



**HAL**  
open science

## Beedroids: How Luminous Autonomous Swarms of UAVs Can Save the World?

Quentin Bramas, Stéphane Devismes, Anaïs Durand, Pascal Lafourcade,  
Anissa Lamani

### ► To cite this version:

Quentin Bramas, Stéphane Devismes, Anaïs Durand, Pascal Lafourcade, Anissa Lamani. Beedroids: How Luminous Autonomous Swarms of UAVs Can Save the World?. FUN: Conference on Fun with Algorithms, May 2022, Island of Favignana, Sicily, Italy. 10.4230/LIPIcs.FUN.2022.7 . hal-03746703

**HAL Id: hal-03746703**

**<https://uca.hal.science/hal-03746703v1>**

Submitted on 28 Oct 2022

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

# Beedroids: How Luminous Autonomous Swarms of UAVs can save the World?

**Quentin Bramas** ✉ 

University of Strasbourg, ICUBE, CNRS

**Stéphane Devismes** ✉ 

Université de Picardie Jules Verne, MIS UR 4290, Amiens

**Anaïs Durand** ✉ 

University Clermont Auvergne, CNRS UMR 6158, LIMOS

**Pascal Lafourcade** ✉ 

University Clermont Auvergne, CNRS UMR 6158, LIMOS

**Anissa Lamani** ✉ 

University of Strasbourg, ICUBE, CNRS

---

## Abstract

Bee extinction is a great risk for humanity. To circumvent this ineluctable disaster, we propose to develop beedroids, *i.e.*, small UAVs mimicking the behaviors of real bees. Those beedroids are endowed with very weak capabilities (short-range visibility sensors, no GPS, light with a few colors, ...). Like real bees, they have to self-organize together into swarms. Beedroid swarms will be deployed in cuboid-shaped greenhouse. Each beedroid swarm will have to indefinitely search for flowers to pollinate in its greenhouse. We model this problem as a perpetual exploration of a 3D grid by a swarm of beedroids. In this paper, we propose two optimal solutions to solve this problem and so to save humanity.

**2012 ACM Subject Classification** Theory of computation → Distributed algorithms

**Keywords and phrases** Bee extinction, luminous swarms of beedroids, perpetual flower pollination problem, greenhouse.

**Digital Object Identifier** 10.4230/LIPIcs.CVIT.2016.23

**Funding** This study was partially supported by the French ANR project ESTATE (ANR-16 CE25-0009-03).

## 1 Introduction

Bees are crucial for human beings. They have limited life span: only few weeks for the workers and up to 6 years for the queen. The United Nation (UN) dedicates the 20th of May as the “Bee day”.<sup>1</sup> UN says that “we all depend on the survival of bees”. Indeed, pollination is a fundamental process for the survival of our ecosystems. Nearly 90% of the world’s wild flowering plant species depend on pollination. There are more than 800 wild bee species, seven of which are classified by the International Union for Conservation of Nature<sup>2</sup> (IUCN) as critically endangered. A further 46 are endangered, 24 are vulnerable, and 101 are near threatened. Many associations like Greenpeace<sup>3</sup> or World Wild Foundation<sup>4</sup> (WWF) are protecting bees and helping to avoid their extinction.

Our goal here is to anticipate by considering the worst-case: the bee extinction. We propose solutions to save humanity using *Beedroids*. Beedroids are artificial bees that aim at pollinating

---

<sup>1</sup> <https://www.un.org/en/observances/bee-day>

<sup>2</sup> <https://www.iucn.org/>

<sup>3</sup> <https://cdn.buglife.org.uk/2019/08/CM-EofE-bee-report-2019-Headlines-FINAL-CJ.pdf>

<sup>4</sup> <https://www.greenpeace.org/static/planet4-international-stateless/2013/04/66f3eb6b-beesindecline.pdf>

## 23:2 Beedroids: How Luminous Autonomous Swarms of UAVs can save the World?

flowers autonomously in a greenhouse. Such a technology can also be used in the Mars colonization. For example, the Biosphere 2 project<sup>5</sup> was meant to demonstrate the viability of closed ecological systems to support and maintain human life in outer space. Beedroids are mandatory to implement such a project since it is not yet clear whether bees can survive interstellar trips.

A beedroid is a small autonomous Unmanned Aerial Vehicles (UAV) that mimics the behavior of a real bee. A famous ability of bees is stigmergy, *i.e.*, indirect communication through the movements. Implementing stigmergy requires perfect synchronization and visibility sensors. Consequently, we consider here a fully synchronous look-compute-move model of computation [20] (FSYNC). As explained before, beedroids will be deployed in greenhouses. So, each beedroid swarm will have to perpetually explore its greenhouse to find flowers and pollinate them. We assume greenhouses are finite cuboids. Each of these cuboids should be divided into cells to visit. Thus, we conveniently discretized a cuboid-shaped greenhouse as a finite 3D grid. Then, the problem we have to solve consists in coordinating a swarm of beedroids to perpetually explore a finite 3D grid in an exclusive manner (meaning that two beedroids cannot occupy the same place simultaneously).

The world bee population constantly decreases but is still incredibly huge as compared to human population. As a matter of facts, a bee colony is constituted of around 50 000 bees and in 2018 it was estimated that there were around 800 000 bee colonies in Canada for example. So, to eventually be able to replace bees, we should (1) solve pollination of greenhouses with the smallest possible number of beedroids and (2) also be able to massively produced them. Consequently, the design of beedroids should be as simple as possible. Below we list and motivate our main design choices.

**Visibility Sensors:** To maximize their flight time, we should save energy. Hence, the visibility range of beedroids should be as small as possible.

**Communication:** As previously explained, the communication between beedroids, like real bees, is indirect and based on positions of other bees. To vastly increase stigmergic communication, without compromising production cost and energy consumption, we have only endowed them with LED lights of a few colors that can be sensed by other beedroids within a short distance.

**Memory:** Still to save energy and manufacturing costs, beedroids have no permanent memory, except the color of 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 and energy consumption also prevent us from endowing beedroids with GPS. Instead, we use chirality facilities making beedroids able to distinguish the two sides of a symmetrically reflexive panorama.

**Contribution.** To bring our own stone to the world safeguarding, we propose to implement pollination into cuboid-shaped greenhouses using swarms of artificial bees, so-called beedroids. Solving this problem requires to coordinate each swarm so that it perpetually explores a 3D grid. As motivated before, we should both minimize the size of the swarm (*i.e.*, the number of beedroids that compose it) and the capabilities used by those beedroids.

We first study the problem in the FSYNC model assuming the optimal visibility range one. Under this assumption, we show that three beedroids are necessary and sufficient to solve the problem. For the sufficient part, we propose an algorithm that requires only five colors. Then, we look for another solution optimal in terms of colors, still in FSYNC model. Actually, the solution we propose works with oblivious beedroids, *i.e.*, beedroids endowed with only one light color. This second solution requires five beedroids and visibility range two.

In order to help the reader, online animations illustrating the behavior of our algorithms are available for our two solutions: [3] and [4].

---

<sup>5</sup> <https://biosphere2.org>

**Roadmap.** In the next section, we formally define the model, the beedroid skills, and the problem to solve, so-called the *Perpetual Flower Pollination Problem* (PFPP). In Section 3, we present the lower bound on the number of beedroids necessary to solve the PFPP under visibility range 1. In Sections 4 and 5, we present two algorithms solving the PFPP. Section 6 is dedicated to related work. Finally, we make concluding remarks in Section 7. Due to the lack of space, several proofs have been moved into the appendix.

## 2 Preliminaries

We consider a swarm of  $n > 0$  beedroids (*n.b.*,  $n$  is *a priori* unknown by beedroids) evolving in a greenhouse modeled as a *finite 3D grid* of size  $S_x \times S_y \times S_z$ , with  $S_x \geq n$ ,  $S_y \geq n$ ,  $S_z \geq n$ , *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  $E = \{\{(i, j, k), (i', j', k')\} : |i - i'| + |j - j'| + |k - k'| = 1\}$ . Note that coordinates are used for the analysis only, *i.e.*, beedroids cannot access them.

We assume discrete time and at each *round*, the beedroids synchronously perform a *Look-Compute-Move* cycle. In the *Look* phase, a beedroid gets a snapshot of the subgraph induced by the nodes within distance  $\Phi \in \mathbb{N}^*$  from its position.  $\Phi$  is called the *visibility range* of the beedroids. The snapshot is not oriented in any way as the beedroids do not agree on the orientation of any of the three axes of the coordinate system. However, it is implicitly ego-centered since the beedroid that performs a Look phase is located at the center of the subgraph in the obtained snapshot. Then, each beedroid *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 beedroids to occupy the same node simultaneously. A node is *occupied* when a beedroid is located at this node, otherwise it is *empty*. Beedroids have *lights* with maybe different colors that can be seen by beedroids within distance  $\Phi$  from them. We denote by  $Cl$  the set of all possible colors ( $|Cl| = 1$  corresponds to the case of oblivious beedroids).

The *state* of a node is either the color of the light of the beedroid 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  $\Phi$ ). During the compute phase, a beedroid may decide to change the color of its light (of course, if  $|Cl| > 1$ ).

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

**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 beedroid 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 beedroid  $r$ , *i.e.*, the subset of the configuration containing the states of the nodes at distance at most  $\Phi$  from  $r$ , translated so that the coordinates of  $r$  is  $(0, 0)$ . We use this globally oriented view in our analysis to describe the movements of the beedroids (see, for example, Figure 1): when we say “the beedroid moves Left”, it is according to the globally oriented view. However, since beedroids do not agree on any axis, they have no access to the globally oriented view. When a beedroid looks at its surroundings, it instead obtains a *local view*. To model this, we assume that the local view acquired by a beedroid  $r$  in the Look phase is the result of an arbitrary *indistinguishable transformation* on  $G_r$ . Here, we assume that beedroids 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. Nevertheless, beedroids share a common chirality, which means that their local view is not a reflection (also called mirroring) of the global view. Having a common chirality allows a beedroid to distinguish a local view from its reflection and so take different decisions in such cases (e.g., chirality allows to discriminate Above from Below in Rule  $R_1$  of Figure 1). In other words, having a common chirality allows, given two axes, to determine a third one using the right-hand rule.

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

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

**Algorithm.** An algorithm  $A$  is a tuple  $(Cl, \text{Init}, T)$  where  $Cl$  is the set of possible colors,  $\text{Init}$  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  $\text{Views} \rightarrow \{\text{Idle}, \text{Front}, \text{Back}, \text{Left}, \text{Right}, \text{Above}, \text{Below}\} \times Cl$ , where  $\text{Views}$  is the set of local views. When the beedroids 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  $f^{-1} \circ T \circ f \circ V_C(i, j, k) = (\text{Idle}, c')$ ,
- $((i - 1, j, k), c) \in C$  and  $f^{-1} \circ T \circ f \circ V_C(i - 1, j, k) = (\text{Right}, c')$ ,
- $((i + 1, j, k), c) \in C$  and  $f^{-1} \circ T \circ f \circ V_C(i + 1, j, k) = (\text{Left}, c')$ ,
- $((i, j - 1, k), c) \in C$  and  $f^{-1} \circ T \circ f \circ V_C(i, j - 1, k) = (\text{Front}, c')$ ,
- $((i, j + 1, k), c) \in C$  and  $f^{-1} \circ T \circ f \circ V_C(i, j + 1, k) = (\text{Back}, c')$ ,
- $((i, j, k - 1), c) \in C$  and  $f^{-1} \circ T \circ f \circ V_C(i, j, k - 1) = (\text{Above}, c')$ , or
- $((i, j, k + 1), c) \in C$  and  $f^{-1} \circ T \circ f \circ V_C(i, j, k + 1) = (\text{Below}, c')$ .

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  $C_0 \in \text{Init}(G)$  and  $\forall i \geq 0, C_i \rightsquigarrow C_{i+1}$ .

**The Perpetual Flower Pollination Problem.** An execution  $(C_i)_{i \in \mathbb{N}}$  in a 3D grid  $G = (V, E)$  achieves the *Perpetual Flower Pollination Problem* (PFPP) 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  $n$  beedroids solves the PFPP problem if, for every finite 3D grid  $G = (V, E)$  of size at least  $n \times n \times n$  and every initial configuration  $C_0 \in \text{Init}(G)$ , we have every execution of  $A$  in  $G$  starting from  $C_0$  that achieves the PFPP.

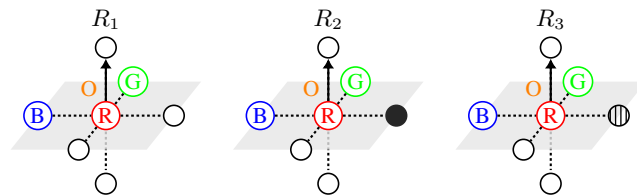
**An Algorithm as a Set of Rules.** We write an algorithm as a set of rules, where a *rule* is a triplet  $(V, d, c) \in \text{Views} \times \{\text{Idle}, \text{Front}, \text{Back}, \text{Left}, \text{Right}, \text{Above}, \text{Below}\} \times Cl$ . We say that an algorithm  $(Cl, \text{Init}, T)$  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 beedroids having a visibility range of one. This rule means that, when a beedroid sees two beedroids, one with Color  $B$  on its left and the other with color green in front of it, then the red beedroid is dictated to move Above and change its color to  $O$ . By extension, the same rule applied if the view is

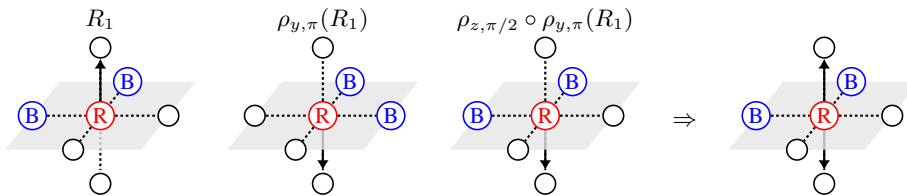
rotated by  $\pi$  on the  $z$ -axis, but in that case, the destination would be Below.

In the same figure, Rule  $R_2$  is a rule where a black node represent 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 several time rules with very similar views, we propose a notation to represent several rules in just one picture. For example, Rule  $R_3$  in 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 beedroid has a symmetric view so that, depending on the transformation  $f$  chosen by the adversary, the beedroid 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.



■ **Figure 1** Examples of rules. Colored letters inside nodes indicate the color of the beedroids occupying the nodes. The arrow indicates the destination and when a colored letter is given next to an arrow, this means that the rule dictates the beedroid to switch to that color.



■ **Figure 2** An ambiguous rule. Since the destination depends on the choice of the adversary, we represent the rule abusively with multiple destinations, as illustrated on the right. Observe that if blue beedroids had different colors (as in Figure 1) the rule would not have been ambiguous. Indeed, having a common chirality allows the beedroid to determine a third direction, given two direction obtained from the view.

### 3 Impossibility Results

In this section, we establish that there is no algorithm solving the PFPP problem in 3D grids using two beedroids with visibility range one, whatever the finite number of available colors.

We first observe that in large enough grids, if beedroids travel a long distance without seeing a wall, then they execute a periodic sequence of movements. Indeed, in our settings, there are at most  $B = \binom{|Cl|}{2} = \frac{|Cl|(|Cl|+1)}{2}$  different views without wall, and so at most  $B$  associated rules, where  $Cl$  is the set of available colors. Thus, if the two beedroids travel a distance at least  $B$  without seeing a wall, then they are executing a periodic sequence of movements. The definition of  $B$ , which depends on the algorithm, will be used throughout this section.

The above observation is important to prove our impossibility results. Actually, the outline of our proof is similar to the 2D case for luminous non-chiral robots studied in [19]. However, the



existence of an axis of symmetry is replaced here by an axis of rotational symmetry. Indeed, robots in a 2D grid that do not agree on a common chirality cannot distinguish between two destinations that are symmetric with respect to an axis of symmetry. Similarly, two beedroids that do not agree on a common coordinate system, but agree on a common chirality, cannot distinguish between four destinations that are symmetric with respect to a rotational symmetry. Nevertheless, the proof of the main argument, given in Lemma 1 of [19], cannot be adapted directly since it is more complex than in the 2D case.

The overview of the proof is as follows. We proceed by contradiction assuming that an algorithm solving the PFPP in 3D grids exists. Then, we show that once beedroids move far away from the walls, their possible movements are restricted. In more details, they can only move straight, otherwise they may not explore the whole grid. Next, we show that, once the exploration has reached specific places, the beedroids must always stay close to at least one wall (Lemma 1), leading to the final contradiction (Theorem 2). Due to the lack of space, several technical lemmas and proofs have been moved in Appendix A.

The lemma below is the cornerstone of our impossibility result. It states that there are configurations from which the two beedroids can remain forever at bounded distance from walls. To see this, we need to define a few concepts beforehand. We say that a beedroid is *2-close* if it is at distance at most  $2B$  from at least two walls. We say that beedroids are in a *T-configuration* if there is a line  $L$  of the 3D grid such that

- $L$  contains the two beedroids and
- one of them is adjacent to a wall that is orthogonal to  $L$ .

► **Lemma 1.** *Let  $A$  be an algorithm solving the PFPP in 3D grids using two beedroids under visibility range one. Assume some execution  $E$  reaches at a given time  $t$  (i) a configuration where a beedroid is 2-close or (ii) a T-configuration. Let  $C$  be the configuration of  $E$  at time  $t$ . Then, there exists an execution  $E'$  and a time  $t' > t$  such that*

- $C$  is reached in  $E'$  at time  $t$ ,
- at time  $t'$  in  $E'$ , a beedroid is 2-close or the system is in a T-configuration, and
- between time  $t$  and  $t'$  in  $E'$ , the beedroids remain at distance at most  $4B$  from a wall.

**Sketch of proof.** The following sketch is illustrated with Figures 3 and 4. Consider first the case where a beedroid is 2-close *i.e.*, at distance at most  $2B$  from two walls  $W_1$  and  $W_2$ . Due to the visibility range, the other beedroid must be located at a neighboring node. Then, we can show that to explore the whole 3D grid, the two beedroids must eventually be not 2-close. Now to become not 2-close, they have to move away from one wall, say  $W_1$ . Consider the first time they become not 2-close. We can show that they are in a line parallel to another wall say  $W_2$  and at distance at most  $2B$  from that wall; see the solid line in Figure 3.

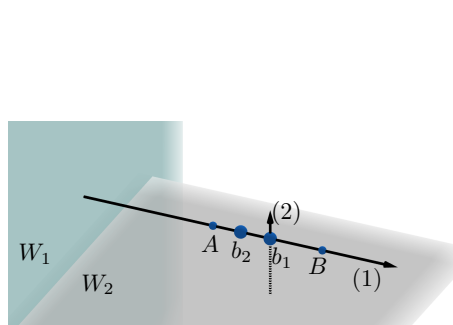
Assume that both beedroids do not see  $W_2$  when they become not 2-close. In this case, they see no wall at all (otherwise they are still 2-close). Two situations can happen. Either the beedroids move in a straight line for a distance  $B$  (Case (a)), or they eventually move away from the line (Case (b)).

In Case (a), they move in a straight line and, after  $B$  rounds without seeing any wall, they perform a periodic sequence of movements. So, they continue to move straight until reaching the wall opposite to  $W_1$ . Consequently, one of them eventually becomes 2-close again while the two beedroids were constantly at distance at most  $2B$  from  $W_2$ , and the lemma holds.

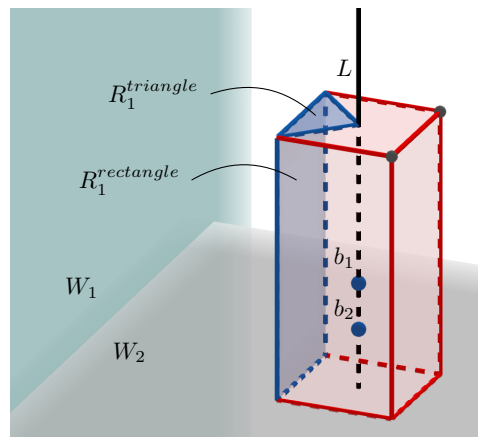
In Case (b), at least one of them move away from the line by performing an ambiguous move before  $B + 1$  rounds elapse. We can show that in this case the adversary can enforce them to reach a line  $L$  orthogonal to  $W_2$ . We consider the square cuboid for which (1) one face is at distance 1 from  $W_2$ , (2) the opposite one is at distance  $2B$  from  $W_2$ , (3) the closest face parallel to  $W_1$  is at distance  $2B$  from  $W_1$ , and (4)  $L$  is an axis of rotational symmetry; see Figure 4. We can then show

that there is an execution where the beedroids leave the square cuboid either from  $R_1^{rectangle}$  or from  $R_1^{triangle}$ . In the former case, a beedroid becomes 2-close again while the two beedroids remain at distance at most  $2B$  from a wall, and the lemma holds. In the latter case, the beedroids traveled  $B$  rounds without seeing a wall. As a consequence, they perform a periodic sequence of movements in a straight line, parallel to  $W_1$ , until reaching the wall opposite to  $W_2$  while remaining at distance less than  $3B$  from  $W_1$ , and reach a T-configuration. So, the lemma also holds in this case.

The case where the beedroids see  $W_2$  when they become not 2-close is treated similarly but requires to increase the distance from  $W_1$  by  $B$  in Case (b). Finally, if we start from a T-configuration, again the proof is similar. ◀



■ **Figure 3** Position of the beedroids when they become not 2-close.



■ **Figure 4** Position of the beedroids if they execute an ambiguous move after becoming not 2-close.

Using the previous lemma, we can now conclude.

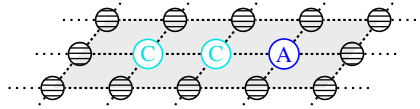
► **Theorem 2.** *The PFPP is not solvable using two beedroids under visibility range 1 and any finite number of colors.*

**Proof.** Assume that algorithm A solves the problem. Consider a grid of size  $10B \times 10B$ . Since the beedroids explore the entire grid, there exists a round where a beedroid is 2-close. By applying Lemma 1 repeatedly, we can construct an execution from there where beedroids forever remain at distance at most  $4B$  from a wall, so that nodes at distance more than  $4B$  from all the walls are not visited anymore, a contradiction. ◀

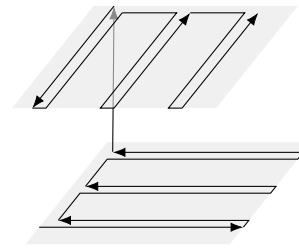
**4 Visibility range one:  $Vone_5^3$**

In this section, we address the PFPP using beedroids under visibility range one. We present an algorithm, denoted by  $Vone_5^3$ , that solves PFPP using three beedroids endowed with five colors. By Theorem 2, under visibility range one,  $Vone_5^3$  is optimal with respect to the number of beedroids. We encourage the reader to follow the overview of  $Vone_5^3$  while looking at the animations available online [3], published as an additional material. The 3D grid can be seen as a building with several floors. The overall idea of the proposed algorithm is to make the beedroids explore the 3D grid floor by floor, as illustrated in Figure 6.



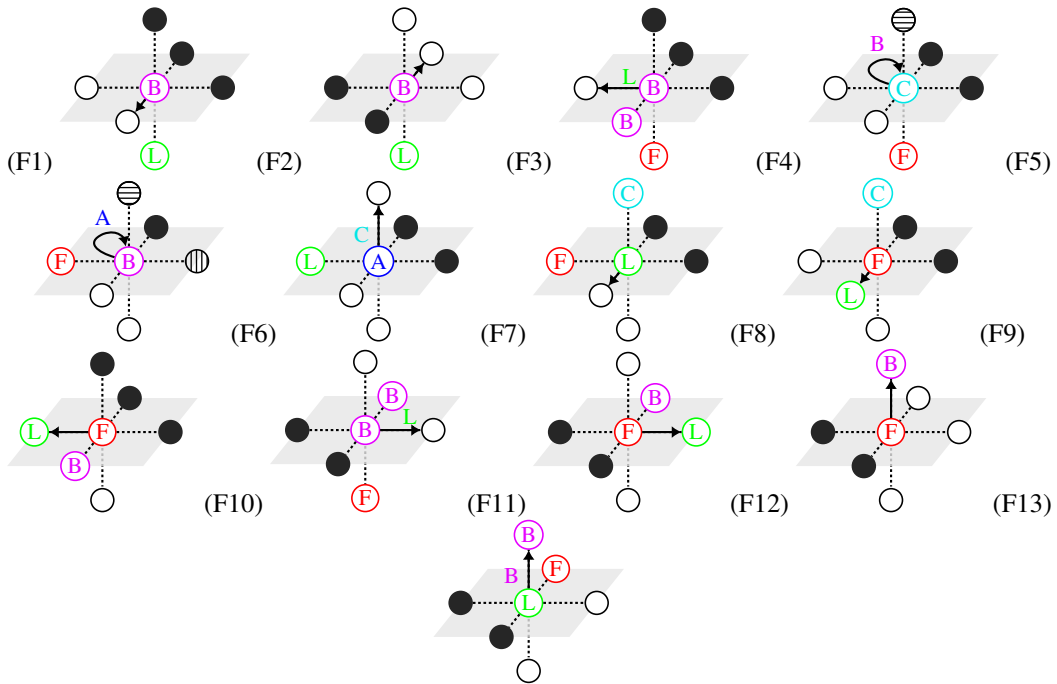


■ **Figure 5** Initial configuration of the beedroids



■ **Figure 6** Overview of the journey made by the exploring beedroids

While on a given floor, two beedroids will be in charge of exploring the floor line by line while the third one will be used as a landmark to keep track of the exploring direction: it will either designate the next line or the next floor to explore. Thus, the algorithm defines three main roles for the beedroids using colors: *Leader* (L), *Follower* (F) and *Landmark*. We use three different landmarks A, B, and C to distinguish different situations. Notice that a beedroid can change its role several times during the execution.

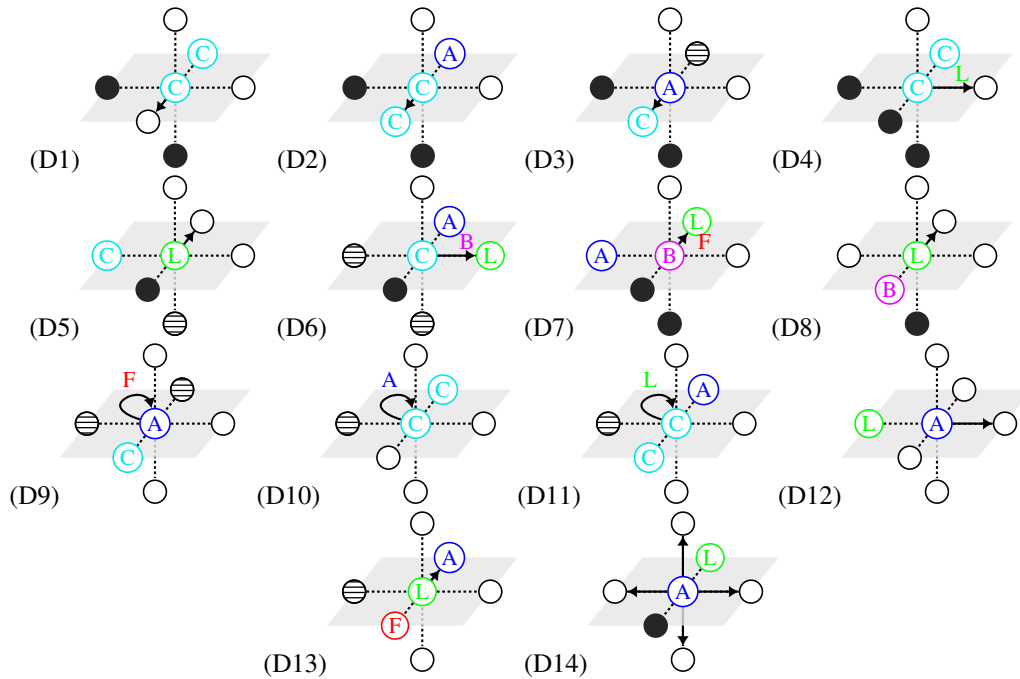


■ **Figure 7** Moving to the next floor.

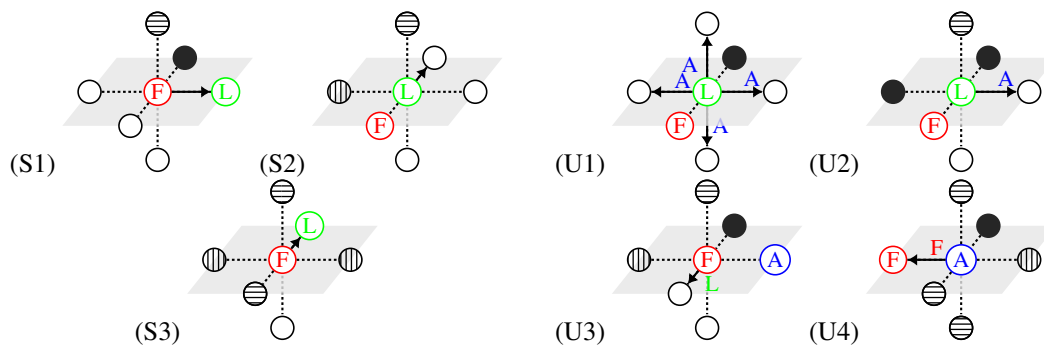
Initially, beedroids respectively have colors C, C, and A, as shown in Figure 5. They are aligned and one beedroid with color C should be adjacent to the two others. This pattern can be arbitrary placed on the 3D grid. In that sense, the set of all possible initial configurations is locally-defined [6]. Starting from any such locally-defined initial configuration, beedroids first move towards a wall. If they are aligned along an edge of the 3D grid, they do so while keeping their respective color. Otherwise, they first switch to the color sequence A, L, F. Once a wall is reached, beedroids coordinate together to reach a particular kind of configurations, denoted by  $C_p$  in the following, which will correspond to the effective start of the exploration. In other word, the initial prefix leading to a configuration  $C_p$  occurs only once. Then, the system periodically goes through Configurations  $C_p$

and all nodes are visited between two occurrences of them in the execution.

A configuration is of type  $C_p$  if the beedroids are located on two adjacent lines  $\ell_i$  and  $\ell_{i+1}$  of the 3D grid such that  $\ell_i$  hosts two adjacent beedroids colored  $F$  and  $L$  respectively at distance 2 and 3 from the same wall  $\mathcal{L}$ ; and  $\ell_{i+1}$  hosts a single beedroid with color  $A$  and adjacent to  $\mathcal{L}$ . The rules to reach a configuration  $C_p$  from a locally-defined initial configuration are given in Figure 8.



■ **Figure 8** Rules executed from a locally-defined configuration. Colored letters inside nodes indicate the color of the beedroids occupying the nodes. The arrow indicates the destination and when a colored letter is given next to an arrow, this means that the rule dictates the beedroid to switch to that color. Finally, self-loops indicate when a beedroid stays idle.

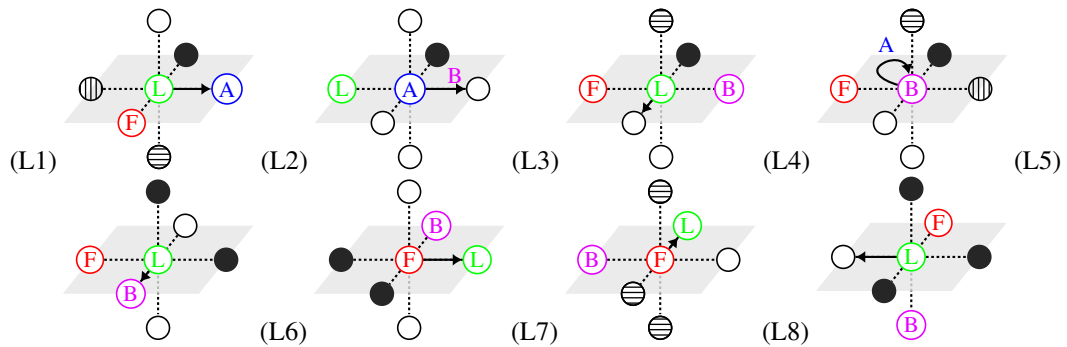


■ **Figure 9** Moving in a straight line.

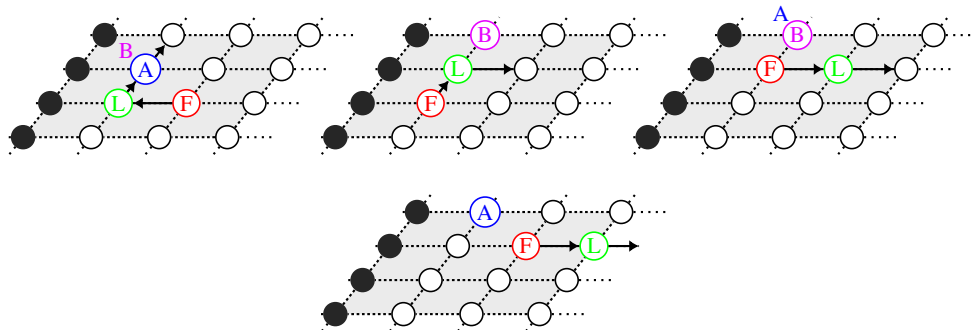
■ **Figure 10** U-turn.

As explained before, from configuration  $C_p$ , the beedroids will perform a periodic exploring journey around the 3D grid visiting all its nodes floor by floor. As each floor of the 3D grid is a finite 2D grid, the strategy used to explore a given floor is similar to the one of [19] *i.e.*, two beedroids move and explore a given line while the third beedroid remains idle next to a wall to indicate the next line to explore. More precisely, let  $\ell_i$  be the current line of the floor  $f_i$  being explored. The

leader moves away from the follower and the follower just follows the leader using the rules given in Figure 9. At the beginning, both the leader and the follower move away from the landmark which has color A, and move along the nodes of  $\ell_i$  until reaching a wall. When the leader sees the wall, it does not see the landmark (since the landmark was left on the opposite wall), then the leader and the follower exchange their respective roles and move back along the same line  $\ell_i$ . This role exchange is done in two rounds: first, the leader moves to one of its adjacent nodes changing its color to A to notify the follower that they have to change their role. As the follower does not sense the wall yet, it continues to follow the leader and hence it becomes neighbor to a wall. Next, by observing a beedroid with color A, the follower moves back to its previous position and changes its color to the leader's one L, while the ex-leader, the beedroid with color A, starts following the new leader and updates its color to become a follower. This is done by executing the rules of Figure 10.



■ **Figure 11** Moving to the next line.

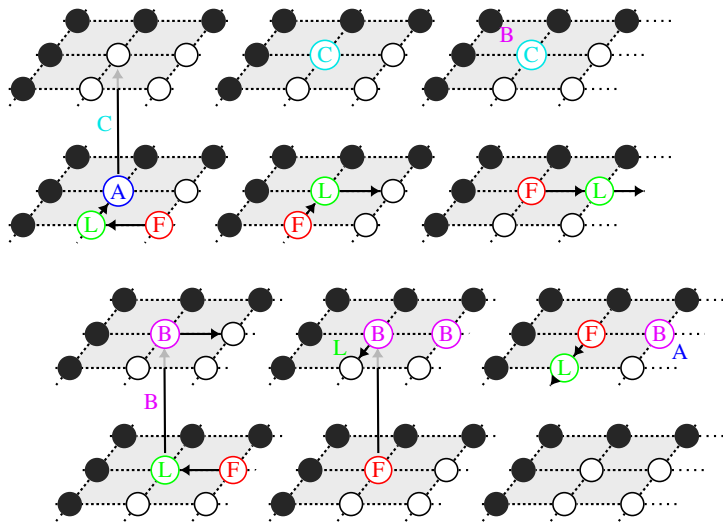


■ **Figure 12** Sequence of configurations during a line change on a simple floor.

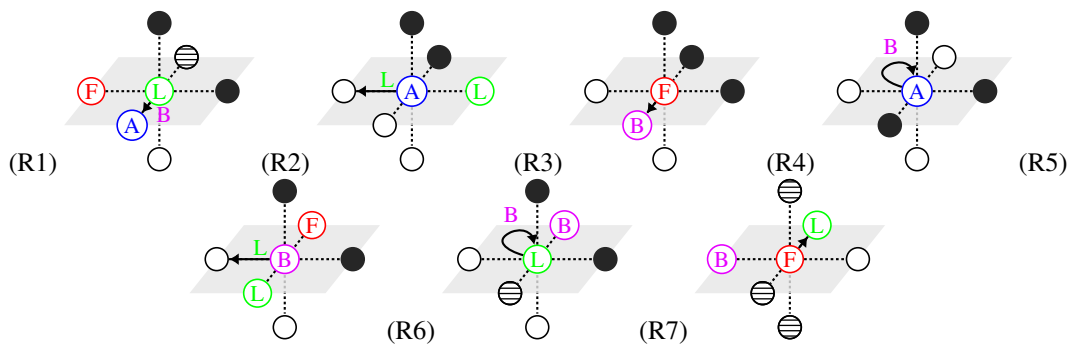
As the beedroids have switched their roles, they proceed again at the exploration of  $\ell_i$  but this time, in the reverse direction. When the leader reaches the opposite wall, it sees this time the landmark and hence knows that the current line  $\ell_i$  has been fully explored. Let  $\ell_{i+1}$  be the line that hosts the landmark. Line  $\ell_{i+1}$  is the next line to be explored. For this purpose, both the leader and the follower need to move to line  $\ell_{i+1}$  while the landmark moves to another line  $\ell_{i+2}$ , the line to visit after  $\ell_{i+1}$ . This line switch is done in three rounds by executing the rules of Figure 11. Figure 12 illustrates the sequence of configurations reached during this latter process. The leader and the follower then simply proceed at the exploration of line  $\ell_{i+1}$  in the same manner as line  $\ell_i$ .

In the case where  $\ell_{i+1}$  is the last line to be explored on  $f_i$ , the landmark moves to the next floor  $f_{i+1}$  to be explored when the leader moves to  $\ell_{i+1}$ . Note that  $f_{i+1}$  is determined thanks to chirality.

Indeed, as the landmark is at a corner and sense the leader at one side, it can identify the upper floor from the lower floor. Both the leader and the follower then proceed in the same manner as previously. That is, they explore the nodes of  $l_{i+1}$ , exchange their roles and then move back on  $l_{i+1}$  until they reach the wall again. When the leader and the follower reach the wall after exploring  $l_{i+1}$ , they also move to Floor  $f_{i+1}$  indicated by the landmark. This is done by executing the rules of Figure 7. The beedroids will repeat the same process to explore Floor  $f_{i+1}$ . Note that in order to keep the same exploring direction, the landmark moves to the line such that the leader remains on the same side. This direction is again chosen using the chirality (recall that the beedroid is at the corner and senses a beedroid at Floor  $f_i$ ). Figure 13 shows the sequence of configurations of this floor change.

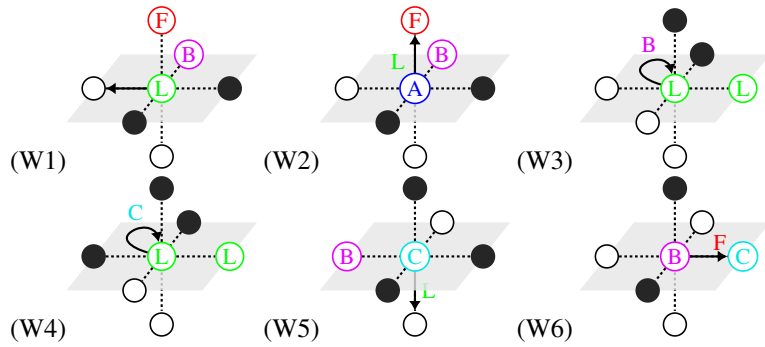


■ **Figure 13** Sequence of configurations during a change of floor. The three first configurations replace the line change that cannot occur because the beedroid  $A$  is in a corner, then the moving group explore the last line of the floor. The three last configurations occurs when the moving group comes back.



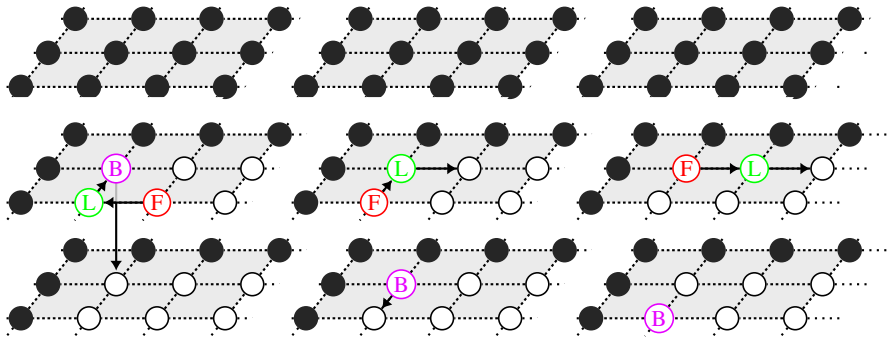
■ **Figure 14** Changing line on a roof.

By doing so, the beedroids eventually explore each floor until they reach the last floor, called a roof in the sequel. This latter is explored similarly but the beedroids update their respective color differently when changing lines (this is done to ensure the beedroids remain on the roof without adding additional colors) using the rules shown in Figure 14.



■ **Figure 15** Switching exploring direction.

The beedroids then change the exploring direction to explore the 3D grid in the reverse direction. This is done by executing the rules of Figure 15. Figure 16 presents a sequence of configurations during the initiation of the exploring direction change and Figure 17 presents the sequence of configurations to terminate this process once the leader and the follower come back from exploring the last line of the roof.



■ **Figure 16** Sequence of configurations during the initiation of the exploring direction change.

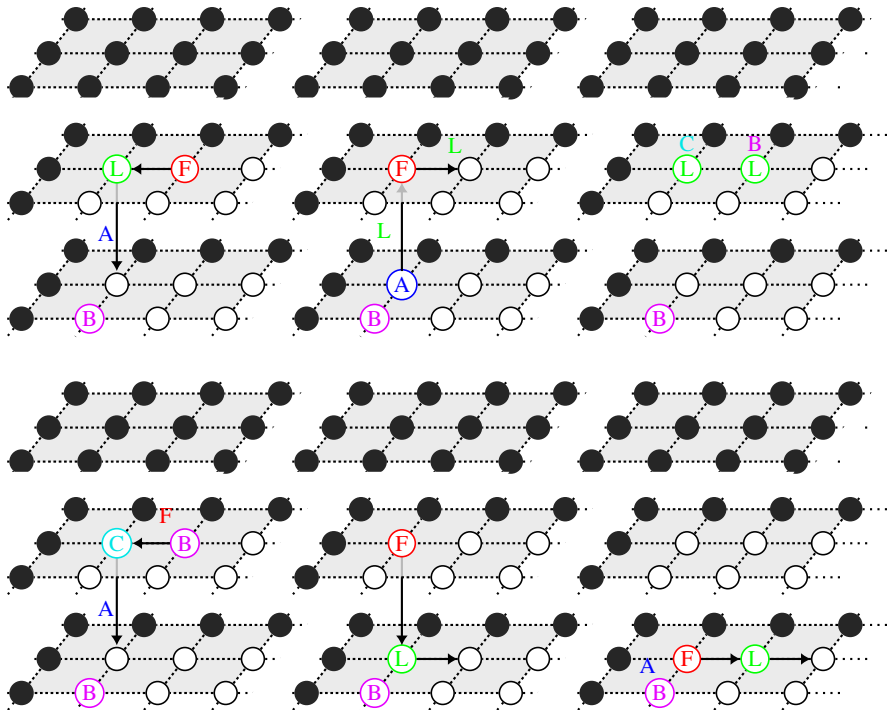
We have validated several base cases using our simulation tool [3]. Then, we have generalized the reasoning by induction to show that  $Vone_3^3$  solves the PFPP. Due the lack of space the proofs leading to the theorem below are omitted but can be found in Appendix B.

► **Theorem 3.** *Under visibility range one,  $Vone_3^3$  solves the perpetual flower pollination problem with three beedroids endowed with five colors.*

## 5 Visibility range two: $Vtwo_1^5$

In this section, we present an algorithm that uses five oblivious beedroids to solve the PFPP. The beedroids are oblivious in the sense that the lights of all the beedroids have the same color and cannot change, so they do not have any bit of persistent memory.

This algorithm works with a specific set of initial configurations that are not locally-defined. The initial configurations are those where the beedroids are close to a wall, like in the first configuration of Figure 18 but they must not be all adjacent to the same wall. Then, the principle is similar to  $Vone_3^5$  i.e., beedroids explore the 3D grid floor by floor. Within a given floor, beedroids are able to

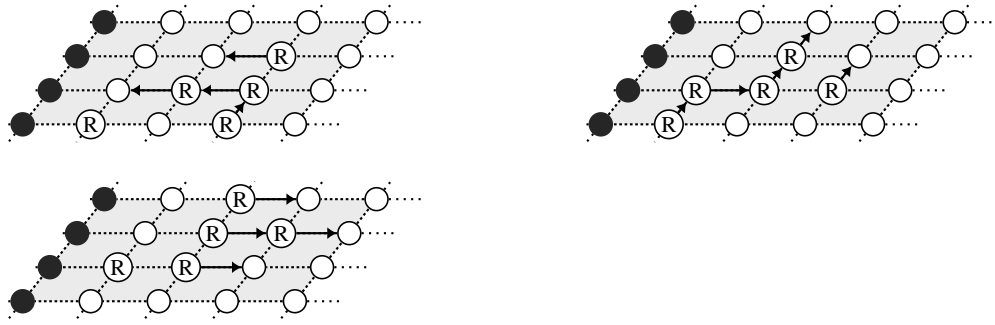


■ **Figure 17** Sequence of configurations once the leader and follower come back to complete the exploring direction change.

fly in straight line when they form a  $\perp$  pattern, called moving group, composed of four beedroids. Initially, the moving group makes a U-turn while changing line (Figure 18). Then, they explore one line leaving a landmark beedroid adjacent to the wall. Upon reaching the opposite wall, they make a U-turn again and explore the same line again (Figure 19). When they meet the landmark again, beedroids are in the same configuration as initially, translated by one node.

Once they reach the corner, they move one floor above and explore this floor with the same pattern, rotated by  $\frac{\pi}{2}$  (refer to Figure 20). Eventually, the beedroids finish exploring the roof and then switch the exploring direction to start exploring the floors of 3D grid in the reverse sense (refer to Figure 21). Due to space constraints, the set of rules can be found in the appendix. The animation is available online [4].

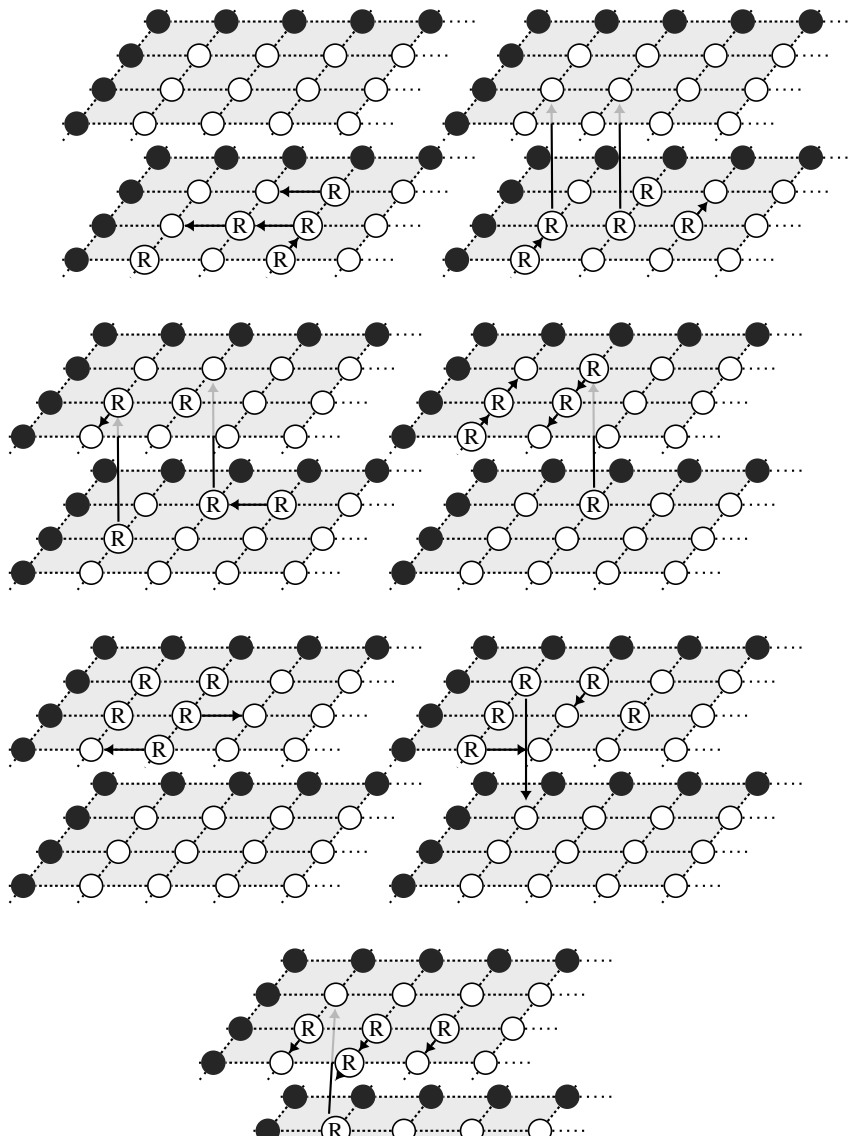
23:14 **Beedroids: How Luminous Autonomous Swarms of UAVs can save the World?**



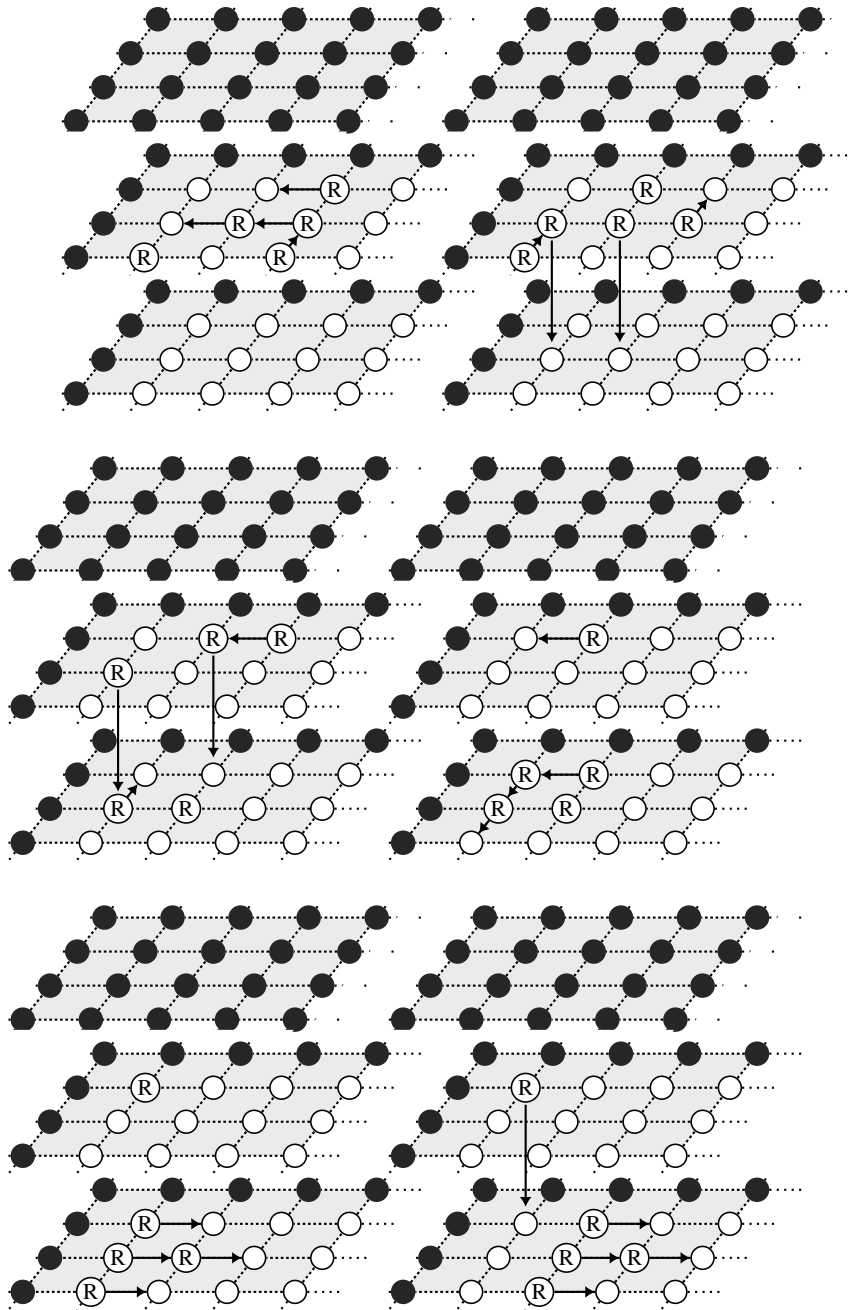
■ **Figure 18** In the initial configuration, beedroids should near a wall, as in the leftmost configuration. The beedroids then start a line change.



■ **Figure 19** Sequence of configuration during a U-turn.







■ **Figure 21** Sequence of configurations when beedroids switch the exploring direction.

► **Theorem 4.** *Under visibility range two,  $\forall \text{two}_1^5$  solves the perpetual flower pollination problem with five oblivious beedroids.*

The proof of Theorem 4 is an induction similar to the one of Theorem 3 (actually it is even simpler due to the absence of any ambiguous rule).

## 6 Related Work

The problem of perpetual flower pollination by a beedroid swarm is actually known in the literature as the *perpetual exploration* of a 3D grid by a swarm of *luminous robots* [18]. Exploration of discrete environment by a robot swarm has been widely studied. Various topologies have been already considered including lines [15], rings [1, 8, 11, 16, 17], trees [14], torus [10], finite [2, 6, 9, 19], infinite 2D grids [5, 7], and even infinite  $n$ -dimensional grids [13]. (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 [8–11, 14–16], while [1, 2, 6, 19] deal with the perpetual exploration problem. Notice that Ooshita and Tixeuil consider the two variants of the problem in [17]. 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, 9–11, 14–16]. 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. Exploration algorithms satisfying exclusiveness are proposed in both finite [1, 2, 6, 19] and infinite graphs [5, 7]. Assuming a common chirality is pretty usual in the 2D Euclidean plan; see *e.g.*, [12]. However, up to now only a few works dedicated to discrete environments, *e.g.*, infinite [7] and finite [6] 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 [6, 19]. To the best of our knowledge, perpetual exploration has been never addressed in finite 3D grids. However, the exploration of an infinite  $n$ -dimensional grid has been investigated in [13]. 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

We have studied how typically small swarms of chiral luminous beedroids can solve the perpetual flower pollination problem in 3D grids assuming the FSYNC model. Under the optimal visibility range one, we have shown that three beedroids are necessary and sufficient to solve the problem. For the sufficient part, we have proposed an algorithm that requires only five colors. Then, we have proposed another solution that is optimal in terms of colors: an algorithm working with five oblivious beedroids under visibility range two.

However, our industrial partners are still not fully satisfied by our proposal. Even if our solutions require a very few number of weak beedroids, they believe that we can still achieve some economies of scale. Like the character Peter Isherwell in the movie “*Don’t look up*”, they want to both save humanity and win money... So, we have to study whether we can reduce the number of colors used by the first algorithm. We should also study whether the number of beedroids and the visibility range

of the second algorithm can be decreased. For this latter, we are pessimistic: we conjecture that the visibility range cannot be lowered to one in the oblivious case. Our idea is that skills necessary to solve the perpetual flower pollination problem in 3D grids with chiral oblivious beedroids are similar to those necessary to solve the 2D grid exploration problem with non-chiral oblivious beedroids. So, we expect that the impossibility proof given in [7] can be adapted to the context of chiral oblivious beedroids evolving in a 3D grid.

---

## 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.
- 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. URL: <http://www.springerlink.com/content/913v424157681707/>.
- 3 Quentin Bramas. Animation of the first algorithm, 2022. <https://bramas.pages.unistra.fr/robot-grid-exploration-simulator/?/robot-grid-exploration-simulator/algo/finite-grid/chirality/range-1/3-robots-5-colors.web-algo>.
- 4 Quentin Bramas. Animation of the second algorithm, 2022. <https://bramas.pages.unistra.fr/robot-grid-exploration-simulator/?/robot-grid-exploration-simulator/algo/finite-grid/chirality/range-2/algo-5-robots-oblivious.web-algo>.
- 5 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. URL: [https://doi.org/10.1007/978-3-030-67087-0\\_9](https://doi.org/10.1007/978-3-030-67087-0_9).
- 6 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.
- 7 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. URL: <https://doi.org/10.4230/LIPICs.FUN.2021.6>.
- 8 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.
- 9 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.
- 10 Stéphane Devismes, Anissa Lamani, Franck Petit, and Sébastien Tixeuil. Optimal torus exploration by oblivious robots. *Computing*, 101(9):1241–1264, 2019.
- 11 Stéphane Devismes, Franck Petit, and Sébastien Tixeuil. Optimal probabilistic ring exploration by semi-synchronous oblivious robots. *Theoretical Computer Science (TCS)*, 498:10–27, 2013.

- 12 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.
- 13 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.
- 14 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.
- 15 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.
- 16 Paola Flocchini, David Ilcinkas, Andrzej Pelc, and Nicola Santoro. Computing without communicating: Ring exploration by asynchronous oblivious robots. *Algorithmica*, 65(3):562–583, 2013.
- 17 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.
- 18 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.
- 19 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.
- 20 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.

## A Omitted Proofs of Section 3

We first recall the overview of the proof. We proceed by contradiction assuming that an algorithm solving the PFPP in 3D grids exists. Then, we show that once beedroids move far away from the walls, their possible movements are restricted. In more details, they can only move straight (Lemma III), otherwise they may not explore the whole grid (Lemma IV). Next, we show that, once the exploration has reached specific places, beedroids must always stay close to at least one wall (Lemma 1), leading to the final contradiction (Theorem 2).

The two first lemmas are technical results that are in particular used in the main lemma, Lemma 1. The first one states that to explore the 3D grid, beedroids should stay neighbors when they do not see any wall. The second one shows that if a beedroid  $b_1$  is lost, at distance 2 from a wall, and at distance at least 4 from other walls, then the other beedroid  $b_2$  should be adjacent to a wall; moreover  $b_1$  should wait for  $b_2$  which, in turn, should eventually leave the wall to meet  $b_1$ .

► **Lemma I.** *Let  $A$  be an algorithm solving the PFPP in 3D grids using two beedroids under visibility range one. In a 3D grid of size at least  $4 \times 4 \times 4$ , no execution of  $A$  reaches a configuration where the two beedroids are lost.*

**Proof.** Assume by the contradiction that an execution  $E$  of  $A$  contains a configuration  $C$  where the two beedroids are lost. In  $C$ , both beedroids, respectively denoted by  $b_1$  and  $b_2$ , see neither a wall nor a beedroid.

Assume that only one beedroid, say  $b_1$  without the loss of generality, moves from  $C$ . The 6 possible destinations of  $b_1$  look identical, so the adversary can choose any of them. Among those 6 destinations, at least 3 are at distance at least two from any wall, and among them at least one is at distance at least 2 from  $b_2$ . So, there is a possible step from  $C$  that reaches a configuration where the two beedroids are still lost.

Assume that both beedroids moves from  $C$ . Again, for each beedroid, the 6 possible destinations look identical. Each beedroid  $r_i$  has a set  $D_i$  of at least three destinations that are at distance at least 2 from any wall. After choosing an arbitrary destination in  $d_1 \in D_1$  for the first beedroid, there exists in  $D_2$ , at least one destination  $d_2$  that is also at distance at least 2 from  $d_1$  (indeed, each pair of destination has only one common neighbor, and this common neighbor is distinct for each pair of destination). After their movement, the two beedroids are lost again.

In both cases, during the next step, the adversary can make any moving beedroid coming back to its position in  $C$ . So, there exists an execution where beedroids do not explore all nodes infinitely often, a contradiction. ◀

► **Lemma II.** *Let  $A$  be an algorithm solving the PFPP in 3D grids using two beedroids under visibility range one. Consider an execution  $E$  of  $A$  in a 3D grid of size at least  $8 \times 8 \times 8$ . If  $E$  contains a configuration  $C$  where a beedroid  $b_1$  is lost, at distance 2 from a wall, and at distance at least 4 from the other walls, then*

- $b_1$  is idle, moreover
- the other beedroids  $b_2$  is adjacent to a wall and is either idle or moves away from the wall during the next step.

**Proof.** First,  $b_2$  is not lost in  $C$  by Lemma I. So

**Claim 1:**  $b_2$  is adjacent to a wall in  $C$ .

To complete the exploration,  $b_2$  has to meet  $b_1$ . Indeed, since  $b_1$  is lost, whether or not  $b_1$  is idle, the adversary can keep it at distance 2 from the same wall and at distance at least 4 from the other walls while it does not meet  $b_2$ . In this situation,  $b_2$  cannot move away from a wall without meeting  $b_1$ , indeed otherwise the execution reaches a configuration where both beedroids are lost, which is

forbidden (by Lemma I). On the other hand, still in the aforementioned situation, if  $b_2$  remains forever adjacent to a wall without meeting  $b_1$ , then some nodes will be not visited anymore, a contradiction.

Hence,  $b_2$  eventually meets  $b_1$  after  $C$ . Consider now any execution  $E'$  containing  $C$  where  $b_2$  meets  $b_1$  after  $C$  and where  $b_1$  remains at distance 2 from the same wall and at distance at least 4 from the other walls until it meets  $b_2$  (we have just seen that such execution exists). Let  $t_C$  be the time at which  $C$  occurs in  $E'$ . Let  $t_m$  be the first time  $b_1$  and  $b_2$  meet after  $C$  in  $E'$ .

**Claim 2:** *In  $E'$ ,  $b_2$  is at distance at most 2 from  $b_1$  at every time  $t$  such that  $t_C \leq t < t_m$ .*

*Proof of the claim:* Assume by contradiction that  $b_2$  is at distance at least 3 from  $b_1$  in a configuration  $C_3$  occurring at time  $t'$  in  $E'$  with  $t_C \leq t' < t_m$ . We now show that there is a possible infinite suffix from  $C_3$  where  $b_2$  remains at distance at least 3 from  $b_1$ .

Actually, we can construct an execution where, after the prefix of  $E'$  leading to  $C_3$ , every time  $b_2$  is dictated to move from distance 3 to distance 2 from  $b_1$ , then the adversary decides that  $b_2$  moves at distance 4 from  $b_1$  since the move of  $b_2$  is ambiguous.

Indeed, in  $C_3$ ,  $b_2$  is at distance 3 from  $b_1$ , which is lost. So,  $b_2$  is adjacent to a wall otherwise we get a contradiction by Lemma I. In this case,  $b_2$  is at distance at least 2 from any other wall since  $b_1$  at distance 2 from the wall adjacent to  $b_2$  and at distance at least 4 from the other walls. So, its view has an axis of rotational symmetry with respect to the line orthogonal to the wall. Hence, the adversary can rotate the local view of  $b_2$  in such way that  $b_2$  does not come closer to  $b_1$ , but instead remains at distance at least 3 from it (*n.b.*,  $b_1$  may also move simultaneously). By repeating the same argument, we can create an infinite execution where  $b_2$  remains at distance 3 from  $b_1$ , so beedroids never meet, a contradiction.

**Claim 3:** *In  $C$ ,  $b_2$  is idle or moves away from the wall.*

*Proof of the claim:* From Claim 2, we know that  $b_2$  is at distance 2 from  $b_1$  in  $C$ . Consequently,  $b_2$  is at distance at least 2 from all the other walls.  $b_2$  cannot move along its adjacent wall because its view has an axis of rotational symmetry and the adversary can decide that a destination at distance 3 from  $b_1$ . In this case, if  $b_1$  is idle, then the distance becomes 3. Otherwise, the adversary can also decide the destination of  $b_1$  since it is lost so that the distance becomes 4. Hence, this contradicts Claim 2 in both cases. Thus,  $b_2$  can either stay idle or move away from the wall, and cannot do anything else.

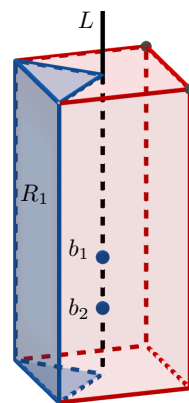
**Claim 4:** *In  $C$ ,  $b_1$  is idle.*

*Proof of the claim:* If it is not, then, the adversary can decide that its destination is a node at distance 3 from  $b_2$  (from Claim 2, we know that  $b_2$  is at distance 2 from  $b_1$  in  $C$ ). If  $b_2$  moves in the same step, then  $b_2$  moves away from its adjacent wall, by Claim 3. By Claim 2, both beedroids are lost in the reached configuration, a contradiction to Lemma I. Otherwise,  $b_2$  stays idle, and the distance between  $b_1$  and  $b_2$  becomes 3, contradicting Claim 2.

The lemma directly follows from Claims 1,3, and 4. ◀

► **Lemma III.** *Let  $A$  be an algorithm solving the PFPP in 3D grids using two beedroids under visibility range one. If there exists an execution that reaches a configuration  $C$  where beedroids are at distance at least  $2B$  from any wall and, from  $C$ , the beedroids perform a periodic sequence of movements without ambiguous rules, then there is a straight line of the 3D grid that contains the two beedroids while none of them sees a wall.*

**Proof.** When a beedroid does not see any wall, a rule is unambiguous only if the beedroid sees the other one at distance one and the destination of the rule is on the axis passing through the other beedroid. Consequently, each beedroid can only stay idle, move away, or move towards the other beedroid. The formal proof is exactly the same as in [19] (*i.e.*, 2D grids without chirality). ◀



■ **Figure 22** If beedroids are on a line  $L$ , the adversary can decide from which side of the square cuboid the beedroids will exit. In particular, we can decide that the beedroids will exit toward  $R_1$ , the blue area.

► **Lemma IV.** *Let  $A$  be an algorithm solving the PFPP in 3D grids using two beedroids under visibility range one. There exists no execution that reaches a configuration  $C$  where beedroids are at distance at least  $2B$  from any wall and, from  $C$ , the beedroids perform a periodic sequence of movements that includes an ambiguous rule.*

**Proof.** Assume, by the contradiction, that there is an execution that reaches a configuration  $C$  where beedroids are at distance at least  $2B$  from any wall and, from  $C$ , the beedroids perform a periodic sequence of movements which includes an ambiguous rule.

First, by definition, the ambiguous rule is executed during the  $B$  first steps from  $C$ , and when it is done, beedroids are still at least at distance  $B$  from any wall.

Assume that there is a configuration  $C$  reached during the  $B$  where the two beedroids are lost. Since beedroids are initially at distance  $2B$  from any wall, the size of the 3D grid is at least  $4 \times 4 \times 4$  and we obtain a contradiction by Lemma I.

So, the two beedroids remains neighbors while they do not see any wall. Let  $b_1$  be the beedroid executing the ambiguous rule and  $b_2$  be the other one. The ambiguous rule executed by  $b_1$  necessarily cause beedroids to either shift to a parallel line or make a turn. In the former case, both beedroids execute ambiguous rules, so the adversary can lost them instead of performing the shift, and we obtain a contradiction as previously. In the latter case, the adversary decides on which side the beedroids will turn among the at least four possibilities.

Hence, if the periodic sequence of movements contains an ambiguous move, the beedroids may make at least one turn per period and the adversary can make the beedroids remain in the same square grid of size  $B$  (the period of the sequence is at most  $B$ ). While doing so, the beedroids do not see any wall, and do not explore the whole 3D grid, a contradiction. Again, the formal proof is similar to the proof in the 2D grid case without chirality [19]. ◀

The next lemma states that if two beedroids are on the same line and inside an area that is rotationally symmetric, then they cannot break this symmetry without executing an ambiguous rule (due to the lack of agreement on the coordinate system). Hence, the adversary can decide on which side of the line the beedroids move. More formally, if beedroids move out of the area through a node  $u$ , then there exists an execution where the beedroids move out of the area through another node  $u'$  that is symmetric to  $u$ .

We need to define a few concepts beforehand; see Figure 22 for an illustration. Consider a



configuration  $C$  where the two beedroids are located in some line of the 3D grid. We call *blurred area* of  $C$  any subset  $S$  of nodes such that the subgrid induced by  $S$  shapes a square cuboid for which  $L$  is an axis of rotational symmetry. A blurred area is non-trivial if it does not contain all nodes of the 3D grid. We denote by  $EH(S)$  the external hull of a blurred area  $S$ , i.e., the set of nodes in  $V \setminus S$  at distance one from a node of  $S$ .

► **Lemma V.** *Let  $A$  be an algorithm solving the PFPP in 3D grids using two beedroids under visibility range one. Assume an execution  $E$  reaching a configuration  $C$  where the two beedroids are on the same line  $L$ . Let  $S$  be a blurred area of  $C$ . Let  $R_1 \subseteq EH(S)$  such that the union of  $R_1$  and its symmetric w.r.t. the rotations around  $L$  is equal to  $EH(S)$ . If  $S$  is non-trivial and  $L$  is also an axis of rotational symmetry of  $EH(S)$ , then there exists an execution from  $C$  where a beedroid reaches  $EH(S)$  for the first time at a node  $u \in R_1$ .*

**Proof.** Once  $E$  has reached  $C$ , the beedroids have to explore the nodes outside  $S$  since  $S$  is non-trivial. So, in  $E$ , eventually a beedroid goes outside  $S$  for the first time from  $C$  by reaching a node  $u$  in  $EH(S)$ , say at time  $t$ .

If  $u \in R_1$ , we are done. Otherwise  $u \in EH(S) \setminus R_1$  and, by assumption, there exists a node  $u' \in R_1$  symmetric to  $u$  with respect to a rotation around  $L$ , i.e.,  $u' = \phi(u)$  where  $\phi$  is a rotation around  $L$ . First,  $u' \notin L$ , because otherwise  $u = u' \in R_1$ . So, to reach  $u$ , a beedroid has to move away from  $L$  at some round before  $t$ . Let  $t_0$  be the first time, after configuration  $C$ , such that a beedroid, say  $r$ , moves away from  $L$ . In the configuration at time  $t_0$ ,  $C(t_0)$ , the view of both beedroids have a rotational symmetric around axis  $L$  so that the rule executed by  $r$  is ambiguous. Consider now the execution  $E'$  similar to  $E$ , but where at time  $t_0$  the adversary rotates the view of  $r$  by rotation  $\phi$ , so that the beedroid  $r$  moves away from  $L$  on another side (and we do the same for the other beedroid if it moves away from  $L$  at time  $t_0$ ). After that, we consider that the execution is similar to  $E$  but where everything is rotated by  $\phi$ . Hence, at time  $t$ , a beedroid reaches the node  $u'$  and we are done. ◀

In particular, we can choose  $R_1$  to be the union of one side and two triangles, as shown in Figure 22 (page 21), where  $R_1$  and its symmetric form the red external hull of the square cuboid. Remark that if the square cuboid leans against a wall (as illustrated in Figure 22 if the gray plan is a wall), then  $R_1$  can consist only in one face and one triangle. By applying the lemma, we obtain that there is an execution where the beedroids escape from the blurred area either through a rectangular face or the top triangle.

► **Lemma 1.** *Let  $A$  be an algorithm solving the PFPP in 3D grids using two beedroids under visibility range one. Assume some execution  $E$  reaches at a given time  $t$  (i) a configuration where a beedroid is 2-close or (ii) a  $T$ -configuration. Let  $C$  be the configuration of  $E$  at time  $t$ . Then, there exists an execution  $E'$  and a time  $t' > t$  such that*

- $C$  is reached in  $E'$  at time  $t$ ,
- at time  $t'$  in  $E'$ , a beedroid is 2-close or the system is in a  $T$ -configuration, and
- between time  $t$  and  $t'$  in  $E'$ , the beedroids remain at distance at most  $4B$  from a wall.

**Proof.** We consider a 3D grid whose size is more than  $8B \times 8B \times 8B$ . The lemma otherwise trivially holds: by definition of  $A$ , a beedroid is infinitely often 2-close; moreover every beedroid is always at distance at most  $4B$  from a wall in a 3D grid where at least one side is less or equal to  $8B$ .

Assume first that a beedroid is 2-close in a configuration  $C$  (the case where beedroids are in a  $T$ -configuration will be treated in the last paragraph of this proof). To explore the 3D grid, the two beedroids must sometimes be not 2-close. Indeed, if a beedroid remains 2-close forever, the other beedroid should in particular explore nodes at distance more than  $2B + 2$  from every wall. In this case, it would be lost and consequently, the adversary can make it alternating between two nodes forever, making the exploration fail.

Consider now an execution  $E'$  such that, whenever a beedroid executes an ambiguous moves that could make it not 2-close, then the adversary chooses a destination that is 2-close. This is possible because, in case of an ambiguous move, the adversary can chose among at least 4 destinations and if one destination would make the beedroid not 2-close, then the opposite destination (with respect to the beedroid's position) would keep it 2-close (because when a beedroids makes a move, only the distance to one wall increases).

In  $E'$ , let  $t_0 > t$  be the first time when no beedroid is 2-close. By assumption, at least one beedroid that is 2-close at time  $t_0 - 1$ , say  $b_2$ , makes an unambiguous move. To make this unambiguous move,  $b_2$  necessarily moves toward the other one,  $b_1$  that is not 2-close. So, at time  $t_0 - 1$ , only  $b_2$  is 2-close, *i.e.*, at distance at most  $2B$  from two walls  $W_1$  and  $W_2$ . Without loss of generality, at time  $t_0$ ,  $b_2$  is at distance  $2B + 1$  from wall  $W_1$  and at distance at most  $2B$  from  $W_2$  ( $b_2$  is at distance more than  $2B$  from other walls).

Since at time  $t_0 - 1$ ,  $b_2$  is moving towards  $b_1$ , then, at time  $t_0 - 1$ , the two beedroids are on a line parallel to the wall  $W_2$ . Assume first that beedroids are not adjacent to  $W_2$ . Two cases can occur (both cases are represented in Figure 3) (page 7).

**Case (1):** They remain on the same line parallel to  $W_2$ , moving away from  $W_1$ , until a beedroid is at distance  $3B + 1$  from  $W_1$ .

If they do so, since they traveled a distance  $B$  since  $t_0 - 1$ , they are executing a periodic sequence of movements, hence, they continue to move on the same line until reaching the wall opposite to  $W_1$  (Lemmas III and IV), in a  $T$ -configuration, while remaining at distance at most  $2B$  from  $W_2$ , and the lemma holds in this case.

**Case (2):** Before being at distance  $3B + 1$  from  $W_1$ , one or two beedroids move away from the line they were traveling through.

These moves are necessarily ambiguous. If two beedroids moves away simultaneously, we get a contradiction because the adversary can choose the destination so that the two beedroids become lost (Lemma I). So, only one beedroid, say  $b_1$ , moves away from the line. Again, since beedroids cannot become lost (Lemma I) and the destination of  $b_1$  is chosen by the adversary, we can consider the case where the two beedroids end up, at time  $t_1 \geq t_0$ , in a line  $L$  orthogonal to  $W_2$ .

Consider now the case where the beedroids are adjacent to  $W_2$  at time  $t_0 - 1$  when moving away from  $W_1$ . Similar things occur.

**Case (a):** They perform a periodic movement while remaining adjacent to  $W_2$  and travel along the wall (but not necessarily in straight line) until reaching another wall (so they become 2-close), and the lemma holds in this case;

**Case (b):** a beedroid moves away from  $W_2$  and forms with the other beedroid a line  $L$  orthogonal to  $W_2$  before being at distance  $3B + 1$  from  $W_1$ ; or

**Case (c):** both beedroids move away from  $W_2$  (simultaneously or one after the other, as a lost beedroid must wait the other beedroid, by Lemma II) and they end up in a line  $L$  parallel to  $W_2$ . In this case, since they traveled at most  $B$  from  $t_0$ , they can travel again at most  $B$  before either performing a periodic movement (Case (1)) while staying at distance 2 from  $W_2$ , and the lemma holds in this case.

Otherwise, they make an ambiguous move to end up in a line  $L$  orthogonal to  $W_2$  (Case (2)). In this latter case, the beedroids end up at distance at most  $4B$  from  $W_1$ .

So, the only cases that remains to consider are those where the beedroids are in a line  $L$  orthogonal to  $W_2$  at distance at most  $4B$  from  $W_1$ .

Consider the set of nodes  $R_1 = R_1^{rectangle} \cup R_1^{triangle}$  where  $R_1^{rectangle}$  is a rectangle of nodes at distance  $2B$  from the wall  $W_1$  and  $R_1^{triangle}$  a triangle of nodes at distance  $2B$  from  $W_2$  and at most  $4B$  from  $W_1$ , such that the union of  $R_1$  and its image by the three rotations around  $L$  form the external hull of a square cuboid for which (1) one face is at distance 1 from  $W_2$  and (2)  $L$  is axis

of rotational symmetry; see Figure 4 for an illustration (page 7). Using Lemma V, there exists an execution such that a beedroid reaches  $R_1$ .

If a beedroid reaches  $R_1^{rectangle}$ , then a beedroid becomes 2-close and the lemma is proven. If a beedroid reaches  $R_1^{triangle}$ , then the beedroids have traveled a distance at least  $B$  without seeing a wall, hence are executing a periodic sequence of movements. The sequence cannot contain an ambiguous rule (using Lemma IV) because the beedroids are at distance at least  $2B$  from any wall, so they are moving in a straight line (by Lemma III), and they end up in the wall opposite to  $W_2$  and reach a  $T$ -configuration, while remaining at distance at most  $4B$  from  $W_1$ .

Now we consider that the beedroids are in a  $T$ -configuration in configuration  $C$ . Then, they are on a line  $L$  orthogonal to a wall, say  $W_2$ . Using a similar argument, we know that either the beedroids become 2-close, or move in a straight line to the opposite wall until they reach a  $T$ -configuration. ◀

## B Correctness of Algorithm $Vone_5^3$

► **Theorem 3.** *Under visibility range one,  $Vone_5^3$  solves the perpetual flower pollination problem with three beedroids endowed with five colors.*

**Proof.** We consider 3D grids of size  $S_x \times S_y \times S_z$  where  $S_x \geq 3$ ,  $S_y \geq 3$ ,  $S_z \geq 3$  since  $Vone_5^3$  uses three beedroids. In the following, we will respectively refer to  $S_x$  as the length,  $S_y$  as the width, and  $S_z$  as the height of the 3D grid.

As a preliminary remark, we can observe that  $Vone_5^3$  proceeds by phases. First, a default prefix where beedroids move toward a wall and then self-organize together to reach a configuration  $C_p$  corresponding to the effective start of the exploration. Then, the system periodically goes through Configurations  $C_p$  and all nodes should be visited at least once between two occurrences of  $C_p$  in the execution. Between two occurrences of  $C_p$ , a finite exploration is split as follows. Nodes are visited floor by floor until reaching the roof, *i.e.*, the  $S_z^{th}$  floor. After the visit of the roof, the exploration direction changes (from bottom-up to top-down or the reverse), and so on.

We have defined the rules in such a way that the processing of each phase is independent from the size of the grid (for example, colors are not used to compute any distance). In particular, the default prefix widely depends on the initial positions of beedroids. The grid size only impacts the duration of this prefix, precisely the time necessary to reach a wall.

The exploration of each floor is made line by line. The number of lines per floor as well as the length of lines do not impact the way lines are explored except from increasing the exploration time.

Concerning now the number of floors. The exploration of two consecutive non-roof floors is analogous up to a  $\frac{\pi}{2}$  rotation. Hence, the exploration of the  $i^{th}$  and  $j^{th}$  floor is analogous provided that none of them is the roof and  $i \bmod 4 = j \bmod 4$  (every 4 rotations of  $\frac{\pi}{2}$ , we retrieve an analogous situation). Similarly, the exploration of the roof is similar for any two 3D grids with the same length, same width and whose respective heights  $i$  and  $j$  satisfy  $i \bmod 4 = j \bmod 4$ .

Hence, we proceed by induction on  $S_x \times S_y \times S_z$ . We have validated several base cases, including heights  $h$  such that  $h \bmod 4 = x$ , for each value  $x \in \{0, 1, 2, 3\}$ . Precisely, we have checked that  $Vone_5^3$  solves the PFPP on every 3D grid of size  $S_x \times S_y \times S_z$  with  $S_x \in \{3, 4, 5\}$ ,  $S_y \in \{3, 4, 5\}$ , and  $S_z \in \{3, 4, 5, 6, 7, 8\}$ .<sup>6</sup>

Such a checking is easy since, from a given initial configuration, we only need to check up to four executions. Actually, two rules of  $Vone_5^3$  are ambiguous: D14 and U1. D14 is executed at most once during a default prefix and four destinations are possible. Hence, we add to check the four possible

<sup>6</sup> Those choices have been motivated by the fact that values 3 and 4 are very specific since in those cases, 1 or 2 beedroids are necessarily adjacent to a wall initially.

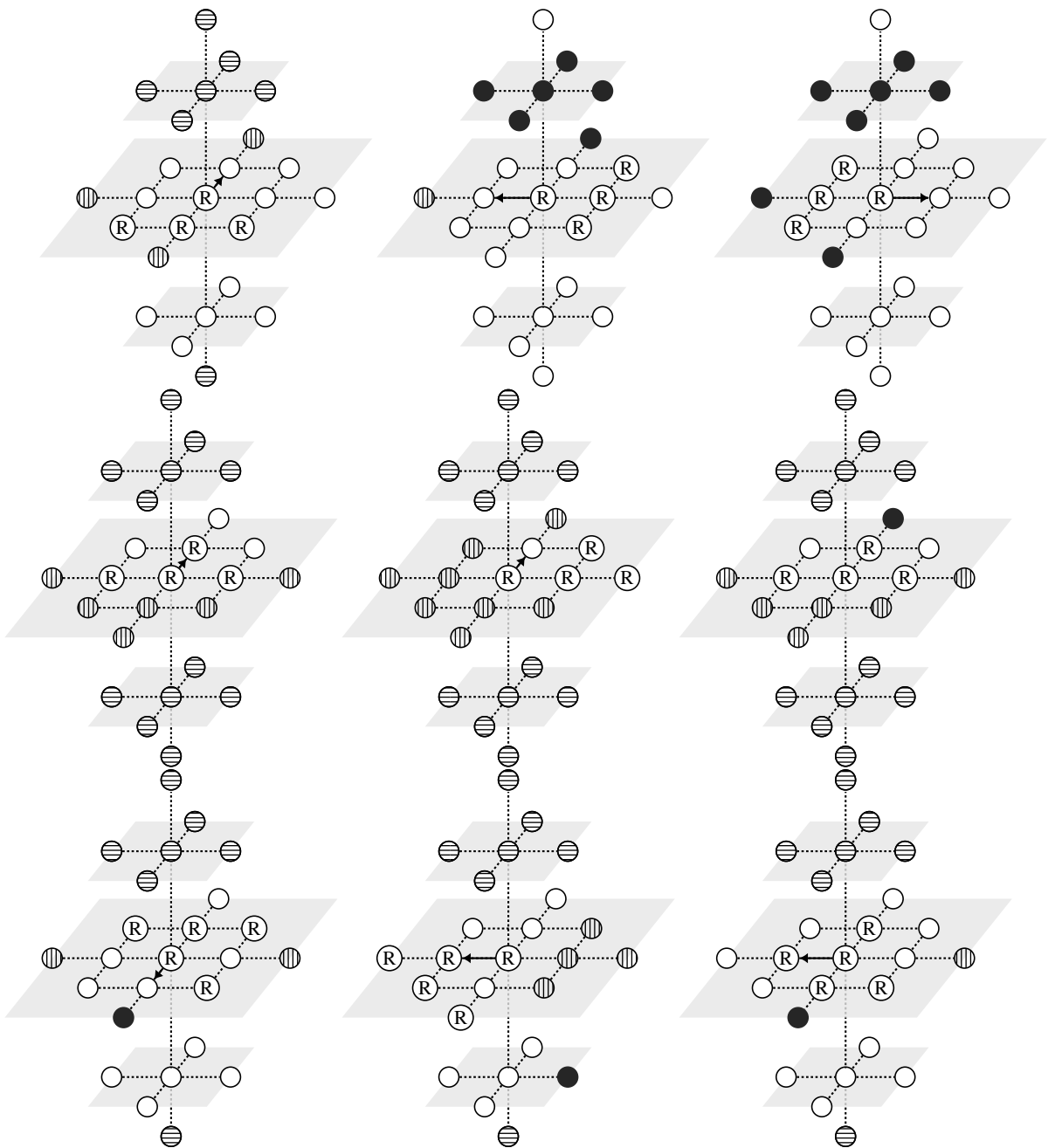
destinations each time D14 occurs in a prefix. The case of Rule U1 is less problematic. Indeed, U1 is executed only once per line during a floor exploration. Moreover, when U1 is executed by a beedroid, then, whatever be the destination, the beedroid goes back to its initial position rightafter by executing U4. Consequently, whatever be the destination chosen by the adversary, the reached configuration one round after U1 is the same. Hence, we just had to check four executions per initial configuration since the system is synchronous and so only influenced by the nondeterministic choice made by D14 during the default prefix. We ran those executions until reaching an already encountered configuration (actually  $C_p$ ): at that time all nodes should have been visited.

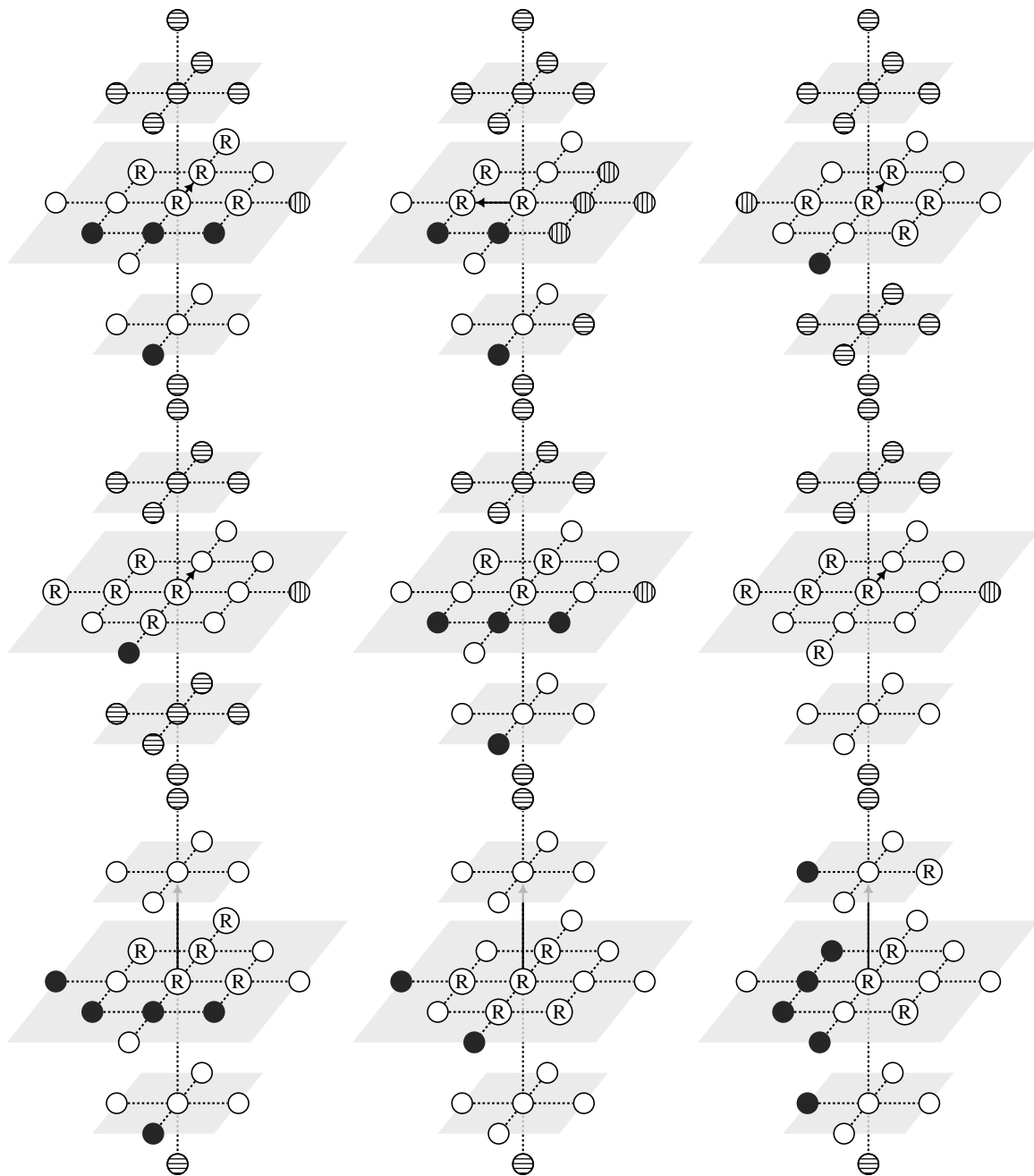
We assume now that  $\text{Vone}_5^3$  solves the PFPP problem in all 3D grids of size  $S_x \times S_y \times S_z$  with  $3 \leq S_x \leq \ell$ ,  $3 \leq S_y \leq w$ , and  $3 \leq S_z \leq h$  for some values  $\ell \geq 5$ ,  $w \geq 5$ , and  $h \geq 8$ .

From the previous discussion, the induction holds for the 3D grid of size  $(\ell + 1) \times w \times h$ ,  $\ell \times (w + 1) \times h$ , and  $\ell \times w \times (h + 1)$  since we can apply the induction on 3D grids of size  $\ell \times w \times h$  and  $\ell \times w \times (h - 3)$ . ◀

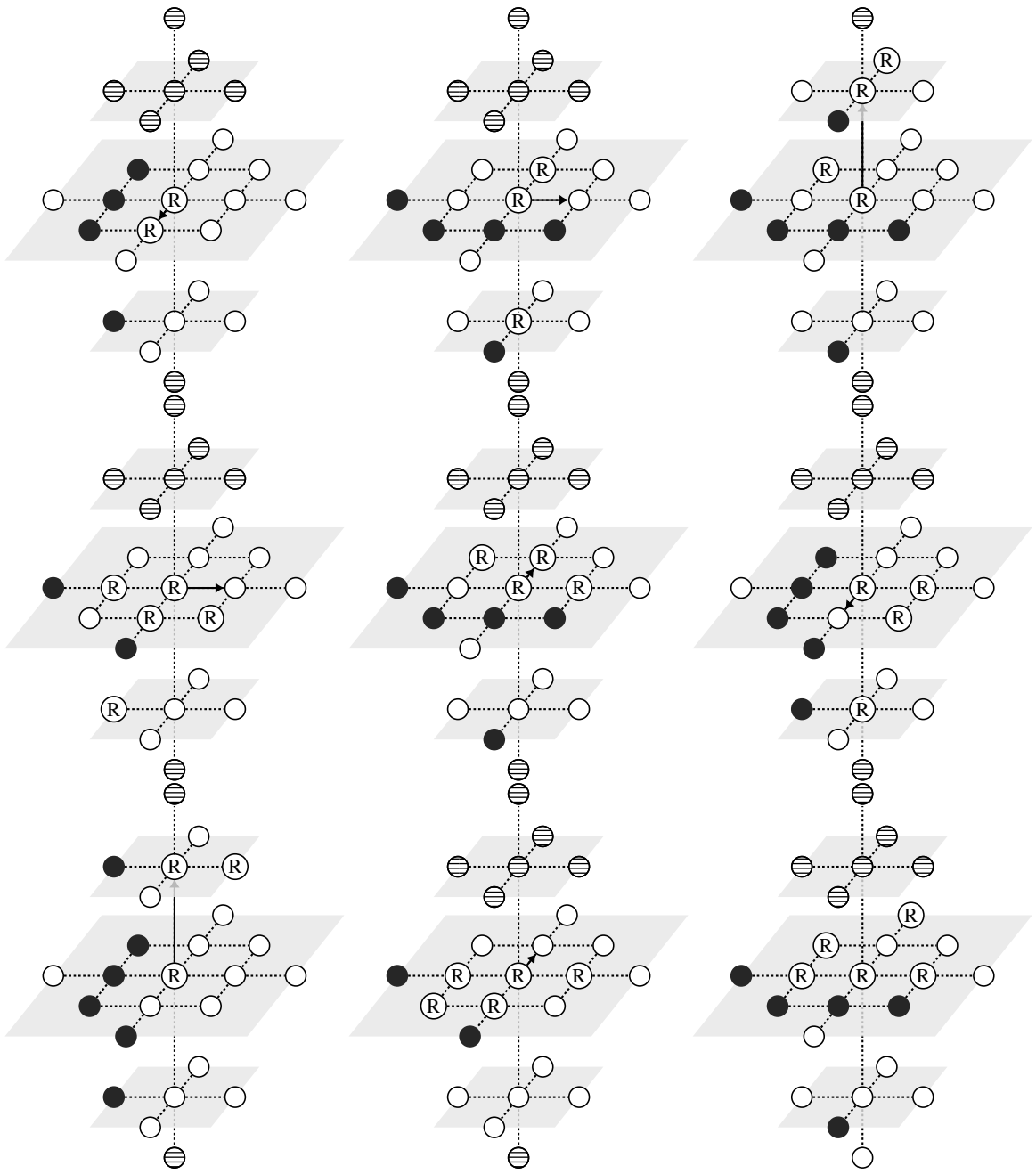
**C** Algorithm  $V_{two}^5_1$

Here are the set of rules.

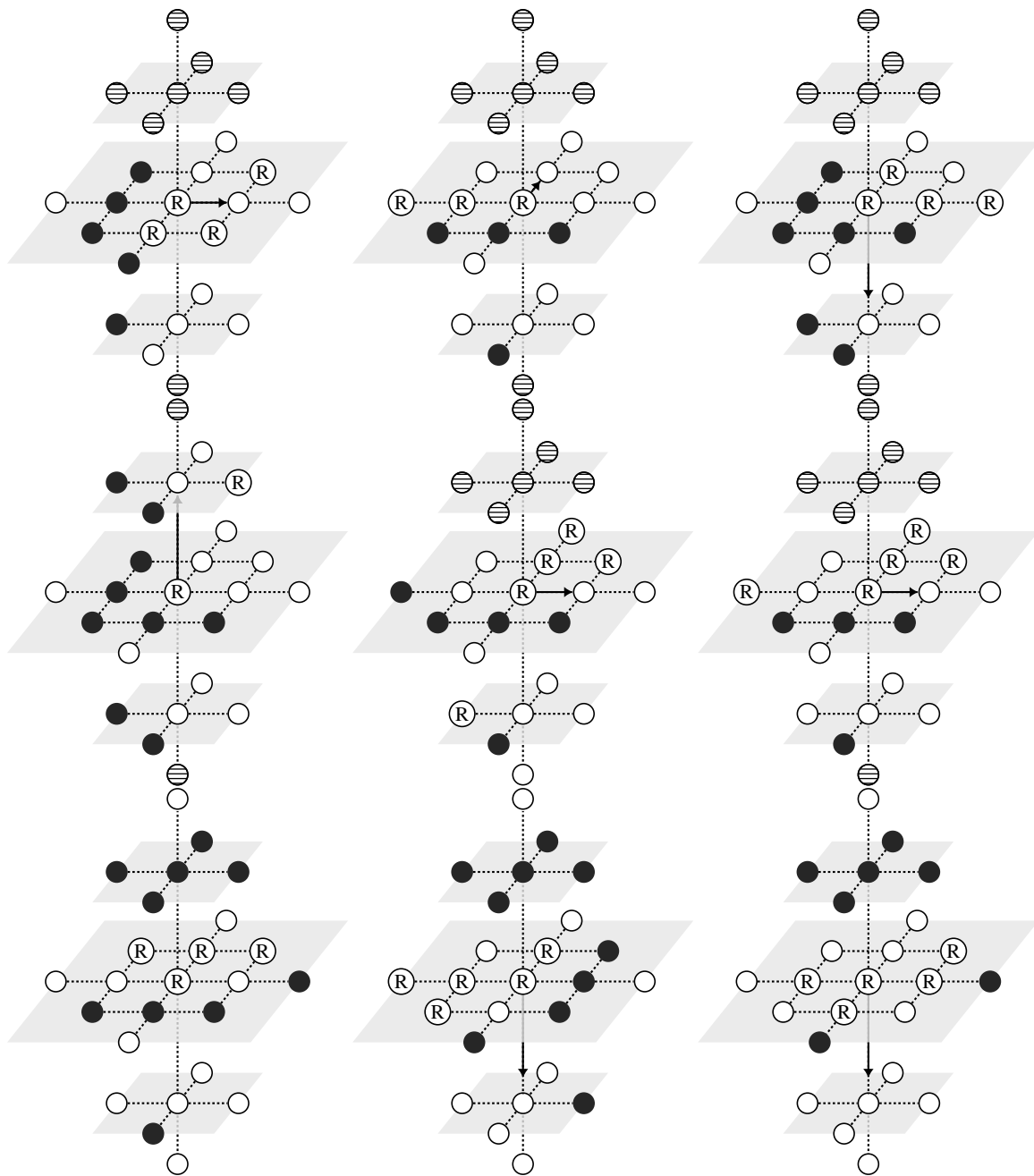




23:28 **Beedroids: How Luminous Autonomous Swarms of UAVs can save the World?**







23:30 **Beedroids: How Luminous Autonomous Swarms of UAVs can save the World?**

