreover, at the second round of the turn, the leader changes its color (*i.e.*, it either switches from R to L or from L to R) while it leaves the wall. The corresponding rules are given in Fig. 5. Moreover, an illustrative example is proposed in Fig. 6.

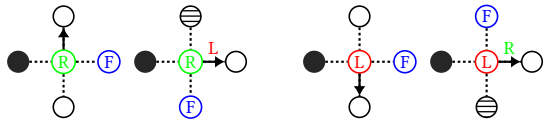

Figure 5: Turning in front of a wall for R and L leader.

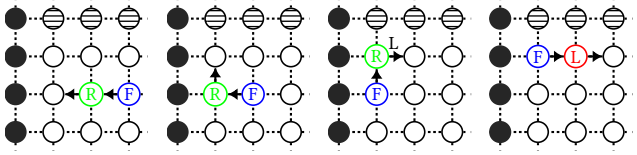

Figure 6: Sequence of configurations during a right turn in an ascending phase.

The last case to study is when the leader reaches a corner after that the group has traversed the line along the top (resp. bottom) wall in an ascending (resp. descending) phase. In this case, the leader detects that the group cannot further move Up (resp. Down). For example, the leader has color L but cannot turn Left because the top wall is at its Left. In this case, the leader moves away from the top wall and changes its color in the same step, *i.e.*, it moves Right and takes color R if its color was L, it moves Left and takes color L otherwise; see the rules in Fig. 7. Notice also that, after this turn, the leader again changes its color while leaving the wall, using the

previous rules given in Fig. 5. Overall, after turning at such a corner, the leader has changed its color twice, and consequently starts the descending (resp. ascending) phase. An example where the group switches from an ascending to a descending phase (resp. from a descending to an ascending phase) is given in Fig. 8 (resp. Fig. 9).

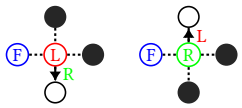

Figure 7: Moving at a corner.

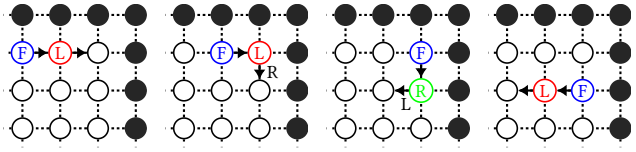

Figure 8: Sequence of configurations during a turn at a corner.

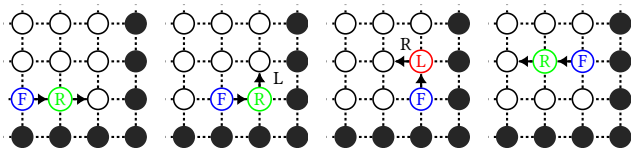

Figure 9: Sequence of configurations during a turn at another corner.

Theorem 4.1. Vone_3^2 *solves the PGFE problem with two robots and three colors.*

Proof. By induction on $m \times n$, where m is the number of lines and n is the number of columns of the grid. We have validated several base cases, including even and odd values for the number of lines, using our simulation tool [6]. Precisely, we have checked that Vone_3^2 solves the PGFE problem on every grid of size $m \times n$ for $m = 2, 3$ and $n = 2$. Such a checking is easy since, from a given initial configuration, there is only one possible execution (the algorithm is well-defined and the execution is synchronous). So, we just have to execute the algorithm until reaching an already encountered configuration: at that time all nodes should have been visited.

We assume now that Vone_3^2 solves the PGFE problem in all grids $x \times y$ with $2 \leq x \leq m$ and $2 \leq y \leq n$ for some values $m \geq 3$ and $n \geq 2$. We should show that Vone_3^2 solves the PGFE problem in the grids of size $m \times (n+1)$ and $(m+1) \times n$.

Consider first the grid of size $m \times (n+1)$. Then, it is easy to see that after adding one column, our algorithm still solves the PGFE problem. Indeed, when robots are traveling from the left to the right, they move in a straight line periodically until they reach a wall, so adding one column just increases by one the number of times they perform their periodic movement.

Consider now the grid of size $(m+1) \times n$. Since $m - 1 \geq 2$, the induction hypothesis applies to the case $(m-1) \times n$, *i.e.*, Vone_3^2 solves the PGFE problem in the grid $(m-1) \times n$. Now, we can remark that, after adding two lines, Vone_3^2 still solves the PGFE. Indeed, every time the robots travel horizontally from one wall to the other and then come back, they are located exactly two lines above their previous location when they left the wall the first time. For example, assume $(0,0)$ represents the node at the bottom left corner and consider the grid of size $(m+1) \times n$. If the configuration is $C_1 = \{((0,j),F),((1,j),L)\}$ at a given time, with $1 \leq j \leq m-1$, then it is $C_2 = \{((0,j+2),F),((1,j+2),L)\}$ after $2n$ rounds. So, the execution is similar from C_1 in the grid of size $(m-1) \times n$ and from C_2 in the grid of size $(m+1) \times n$, *i.e.*, adding two lines just implies that they perform this periodic movement one more time. □

5 VISIBILITY RANGE TWO : Vtwo_2^2 AND Vtwo_3^1

Under visibility range of two, we can reduce the number of colors to two still using two robots; see Subsection 5.1. We can also use only one color, yet with three robots; see Subsection 5.2.

5.1 Algorithm Vtwo_2^2

The principle of Algorithm Vtwo_2^2 is similar to that of Algorithm Vone_2^3. For example, initially, the leader robot, with color L, should be adjacent to a follower, with color F (*n.b.*, as a consequence, the set of all possible initial configurations is still locally-defined). However, instead of using two colors at the leader to distinguish the right and left wall, we exploit the distance between the leader and the follower. More precisely, when the leader reaches a wall while being adjacent to the follower, the two robots perform a left turn, as illustrated in Fig. 10. Conversely, if the leader is at distance two from the follower when reaching a wall, the two robots perform a right turn, as illustrated in Fig. 11.

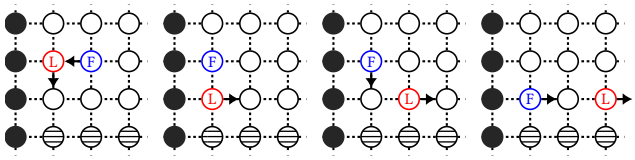

Figure 10: Sequence of configurations during a left turn. The same sequence occurs regardless of the presence of a wall at the hatched nodes.

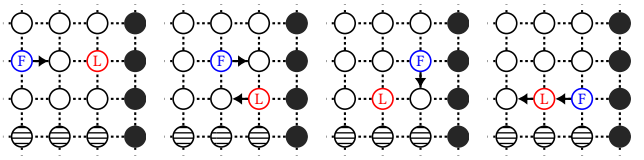

Figure 11: Sequence of configurations during a right turn. The same sequence occurs regardless of the presence of a wall at the hatched nodes.

When traveling from wall to wall, the two robots preserve their distance. The leader moves away from the follower, regardless its distance to the follower, the presence of walls at distance two, or the presence of a wall at distance one on a specific side, as illustrated in Fig. 12. The follower always follows the leader regardless of their distance and the presence of walls, as dictated by rules shown in Fig. 13. The only exception occurs when the follower is adjacent to the leader, and there is a wall adjacent on its right. In that case, the follower stays idle. This is done to separate the leader and the

follower during a left turn. We can see that in the second configuration in the sequence shown in Fig. 10, the follower is adjacent to the leader, but stays idle while the leader continues to move. After that, the leader and the follower are at distance two.

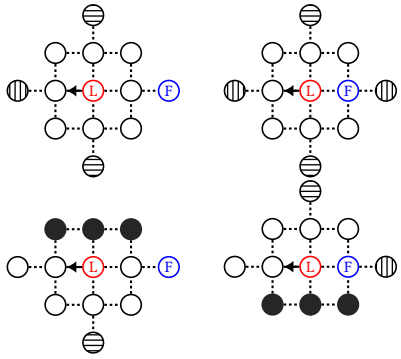

Figure 12: When it sees no walls, the leader moves away from the follower.

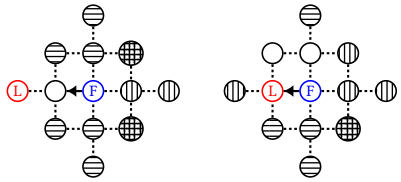

Figure 13: The follower always follows the leader, except when there is a wall immediately on its right.

The rules executed by the leader when reaching a wall are more specific and depend on its distance to the follower. When the leader reaches a wall and is at distance two (resp. distance one) from the leader, then it turns right (resp. turns left); as shown in Fig. 14.

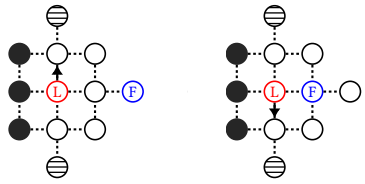

Figure 14: When reaching a wall, the leader either turns right or left depending on its distance with the follower.

If the leader has turned right, the rules shown in Fig. 15 dictate the follower to move towards the wall, while the leader moves away from the wall. Then, the follower moves along the wall towards the leader, and the leader does not move, so that the leader and the follower become adjacent. After that, the leader and the follower are in a configuration from which they move straight towards the opposite wall. The entire sequence is shown in Fig. 11.

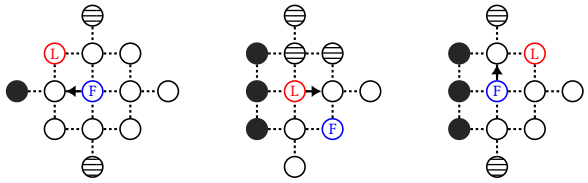

Figure 15: The rules to complete a right turn. After the turn the follower and the leader are adjacent.

If the leader has turned left when it reached the wall, the rules shown in Fig. 16 dictate the leader to move away from the wall while the follower stays idle. Then, the follower moves along the wall towards the leader while the leader moves away from the wall. After that the leader and the follower are at distance two. The entire sequence is shown in Fig. 10.

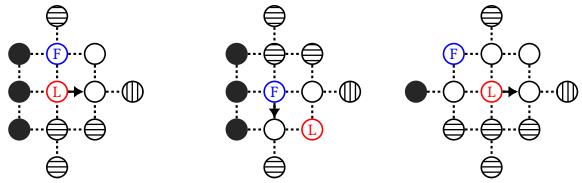

Figure 16: The rules to complete a left turn. After the turn the follower and the leader are at distance two.

The remaining rules apply when the robots reach a corner and the phase should change (from ascending to descending, or conversely). When robots are adjacent and move along a wall on their left side, they move while ignoring the wall until the leader sees a wall in front of it. When this happens, the first rule in Fig. 17 dictates the leader to change its color. After that, both robots have color *F*. Then, they perform a special move to (1) turn around in one round (the second and third rules in Fig. 17) by moving away from the wall, and (2) reverse the leader and follower roles. The complete sequence is illustrated in Fig. 18.

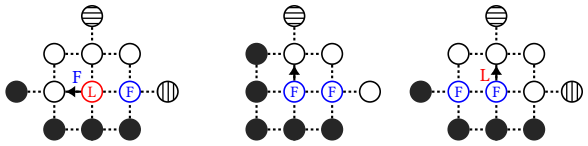

Figure 17: The rules to move along a wall when robots are adjacent, until reaching a corner and then performing a right turn by switching the leader and follower roles.

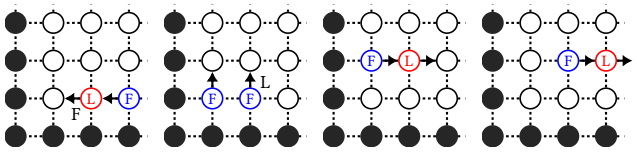

Figure 18: Sequence of configurations at a corner.

The same thing occurs in the opposite direction when robots are not adjacent, using the rules shown in Fig. 19.

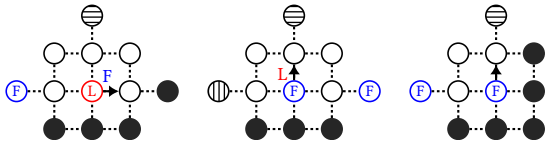

Figure 19: The rules to move along a wall when robots are not adjacent, until reaching a corner and then performing a left turn by switching the leader and follower roles.

Finally, there are three rules that are used in specific cases. The first rule in Fig. 20 applies only once in specific initial configurations.

The second rule in Fig. 20 is used when the number of columns or the number of lines is two. When this rule is executed, the robots will move along the wall and the other rules apply as usual. The last rule in Fig. 20 applies only in the grid of size 2×2. After this rule is executed, the leader moves to an empty node, while the follower moves to another empty node (by applying an existing rule shown in Fig. 16), hence after one round, the configuration is rotated by $\pi/2$ so that this occurs infinitely often while each node is visited every two rounds. Those special cases can be visualized in the animation in our complementary material [6].

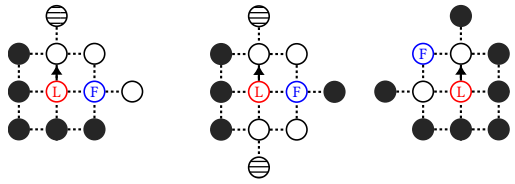

Figure 20: Three rules used in specific cases.

Theorem 5.1. Vtwo_2^2 *solves the PGFE problem with two robots, having two colors and visibility range two.*

Proof. The induction proof is similar to Theorem 4.1 for Vone_2^3. In particular, the base cases have been checked automatically using the same simulation tool [6]. □

5.2 Algorithm Vtwo_3^1

Our last algorithm, Algorithm Vtwo_3^1, uses three anonymous oblivious robots (*i.e.*, three robots with only one color) under visibility range two. The algorithm uses the same principle as the previous ones, but two shapes are used to distinguish what to do when reaching a wall. We respectively denote these two shapes by "<" and "L". We define the set of locally-defined initial configurations as the set of configurations where the robots form an "<" shape. Fig. 21 (resp. Fig. 22) shows the rules executed by the robots to move straight when in the "<" shape (resp. when in the "L" shape). With those rules, robots are able to move straight, regardless of the presence of wall "behind" them, or on their "side".

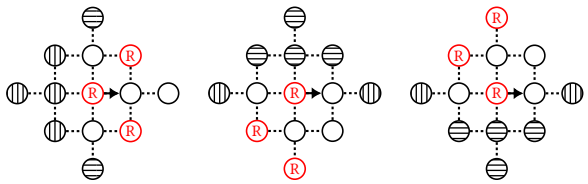

Figure 21: Rules to move straight for an "<" shape.

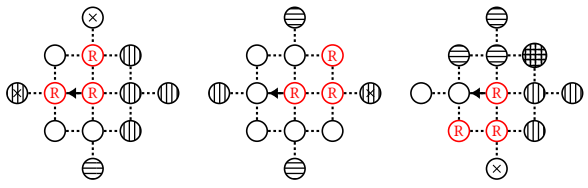

Figure 22: Rules to move straight for a "L" shape.

When a robot reaches a wall, the robots "turn around". When two robots become adjacent to a wall while robots are in an "<" shape, two robots move down (if the wall is on the right in the global view), so that the robots now form an "L" shape; as illustrated in the sequence of configurations in Fig. 23. The rules executed by the robots during this turn are shown in Fig. 24.

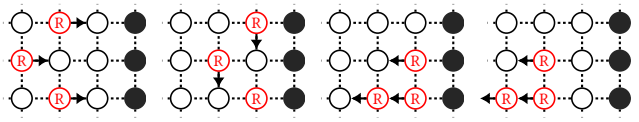

Figure 23: Sequence of configurations when robots in an "<" shape reach a wall.

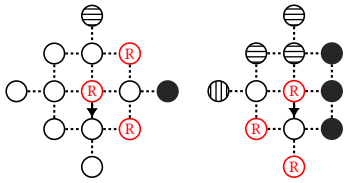

Figure 24: Rules to turn when robots are in an "<" shape.

When a robot becomes adjacent to a wall while robots are in an "L" shape, it moves down (if the wall is left in the global view) while another robot remains idle and the last one continues to move towards the wall (because it does not see the wall as it is obstructed). After that, the bottommost robot moves away from the wall so that the robots form an "<" shape; as illustrated in the sequence of configurations in Fig. 25. The rules executed by the robots during this turn are shown in Fig. 26.

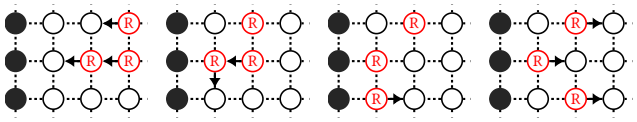

Figure 25: Sequence of configurations when robots in a "L" shape reach a wall.

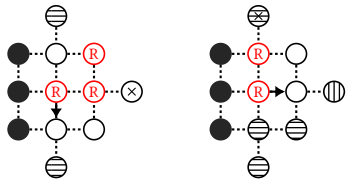

Figure 26: Rules to turn when robots are in a "L" shape.

When the robots cannot perform a turn at a corner, they perform a turn around without changing their shape. This always occurs when they are in an "<" shape. The robot at the corner moves away towards the position from it came from, the robot in the middle moves up (instead of down during a standard turn), and the topmost robot, which does not see the corner, moves down as usual; as illustrated in the sequence of configurations in Fig. 27. The rules executed by the robots during this turn are shown in Fig. 28.

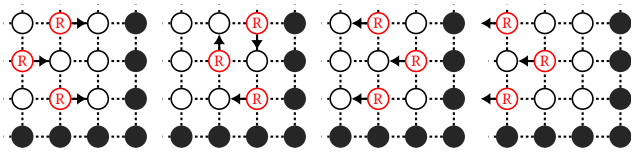

Figure 27: Sequence of configurations when robots in an "<" shape reach a corner.

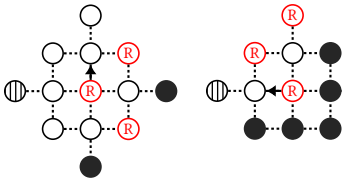

Figure 28: Rules to turn at a corner.

Theorem 5.2. Vtwo_3^1 *solves the PGFE problem with three robots, having only one color and visibility range two.*

Proof. The induction proof is similar to Theorem 4.1 for Vone_2^3. In particular, the base cases have been checked automatically using the same simulation tool [6]. □

6 CONCLUSION

We have proposed optimal synchronous solutions to the exclusive PFGE problem for swarms of luminous myopic opaque robots. The optimal algorithm Vone_2^3 for visibility range one requires only two robots and three colors. Yet, if we increase the visibility range to two, we can either decrease the number of colors to two (Algorithm Vtwo_2^2), or decrease the number of colors to one (Algorithm Vtwo_3^1) but at the price of using one additional robot. We now present some extensions and perspectives.

Asynchronous version of Algorithm Vone_2^3. In the asynchronous model, a Look-Compute-Move cycle is not atomic, yet each of the three phases of a cycle is. At each round, a robot executes at most one of the phases. The time between Look, Compute, and Move phases at a given robot is finite yet unbounded.

Algorithm Vone_2^3 can be modified to work in the asynchronous model, yet using visibility range two. For this, the rules of the leader remain the same: the leader moves if and only if it sees that the follower was its neighbor in its previous Look phase. In contrast, the follower now moves if and only if it sees the leader at distance two in its previous look phase. In this case, it moves toward the leader, as in the synchronous algorithm. It is easy to see that this version works in the asynchronous model since all the robot movements are sequential: each move of the synchronous algorithm is emulated with exactly two moves in the asynchronous one.

From perpetual to terminating exploration. We can simply modify our non-oblivious algorithms so that they terminate once all nodes have been visited. We just have to duplicate states of the follower (four times) to encode in them the number of corners it has already visited and modify the rules accordingly. Using this encoding, when the follower detects that it has reached a corner for the fourth time, it definitely stops. Seeing the follower in a terminating state, the leader also stops. Since the follower has visited the four corners, we have the guarantee that all nodes have been visited meanwhile.

Perspectives. First, one can study the impact of removing the common chirality on the lower bounds. Then, optimality for the terminating variants needs to be investigated. In particular, finding an optimal (w.r.t. the number of robots) anonymous oblivious terminating solution under the small visibility range two seems to be quite challenging. Finally, even if we have already given some hints, the designing of asynchronous efficient solutions, still in the myopic context, is definitely the most exciting perspective.

REFERENCES

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

[2] Lélia Blin, Alessia Milani, Maria Potop-Butucaru, and Sébastien Tixeuil. 2010. Exclusive Perpetual Ring Exploration without Chirality. In *Distributed Computing, 24th International Symposium, DISC 2010, Cambridge, MA, USA, September 13-15, 2010. Proceedings*, Nancy A. Lynch and Alexander A. Shvartsman (Eds.), Vol. 6343. Springer, Boston, Massachusetts, USA, 312–327.

[3] François Bonnet, Alessia Milani, Maria Potop-Butucaru, and Sébastien Tixeuil. 2011. Asynchronous exclusive perpetual grid exploration without sense of direction. In *Proceedings of International Conference on Principles of Distributed Systems (OPODIS 2011)*, Antonio Fernández Anta (Ed.). Springer Berlin / Heidelberg, Toulouse, France, 251–265. http://www.springerlink.com/content/9l3v424157681707/

[4] Quentin Bramas, Stéphane Devismes, and Pascal Lafourcade. 2020. Finding Water on Poleless using Melomaniac Myopic Chameleon Robots. In *FUN 2020, 10th International Conference on Fun with Algorithms*. LiPICs, Favignana, Sicily, Italy. to appear.

[5] Quentin Bramas, Stéphane Devismes, and Pascal Lafourcade. 2020. Infinite Grid Exploration by Disoriented Robot. In *NETYS'2020, the 8th International Conference on NETworked sYStems*. Springer, Marrakech, Morocco. to appear.

[6] Quentin Bramas, Stéphane Devismes, and Pascal Lafourcade. 2020. Optimal Exclusive Perpetual Grid Exploration by Luminous Myopic Opaque Robots: The Animations. https://doi.org/10.5281/zenodo.3947756.

[7] Ajoy Kumar Datta, Anissa Lamani, Lawrence L. Larmore, and Franck Petit. 2015. Enabling Ring Exploration with Myopic Oblivious Robots. In *2015 IEEE International Parallel and Distributed Processing Symposium Workshop, IPDPS 2015*. IEEE Computer Society, Hyderabad, India, 490–499.

[8] Stéphane Devismes, Anissa Lamani, Franck Petit, Pascal Raymond, and Sébastien Tixeuil. 2020. Terminating Exploration of a Grid by an Optimal Number of Asynchronous Oblivious Robots. *Comput. J.* 1 (03 2020). to appear.

[9] Stéphane Devismes, Anissa Lamani, Franck Petit, and Sébastien Tixeuil. 2019. Optimal torus exploration by oblivious robots. *Computing* 101, 9 (2019), 1241–1264.

[10] Stéphane Devismes, Franck Petit, and Sébastien Tixeuil. 2013. Optimal Probabilistic Ring Exploration by Semi-synchronous Oblivious Robots. *Theoretical Computer Science (TCS)* 498 (2013), 10–27.

[11] Yoann Dieudonné, Franck Petit, and Vincent Villain. 2010. Leader Election Problem versus Pattern Formation Problem. In *Distributed Computing, 24th International Symposium, DISC 2010*, Nancy A. Lynch and Alexander A. Shvartsman (Eds.), Vol. 6343. Springer, Cambridge, MA, USA, 267–281.

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

[13] Paola Flocchini, David Ilcinkas, Andrzej Pelc, and Nicola Santoro. 2011. How many oblivious robots can explore a line. *Inf. Process. Lett.* 111, 20 (2011), 1027–1031.

[14] Paola Flocchini, David Ilcinkas, Andrzej Pelc, and Nicola Santoro. 2013. Computing Without Communicating: Ring Exploration by Asynchronous Oblivious Robots. *Algorithmica* 65, 3 (2013), 562–583.

[15] Fukuhito Ooshita and Sébastien Tixeuil. 2018. Ring Exploration with Myopic Luminous Robots. In *Stabilization, Safety, and Security of Distributed Systems - 20th International Symposium, SSS 2018*, Taisuke Izumi and Petr Kuznetsov (Eds.), Vol. 11201. Springer, Tokyo, Japan, 301–316.

[16] David Peleg. 2005. Distributed Coordination Algorithms for Mobile Robot Swarms: New Directions and Challenges. In *Distributed Computing – IWDC 2005*, Ajit Pal, Ajay D. Kshemkalyani, Rajeev Kumar, and Arobinda Gupta (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–12.

A OMITTED PROOFS

We first give a simple technical lemma that makes easier our impossibility proofs.

Lemma 3.1. *Let* $\mathsf{A} = (Cl, Init, T)$ *be an algorithm that solves the PFGE problem using n robots. For every grid G with at least n lines and n columns, for every node u in G, there exist a configuration C where a robot is located at u and an algorithm* $\mathsf{A}' = (Cl, Init', T)$ *where* $Init'$ *is identical to Init unless* $C \in Init'(G)$ *such that* A' *solves the PFGE problem using n robots.*

Proof. Let u be a node of G. If A solves the PFGE problem in G, then there is an execution $C_1, C_2, \ldots, C_k, \ldots$ starting from $C_1 \in Init(G)$ such that u is occupied in C_k.

Let $\varepsilon' = C'_1 (= C_k), C'_2, \ldots$ an arbitrary execution suffix starting from C_k. Then, $C_1, \ldots, C_k, C'_2, C'_3, \ldots$ is a possible execution of A starting from C_1. By hypothesis, each node is visited infinitely often in this execution, and so is in ε'. Hence, if we add C_k as a possible initial configuration, we still have an algorithm that solves the PFGE problem in G. □

Lemma A.1. *Consider any algorithm* A *that solves the PFGE problem with three robots, one color, and visibility range one.*

No execution of A *contains a configuration where the three robots are isolated and do not see any wall.*

Proof. Assume, by the contradiction, that an execution of A reaches a configuration C where the three robots are isolated and do not see any wall. By Lemma 3.1, we can assume, without the loss of generality, that C is this initial configuration.

Assume first that a robot remains idle in C. Now, since each robot has the same color and the same local view. All robots are idle forever, a contradiction.

Assume now that a robot is enabled to move in C. Again, each robot has the same color and the same local view, so all robots are enabled to move. Moreover, for each robot, each possible destination looks identical. Consider then the smallest rectangle R that encloses the three robots. There exist three transformations such that applying each one to a robot make them move in such a way that they are still isolated and inside R in the reached configuration C'. In particular, they still do not see any wall in C'. So, by repeating the same mechanism, we construct a possible execution starting from C where several nodes are never visited (precisely, all nodes outside R), a contradiction. □

Theorem 3.6. *The PFGE problem is not solvable with three robots, one color, and visibility range one.*

Proof. Assume, by the contradiction, that an algorithm A solves the PFGE problem with three robots, one color, and visibility range one. Let r_1, r_2, and r_3 be the three robots. Consider a grid G of size at least 22×22 whose number of lines or columns is even. By Lemma 3.1 (recall that $\Phi = 1$), we can assume, without the loss of generality, that A solves the PFGE problem starting from an initial configuration C where a robot is located at a center of G.

Claim 1: *There is an execution starting from C containing a configuration where the robots are all at distance at least* 6 *from any wall.*

Proof of the claim: Let r_1 be the robot located at a center of G in C, hence at distance at least 11 from any wall. We have two possibles cases in C:

- *In C,* r_1 *sees a robot, say* r_2. So, r_2 is at distance at least 10 from any wall. So, if r_3 is at distance at most 4 from r_2, we are done. Assume now that r_3 is at distance at least 5 from r_2. Then, in the next step, r_1 and r_2 either stay idle, swap their positions, or become isolated, yet still at distance at least 10 and 9 respectively from any wall. After this step, if r_3 is at distance at least 6 from a wall, we are done. Otherwise, r_3 is at least at distance 5 and 4 from r_1 and r_2 respectively. If r_1 and r_2 are isolated, either they remain idle, or they move but all positions look identical in their respective snapshot. For each of them, there is a transformation that makes them retrieves their initial position. If they are adjacent (they were idle or swapped their position), then they remains adjacent by performing again the same move. Hence, there exists a possible execution where while r_3 is not at distance at least 6 from a wall, r_1 (resp. r_2) can at most alternate between two neighboring positions. In this execution, a node at distance 6 from a wall is visited first by r_3, and when it happens, all robots are at distance at least 6 from any wall.
- r_1 *is isolated in* C. While isolated, r_1 is either idle, or there is a possible step where it moves to a neighboring center, thus remaining at distance 11 from any wall. Hence, in such an execution, there is a node at distance 9 from any wall that is visited first by another robot, say r_2. When it happens, either all robots are at distance 6 from a wall and we are done, or r_1 and r_2 are both isolated and there exists a possible execution suffix where they both alternates between two positions without seeing a wall while they are isolated: the two centers at distance 11 from any wall for r_1 and nodes at distance 9 and 8 from any wall for r_2. In such an execution, a node at distance 7 from a wall is then eventually visited by r_3 and when it happens, we are done.

Claim 2: *If initially no robot is isolated and all robots are at distance at least 6 from any wall, then there is a reachable configuration where all robots are at distance at least 4 from any wall and at least one of them is isolated.*

Proof of the claim: Since initially no robot is isolated, there are only two kinds of possible initial configurations for the three robots: either they form a line, or an L shape, as illustrated in Fig. 29.

We now consider an execution with well-chosen transformations such that every time a robot sees only one another robot, its local view is exactly V_1 in Fig. 29. In the two kinds of possible initial configurations, there are two robots, say r_1 and r_3, with the same view V_1, so their destinations are opposite in a line configuration, and rotated by $\pi/2$ in the L configuration. Hence, they cannot move Up in V_1, *i.e.*, towards r_2, otherwise they create a collision. If they move Down in V_1, *i.e.*, away from r_2, then at least one of them becomes isolated, and we are done. We now show that moving Left or Right in V_1 also leads to at least one robot becoming isolated. If the three robots initially form a line and if r_1 and

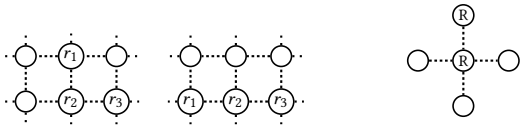

Figure 29: The two possible configurations where three robots are connected, and the view of the robots r_1 and r_3 in both cases.

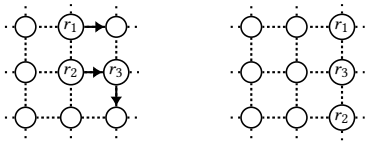

Figure 30: In an L shape, if robots r_1 and r_3 with view V_1 move left, then, r_2 has to move right not to isolate r_1.

r_3 move on the Right or the Left in V_1, then at least one of them becomes isolated. If they do not move and r_2 moves without colliding, r_2 becomes isolated. If no robots move, we have a deadlock and so the exploration fails, a contradiction. If the three robots initially form an L shape and if r_1 and r_3 move Left on V_1, then in the global view illustrated in Fig. 30, r_3 moves down and r_1 moves right. In this case, if r_2 moves towards r_3, the robots form a line and we retrieve a previous case. Otherwise, r_1 or r_3 becomes isolated; see Fig. 30. The same thing happens if r_1 and r_3 move Right in V_1. Overall, since the robots were initially all at distance at least 6 from any wall, after one or two rounds they are at distance at least 4 from any wall and at least one of them is isolated.

Claim 3: *In all reachable configurations where all robots are at distance at least 4 from any wall, no robot is isolated.*
Proof of the claim: Assume by the contradiction that there is an execution where the configuration at time t satisfies the following two conditions: (1) all robots are at distance at least 4 from any wall and (2) one robot, say r_3, is isolated. Then, the two other robots r_1 and r_2 are adjacent, otherwise we obtain a contradiction Lemma A.1. Thus, in the next step, r_1 and r_2 are either idle, swap their positions, or move to be at distance 3 from one another. In the first two cases, r_3 either stays idle or can alternate between two nodes (using the appropriate transformations) without seeing the two other robots, so there is an execution suffix where at most 4 nodes are visited, a contradiction. In the third case, while r_1 and r_2 are moving away from each other, r_3 either moves or is idle. We assume that, if r_3 moves between t and $t+1$, then we apply a transformation that makes it move away from r_1 and r_2 so that, in this case, it remains isolated at time $t+1$. So, at time $t+1$, either each robot becomes isolated, or a robot, say r_2, becomes adjacent to r_3 at time $t+1$, but in this case r_3 was necessarily idle in the step from t to $t+1$. In the former case, we obtain a contradiction by Lemma A.1 again since no robot see a wall. In the latter case, r_2 and r_3 are adjacent at time $t+1$ and are necessarily both at distance at least 3 from r_1 (*n.b.*, as r_3 has stand idle, it was at distance 2 from r_2 and so at distance at least 3 from r_1 and by moving in this opposite direction from r_2, r_1 has necessarily increased its distance to r_3 by one). So, r_2 and r_3 at time $t+1$ have exactly the same view as r_1 and r_2 at time t hence execute the same rules and move at distance 3 from each other. Moreover, r_1 is necessarily idle at time $t+1$ since it is in the same situation as r_3 at time t. So all the robots become isolated at time $t+2$ (recall that r_1 and r_2 are at distance 3 at time $t+1$). Overall, it is possible to make all robots isolated within at most two steps from time t. Now, all robots were at distance at least 4 from any wall at time t. So, in such a case, they still not see any wall after the two steps and we obtain a contradiction by Lemma A.1.

By Lemma 3.1 and Claim 1, we can assume, without loss of generality, robots are initially in a configuration C_0 where they are all at distance at least 6 from any wall. By Lemma A.1, not all robots are isolated in C_0. Moreover, by Claim 2, there is a configuration C_i reachable from C_0 where (1) all robots are at distance at least 4 from any wall and (2) at least one of them is isolated, contradicting then Claim 3.

□