77 Shades of Grey

Quentin Bramas
University of Strasbourg, ICUBE, CNRS, France

Stéphane Devismes
Université de Picardie Jules Verne, MIS UR 4290, Amiens, France

Anaïs Durand
Université Clermont Auvergne, CNRS UMR 6158, LIMOS, Clermont-Ferrand, France

Pascal Lafourcade
Université Clermont Auvergne, CNRS UMR 6158, LIMOS, Clermont-Ferrand, France

Anissa Lamani
University of Strasbourg, ICUBE, CNRS, France

Abstract

Bruce Wayne contacted us to help him develop a new surveillance technology for dark environments such as caves, using a swarm of Unmanned Aerial Vehicles (UAVs), called *Batdroids*. A Batdroid has no chirality, limited visibility, and a perfect clock to synchronize with the others. A Batdroid can produce 77 shades of grey in dark mode and four colors in light mode.

In this paper, we propose two algorithms using three Batdroids to perpetually explore a finite 3D grid modeling a cave. The first algorithm operates in darkness, uses 77 shades of grey, and requires visibility range one. The second operates in light, uses four colors and visibility range two. We also prove that three is the optimal number of Batdroids required to solve Bruce Wayne's challenge.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Mobile robots, grid exploration, perpetual exploration

Digital Object Identifier 10.4230/LIPIcs.FUN.2026.10

Funding This study was partially supported by the French ANR projects SkyData (ANR-22-CE25-0008) and PRIVA-SIQ (ANR-23-CE39-0008).

1 Introduction

Six months ago, we received an email from Alfred, Bruce Wayne's butler. In this letter, Alfred explained that his master discovered our previous work, published at FUN 2022 [5], while he was reading a scientific article on the strategic enhancement of pollination through smart agriculture to counteract the decline of natural pollinators [22]. In that work, we designed the concept of *Beedroids*, artificial bees that aim to autonomously pollinate flowers in a greenhouse through perpetual exploration with synchronized movements. A Beedroid is a small autonomous Unmanned Aerial Vehicle (UAV) that mimics the behavior of a real bee. One famous ability of bees is *stigmergy*, *i.e.*, indirect communication through movements. Implementing stigmergy requires perfect synchronization and visibility sensors. Consequently, we used the fully synchronous Look-Compute-Move (FSync) model of computation [21].

Surprisingly, Bruce Wayne is very fond of bats and owns a large cave beneath his manor house. He immediately noticed that our concept can be used to patrol caves and detect any intrusions. Consequently, he asked us to design a new prototype of UAV to be produced by Wayne Enterprises. He also requested that the associated algorithms ensure perpetual exploration of his house. The code name of this UAV is naturally *Batdroid*. The first constraint is that the Batdroids should be in carbon-titanium, be silent, and resemble a bat with a size of 28 cm and a weight of 420 g. Moreover, there are specific constraints

due to the nature of the rock surrounding the cave. Indeed, while our Beedroids rely on chirality (*i.e.*, ability to distinguish the two sides of a symmetrically reflexive panorama), Bruce Wayne's cave is mainly made of basalt stone with a high concentration of magnetite, preventing any compass from functioning. Consequently, Batdroids do not have chirality. A second constraint is that Batdroids should operate both when the cave is lit and when there is almost no light. To address this, Wayne Enterprises owns a patent on the concept of *grey lights*. Thanks to this new top-secret technology, Batdroids can be endowed with lights and sensors that can emit and distinguish exactly 77 shades of grey.

To summarize the constraints of this ambitious project:

Visibility Sensors: To maximize flight time, energy consumption must be minimized. Hence, the visibility range of Batdroids should be as small as possible.

Communication: Communication of Batdroids, like real bats, is indirect and based on positions of other Batdroids. To significantly increase stigmergic communication without compromising production cost and energy consumption, Batdroids are endowed only with LEDs that can produce 77 levels of grey or four colors, which can be sensed by other Batdroids within a short distance. While the shades of grey can be distinguished in darkness, the colors can only be used when there is some light.

Memory: To further reduce energy consumption and manufacturing costs, Batdroids have no permanent memory, except the color or shade of grey emitted by their lights. They only have a short-term working memory allowing them to compute a decision (destination and new light color) at each step of their algorithm.

Orientation: Manufacturing costs, energy consumption, and magnetic interference in the cave also prevent us from endowing Batdroids with GPS, orientation, or any form of chirality.

Synchronization. Batdroids are synchronous UAVs equipped with a perfect clock designed by Wayne Enterprises for previous high-tech products.

Contribution. These constraints clearly constitute a new challenge for us. Nevertheless, we accepted this mission and designed two algorithms; one operates in the dark, while the other operates in the light mode. In this paper, we describe the details of our algorithms for Batdroids, naturally with the authorization of Bruce Wayne. The design of the Batdroids themselves is confidential and protected by patents in order to preserve the intellectual property of Wayne Enterprises; consequently, we do not disclose any information regarding their conception. Our two algorithms operate in the FSync model and enable Batdroids to perpetually explore a three-dimensional grid representing the cave. As motivated above, our goal is to minimize both the size of the colony (*i.e.*, the number of Batdroids that compose it) and the capabilities required by those Batdroids.

We first investigate the problem under visibility range one in dark mode. Under this assumption, it is already known from [5] that three Batdroids are necessary to solve the problem. We show that this number is also sufficient in our context, despite the absence of common chirality. Specifically, we present an algorithm that uses only three Batdroids with visibility range one and requires 77 shades of grey. However, this algorithm has a shortcoming: although it successfully ensures perpetual and complete patrolling of the cave, its executions are not fully deterministic. Some rules are ambiguous and rely on symmetry breaking that is arbitrarily resolved by an adversary. As a result, the patrol behavior is not fully under control – an unsatisfactory property for a highly self-disciplined individual such as Bruce Wayne. In other words, Bruce Wayne requires a *well-defined algorithm* (*i.e.*, non-ambiguous) for his Batdroids.

Nevertheless, we prove that, regardless of the number of available colors, no well-defined algorithm allows three Batdroids to perpetually patrol caves under visibility range one without common chirality. Bruce Wayne is willing to accept this limitation in dark mode, acknowledging that darkness naturally restricts Batdroid perception. However, he argues that in light mode, increasing the visibility range is a reasonable way to circumvent this impossibility.

Motivated by this observation, we investigate a well-defined solution under visibility range two for light mode. This second solution also requires only three Batdroids. Moreover, the increased visibility range allows us to drastically reduce the required capabilities: in light mode, only four colors are sufficient to implement our algorithm.

Roadmap. In the next section, we formally define the model, the Batdroid capabilities, and the problem to solve, the so-called *Perpetual Exploration Problem* (PEP). In Section 3, we present a lower bound on the number of Batdroids necessary to solve the problem with a well-defined algorithm and under visibility range one. In Sections 4 and 5, we present our two algorithms. We give a detailed related work in Section 6. Finally, we conclude in Section 7. Notice also that, to help the reader, online animations illustrating the behavior of our algorithms are available for both solutions: [3] and [4].

2 Preliminaries

We consider a colony of $n > 0$ Batdroids (*n.b.*, n is *a priori* unknown by Batdroids) evolving in a cave modeled as a *finite 3D grid* of size $S_x \times S_y \times S_z$ *i.e.*, an undirected graph $G(V, E)$ where $V = \{(i, j, k) \ : \ i \in [0, S_x - 1], j \in [0, S_y - 1], k \in [0, S_z - 1]\}$ and $E = \big\{\{(i, j, k), (i', j', k')\} \ : \ |i - i'| + |j - j'| + |k - k'| = 1\big\}$. Note that coordinates are used for the analysis only, *i.e.*, Batdroids cannot access them. We assume in this work that $S_x \geq 4$, $S_y \geq 4$, $S_z \geq 4$ to reduce the number of particular cases in small grids, despite the literature allowing smaller grids, which usually require specific considerations.

We assume discrete time and at each *round*, the Batdroids synchronously perform a *Look-Compute-Move* cycle. In the *Look* phase, a Batdroid 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 Batdroids. The snapshot is not oriented in any way as the Batdroids do not agree on the orientation of any of the three axes of the coordinate system. However, it is implicitly ego-centered since the Batdroid that performs a Look phase is located at the center of the subgraph in the obtained snapshot. Then, each Batdroid *computes* a destination in its local coordinate system (either Front, Back, Left, Right, Above, Below, or Idle) based on the received snapshot only. Finally, it *moves* towards its computed destination.

We forbid any two Batdroids to occupy the same node simultaneously. A node is *occupied* when a Batdroid is located at this node, otherwise it is *empty*. Batdroids have *lights* with maybe different colors that can be seen by Batdroids within distance Φ from them. We denote by Cl the set of all possible colors ($|Cl| = 1$ corresponds to the case of oblivious Batdroids).

The *state* of a node is either the color of the light of the Batdroid 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 Batdroid may decide to change the color of its light (of course, if $|Cl| > 1$).

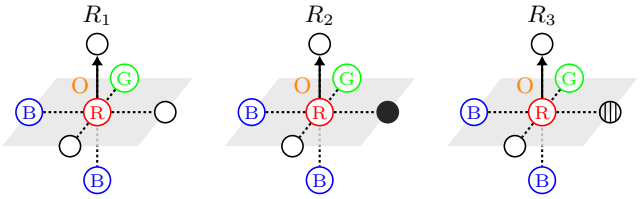

Figure 1 Examples of rules. Colored letters inside nodes indicate the color of the Batdroids occupying the nodes. The arrow indicates the destination and when a colored letter is given next to an arrow, this represents the new color taken by the Batdroid.

In all our algorithms, we also prevent any two Batdroids from traversing the same edge simultaneously. Since we already forbid them to occupy the same position simultaneously, this means that we additionally prevent Batdroids from swapping their positions. Algorithms verifying this property are said to be *exclusive*. However, to remain as general as possible, our impossibility result does not rule out executions in which Batdroids swap their positions.

Configurations. A *configuration* C in a 3D 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 Batdroid 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 Batdroid 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 are $(0, 0, 0)$. We use this globally oriented view in our analysis to describe the movements of the Batdroids (see, for example, Figure 1): when we say "the Batdroid moves Left", it is according to the globally oriented view. However, since Batdroids do not agree on any axis, they have no access to the globally oriented view. When a Batdroid looks at its surroundings, it instead obtains a *local view*. To model this, we assume that the local view acquired by a Batdroid r in the Look phase is the result of an arbitrary *indistinguishable transformation* on G_r. Here, we assume that Batdroids are *self-inconsistent*, meaning that different transformations may be applied at different rounds. An indistinguishable transformation consists of applying to each of the three axes (x-axis, y-axis, and z-axis) passing through r a rotation (maybe different for each axis) picked in the set $\{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$. We denote by $\mathcal{IT}$ the set of all possible indistinguishable transformations. Contrary to Beedroids in [5], Batdroids do not share a common chirality. That is, they cannot distinguish a local view from its reflection and hence take different decisions in each case. For instance, in Figure 2, Batdroids cannot distinguish Above from Below as they cannot use the right-hand rule to determine the third axis knowing two other axes.

It is important to note that when a Batdroid 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 Batdroid in the *globally oriented view* is $f^{-1}(d)$. For example, if $d = Above$ but the Batdroid sees the 3D grid upside-down (f is the π-rotation along the y-axis), then the Batdroid moves $Below = f^{-1}(Above)$. In a configuration C, $V_C(i, j)$ denotes the globally oriented view of a Batdroid located at (i, j).

A Batdroid is said to be *lost* when it sees no wall and no other Batdroids. Observe that in this case, if the Batdroid decides to move, the destination is entirely determined by the choice of the transformation f done by the adversary.

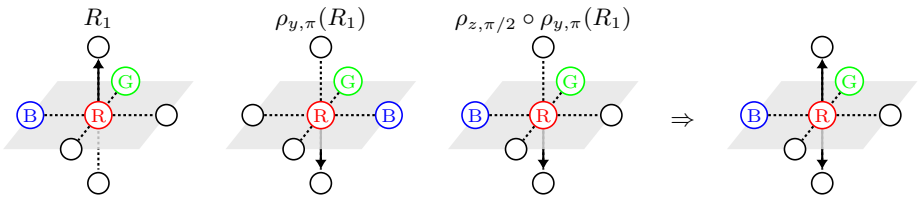

Figure 2 An ambiguous rule. Since the destination depends on the choice of the adversary, the rule is abusively represented with multiple destinations, as illustrated on the right. Observe that despite the Batdroids having different colors, the rule is ambiguous since the Batdroids cannot determine a third direction, given the two directions obtained from the view.

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 3D grid to a non-empty set of initial configurations in that 3D grid, and T is the transition function $Views \to \{Idle, Front, Back, Left, Right, Above, Below\} \times Cl$, where $Views$ is the set of local views. When the Batdroids are in Configuration C, a configuration C' obtained after one round satisfies: $((i,j,k),c') \in C'$, if and only if there exists a color $c \in Cl$ and a transformation $f \in \mathcal{IT}$ such that one of the following conditions holds:

- $((i,j,k),c) \in C$ and $f^{-1} \circ T \circ f \circ V_C(i,j,k) = (Idle, c')$,
- $((i-1,j,k),c) \in C$ and $f^{-1} \circ T \circ f \circ V_C(i-1,j,k) = (Right, c')$,
- $((i+1,j,k),c) \in C$ and $f^{-1} \circ T \circ f \circ V_C(i+1,j,k) = (Left, c')$,
- $((i,j-1,k),c) \in C$ and $f^{-1} \circ T \circ f \circ V_C(i,j-1,k) = (Front, c')$,
- $((i,j+1,k),c) \in C$ and $f^{-1} \circ T \circ f \circ V_C(i,j+1,k) = (Back, c')$,
- $((i,j,k-1),c) \in C$ and $f^{-1} \circ T \circ f \circ V_C(i,j,k-1) = (Above, c')$, or
- $((i,j,k+1),c) \in C$ and $f^{-1} \circ T \circ f \circ V_C(i,j,k+1) = (Below, c')$.

An Algorithm as a Set of Rules. We write an algorithm as a set of rules, where *a rule* is a triplet $(V,d,c) \in Views \times \{Idle, Front, Back, Left, Right, Above, Below\} \times Cl$. 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 $f \in \mathcal{IT}$.

As an illustrative example, consider the rule R_1 given in Figure 1. This rule is defined for Batdroids having a visibility range of one. This rule means that, when a Batdroid sees three Batdroids, one with Color B on its left, one with color G in front of it, and the third one with Color B below it, then the red Batdroid is dictated to move Above and change its color to O.

In the same figure, Rule R_2 is a rule where a black node represents a part of the outer boundary of the 3D grid, that we call *a wall* in the remaining of the paper. 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 the same rules multiple times with very similar views, we propose a notation to represent several rules in just one picture. For example, Rule R_3 in Figure 1 has one node hatched with vertical lines, which means that the rule applies regardless of the presence of a wall located at this node. 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 Figure 1 is a concise representation of Rules R_1 and R_2.

Figure 2 shows an ambiguous rule. The Batdroid has a symmetric view so that, depending on the transformation f chosen by the adversary, the Batdroid executing this rule moves either above or below. In the following, we represent in ambiguous rules all possible destinations that can be dictated by the adversary.

Well-defined Algorithms. Recall that Batdroids 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 Batdroid 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 $T(V) = f^{-1}(T(f(V)))$. An algorithm that is not well-defined is said to be *ambiguous*, as it may lead to different movements depending on the choice of the transformation f done by the adversary (see Figure 2 for an example). Note that a well-defined algorithm does not contain any ambiguous rule.

We denote by $C \rightsquigarrow C'$ the fact that C' can be reached in one round from C (*n.b.*, $\rightsquigarrow$ is then a binary relation over configurations).

An *execution* of Algorithm A in a 3D grid G is then a sequence $(C_i)_{i\in\mathbb{N}}$ of configurations such that $C_0 \in Init(G)$ and $\forall i \geq 0$, $C_i \rightsquigarrow C_{i+1}$.

The Perpetual Exploration Problem. An execution $(C_i)_{i\in\mathbb{N}}$ in a 3D grid $G = (V, E)$ *achieves the Perpetual Exploration Problem* (PEP) 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 Batdroids solves the PEP if, for every cave (*i.e.*, any 3D grid with $S_x \geq 4$, $S_y \geq 4$, $S_z \geq 4$) and every initial configuration $C_0 \in Init(G)$, we have every execution of A in G starting from C_0 that achieves the PEP.

3 Impossibility Results

From [5], we know that solving the perpetual exploration problem in a cave under visibility range one requires at least three Batdroids, regardless the number of available colors and even if they share a common chirality and the well-definedness requirement is removed. Below, we complete the picture by showing that three Batdroids are not sufficient to solve the problem with a well-defined algorithm under visibility range one when Batdroids do not share a common chirality. In the next two sections, we will show that this impossibility can be circumvented in two ways: three Batdroids are sufficient if either the visibility range is increased by one (Section 4) or the well-definedness requirement is dropped (Section 5).

In the following, a Batdroid is said to be *isolated* if no other Batdroid is located on a node adjacent to its current position. Notice that an isolated Batdroid is not necessarily lost as it may be neighbor of a wall. Let $\mathcal{C}$ be a cave (*i.e.*, any 3D grid with $S_x \geq 4$, $S_y \geq 4$, $S_z \geq 4$). We call *internal node* any node of $\mathcal{C}$ of degree 6; any other node is said to be a *border node*. In particular, nodes of degree 3 are called *corners*. We call *slice* of $\mathcal{C}$ any subgraph of $\mathcal{C}$ that is isomorph to a 2D grid, contains four corner nodes, and whose nodes of degree at most 3 are border nodes of $\mathcal{C}$.

▶ **Theorem 1.** *There exists no well-defined algorithm allowing three Batdroids, under visibility one, to perpetually explore caves, regardless of the number of available light colors.*

Proof. Assume, by the contradiction, that there exists a well-defined algorithm allowing three Batdroids under visibility one to perpetually explore caves, using an arbitrary large number of light colors. In the following, we denote by B_1, B_2, and B_3 the three Batdroids.

Consider any cave $\mathcal{C}$ of size $S_x \times S_y \times S_z$ with $S_x = S_y = S_z \geq 13$ and S_x is even. First, remark that $\mathcal{C}$ has four centers.[1] Observe that *(*) if at some time, there exists a slice S of $\mathcal{C}$ containing a center and all three Batdroids, then exploration with a well-founded algorithm becomes impossible.* Indeed, (1) there are nodes of $\mathcal{C}$ outside S, and (2) any destination allowing a Batdroid to leave S is ambiguous. As a consequence, the three Batdroids remain forever trapped to S making the perpetual exploration fail.

Next, as there are three Batdroids and four centers in the cave, at least one of them, say c, is initially empty. So, at least one Batdroid, say B_1, should eventually visit c, say at time $t > 0$. Remark then that at time $t-1$, B_1 was on a neighboring node of c. Moreover, B_1 was not lost since otherwise the move would have been ambiguous. Hence, *(**) at least one Batdroid was neighbor of B_1 at time $t-1$.* At time t, four cases are possible:

1. *A Batdroid is neighbors of the two others.*
   Then, the three Batdroids are on the same slice of $\mathcal{C}$ containing c, and we obtain a contradiction from *(*)*.
2. *One Batdroid, say B_2, is neighbor of B_1 and the other B_3 is isolated.*
   In this case, during the next round, either B_3 is idle, or it is neighbor of a wall and move away from the wall. In this latter case, B_3 become lost: due to the size of $\mathcal{C}$, it is still not neighbor B_1 or B_2 after the round, whatever be the behavior of B_1 and B_2. Thus, at time $t+1$, B_3 is necessarily idle. Then, let L be the line passing through B_1 and B_2 at time t. For each of the two Batdroids B_1 and B_2, the only unambiguous destinations are located on L. So, they can only move along L. If, a time $t+1$, B_3 is neither on L, nor located on a node neighbor to a node of L, then B_1 and B_2 remain forever on L and B_3 is idle forever. Now, as there exist nodes of the cave different from the final location of B_3 and outside L, the perpetual exploration fails, a contradiction. Otherwise, to eventually leave L, B_1 and B_2 must move so that eventually at least one of them becomes neighbor of B_3. But, if this happens, the three Batdroids are on a slice S of $\mathcal{C}$ containing L: S contains a center of $\mathcal{C}$ (namely, c) and we obtain a contradiction by *(*)*.
3. *B_1 is isolated, yet B_2 and B_3 are neighbors.*
   In this case, while B_1 is isolated, it stays idle at c since all its possible destinations are ambiguous. Then, let L be the line passing through B_2 and B_3 at time t. At time $t-1$, before arriving at c, B_1 was on a neighboring node of c. Thus, another Batdroid, say B_2, was at distance at most 2 from c at time $t-1$ by *(**)* and so is at distance at most 3 from c at time t. B_3 being neighbor of B_2 at time t, B_3 is then at distance at most 4 from c at time t. Consequently, for each of the two Batdroids B_2 and B_3, the only unambiguous destinations are located on L: they can only move along L. If B_1 is neither on L, nor located on a node neighbor to a node of L, then B_2 and B_3 remain forever on L, and as there are nodes of $\mathcal{C}$ different from c and outside L, the perpetual exploration fails, a contradiction. Otherwise, to eventually leave L, they must move so that at least one of them eventually becomes neighbor of B_1, but in this case, we retrieve the first or second case, a contradiction.
4. *The three Batdroids are isolated.*
   In this case, B_1 is necessarily idle since lost. Another Batdroid, say B_2, is at distance at most 3 from c by *(**)*. So, due to the size of $\mathcal{C}$, B_2 is lost too and so idle. If B_3 is also idle, then the exploration fails, a contradiction. Otherwise, B_3 is neighbor of a wall at time t and so at distance at least four (resp. three) from B_1 (resp. B_2), it necessarily moves away from the wall, and it is lost at time $t+1$. Hence, all three Batdroids are idle at time $t+1$, a contradiction.

Hence, in all cases we obtain a contradiction and so the theorem holds. ◀

[1] A center is any node of minimum eccentricity.

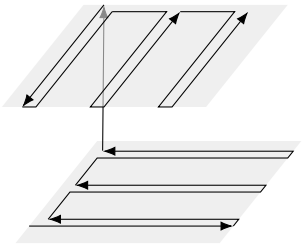

Figure 3 Overview of the route taken by the Batdroids.

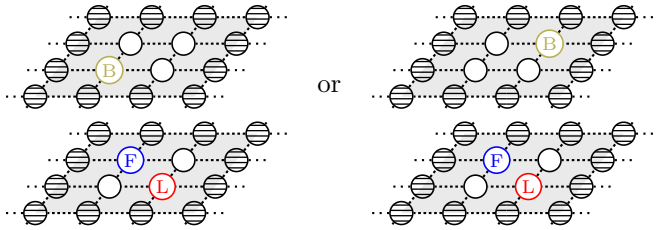

Figure 4 Initial configurations of $\mathsf{A}_{\mathsf{light}}$.

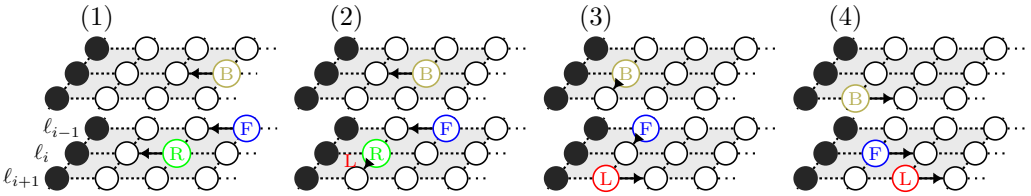

Figure 5 Sequence of configurations during a line change.

4 Well-defined Algorithm for 3 Batdroids, Visibility 2, and 4 Colors

In this section, we describe an exploration algorithm, called $\mathsf{A}_{\mathsf{light}}$, for the light mode of the Batdroids. It is well-defined, uses three Batdroids under visibility range two, and requires four colors. By Theorem 1, $\mathsf{A}_{\mathsf{light}}$ is optimal with respect to the number of Batdroids and the visibility range, given that it is well-defined. The reader is encouraged to follow the description of $\mathsf{A}_{\mathsf{light}}$ while looking at the online animations [3].

The idea of this algorithm is to perform the exploration of the 3D grid floor by floor. Each floor will be explored row by row, see Figure 3. To keep track of the exploration direction, the Batdroids use their relative positions and colors. Each Batdroid has a role, determined by its color: the *leader* (of color L or R), the *follower* (of color F), and the *back* Batdroid (of color B). Notice that a Batdroid may change its role during the exploration.

Initially, each Batdroid is placed at distance two from the other two. The leader has color L and is on the same floor as the follower. The back Batdroid is on the next floor, see Figure 4. This pattern can be arbitrarily placed on the 3D grid, thus the set of all possible initial configurations of $\mathsf{A}_{\mathsf{light}}$ is locally-defined [20]. By convention, let us say that the leader and the follower are on floor f_i while the back Batdroid is on floor f_{i-1}, and that the leader and back Batdroid are on line ℓ_i while the follower is on line ℓ_{i-1}. Notice that this pattern allows the robots to build and agree on a common orientation system based on their relative positions.

From such a position, the Batdroids start exploring line ℓ_i: the leader moves away from the two other Batdroids, while staying on the same floor (Rule R_1). The follower and back Batdroids follow in the same direction (Rules R_2 and R_3, respectively). Once they reach a wall, they turn back and the leader takes color R (Rule R_4). They return in the same way until they reach the opposite wall (Rules R_5-R_8).[2]

[2] Notice that doing so, line ℓ_{i-1} on floor f_i and line ℓ_i on floor f_{i-1} are explored by the follower and the back Batdroid, respectively, at the same time as line ℓ_i on floor f_i is explored by the leader.

Then, the Batdroids must move to the next line to explore. Thanks to their relative positions, they can determine it without ambiguity: the leader moves to the line on the opposite direction to the follower, *i.e.*, ℓ_{i+1}, and switches back to color L (Rules R_9 and R_{10}). The two others follow to return in their exploration pattern (Rules R_{11} and R_{12}, respectively). The sequence of configurations reached during this change of line is illustrated on Figure 5. They continue on their exploration of the floor line by line.

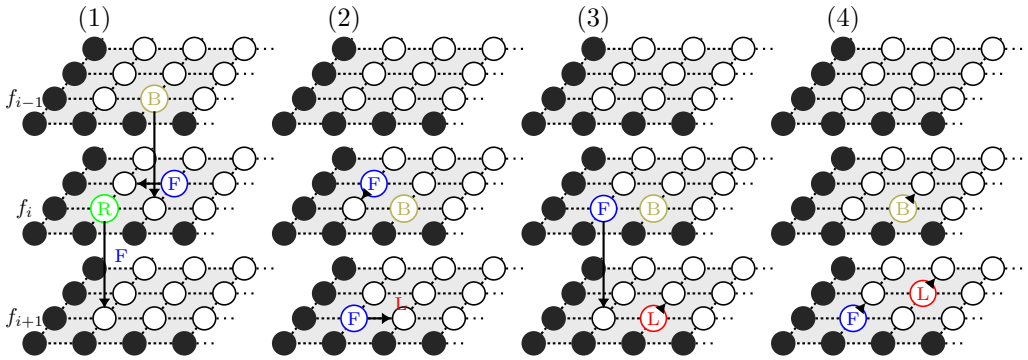

Figure 6 Sequence of configurations during a floor change.

After finishing the exploration of the last line of the floor, the Batdroids move to the next floor, in the opposite direction to the one pointed by the back Batdroid, *i.e.*, floor f_{i+1}. Again, thanks to their relative positions and using the walls as landmark, the Batdroids can determine their directions without ambiguity (Rules R_{13}-R_{18}). Figure 6 illustrates the sequence of configurations encountered during this change of floor. Then, they can start exploring this floor just as the previous one, but they are rotated by $\frac{\pi}{2}$.

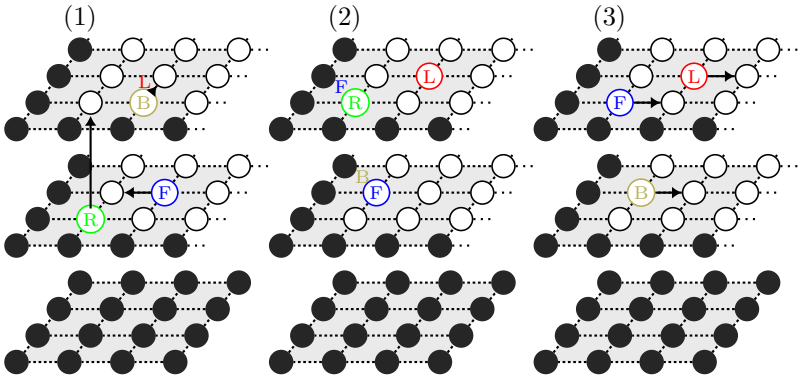

Figure 7 Sequence of configurations to switch the exploration direction in the corner of the grid.

The Batdroids eventually explore each row repeating this pattern. When they have finished the exploration of the bottom (or top) floor, they change their exploration direction. The sequence of configurations used is described on Figure 7. Notice that the Batdroids switch their roles: the back Batdroid becomes the leader (Rule R_{19}), the leader becomes follower (Rules R_{20} and R_{21}), and the follower becomes the back Batdroid (Rule R_{22}). The rules of Algorithm $\mathsf{A}_{\mathsf{light}}$ are listed in Figures 8 and 9.

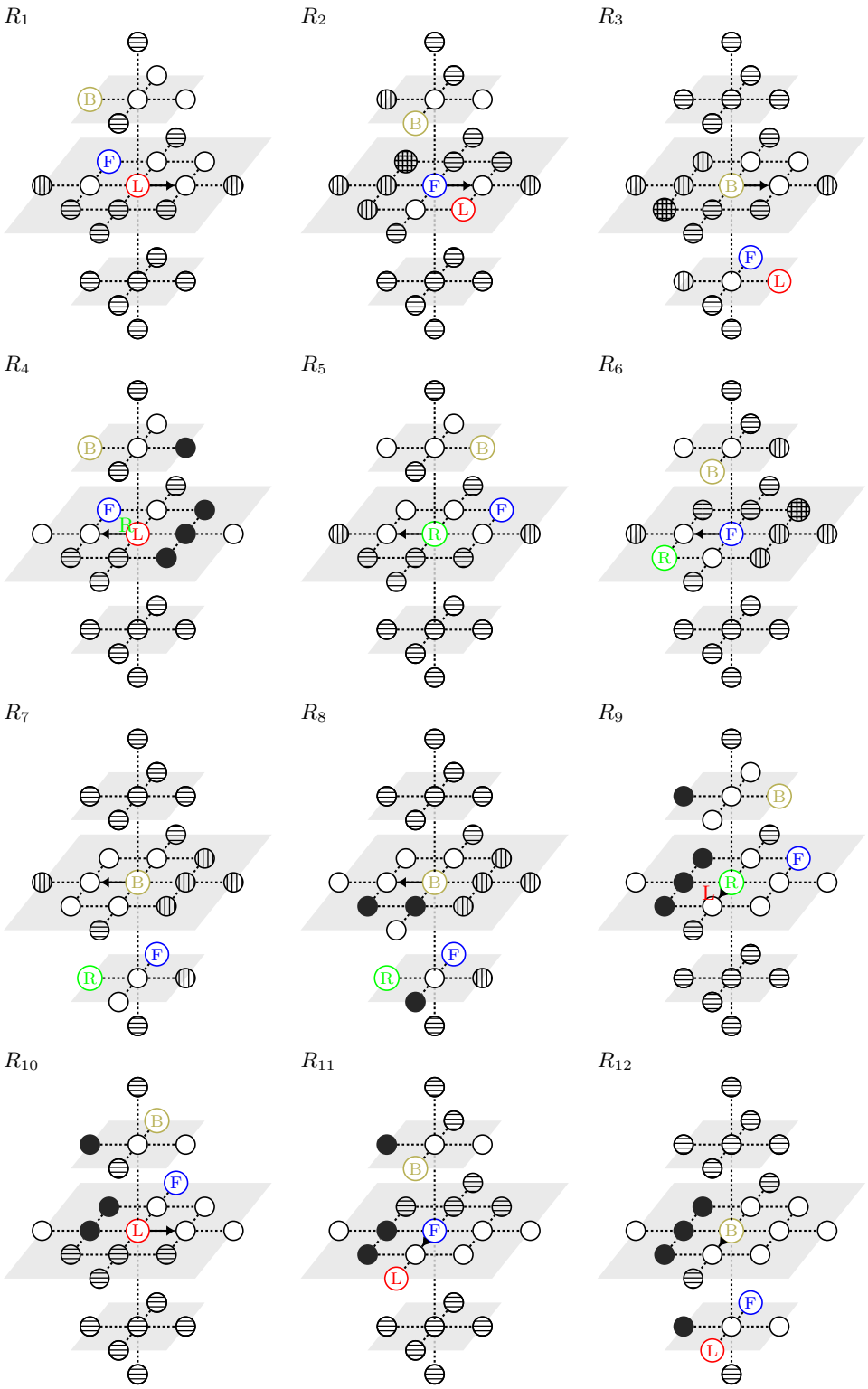

Figure 8 Rules of Algorithm $\mathsf{A}_{\mathsf{light}}$.

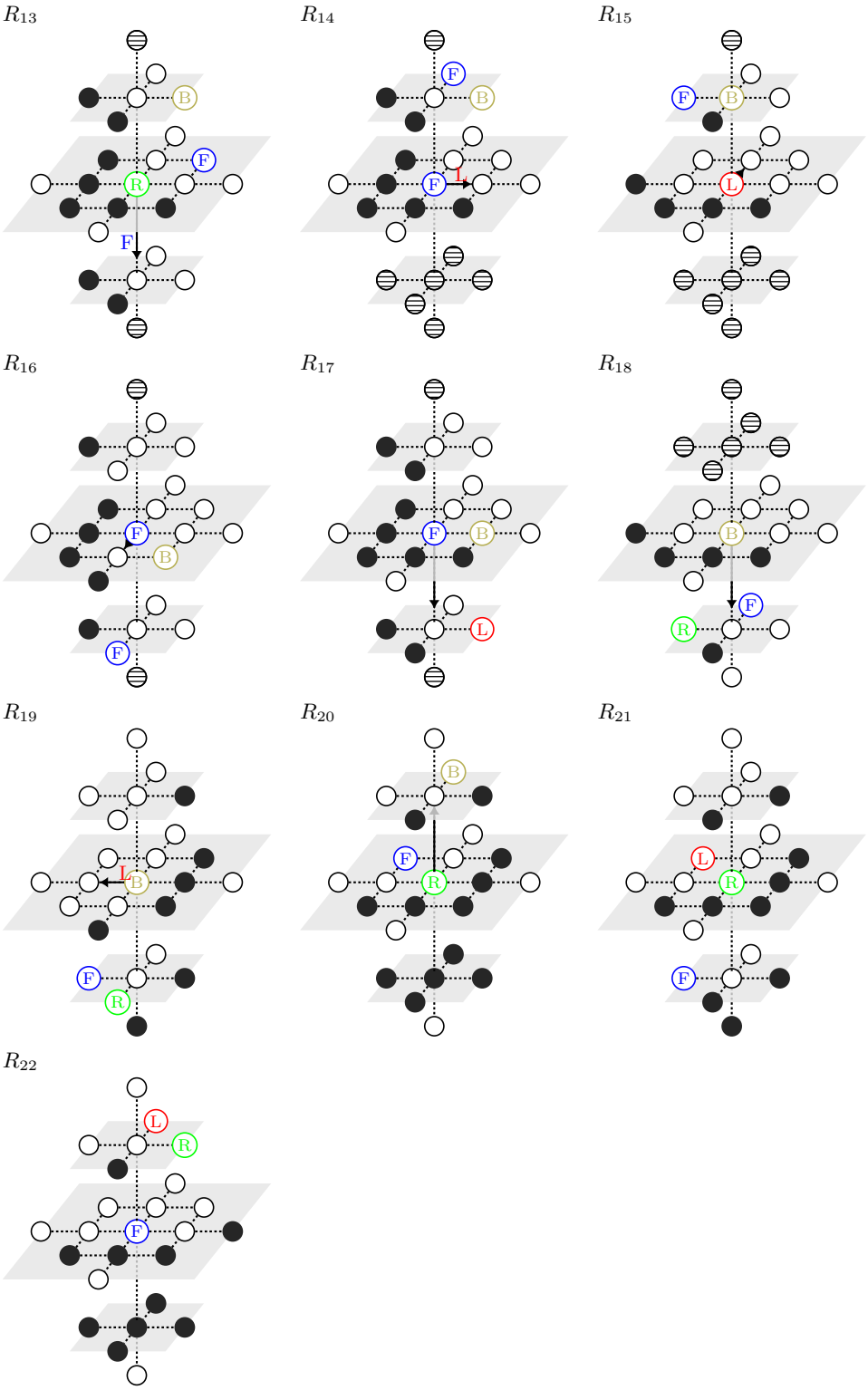

Figure 9 Rules of Algorithm $\mathsf{A}_{\mathsf{light}}$ (next).

▶ **Theorem 2.** $\mathsf{A}_{\mathsf{light}}$ *solves PEP with three Batdroids, four colors, under visibility range two and without common chirality.*

Proof. Without loss of generality, let assume that S_x, S_y, and S_z are respectively the width, the length, and the height of the 3D grid, and that the Batdroids start exploring the grid floor by floor in a bottom-up direction.

First, remark that $\mathsf{A}_{\mathsf{light}}$ proceeds in repetitive phases that are independent from the grid size. Let $\mathcal{C}$ be the initial configuration. This configuration is periodically reached and, in between two occurrences of $\mathcal{C}$, every node of the grid must have been visited at least once. The finite exploration that happens in between two occurrences of $\mathcal{C}$ is composed of the following phases: The grid is explored floor by floor, until reaching the last one (*i.e.*, the S_z^{th}), from where it changes its exploration direction (from bottom-up to top-down, or conversely). Each floor is explored line by line. The size of the floor, *i.e.*, the length of a line and the number of lines, does not impact the way the exploration is performed. It only impacts the exploration time since Batdroids must repeat the same patterns for an additional number of times. However, the Batdroids do a $\frac{\pi}{2}$-turn when changing of floor. Thus, the exploration of two consecutive floors is analogous up to a $\frac{\pi}{2}$ rotation, meaning that the exploration of the i^{th} and j^{th} floors is analogous if $i \pmod 4 = j \pmod 4$.

Hence, we proceed by induction on $S_x \times S_y \times S_z$.

- *Base cases:* We have validated several base cases by simulation, namely every grid of size $S_x \times S_y \times S_z$ with $4 \leq S_x \leq 7$, $4 \leq S_y \leq 7$, and $4 \leq S_z \leq 7$. Since no rule is ambiguous and we consider a synchronous scheduler, this check is quite easy: We only have to perform one execution for each grid and each initial configuration $\mathcal{C}$. Once the Batdroids reach $\mathcal{C}$ again, we just have to check that every node of the grid has been visited.
- *Induction hypothesis:* Assume that $\exists X \geq 7, Y \geq 7, Z \geq 7$, such that $\mathsf{A}_{\mathsf{light}}$ solves the perpetual exploration in every 3D grid of size $S_x \times S_y \times S_z$ where $4 \leq S_x \leq X$, $4 \leq S_y \leq Y$, and $4 \leq S_z \leq Z$.
- *Induction step:* Using the previous arguments, we can easily show that the exploration of a grid G of size $(X+1) \times Y \times Z$ or $X \times (Y+1) \times Z$ is analogous to the exploration of a smaller grid G' of size $X \times Y \times Z$. Similarly the exploration of a grid G of size $X \times Y \times (Z+1)$ is analogous to the exploration of a smaller grid G' of size $X \times Y \times (Z-3)$. In both cases, the induction hypothesis holds on the smaller grid G' so $\mathsf{A}_{\mathsf{light}}$ also performs the perpetual exploration on G. ◀

5 Ambiguous Algorithm for 3 Batdroids, Visibility 1, and 77 Shades of Gray

In this section, we describe an exploration algorithm, called $\mathsf{A}_{\mathsf{dark}}$, for the dark mode of the Batdroids. It uses three Batdroids under visibility range one, and requires 77 shades of grey. Before describing and looking at the full algorithm, we explain the overall idea and techniques we use. The order of exploration is similar to the previous algorithm: the Batdroids explore the 3D grid floor by floor, each floor being explored row by row. However, since the visibility range is one, it becomes very challenging to keep track of the exploration progress. Indeed, the Batdroids can only see their adjacent nodes, which makes difficult to store and retrieve information about which parts of the grid are yet to be explored (what is the next row and the next floor to explore).

Informally, if the three Batdroids move together to explore a line like in the previous algorithm, they cannot "store" using their relative position three directions, which is necessary to remember in which direction they are exploring the current line, but also the direction to

the next line, and the direction to the next floor. This is because they can only be aligned or form an L shape, which only gives two directions. So one of the Batdroids has to stay behind to act as a beacon to store this information, while the two other Batdroids (called the traveling group) move together to explore the current line of the current floor.

Even with this trick (already used by Beedroids [5]), the beacon cannot be directly next to the traveling group when they return to the wall after exploring a line, because in this case, they would form an L-shape at some point and, again, they would be unable to distinguish between the next line and next floor directions. To overcome this issue, $\mathsf{A}_{\mathsf{dark}}$ employs a unique strategy that leverages a beacon robot to guide the exploration process. Both Batdroids of the traveling group reach the wall (which gives them one direction), and only after a few steps (detailed below) become a neighbor of the beacon Batdroid to get the other two directions.

The beacon search algorithm. We now describe the main idea of the algorithm in more detail. This technique takes place in a slice of the grid at the border of the 3D grid, which we represent as a 2D grid in Figure 10. In this grid, a number represents the shade of grey of the Batdroid (for simplicity we used numbers that differ slightly from the full algorithm). The beacon Batdroid has shade 10 and is initially placed diagonally at distance 2 in one direction and distance 1 orthogonally from the line explored by the traveling group (this line is orthogonal from the slice we are seeing).

In the first grid (Step 1 of Figure 10), we see the first Batdroid of the travelling group with shade 1, denoted ① (the other is not visible in this slice but is adjacent to it, in the opposite direction from the wall). When the Batdroid ① reaches the wall and has to move, it cannot distinguish any direction in the slice, so this ambiguous move is resolved by the adversary.

The key idea is that, regardless of the direction, the travelling group will search for the beacon Batdroid ⑩ in all four directions. Interestingly, we eventually and deterministically reach a configuration where the traveling group is adjacent to the beacon as in the last step of Figure 10, *regardless of the choices made by the adversary during the search process.*

For the sake of this example, consider that Batdroid ① first moves up, followed by Batdroid ⑪ (Step 2 in Figure 10). After two steps, counted by Batdroid ① that becomes ②, and since the beacon is not found, Batdroid ② becomes ③ to signal the follower that they have to search in the opposite direction. Batdroids exchange their roles: former leader is now follower with color ⑪, and former follower becomes leader taking color ④. After three steps in the opposite direction (counted by the Batdroid ④ that becomes ⑤ and then ⑥), since the beacon is still not found, Batdroid ⑥ signals the follower that they have to search in the orthogonal direction by taking color ⑦.

In Step 9 of Figure 10, Batdroid ⑧ moves orthogonally with an ambiguous move. Its next color is ① and we reach exactly the configuration that we would have reached if the first move of the search had been in this orthogonal direction. Thus, by repeating the same sequence of moves, we eventually reach the configuration where the travelling group is adjacent to the beacon Batdroid ⑩ (Step 11 of Figure 10). When this happens, Batdroid ② becomes ⑨ to signal the follower that they can start the next phase of the exploration. This last configuration is reached regardless of the choices made by the adversary during the search process (in step 1 and 9), which is crucial for the correctness of the algorithm.

Exploring the next line. Once the traveling group is adjacent to the beacon Batdroid, they can determine the three necessary directions to continue their exploration: order of exploration of nodes in each line (away from the adjacent wall), the order in which lines are

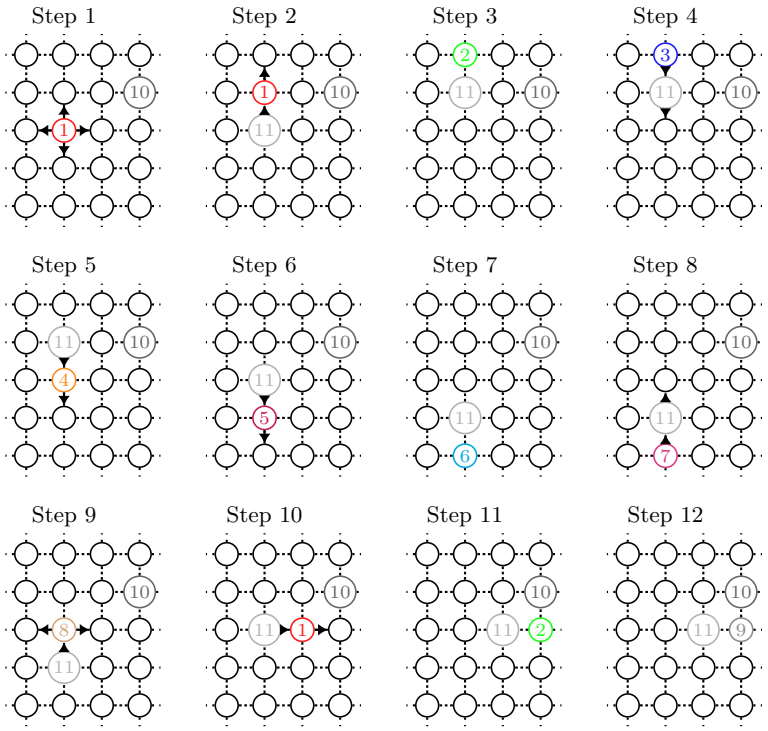

Figure 10 Search for the beacon robot by the two traveling robots when they arrive at a wall. The 2D grid we see is a slice at the border of the 3D grid.

explored (from left to right in the figure), and the order in which floors are explored (from bottom to top in the figure). From step 11, the next line to be explored is the line that is orthogonal to the slice we are seeing and passing through batdroid (11), which is in fact the line that is parallele and at distance one from the line the travelling group just explored in step 1.

After a sequence of moves, the three Batdroids reach the same configuration as when they started exploring the previous line, but translated to the next line. So the exact same sequence of moves can be repeated to explore this new line. This process is repeated until the entire floor is explored.

Exploring the next floor. After exploring the penultimate line of the floor, the traveling group detects it as the beacon is now adjacent to a wall. After exploring the last line of the floor, a sequence of moves (again containing some ambiguous moves) is performed to move the three Batdroids to the next floor and start exploring the first line of this new floor. Here the Batdroids reach one of two possible configurations that are mirrored, but both are similar to the configuration formed when they started exploring the previous floor (but translated to the next floor and rotated by $\frac{\pi}{2}$), the exact same sequence of moves can be repeated to explore this new floor. This process is repeated until the entire 3D grid is explored.

Exploration of the last floor. The Batdroids detect when they are exploring the last floor. At this point, each sequence we described previously is slightly modified (actually simplified since we have the wall located above to help), eventually the beacon Batdroid reaches the last corner of the last floor. After exploring the last line of the last floor, a final sequence of moves is performed to turn the Batdroids around and start exploring the 3D grid in the opposite direction, repeating the same process as before.

When putting all the steps together, we obtain the full algorithm $\mathsf{A}_{\mathsf{dark}}$ that can be executed in the online simulator [4], where one can see the exploration and check what rule is executed by which Batdroid.

▶ **Theorem 3.** $\mathsf{A}_{\mathsf{dark}}$ *solves PEP with three Batdroids, 77 shades of gray, under visibility range one and without common chirality.*

Proof. The proof of this theorem follows the same structure as the proof of Theorem 2 in Section 4. The main difference is that the execution is not deterministic due to the ambiguous moves. It may seem intractable to check all the possible execution, however, as explained in the description of the algorithm, the key property that ensures the correctness of the algorithm is that, whenever an ambiguous move is performed at a time t, there exists a time $t' > t$ when we reach a set of configurations regardless of the choice of the adversary. Informally, if we reach a single configuration, we can ignore the configurations between t and t' every time an ambiguous move occurs. In some cases, for instance when we change floor, we may reach two configurations that differ by a rotation or a mirroring, but both configurations are equivalent for the exploration purpose.

The induction step of the proof, similar to the one of Theorem 2, can then be applied similarly since the exploration of a line does not depend on its length, nor does the exploration of a floor depend on its size, nor does the exploration of the 3D grid depend on its height.

So the correctness of the algorithm relies on the fact that an ambiguous move is followed by a sequence of moves that eventually leads to a configuration or a set of configurations that does not depend on the choice of the adversary. This property can be proved by simply checking that the configuration graph starting from a configuration C (where a Batdroid has an ambiguous move), is a Directed Acyclic Graph (DAG) with a sink C'.

For instance, there is a configuration in the execution of our algorithm where the three Batdroids form an L-shape (all adjacent to a wall) and we want to reach a configuration where one Batdroid is at distance two and three from the two other Batdroids respectively (to be adjacent to the beacon), as shown in Figure 11. The graph of configuration from this configuration is shown in Figure 12.

A similar graph can be built for the ambiguous moves shown in Figures 10. In all these cases, we can check that the configuration graph is a DAG with a unique sink, which proves the desired property.

Therefore, by checking all the ambiguous moves of the algorithm, we can conclude that the algorithm eventually reaches configurations that do not depend on the choices of the adversary after each ambiguous move.

Thus, the exploration proceeds as in a deterministic execution, which concludes the proof. ◀

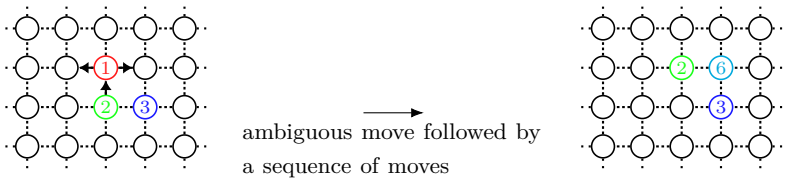

Figure 11 Example of configuration with an ambiguous move (left) that eventually leads to a configuration that does not depend on the choice of the adversary (right). The numbers are not the ones used by the algorithm.

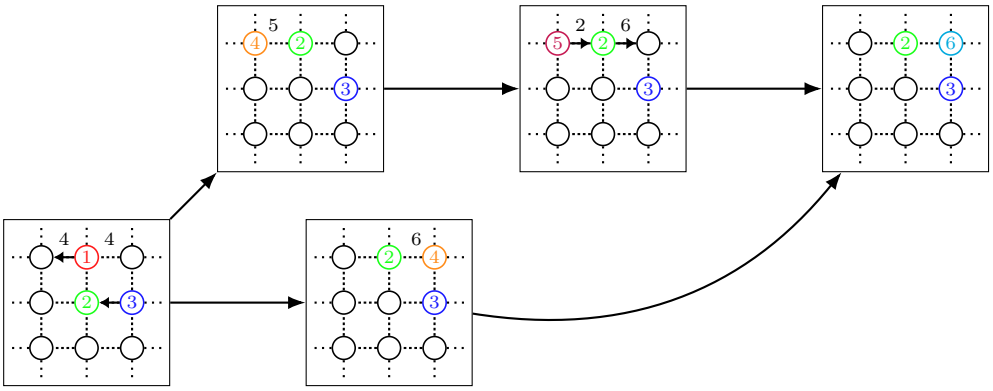

Figure 12 Example of configuration graph starting from a configuration with an ambiguous move.

6 Related Work

Colonies of Batdroids correspond to what are referred to as *swarms of luminous robots* in the literature [19]. Exploration of discrete environment by a robot swarm has been widely studied. Various topologies have been already considered including lines [16], rings [1, 9, 12, 17, 18], trees [15], torus [11], finite [2,7,10,20], infinite 2D grids [6,8], infinite n-dimensional grids [14], and 3D grids [5]. (In the infinite case, the exploration problem requires that each node is visited within finite time by at least one robot.)

In the context of finite graphs, two main variants of the exploration problem have been studied: the *terminating* and *perpetual* exploration. 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 [9–12, 15–17], while [1, 2, 5, 7, 20] deal with the perpetual exploration problem. Notice that Ooshita and Tixeuil consider the two variants of the problem in [18].

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 [1, 2, 10–12, 15–17]. 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, chirality has never been considered under such settings.

Assuming a common chirality is pretty usual in the 2D Euclidean plan; see *e.g.*, [13]. However, up to now only a few works dedicated to discrete environments, *e.g.*, infinite [8] and finite [5, 7] 2D grids, assume robots have a common chirality. Now, the common chirality has an impact on the number of robots necessary to solve exploration: for example, with visibility range one and three colors, two (resp. three) synchronous robots are necessary and sufficient to explore a finite 2D grid with (resp. without) the common chirality assumption [7, 20].

To the best of our knowledge, perpetual exploration of finite 3D grids has been investigated only in [5]. However, the robots in [5] are assumed to share a common chirality, whereas we do not make such an assumption here. Under the common chirality assumption, three robots with visibility range 1 are both necessary and sufficient to explore 3D grids using a well-defined algorithm. For sufficiency part, an algorithm requiring only five colors is proposed. In addition, another well-defined solution that is optimal with respect to the number of colors – using only a single color – is also presented; however, it requires a visibility range of 2.

Notice also that the exploration of an infinite n-dimensional grid has been investigated in [14]. In that paper, authors consider robots operating in two models: the semi-synchronous and synchronous ones. However, they do not impose the exclusivity at all since their robots can only sense the states of the robots located at the same node (in that sense, the visibility range is zero). Moreover, in contrast with our work, they assume all robots agree on a *global compass*, *i.e.*, they all agree on the same directions North-South and East-West. They propose several solutions and bounds, in particular they show that, in the semi-synchronous model, four deterministic robots are necessary and sufficient to explore an infinite 3D grid.

7 Conclusion

Upon a request of Bruce Wayne, we have studied how a small swarm of Batdroids can solve the perpetual exploration of a cave modeled by a 3D grids assuming the FSync model, without a common chirality. We first show that three Batdroids under the optimal visibility range one are not sufficient to solve the problem with a well-defined algorithm. We thus proposed an optimal solution in terms of number of robots and visibility range: a well-defined algorithm that requires three Batdroids, four colors, and visibility range two. Then, we have proposed another solution under the optimal visibility range of one, but at the cost of ambiguity. It is an ambiguous algorithm for three Batdroids that requires 77 shades of gray. However, Bruce Wayne is not completely satisfied yet. The gray lights produced by Wayne Enterprises are highly complex and fragile. Reducing the number of shades of gray required by our algorithm would allow to use more durable lights and ensure a better longevity of the Batdroids. So, we have to study whether we can reduce the number of shades of gray used by the second algorithm. We should also study whether the number of colors used in the first algorithm can be decreased. Finally, we should study whether this problem can be solved with oblivious Batdroids, *i.e.*, single-color Batdroids. This might require a larger visibility range and a greater number of Batdroids.

References

1 Lélia Blin, Alessia Milani, Maria Potop-Butucaru, and Sébastien Tixeuil. Exclusive perpetual ring exploration without chirality. In Nancy A. Lynch and Alexander A. Shvartsman, editors, *Distributed Computing, 24th International Symposium, DISC 2010, Cambridge, MA, USA, September 13-15, 2010. Proceedings*, volume 6343 of *Lecture Notes in Computer Science*, pages 312–327, Boston, Massachusetts, USA, September 2010. Springer. doi:10.1007/978-3-642-15763-9_29.

2 François Bonnet, Alessia Milani, Maria Potop-Butucaru, and Sébastien Tixeuil. Asynchronous exclusive perpetual grid exploration without sense of direction. In Antonio Fernández Anta, editor, *Proceedings of International Conference on Principles of Distributed Systems (OPODIS 2011)*, number 7109 in Lecture Notes in Computer Science (LNCS), pages 251–265, Toulouse, France, December 2011. Springer Berlin / Heidelberg. doi:10.1007/978-3-642-25873-2_18.

3 Quentin Bramas. Animation of the first algorithm, $\mathsf{A}_{\mathsf{light}}$, 2026. URL: https://robots.app.bramas.fr/?unpublished/8.

4 Quentin Bramas. Animation of the second algorithm, $\mathsf{A}_{\mathsf{dark}}$, 2026. URL: https://robots.app.bramas.fr/?unpublished/7.

5 Quentin Bramas, Stéphane Devismes, Anaïs Durand, Pascal Lafourcade, and Anissa Lamani. Beedroids: How luminous autonomous swarms of uavs can save the world? In *11th International Conference on Fun with Algorithms, FUN 2022, Island of Favignana, Sicily, Italy, May 30 - June 3, 2022*, volume 226 of *LIPIcs*, pages 7:1–7:21, 2022. doi:10.4230/LIPIcs.FUN.2022.7.

6 Quentin Bramas, Stéphane Devismes, and Pascal Lafourcade. Infinite grid exploration by disoriented robots. In Chryssis Georgiou and Rupak Majumdar, editors, *Networked Systems - 8th International Conference, NETYS 2020, Marrakech, Morocco, June 3-5, 2020, Proceedings*, volume 12129 of *Lecture Notes in Computer Science*, pages 129–145. Springer, 2020. doi:10.1007/978-3-030-67087-0_9.

7 Quentin Bramas, Stéphane Devismes, and Pascal Lafourcade. Optimal Exclusive Perpetual Grid Exploration by Luminous Myopic Opaque Robots with Common Chirality. In *ICDCN'21: International Conference on Distributed Computing and Networking, Virtual Event*, pages 76–85, Nara, Japan, 5-8 january 2021. ACM. doi:10.1145/3427796.3427834.

8 Quentin Bramas, Pascal Lafourcade, and Stéphane Devismes. Finding water on poleless using melomaniac myopic chameleon robots. In Martin Farach-Colton, Giuseppe Prencipe, and Ryuhei Uehara, editors, *10th International Conference on Fun with Algorithms, FUN 2021, May 30 to June 1, 2021, Favignana Island, Sicily, Italy*, volume 157 of *LIPIcs*, pages 6:1–6:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.FUN.2021.6.

9 Ajoy Kumar Datta, Anissa Lamani, Lawrence L. Larmore, and Franck Petit. Enabling ring exploration with myopic oblivious robots. In *2015 IEEE International Parallel and Distributed Processing Symposium Workshop, IPDPS 2015*, pages 490–499, Hyderabad, India, May 25-29, 2015 2015. IEEE Computer Society. doi:10.1109/IPDPSW.2015.137.

10 Stéphane Devismes, Anissa Lamani, Franck Petit, Pascal Raymond, and Sébastien Tixeuil. Terminating exploration of A grid by an optimal number of asynchronous oblivious robots. *Comput. J.*, 64(1):132–154, 2021. doi:10.1093/comjnl/bxz166.

11 Stéphane Devismes, Anissa Lamani, Franck Petit, and Sébastien Tixeuil. Optimal torus exploration by oblivious robots. *Computing*, 101(9):1241–1264, 2019. doi:10.1007/S00607-018-0595-8.

12 Stéphane Devismes, Franck Petit, and Sébastien Tixeuil. Optimal probabilistic ring exploration by semi-synchronous oblivious robots. *Theoretical Computer Science (TCS)*, 498:10–27, 2013. doi:10.1016/J.TCS.2013.05.031.

13 Yoann Dieudonné, Franck Petit, and Vincent Villain. Leader election problem versus pattern formation problem. In Nancy A. Lynch and Alexander A. Shvartsman, editors, *Distributed Computing, 24th International Symposium, DISC 2010*, volume 6343 of *Lecture Notes in Computer Science*, pages 267–281, Cambridge, MA, USA, september 13-15 2010. Springer. doi:10.1007/978-3-642-15763-9_26.

14 Stefan Dobrev, Lata Narayanan, Jaroslav Opatrny, and Denis Pankratov. Exploration of high-dimensional grids by finite automata. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, volume 132 of *LIPIcs*, pages 139:1–139:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.139.

15 Paola Flocchini, David Ilcinkas, Andrzej Pelc, and Nicola Santoro. Remembering without memory: Tree exploration by asynchronous oblivious robots. *Theor. Comput. Sci.*, 411(14-15):1583–1598, 2010. doi:10.1016/j.tcs.2010.01.007.

16 Paola Flocchini, David Ilcinkas, Andrzej Pelc, and Nicola Santoro. How many oblivious robots can explore a line. *Inf. Process. Lett.*, 111(20):1027–1031, 2011. doi:10.1016/j.ipl.2011.07.018.

17 Paola Flocchini, David Ilcinkas, Andrzej Pelc, and Nicola Santoro. Computing without communicating: Ring exploration by asynchronous oblivious robots. *Algorithmica*, 65(3):562–583, 2013. doi:10.1007/S00453-011-9611-5.

18 Fukuhito Ooshita and Sébastien Tixeuil. Ring exploration with myopic luminous robots. In Taisuke Izumi and Petr Kuznetsov, editors, *Stabilization, Safety, and Security of Distributed Systems - 20th International Symposium, SSS 2018*, volume 11201 of *Lecture Notes in Computer Science*, pages 301–316, Tokyo, Japan, november 4-7 2018. Springer. doi:10.1007/978-3-030-03232-6_20.

19 David Peleg. Distributed coordination algorithms for mobile robot swarms: New directions and challenges. In Ajit Pal, Ajay D. Kshemkalyani, Rajeev Kumar, and Arobinda Gupta, editors, *Distributed Computing – IWDC 2005*, pages 1–12, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. doi:10.1007/11603771_1.

20 Arthur Rauch, Quentin Bramas, Stéphane Devismes, Pascal Lafourcade, and Anissa Lamani. Optimal exclusive perpetual grid exploration by luminous myopic robots without common chirality. In Karima Echihabi and Roland Meyer, editors, *Networked Systems - 9th International Conference, NETYS 2021, Virtual Event, May 19-21, 2021, Proceedings*, volume 12754 of *Lecture Notes in Computer Science*, pages 95–110. Springer, 2021. doi:10.1007/978-3-030-91014-3_7.

21 Ichiro Suzuki and Masafumi Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM J. Comput.*, 28(4):1347–1363, 1999. doi:10.1137/S009753979628292X.

22 Wantong Zhang, Fa Song, and Jiyu Sun. A review of strategic enhancement of pollination with smart agriculture to counteract the decline of natural pollinators. *Biosystems Engineering*, 261:104344, 2026. doi:10.1016/j.biosystemseng.2025.104344.