Exploration of triangular grids by a swarm of synchronous luminous robots with common chirality.

Maurin Gilles
ISIMA
ZZ3 F4 - M2 ICS

March 24, 2026

Spervisor: Anaïs Durand
Defense jury: Violaine Antoine

Abstract

We consider a swarm of luminous myopic opaque robots without common chirality evolving in synchronous Look-Compute-Move cycles. We investigate algorithms to solve the exclusive perpetual exploration in finite triangular grids with such robots. In particular, we propose an algorithm to solve the perpetual exploration in grids whose global shape is a triangle. This algorithm uses only two robots, two colours and a visibility range of one, which is better than known optimals for other types of grids.

Keywords: luminous robots, grid exploration, perpetual exploration, triangular grid

---

Contents

---

1 Introduction

Swarm robotics is the subfield of robotics that aims to solve complex tasks by robots that are not powerful because of their individual capabilities, but their number and coordination. One of the most obvious tasks we could use such robots for is exploration, and a natural abstraction for this problem is to make it discrete by restraining exploration to a finite grid.

We consider a set of robots evolving in Look-Compute-Move cycles. In the *Look* phase, the robots scan neighbour nodes within a certain visibility range, to capture the subjective position of those occupied by other robots and what colour these robots are. In the *Compute* phase, the robots use this perception of their surroundings to compute their next position and next colour following embedded rules. In the *Move* phase, all the robots simultaneously go to their new position and change their colour.

All the robots follow the same set of rules. They are synchronised, so all of them perform each phase of a cycle at the same time. We are also interested here in the exclusive case, when two robots cannot be on a same edge or vertex. Eventually, we focus on an algorithm for robots with common chirality. It means that even though they have no sense of a global North-South or East-West direction, they are able to distinguish a local left from a local right.

Our goal is to find a set of rules that solves the perpetual exploration on certain types of grids while minimising the number of robots, the number of available colours, and the visibility range.

Our major contribution is an algortihm that solves the exclusive perpetual exploration in *Triangle-Shaped Triangular Grids*, a type of triangular grids whose global shape is a triangle (see Figure 1). This algorithm requires 2 robots with common chirality, 2 colours and a visibility range of 1. This is better than best known algorithms for any other type of grid.

In Section 2, we present the state of the art and existing solutions in related problem. In Section 3, we dive into a more formal definition of the problem. Section 5 deals with our contribution and is divided in two parts. In 5.1, we present the rules of our algorithm and explain how it works. In 5.2, we give a proof that the algorithm solves the perpetual exploration for grids of any size.

2 State of the art

The problem of perpetual grid exploration by a swarm of robots has already been adressed in various settings. The closest to our work deals with synchronous robots evolving on rectangular grids, with common chirality [7] or not [9]. Related work also deals with other shapes of grids such as torus [11], 3D rectangular grids [10], trees [3] or rings [1]. To our knowledge, the triangular case has only been studied in [13]. The case of an infinite grid has also been studied in the Euclidian case [6]. Equivalent problems have also been studied for asynchronous models, in rectangular grids [12] and [4] without chirality, and rings [5]. Terminating exploration has also been studied in asynchronous cases, in general networks [2] and later more thoroughly in rectangular grids [8].

2.1 Rectangular grids

The work on 2D rectangular grids provides helpful insights for the problem on triangular grids by two means. Firstly, it demonstrates impossibility results for some parameters with proofs that can − for some of them − easily be transposed to triangular grids. Secondly, it provides optimal algorithms based on strategies that can inspire algorithms to solve the problem on triangular grids.

The following tables summarise existing impossibility results and optimal algorithms for rectangular grids with and without common chirality, depending on three parameters: the range of visibility, the number of robots, and the number of colours robots can have. Note that impossibility results with chirality naturally still stand without chirality and are therefore not repeated.

Visibility	Robots	Colours	
$< \infty$	1	$< \infty$	*Impossible [7]*
$< \infty$	2	1	*Impossible [7]*
1	2	2	*Impossible [7]*
1	3	1	*Impossible [7]*
1	2	3	*Possible [7]*
2	2	2	*Possible [7]*
2	3	1	*Possible [7]*

Table 1: Impossible and optimal parameters for rectangular grids with common chirality.

Visibility	Robots	Colours	
1	2	$< \infty$	*Impossible [9]*
1	3	3	*Possible [9]*
2	5	1	*Possible [9]*

Table 2: Impossible and optimal parameters for rectangular grids without common chirality.

2.2 Triangular grids

To our knowledge, perpetual triangular grid exploration has only been studied by Das et al. in [13], in a synchronous setting and for the specific case of Rectangle-Enclosed Triangular Grids. Also, a gap remains between the parameters they prove unsufficient and the parameters used in the algorithms they propose, so optimality might not have been reached yet.

Visibility	Robots	Colours	
$< \infty$	1	$< \infty$	*Impossible [13]*
2	2	1	*Impossible [13]*
0	2	$< \infty$	*Impossible [13]*
2	2	2	*Possible [13]*

Table 3: Impossible and optimal parameters for RETGs without common chirality.

Visibility	Robots	Colours	
1	2	3	*Possible [13]*

Table 4: Impossible and optimal parameters for RETGs with common chirality.

3 The model

3.1 The grid

In our model, the grid covering space is a finite graph $G = (V, E)$ whose vertices are to be visited by robots that move along its edges. We discretise time in a way such that at any time t every robot r_i is *visiting* a vertex $u_i \in V$. Robots can choose to move along an edge at each step of time (thereby going to a node of distance one), so at $t + 1$ every robot r_i is visiting a vertex $u'_i \in V$ such that $\forall r_i : 0 \leq d(u_i, u'_i) \leq 1$, where $d(.,.)$ is the distance between two nodes. We are here interested in the particular problem of *perpetual* exploration, where algorithms should ensure that every vertex would be visited an infinite amount of times if it was ran for an infinite period of time. So for a graph to explore $G = (V, E)$, we want for all $u \in V$, for any time t, there exists a time $t' \geq t$ such that a robot is visiting u at time t'.

For generalisation purposes, we seek algorithms that solve the perpetual exploration problem for families of graphs. Intuitively, a more "general" family may require more capabilities from the robots, while fewer and more minimal robots can be sufficient for families matching more precise properties. For example, previous work in the field was interested in regular *rectangular* grids, made of $\mathcal{L} > 2 \in \mathbb{N}$ lines and $\mathcal{C} > 2 \in \mathbb{N}$ columns, and defined by the graph $G = (V, E)$ with

$$V = \{(i, j) : i \in [1, \mathcal{C}], j \in [1, \mathcal{L}]\}$$

and

$$\begin{aligned} E = &\{\{(i, j), (k, l)\} : (i, j) \in V \wedge (k, l) \in V \\ &\wedge |i - k| + |j - l| = 1\} \end{aligned}$$

In our case, we are interested in triangular grids, that can be informally imagined as grids made of triangles instead of squares. Multiple families of triangular grids can be isolated depending on the "global shape" of the grids. For example, the family of Triangle-Shaped Triangular Grids (TSTG) is defined by a size $\mathcal{N} > 0 \in \mathbb{N}$ and the graph $G = (V, E)$ with

$$V = \{(i, j) : i \in [1, \mathcal{N}], j \in [1, i]\}$$

and

$$\begin{aligned} E = &\{\{(i, j), (k, l)\} : (i, j) \in V \wedge (k, l) \in V \\ &\wedge max(|i - k|, |j - l|) \leq 1 \\ &\wedge i + j \neq k + l\} \end{aligned}$$

We can also define Hexagon-Shaped Triangular Grids (HSTG) of size $\mathcal{N} > 0 \in 2\mathbb{N} + 1$ defined by the same set of rules for the edges applied to the set of vertices

$$\begin{aligned} V = &\{(i, j) : i \in [1, \mathcal{N}], \\ &j \in [max(1, i - (\mathcal{N}/2)), min(i + (\mathcal{N}/2), \mathcal{N})] \end{aligned}$$

Eventually, we call Rectangle Enclosed Triangular Grid (RETG) grids of size $\mathcal{N} \times \mathcal{M}$ with $\mathcal{N} > 2 \in \mathbb{N}$ and $\mathcal{M} > 2 \in \mathbb{N}$ where the same set of rules for the edges is applied to the set of vertices

$$V = \{(i, j) : i \in [1, \mathcal{N}], j \in [\lceil i/2 \rceil, \lceil i/2 \rceil + \mathcal{M}]\}$$

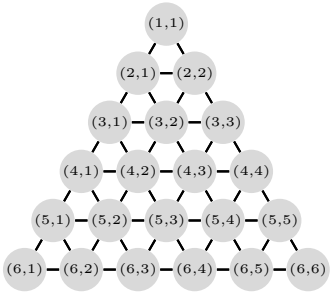


Figure 1: TSTG
$\mathcal{N} = 6$

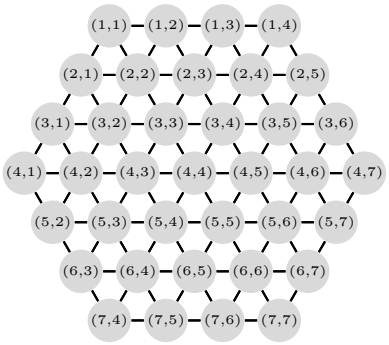


Figure 2: HSTG
$\mathcal{N} = 7$

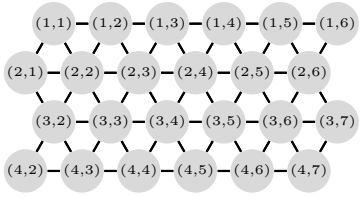


Figure 3: RETG
$\mathcal{N} = 4, \mathcal{M} = 6$

3.2 Robots

Regarding the capabilities of the robots, they are luminous, deaf-mute, myopic and opaque. "Luminous" refers to the fact that all robots have a colour. Robots know their own colour, they can change it, and they can recognise other robots' colours. "Deaf-mute" clarifies that colours are the only mean of communication robots can use. "Myopic" suggests that robots have a limited visibility range. In our graph context, it corresponds to a maximum length ϕ such that a robot only knows the state of the nodes it can reach by a path of maximum length ϕ. Finally, robots are said "opaque" because other robots cannot "see" through them. In other words, the previous definition of the visibility range should be completed as follows: a robot only knows the state of the nodes it can reach by a path of maximum length ϕ made of empty nodes.

Formally, we can define the set of all the nodes a robot visiting a node r can see as follows:

$$\{(i,j) \in V, d(r,(i,j)) \leq \phi,$$

$$\exists (l_k, l_{k+1})_{k=1...K-1}, K \leq \phi, (l_k, l_{k+1}) \in E, \forall k,$$

$$l_1 = r, l_K = (i,j), l_k \in V \wedge \sigma(l_k) = \perp, \forall k = 2...K-1\}$$

Where $d(r,(i,j))$ is the distance between the nodes r and (i,j), and $\sigma(l_k)$ is the state of the node l_k: either the colour of the robot visiting the node, or $\perp$ if the node is empty.

Another crucial parameter influencing robots' capabilities is their "sense of direction". We distinguish three degrees of sense of direction depending on whether or not robots have a global compass and a common chirality. Very informally, a global compass allows robots to "distinguish the North and the South", and a common chirality only allows them to "distinguish their left from their right", where "left" and "right" have the same meaning for all robots.
More precisely, if we represent what the robot sees in a 2D plan centered on the robot, the operations making this representation unchanged for the robot are none if it has a compass, rotations if it has chirality, and reflections with respect to an axis passing through the robot in addition to rotations without chirality.

Robots evolve on the grid by steps of Look-Compute-Move (LCM) cycles. In a single such cycle, a robot captures a representation of his surrounding with respect to its visibility range (Look), it uses this representation to decide where it should go and what colour it should be next (Compute), and it executes this decision (Move). Robots have no persistent memory, so the information captures during the Look phase is all they can use to compute their next move.
In the typical configuration, we consider robots operating synchronously, so at any moment in time they have executed the same number of cycles and are ready to execute the same phase (Look, Compute, or Move) next. However, the asynchronous configuration can also be studied. In this case, the time a robot takes to execute each phase of the cycle and the

number of cycles a robot remains inactive between two cycles is unpredictable (chosen by an adversary in theorical work such as the present one) yet unbounded.

A last constraint is generally taken into consideration for algorithms solving the perpetual exploration problem: exclusivity. It forbids two robots visiting the same node at the same time or moving along the same edge (it thereby prevents two robots swapping positions). This constraint is especially relevant when thinking about real-world applications, because it prevents two robots from colliding.

The robots are autonomous and identical: they cannot be manipulated by other means that the embedded set of rules, and all the robots obey to the same set of rules. Therefore, to find an algorithm solving the perpetual exploration problem can be reduced to finding a set of rules that lets robots explore the grid when applied. We assume that one has minimal control on the initial position: one can choose how the robots are located with resepect to each other, but not their global orientation nor their position inside the grid (for example one cannot choose to start from a specific border or corner). Thanks to this helpful relaxation, a set of rules solving the exploration problem when starting from a chosen locally-defined configuration is a sufficient result.

3.3 Views

Since robots have no sense of the global grid, it is important to think rules at a local level. We therefore introduce the *views*, that define what a robot sees and are logically centered on the robot. In a triangular grid, views have the star shape observable on figure 4. For robots without global compass in a such grid, views unchanged by a rotation of $(k\pi)/3$ radians, $k = 0...5$ are identical, in the sense that robots cannot distinguish between them. Therefore, any of the six views from figure 4 can be chosen to express an unique rule.

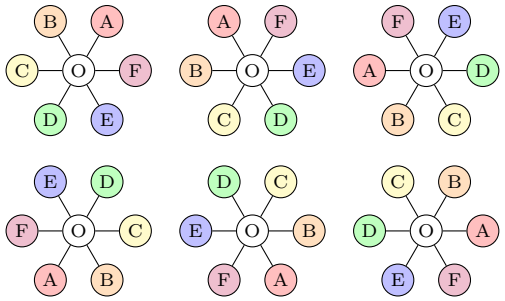


Figure 4: Views unchanged by rotation.

Without common chirality, a view covers 6 additional views equivalent by symetry. With the labels we use in our examples, they correspond to the views where nodes are labelled clockwise instead of counterclockwise. They correspond to those on figure 5

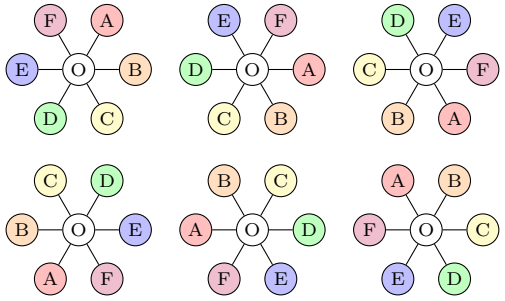
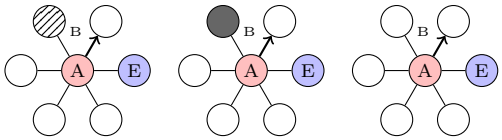


Figure 5: Views unchanged by symetry.

In our representations, each colour a robot can have will be associated with a letter, and the algorithms will be described refering to these letters rather than actual colours. An algorithm is also likely to require special behaviours when robots are near the borders of the grid. This will be handled by creating rules for views including the border as "wall" nodes. Note that such nodes are used only to represent views, and do not correspond to vertices in the graph giving the formal definition of a grid.

Sometimes, being near the border may not influence the expected behaviour of the robots in some cases. If this happens, to avoid creating multiple equivalent rules, we will add another special node to our representations (with black and white stripes), that can interchangeably be a border or an empty node.

For example, rule I of figure 6 covers both rules II and III, and corresponds therefore to 12 different scenarios in the global oriented grid (24 without common chirality).



(a) Rule I (b) Rule II (c) Rule III

Figure 6: Views including walls.

A rule is composed of a view, a node within this view to go next, and a colour to change for. In the previous example on figure 6 for example, the rule could be verbosely described as follows: *"When the robot's colour is A, and it has a robot of colour E among its neighbours with an empty node on the left and three empty nodes on the right, then the robot should move towards the left of the E-coloured robot and switch to colour B."*.

4 Optimality

With chirality, the best known algorithms require two robots, two colours, and a visibility range of two. These parameters have been shown to be optimal in rectangular grids. As regards triangular grids, it has been proven that one robot is not enough for any number of colours, and two robots not enough for one colour. However, the configuration with two robots, two colours and a visibility range of one has not been adressed.

Whether an algorithm with this set of parameters can be found or not in RETGs and HSTGs is still to be determined. For TSTGs, we propose an algorithm that solves the problem. From the existing impossibility results we know that this algorithm is optimal, and better (in terms of parameters) than any algorithm reachable on rectangular grids.

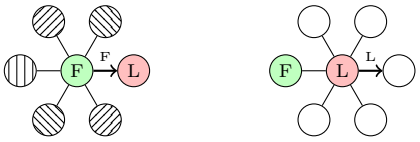
5 Algorithm

5.1 Presentation

Leader and Follower

The two colours used in the algorithm will be named L (red in our paper) and F (green). They correspond respectively to "*Leader*" and "*Follower*", the roles of the robots. Our algorithm uses two robots: one will be the leader and the other the follower.
The follower has the simplest role, implemented by only one rule: it follows the leader. Thanks to it, the leader knows "where it comes from", a piece of information that a robot could not have otherwise without memory.

The leader is in charge of leading the robots along a path that ensures that they visit all the nodes in the grid. Its general behaviour, in the middle of the grid, is to move forward. To implement this behaviour, the Leader simply moves in the opposite direction to the Follower. All the complexiy of the algorithm lies in the behaviour implemented when the leader reaches a border, or a corner.

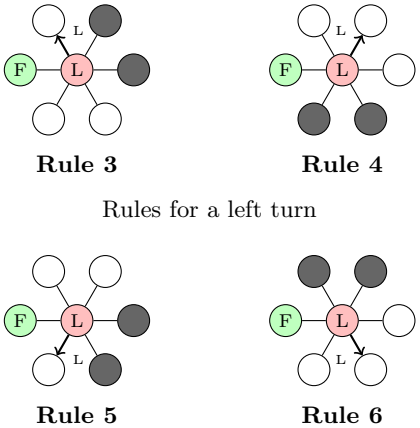


Rule 1 **Rule 2**

Figure 7: General rules: Follower and Leader moving forward.

Sweeping the grid

Because they move forward in the general case, the behaviour of the robots changes only when they reach a border. In a Triangle-Shaped Triangular Grid, a border cannot be encountered perpendicularly. Instead, the leader robot will always form either a 60° angle to the left with the wall and a 120° to the right, or vice versa. By making the robots always turn in the direction of the 60° angle, we thus implement opposite behaviours depending on the angle the robots reach a border with.



Rule 3 **Rule 4**

Rules for a left turn

Rule 5 **Rule 6**

Rules for a right turn

Figure 8: Rules for turns on borders

A consequence of this choice is that the robots will "bounce" between two opposite borders and tend to approach the corner at the intersection of these borders. Take for example the global wiew of a grid in Figure 9 and consider robots that move along the

edges parallel with (BC). When such robots reach the border (AB), the 60° angle let them turn right by applying Rules 5-6, and continue towards the border (BC), where they will turn left by applying Rules 3-4, etc. Eventually, they "sweep" the grid until they reach the corner A.

To solve the perpetual exploration, and ensure that all the nodes including the corners are visited, an intuitive behaviour is to have the robots move towards each of the three corners in a loop. To do so, when the robots reach a corner, they should change their behaviour to bounce between two other borders and thereby start moving towards a new corner. Two solutions can be considered: either the border on which the robots used to turn left becomes the one on which they turn right, or the one on which they used to turn right becomes the one on which they turn left. In the first case the corners are visited counterclockwise, in the second they are visited clockwise.

In our example, the robots used to turn right when they reached the border (AB). If it becomes the border on which they turn left, then they will bounce towards (BC) and turn right, and eventually reach the corner B. Symetrically, if they start turning right when they reach (AC), they will turn right on (BC) and eventually reach C.

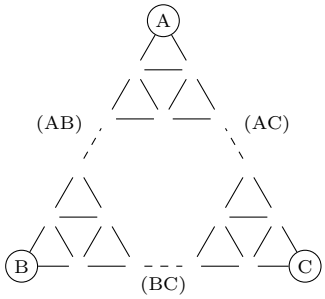


Figure 9: Grid structure

Even and odd grids

The order in which the corners are visited could be of no importance, if it was not to distinguish two cases corresponding to the parity of the grid. By "parity of the grid", we mean whether the number of nodes between two corner is even or odd.

The problem with parity comes from the fact that a unique behaviour in the corners cannot handle both even and odd grids. Consider Figures 12 and 13, where robots are going from a departure corner D to an arrival corner A. Each node on (DA) will either be a node on which they "move out" (blue) or "move in" (red). By moving out we mean that they leave the border and go inside the grid, and we say that they move in when they arrive from inside the grid. Consider applying all the previous rules, plus Rules 7 and 8 from Figure 10 in the corners.

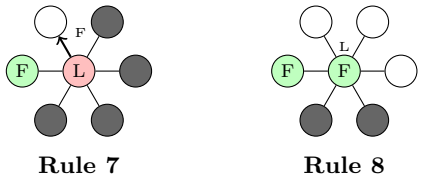


Figure 10: Rules for an even/counterclockwise corner turn

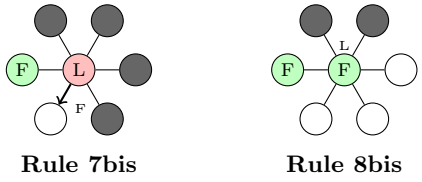


Figure 11: Symmetrical rules for an odd corner turn (imaginary)

In the configuration with an even grid, the corner is reached "by the right", and applying consecutively Rules 7-8-4 will let the robots move in a direction parallel with (DA), and one can convince themself that they will reach the third corner next, then D again, then A, etc, thus solving the perpetual exploration.

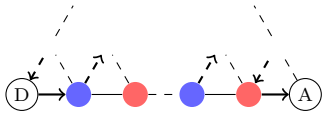


Figure 12: Even grid

On an odd grid, on the other hand, the situation becomes problematic. The robots move out on the last node before the arrival, apply Rule 5 to turn, and find themselves in an uncovered configuration. Thinking symmetrically, one may think of applying Rules 7bis and 8bis from Figure 11. However, the robots would thereby start moving back to the corner D, and alternate between these two, leaving the third corner unexplored.

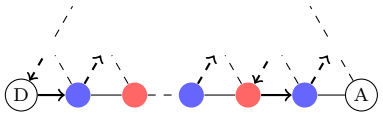
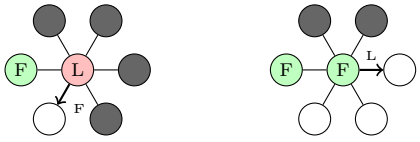


Figure 13: Odd grid

This problem is easy to understand once we realise that in this example, corners D and A were not visited coming from the same side, which can be unformally understood as a "shift" induced by the odd parity of the grid. This can be overcome by retrieving an even parity thanks to a shift included in the corner rules 9 and 10 from Figure 14.



Rule 9 **Rule 10**

Figure 14: Rules for an odd/clockwise corner turn

It produces the behaviour illustrated by Figure 15 with the node produced by the shift in yellow. Note that these rules would conflit with Rules 7 and 8 if we tried to keep the robots visiting the corners counterclockwise, hence the necessity to match parity with directions, which also correspond to visiting corners "coming from" the right or the left.

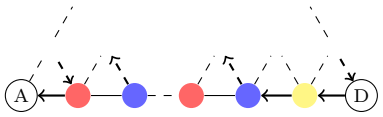


Figure 15: Odd grid, solution

Even and odd positions

We have identified two stable solutions that solve the perpetual exploration with the same set of rules: one on even grids where robots move counterclockwise, one on odd grids where robots move clockwise. When the robots are in a certain position in the middle of a grid of unkwon size and start moving, it is impossible to predict whether they will reach an angle by the left or the right, i.e. if they will perform an even/conterclockwise corner turn or an odd/clockwise one.

However, this impossibility comes from the local view. This is deterministic, and someone with a global view can see for any position of the robots in the grid whether they will move as in the stable solution for an even or an odd grid. We can afford to talk about **even** or **odd** position. With the node numbering from our introduction represented on figure 1, we have immediate laws to identify even and odd positions that depend on the parity of the size $\mathcal{N}$ of the grid, written $p(\mathcal{N})$. These laws are summarised in table 5, and apply only with at most one robot near a border.

Leader (i_L, j_L)		Follower (i_F, j_F)		Position
$p(i_L)$	$p(j_L)$	i_F	j_F	
0	...	i_L	$j_L - 1$	**even**
1	...	i_L	$j_L - 1$	**odd**
0	...	i_L	$j_L + 1$	**odd**
1	...	i_L	$j_L + 1$	**even**
...	$= p(\mathcal{N})$	$i_L - 1$	j_L	**odd**
...	$\neq p(\mathcal{N})$	$i_L - 1$	j_L	**even**
...	$= p(\mathcal{N})$	$i_L + 1$	j_L	**even**
...	$\neq p(\mathcal{N})$	$i_L + 1$	j_L	**odd**
$p(i_L + j_L)$				
$= p(\mathcal{N}))$		$i_L - 1$	$j_L - 1$	**odd**
$\neq p(\mathcal{N}))$		$i_L - 1$	$j_L - 1$	**even**
$= p(\mathcal{N}))$		$i_L + 1$	$j_L + 1$	**even**
$\neq p(\mathcal{N}))$		$i_L + 1$	$j_L + 1$	**odd**

Table 5: Parity of any position inside the grid.

Example: *If the leader is on the vertex* $(7, 4)$ *and the follower on the vertex* $(6, 4)$ *on a grid of size* 11*, then because* $p(j_L) \neq p(\mathcal{N})$ *and* $(i_F, j_F) = (i_L - 1, j_L)$*, we know that the robots are in an even position, which means that they will do an even/counterclockwise turn when they reach the next corner.*

The mechanism behind these laws is not trivial to comprehend. To understand them, one may begin thinking about the last position before the leader reaches the corner, and where the follower should be with respect to the leader for the corner to be reached for an even/counterclockwise turn, for example. These positions are respectively:

- $L = (2, 2), F = (2, 1)$
- $L = (\mathcal{N} - 1, 1), F = (\mathcal{N}, 2)$
- $L = (\mathcal{N}, \mathcal{N} - 1), F = (\mathcal{N} - 1, \mathcal{N} - 1)$

We construct three main laws for these positions to be even for any value of $\mathcal{N}$. All the others are constructed by symmetry from one of these three.

Correcting the direction

Because we have only local choice on the inital position of the robots, it is impossible to guess whether

the parity of the initial position matches the parity of the grid. We want to show that the algorithm inherently solves this problem, and after one corner visited the wrong way the robots will find themselves in a position of proper parity.

See for example the case from Figure 13, where the robots start from corner D applying rules 7 and 8 for even grids, although the grid in which they evolve happens to be odd. Then we can notice that when they enter corner A, they find themselves in the configuration of Figure 15, which has been shown to lead to a stable solution. Similarly, applying the shift of rules 9-10 on an even grid will let the robots access the next corner by the right and continue with the proper stable solution.

A more convincing yet more abstract way to figure this is to think of the positions of the robots after a corner turn. These positions are the following:

- Even Corner Turn
  - $L = (3, 2), F = (2, 1)$
  - $L = (\mathcal{N} - 1, 2), F = (\mathcal{N}, 2)$
  - $L = (\mathcal{N} - 1, \mathcal{N} - 2), F = (\mathcal{N} - 1, \mathcal{N} - 1)$
- Odd Corner Turn
  - $L = (4, 3), F = (3, 3)$
  - $L = (\mathcal{N} - 2, 2), F = (\mathcal{N} - 2, 1)$
  - $L = (\mathcal{N} - 1, \mathcal{N} - 3), F = (\mathcal{N}, \mathcal{N} - 2)$

It can be observed that all these six positions match the parity of $\mathcal{N}$ for any value of $\mathcal{N}$. So no matter what happens in the first corner visited, we are ensured that the robots will be in a position matching the stable position for the grid. It gives an additional intuition on why the exploration is perpetual: robots on an even grid will always perform even corner turns and end up in even postions, while robots on an odd grid will always perform odd corner turns and end up in odd positions.

5.2 Proof

The following proof is done by induction. First, we prove that the algorithm works for grids of size 4 and 5. Then, we show by induction that if the algorithms solves the problem for a grid of size $\mathcal{N}$, it solves the problem for a grid of size $\mathcal{N} + 2$, and thanks to the two base cases, we thus show that the algorithm works in grids of any size.

We use this two-hops recursion to preserve parity. Increading the size of the grid by one changes its parity, so the stable solution in the grid of size $\mathcal{N} + 1$ has the robots moving in the opposite direction, and their behaviour in the subgrid of size $\mathcal{N}$ will not be the same as it is in the grid of size $\mathcal{N}$, which makes induction reasoning more complicated.

Base cases

Thanks to a simulator tool (https://sancy.iut.uca.fr/ durand/simulator/), we have been able to confirm that the algorithm solves the probelm for grids of size $\mathcal{N}$, with:

- $\mathcal{N} = 4$, starting from any position.
- $\mathcal{N} = 5$, starting from any position.

Because these grids are small, we were able to test every possible initial position inside the grid, including those of wrong parity.

Induction

Assume that the algorithm solves the perpetual exploration problem for any grid of size $\mathcal{N}$.

Consider a grid of size $\mathcal{N} + 2$. We can see this grid as a composition of a subgrid of size $\mathcal{N}$, in which the algorithm is assumed to solve the problem, augmented by two additional "rows of triangles" on an arbitrary side, as shows Figure 16.

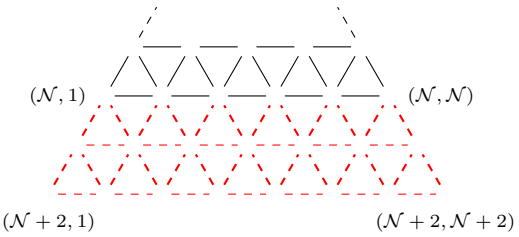


Figure 16: Increasing the size of the grid more

The structure of our reasoning for a grid of size $\mathcal{N} + 2$ will work as follows :

1. The robots will reach a corner.
2. The additional rows do not change the parity of a corner turn.
3. The robots' behaviour inside the subgrid of size $\mathcal{N}$ is not changed so the solution remains stable.
4. Every additional node is visited.

The robots will reach a corner

In our figures, the two additional rows are represented at the bottom. There is no global orientation in the grid, so we can actually declare any side to be the bottom and take these two rows as the additional ones. This allows us to choose that side in a way that ensures that no robot is on a node from these two rows at the initial position, i.e. both robots are inside the subgrid of size $\mathcal{N}$ at the initial position.

The algorithm solves the perpetual exploration problem in this subgrid, so all its vertices will be visited. In particular, the robots will reach a node from the border of the subgrid, i.e. at some point the leader robot will be on a node labelled $(\mathcal{N}, j), j \in [1, \mathcal{N}]$. When this happens in a grid of size $\mathcal{N}$, the robots turn and the leader goes either on node $(\mathcal{N}, j-1)$ or $(\mathcal{N}, j+1)$ depending on the angle they reached the border with, and then start moving back to the inside of the grid.

In the grid of size $\mathcal{N}+2$, the robots turn when they reach a border two nodes later. Since the angle is the same, the robots turn in the same direction, and go back to the inside of the grid in the same configuration as in the grid of size $\mathcal{N}$. This phenomenon that we call **behaviour preservation** on borders is very important for the induction and is illustrated on figuren 17.

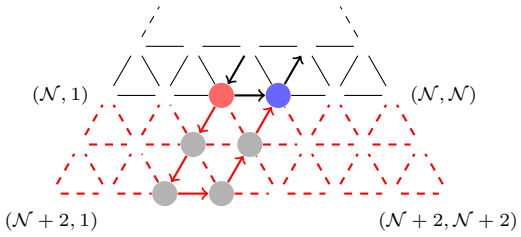


Figure 17: Behaviour preservation on borders

Thanks to this behaviour preservation on borders, we are ensured that the robots keep moving towards a corner.

Behaviour preservation on corners

We want to show that when the robots are in a congiguration that leads to a corner turn of a certain parity in the subgrid of size $\mathcal{N}$, a turn of the same parity will be performed in the actual corner of the grid of size $\mathcal{N}+2$. Additionally, we want to show **behaviour preservation** of such turns, i.e. that the robots will end up in the same configuration as if they had turned on the corner of the subgrid.

Figure 18 shows how it happens for even/counterclockwise corner turns, and Figure 19 shows it for odd/clockwise turns. In these figures, the black arrows show the path that the robots would take if they turned on the corner at $(\mathcal{N}, 1)$, and the red arrows show the path taken by the robots in the grid of size $\mathcal{N}+2$. In both cases, the initial and terminal configuration are the same: $L = (\mathcal{N}, 1), F = (\mathcal{N}-1, 1)$ at the beginning and $L = (\mathcal{N}-1, 2), F = (\mathcal{N}, 2)$ at the end for an even/counterclockwise turn; $L = (\mathcal{N}, 2), F = (\mathcal{N}-1, 1)$ at the beginning and $L = (\mathcal{N}-2, 2), F = (\mathcal{N}-2, 1)$ at the end for an odd/clockwise turn.

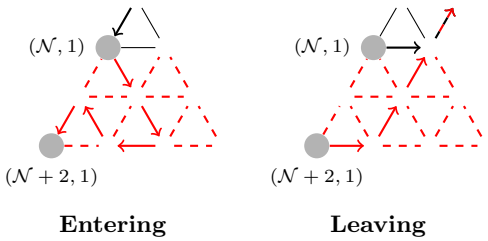


Figure 18: Behaviour preservation for even/counterclockwise corner turns.

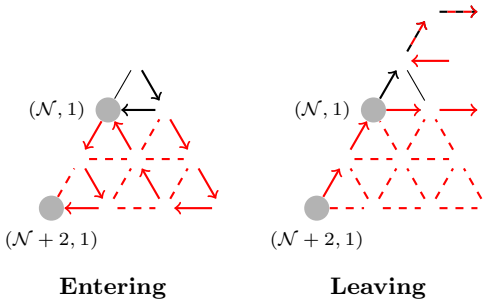


Figure 19: Behaviour preservation for odd/clockwise corner turns.

The stable solution still stands.

From behaviour preservation on borders and in corners for any postition parity, we are allowed to apply an inductive reasoning. Looking only at the behaviour inside the subgrid of size $\mathcal{N}$, one would not be able to guess the presence of the two additional rows. Therefore, because the behaviour in the subgrid implements a stable solution, we are ensured that the algorithm still implements a stable solution in a grid of size $\mathcal{N}+2$, and that all the vertices in the subgrid are visited.

Every additional vertex in visited.

The properties of the algorithm still stand in a grid of larger size, so we know that after at most one corner turn the robots will be in a stable configuration in which every vertex from the subgrid of size $\mathcal{N}$ will be visited. The very last piece of our proof is to show that every vertex from the additional rows is also visited.

First, there are the vertices "in the middle". Because the robots follow the path of a stable solution, we know that every node $(\mathcal{N}, j), j \in [2, \mathcal{N}-1]$, on the bottom border in our figures, will be visited twice every cycle: once during the phase in which the robots are approaching the corner at $(\mathcal{N}+2, \mathcal{N}+2)$, once during the phase in which they are approaching the corner at $(\mathcal{N}+2, 1)$. The first case corresponds to the orientation they have on Figure 17, the second to a rotation of $2\pi/3$ radians of this orientation. Thanks to the mechanism of behaviour preservation illustrated on figure 17, we conclude that every node $(\mathcal{N}+1, j), j \in [2, \mathcal{N}]$ and $(\mathcal{N}+2, j), j \in [2, \mathcal{N}+1]$ will be visited.

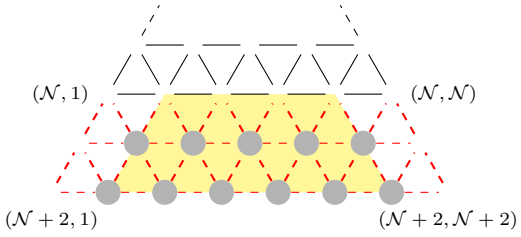


Figure 20: Middle nodes visited.

The remaining nodes are $(\mathcal{N}+1, 1)$, $(\mathcal{N}+2, 1)$, $(\mathcal{N}+1, \mathcal{N}+1)$ and $(\mathcal{N}+2, \mathcal{N}+2)$. Again, because a stable solution is implemented inside the subgrid of size $\mathcal{N}$, we are ensured that corners of this sugrid, $(\mathcal{N}, 1)$ and $(\mathcal{N}, \mathcal{N})$, will be approached. Depending on the parity, the situation will either be similar to Figure 18 or Figure 19. However, in both cases we notice that every node close to the corner is visited during the behaviour preservation mechanism.

6 Perspectives

Our algorithm is a motivating result for the added value of triangular grids for perpetual exploration. However, this specific algorithm works only for the specific case of Triangle-Shaped Triangular Grids. It is yet to be determined if triangular grids of other shapes can be explored with equivalent parameters, or if an additional robot or colour is necessary as it is the case in rectangular grids.

There is also room for investigation in cases with asynchronous robots and/or no common chirality. Another interesting question is to determine if some algorithms could solve multiple shapes of triangular grids with a same set of rules.

References

[1] Lélia Blin et al. "Exclusive perpetual ring exploration without chirality". In: *International Symposium on Distributed Computing*. Springer. 2010, pp. 312–327.

[2] Jérémie Chalopin et al. "Network exploration by silent and oblivious robots". In: *International Workshop on Graph-Theoretic Concepts in Computer Science*. Springer. 2010, pp. 208–219.

[3] Paola Flocchini et al. "Remembering without memory: Tree exploration by asynchronous oblivious robots". In: *Theoretical Computer Science* 411.14-15 (2010), pp. 1583–1598.

[4] François Bonnet et al. "Asynchronous exclusive perpetual grid exploration without sense of direction". In: *International Conference On Principles Of Distributed Systems*. Springer. 2011, pp. 251–265.

[5] Fukuhito Ooshita and Sébastien Tixeuil. "Ring exploration with myopic luminous robots". In: *International Symposium on Stabilizing, Safety, and Security of Distributed Systems*. Springer. 2018, pp. 301–316.

[6] Quentin Bramas, Stéphane Devismes, and Pascal Lafourcade. "Infinite grid exploration by disoriented robots". In: *International Conference on Networked Systems*. Springer. 2020, pp. 129–145.

[7] Quentin Bramas, Pascal Lafourcade, and Stéphane Devismes. "Optimal exclusive perpetual grid exploration by luminous myopic opaque robots with common chirality". In: *Proceedings of the 22nd International Conference on Distributed Computing and Networking*. 2021, pp. 76–85.

[8] Stéphane Devismes et al. "Terminating exploration of a grid by an optimal number of asynchronous oblivious robots". In: *The Computer Journal* 64.1 (2021), pp. 132–154.

[9] Arthur Rauch et al. "Optimal exclusive perpetual grid exploration by luminous myopic robots without common chirality". In: *International Conference on Networked Systems*. Springer. 2021, pp. 95–110.

[10] Quentin Bramas et al. "Beedroids: How Luminous Autonomous Swarms of UAVs Can Save the World?" In: *FUN: Conference on Fun with Algorithms*. 2022.

[11] Omar Darwich et al. "Perpetual torus exploration by myopic luminous robots". In: *Theoretical Computer Science* 976 (2023), p. 114143.

[12] Quentin Bramas et al. "Optimal Asynchronous Perpetual Grid Exploration". In: *International Symposium on Stabilizing, Safety, and Security of Distributed Systems*. Springer. 2024, pp. 89–105.

[13] Raja Das et al. "Rectangle Enclosed Triangular Grid Exploration with Myopic Luminous Robots". In: *Proceedings of the 26th International Conference on Distributed Computing and Networking*. 2025, pp. 249–253.