Gathering and Rendezvous by oblivious robots in arbitrary graphs, rings, and trees *

Jérémie Chalopin † Paola Flocchini ‡ Bernard Mans § Nicola Santoro ¶

Abstract

In this paper we investigate the *Gathering* and the *Exploration* problems by asynchronous, anonymous, oblivious robots in an anonymous edge-labeled graph of arbitrary topology.

If the labeled graph is asymmetric the problems can be easily solved; on the other hand if both the labeled graph and the placement of the robots are symmetric, both problems unsolvable. As a consequence we concentrate on asymmetric placements.

We focus on the Gathering problem and we prove that $k = 2l+1 > 1$ robots can gather in any connected graph with asymmetric placement in finite time. We then turn to the Exploration problem. We show that, when the team is composed by three robots only, not all graphs can be explored by giving a characterization of the topologies that can be explored and the corresponding algorithm. We then prove that $k = 2l + 1 > 1$ robots can explore any connected graph with asymmetric placement in finite time. The case of an even number of robots is left open except for $k = 4$, for which algorithms are given for both gathering and exploration for any graph with asymmetric placement.

We conclude giving a complete characterization for rings and trees.

Keywords: mobile robots, asynchronous, oblivious, exploration, gathering.

1 Introduction

1.1 The Problem and its Framework

Consider a team (or swarm) of identical mobile robots located in a given spatial setting that they can see, each robot operating in a endlessly repeating *look-compute-move* cycle. Consider now the following question:

> *What tasks can such a team perform if the robots do not remember the past and have no explicit means of communication ?*

This computational question has been extensively studied in the *continuous* setting - that is, when the robots operate in two-dimensional space - with respect to a variety of problems, such as *Pattern Formation*, *Gathering*, *Flocking*, etc. (e.g. see [1, 2, 3, 5, 11, 18, 19]).

The same question has been recently posed also in the *discrete* setting - that is, when the robots operate in a network - with respect to two fundamental problems (extensively studied in the past

[1]This work was partially supported by ANR Project SHAMAN, by COST Action 295 DYNAMO, and by NSERC.
[2]LIF, Aix-Marseille Université. jeremie.chalopin@lif.univ-mrs.fr.
[3]SITE, University of Ottawa, Ottawa, Canada. flocchin@site.uottawa.ca
[4]Macquarie University, Sydney, Australia. bernard.mans@mq.edu.au
[5]School of Computer Science, Carleton University, Ottawa, Canada. santoro@scs.carleton.ca

in a variety of other models): *Gathering* (or *Rendezvous*), which requires all robots to move within finite time to the same node (whose location is a priori undetermined); and *Exploration*, which requires every node of the network to be visited by at least one robot within finite time.

For the type of robots considered here, the two problems have been studied in a severely restricted setting, without making any assumptions (except finiteness) on the time elapsed between successive cycles, and making no assumptions on node labels nor on edge labels; that is considering *asynchronous* robots in *anonymous* networks with *unlabeled edges*). The focus of the investigations has been only on two classes of graphs: *rings* [7, 9, 14, 15] and *trees* [10]. The difficulty of the analysis of the question for these well understood classes of graphs has been surprising; furthermore the problems are sometimes solvable only in very limited cases and the solutions are necessarily expensive. No results exist for arbitrary graphs, and the outlook is not encouraging.

The natural question is whether the limits, costs and difficulty are inherent to the problem or rather induced (or amplified) by some of the assumptions (or lack of) in the model. Notice that the computational weakness of the robots - *asynchronous, anonymous, oblivious* - is standard for the continuous case (it is the classical ASYNCH or CORDA model [8, 17]), hence it is meaningful to maintain it in the discrete case. Further notice that *anonymity* of the nodes is also a standard condition when studying computability in networks.

What is not standard is the assumption that the links are *unlabeled*. Indeed, such a situation is explicitly excluded from standard models of networks (anonymous or not) in distributed computing. In fact, the assumption agreed upon is that the links incident on a node x have distinct labels, called *port numbers* (it is the standard *local orientation* assumption). In other words, a network is modeled as an edge-labeled graph (G, λ), where λ is the set of local injective labeling functions λ_x.

Our goal is to investigate the original question under this more standard assumption in graphs of arbitrary topologies.

1.2 Main Contributions

We investigate both the *Gathering* and the *Exploration* problem by asynchronous, anonymous, oblivious robots in an anonymous edge-labeled graphs (G, λ) of arbitrary topology. Let Ψ be the placement function describing the position of the robots. Let (G, λ, Ψ) denote the edge-labeled graph with the placement of the robots. We say that (G, λ, Ψ) is *symmetric* if there exists an automorphism of (G, λ, Ψ), when considering automorphisms that preserve the labels and the placement; *asymmetric* otherwise.

We first observe that if (G, λ, Ψ) is *symmetric* neither *Gathering* nor *Exploration* can be solved. Conversely, if (G, λ) is *asymmetric* both *Gathering* and *Exploration* are of simple resolution. Thus, we analyze the two problems when (G, λ) is *symmetric* but (G, λ, Ψ) is *asymmetric* and shall call such condition an *asymmetric placement*. In the following, unless otherwise stated, we assume that there is such an asymmetric placement.

We focus on the *Gathering* problem. We prove that $k = 2l + 1 > 1$ robots can gather in any connected graph with asymmetric placement in finite time. As for the even case, we prove that 4 robots asymmetrically placed can gather in finite time and we conjecture that any number of asymmetrically placed robots can.

We then turn to the *Exploration problem*. We first consider the case when $k = 3$. Interestingly, in this case not all connected graphs with asymmetric placement can be explored. We give a necessary and sufficient condition for the graph to be explorable. Furthermore, we describe an algorithm that performs the exploration when the condition is verified. On the other hand, when

$k = 2l+1 > 3$ all graphs with asymmetric placement can be explored and the exploration algorithm is based on gathering. We then show that Exploration can be solved in any graph also in the even case of $k = 4$ robots with asymmetric placement. The general even case is left as an open problem.

We conclude giving a complete characterization for rings and trees showing that, in these special topologies, gathering and exploration can be achieved if and only if (G, λ, Ψ) is asymmetric, regardless of the number $k > 2$ of robots. The only exception is the case of three robots in the tree where we have necessary and sufficient conditions for exploration to be achievable.

2 Definitions

Let $G = (V, E)$ be an undirected connected simple graph where V is the set of vertices and E the set of edges, with $|V| = n$, let us indicate with $N(x)$ the neighbors of node x. Let $\lambda_x : N(x) \to \mathcal{L}$ be a local labeling function that associate a label from a set $\mathcal{L}$ to each edge incident on x in such a way that $\lambda_x(y) = \lambda_x(z)$ if and only if $y = z$. Let $\lambda = \{\lambda_x : x \in V\}$ be the global labeling function and let (G, λ) denote the resulting edge-labeled graph. Let $P[x, y]$ denote the set of paths from node x to node y and let Λ denote the extension of λ from edges to paths.

The vertices of (G, λ) can be partitioned according to the equivalence classes they belong to, where an equivalence class $[v_0]$ of vertices of G is such that for each $v \in [v_0]$, there exists an automorphism σ of G that preserves the labels such that $\sigma(v) = v_0$. We say that (G, λ) is *symmetric* if there exists an automorphism of (G, λ) that preserves the edge-labeling λ, *asymmetric* otherwise. Note that, in (G, λ) the equivalence classes can be ordered using the fact that each identifies a different "view" of the graph.

Operating in a system (G, λ) is a set $\mathcal{A}$ of k identical robots; initially there is at most one robot in each node. During the execution of the algorithms, robots move from node to neighboring node, and at any time they occupy some nodes of the graph. Let $\Psi : \mathcal{A} \to V$ be the placement function returning the position of a given robot. Let (G, λ, Ψ) denote the edge-labeled graph with the placement of the robots. We say that (G, λ, Ψ) is *symmetric* if there exists an automorphism of (G, λ, Ψ), when considering automorphisms that preserve the labels and the placement; *asymmetric* otherwise. Clearly, also in (G, λ, Ψ) the corresponding equivalence classes can be ordered using the fact that each identifies a different "view" of the graph taking into account also the placement of the robots.

Each robot operates in Look-Compute-Move cycles, which are performed asynchronously for each robots. When *Looking*, a robot perceives a snapshot of the labeled graph with the current position of the robots (called a *view* of the graph), when *Computing* it decides where to move on the basis of the snapshot, when *Moving* it actually moves to the chosen neighboring node. The time between *Look*, *Compute*, and *Move* operations is finite but unbounded, and is decided by the adversary for each action of each robot. The only constraint is that moves are instantaneous, as in [2, 15, 16], and hence any robot performing a Look operation sees all other robots at nodes of the ring and not on edges. Because of asynchronicity, and thus different delays, robots may move based on significantly outdated perceptions. We assume that the robots can perceive, during the Look operation, if there is one or more robots in a given location; this ability, called *multiplicity detection* is a standard assumption in the continuous model [2, 15, 16]. We will say that there is a *tower* on a node if it is occupied by more than one robot.

Consider a graph (G, λ) where k robots are initially located in k distinct nodes. The *Gathering* problem consists of gathering the k robots in an arbitrary node in finite time. The *Exploration*

problem consists of having every node of the graph visited by at least one robot and all robots entering a quiescent state within finite time.

3 Gathering in Arbitrary Graphs

3.1 Preliminaries

First of all notice that if (G, λ, Ψ) is symmetric, the gathering problem is unsolvable. In fact, a synchronous scheduler would force the robots to perform the same actions thus failing to brake the symmetry. Because of the impossibility result, from now on we assume (G, λ, Ψ) is initially asymmetric.

Let $PATHS(v) = \{\lambda_v(\pi) : \pi \in P[v, x], \forall x \in V - \{v\}\}$ be the set containing all the shortest labeled paths from v to all other nodes, where v is not an articulation point. Let $SORT(PATHS(v))$ be the set of labeled paths sorted first by the length and then by lexicographic order of their associated sequences of labels. Given two vertices v_1 and v_2, we define the following order: $v_1 < v_2$ if $SORT(PATHS(v_1)) <_{lex} SORT(PATHS(v_2))$, where $<_{lex}$ denotes lexicographic order. We have:

Lemma 3.1 *If (G, λ, Ψ) is asymmetric, there is a unique* $\min_v\{SORT(PATHS(v))\}$.

We remind that $\Psi(a)$ denotes the position of a robot $a \in A$ in the graph. Let $SD(v) = \sum_{a \in A} \mathsf{dist}(v, \Psi(a))$ denote the sum of the distances from a node v to all the robots and let $\mathsf{code}(v) = (SD(v), SORT(PATHS(v)))$.

We now introduce the notion of Weber node and minimal Weber node of a configuration (G, λ, Ψ). Recall that $[u]$ denotes the equivalence class of u in (G, λ) and does not depend on the positions of the robots. A *Weber node* of (G, λ, Ψ) is a node u such that $SD(u)$ is minimal among all $SD(v) : v \in [u]$. A *Minimal Weber node* of (G, λ, Ψ) is a node u such that $\mathsf{code}(u)$ is minimal among all $\mathsf{code}(v) : v \in [u]$.

From the definition of $\mathsf{code}()$ and from Lemma 5.1, we can derive the following interesting property of the minimal Weber node:

Property 3.1 *If (G, λ, Ψ) is asymmetric, for any equivalence class $[u]$, its minimal Weber node is unique.*

Given an asymmetric configuration with its minimal Weber node w, we can order the robots as follows. We say that a robot a is closer to w than a robot b if either $\mathsf{dist}(a, w) < \mathsf{dist}(b, w)$, or $\mathsf{dist}(a, w) = \mathsf{dist}(b, w)$ and the label $\Lambda(\Pi_a)$ of a shortest path from a to w is lexicographically smaller (or "weaker") than the label $\Lambda(\Pi_b)$ of any shortest path from b to w.

We now show that in an asymmetric configuration with an odd number of agents, there is always at least one robot that can move towards a minimal Weber node while preserving asymmetry.

Theorem 3.2 *In an asymmetric (G, λ, Ψ) with an odd number of robots k, consider a minimal Weber node r_0. If the closest robot from r_0 that is not on r_0 move towards r_0, then the new configuration (G, λ, Ψ') is asymmetric and r_0 is still a minimal Weber node.*

Proof: Let r_0 be the minimal Weber node for (G, λ, Ψ) and let a be a robot not on r_0 such that $\mathsf{dist}(\Psi(a), r_0)$ is minimal (ties are broken using the label of the paths to r_0). We show that a can always move towards the minimal Weber node maintaining it as a unique minimal Weber node.

Consider first the case when there is no robot on the minimal weber node r_0 for (G, λ, Ψ). Let $\pi(r_0, a)$ be the label of the path from r_0 to $\Psi(a)$ after a has moved. Suppose by contradiction that the movement has created a symmetry and thus there exists an automorphism σ of (G, Ψ). This means that there exists $r_1 \neq r_0 \in [r_0]$ such that $\mathsf{code}(r_1) = \mathsf{code}(r_0)$. Note than after the movement of a $\sum_{a \in A} \mathsf{dist}(r_i, \Psi(a))$ has been reduced by 1 for r_0 and by at most one for any other $r \in R$, and since r_0 was minimum before a moved, we have that $\sum_{a \in A} \mathsf{dist}(r_1, \Psi(a)) = \sum_{a \in A} \mathsf{dist}(r_0, \Psi(a))$. Since $\mathsf{code}(r_1) = \mathsf{code}(r_0)$, there exists a robot a_1 such that the path from r_1 to $\Psi(a_1)$ is $\pi(r_0, a)$ and since $r_0 \neq r_1$ we can ensure that $a \neq a_1$. Since a_1 has not moved, the distance between r_1 and $\Psi(a_1)$ at the previous step was smaller than the distance between r_0 and a, i.e., $\mathsf{code}(r_1) < \mathsf{code}(r_0)$, which is impossible.

Consider now the case when there is a robot a_0 on r_0. Let $a = a_1 \neq a_0$ be the closest robot to r_0 that is not on r_0. Let a move towards r_0 and, by contradiction, let it create a symmetry. Then we have an automorphism of (G, λ, Ψ) mapping r_0 to the new position $\Psi'(a)$ of a. Indeed, since a is the closest robot to r_0, for the same reasons as in the previous case, there cannot be an automorphism that maps r_0 to a vertex different from the new position $r_1 = \Psi'(a)$ of a. However, since we have assumed that the number k of robots is odd, we cannot have $PATHS(r_0) = PATHS(r_1)$. Otherwise, it would imply that at least for one robot the two paths leading from its position to r_0 and to r_1 have the same label. Thus if k is odd, the move of a cannot generate a symmetry. □

All the different "views" (i.e., equivalent classes) of (G, λ) can be ordered. Let R be the minimum equivalence class of (G, λ) according to an arbitrary predefined order (notice that R might contain all the vertices, or it could contain a single one in the case of an asymmetric (G, λ)). If $|R| = 1$, the gathering point is unique regardless of the robots placement or movements. If $|R| > 1$ (i.e., (G, λ) is not asymmetric), there is a unique minimal Weber node of R and Theorem 3.2 suggests a simple gathering algorithm.

Algorithm GATHERING(R)

$r_0 :=$ MINIMALWEBERNODE(G, λ, Ψ) in R
Among all robots that are not on r_0, let a be the closest robot to r_0.
If I am a **then**
    move towards r_0

Theorem 3.3 *Algorithm* GATHERING*(R) terminates correctly, for any odd $k = |R|$ and it terminates correctly regardless of k if (G, λ) is asymmetric.*

When we have an even number of robots, we always have a minimum Weber node. However, the simple algorithm GATHERING may not work. For example, in the asymmetric configuration presented on Figure 1, the minimal Weber node is the node on the cycle hosting a robot. However, if any other robot move towards it, we obtain a symmetric configuration.

When there are 4 robots, there is always a robot that can move in order to reduce $\min\{SD(v) \mid v \in R\}$.

Theorem 3.4 *Consider an asymmetric configuration (G, λ, Ψ) with 4 robots and no tower and let r_0 be the minimal Weber node.*

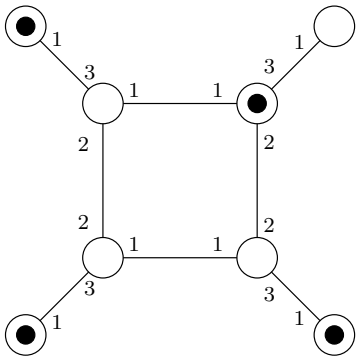

Figure 1: A configuration where no robot can move towards the minimal Weber node without creating a symmetry

(1) *If there is no robot on r_0, the closest robot a to r_0 can move towards r_0 without creating a symmetry.*

(2) *If there is exactly one robot b on r_0, let a be the closest robot to r_0 (that is not on r_0). Either a can move towards r_0 without creating a symmetry, or there exists $r_1 \in [r_0]$ such that b can move towards r_1 without creating a symmetry (this vertex r_1 will be the new minimal Weber node).*

In both cases, $\min\{SD(w') \mid w' \in [w]\}$ *decreases when the move is performed.*

Proof: Let r_0 be the minimal Weber node for (G, λ, Ψ) and let a be a robot not on r_0 such that $\mathsf{dist}(\Psi(a), r_0)$ is minimal (ties are broken using the label of the paths to r_0).

When there is no robot on r_0, for the same reasons as in the proof of Theorem 3.2, a can move towards r_0 without creating a symmetry. It is easy to see that $SD(r_0)$ has decreased. Thus (1) holds.

Suppose now that there is a robot b on r_0 and let a moves towards r_0. Note that such a move decreases the value of $SD(r_0)$. Assume that this move of a creates a symmetry.

Let r_1 be the position of a when it performs such a move. Since it creates a symmetry, for the same reasons as in the proof of Theorem 3.2, $r_1 \in [r_0]$, $\mathsf{code}(r_0) = \mathsf{code}(r_1)$ and there exists an automorphism σ_1 of (G, λ, Ψ_1) such that $\sigma_1(r_0) = r_1$ and $\sigma_1(r_1) = r_0$. Let π be the label of the shortest path from r_0 to r_1; by symmetry, π is also the label of the shortest path from r_1 to r_0 and $(\lambda_{\Psi(a)}(r_1), \lambda_{r_1}(\Psi(a))) \cdot \pi$ is the label of the shortest path from $\Psi(a)$ to r_0.

Let c be the robot different from a, b that is the closest to r_0 and let η be the label of the path from $\Psi(c)$ to r_0. Let d be the remaining robot. Since $\sigma_1(r_0) = r_1$, the shortest path from $\Psi(d)$ to r_1 is labelled by η.

Suppose now that a does not move and that the robot b, located on r_0, move towards r_1 on a vertex u. Since when a moves towards r_0, $SD(r_1) = SD(r_0)$, it implies that even before any move, $SD(r_1) = SD(r_0)$. And thus, when b moves towards r_1, it decreases $SD(r_1)$ and increases $SD(r_0)$. If such a move does not create a symmetry, the new Weber node is r_1.

Suppose that it creates a symmetry, then there exists an automorphism σ_2 of (G, λ, Ψ_2) such that $\sigma_2(u) = \Psi(a)$. Since $\eta \cdot (\lambda_{r_1}(\Psi(a)), \lambda_{\Psi(a)}(r_1))$ is the label of a path from $\Psi(d)$ to $\Psi(a)$,

$\eta \cdot (\lambda_{r_1}(\Psi(a)), \lambda_{\Psi(a)}(r_1))$ is the label of a path from $\Psi(d)$ to u. Thus, $(\lambda_{r_1}(\Psi(a)), \lambda_{\Psi(a)}(r_1)) = (\lambda_{r_0}(u), \lambda_u(r_0))$. Consequently, $\pi(\lambda_{r_1}(\Psi(a)), \lambda_{\Psi(a)}(r_1)) \cdot \pi'$ for some π'. Thus, $(\lambda_{\Psi(a)}(r_1), \lambda_{r_1}(\Psi(a))) \cdot \pi = (\lambda_{\Psi(a)}(r_1), \lambda_{r_1}(\Psi(a))) \cdot (\lambda_{r_1}(\Psi(a)), \lambda_{\Psi(a)}(r_1)) \cdot \pi'$ is not the label of the shortest path from $\Psi(a)$ to r_0, since π' is also a path from $\Psi(a')$ to r_0. We then have a contradiction and (2) holds. □

The preceding theorem enables us to construct an algorithm when there are exactly four robots.

Corollary 3.5 *There exists a gathering algorithm for any asymmetric (G, λ, Ψ) with 4 robots.*

More generally, we believe that there always exists a robot that can move without creating a symmetry and such that $\min\{SD(w') \mid w' \in R\}$ decreases (the minimal Weber node may be different before and after the move).

Conjecture 3.1 (Weber node invariance conjecture) *In an asymmetric (G, λ, Ψ) with at least 3 robots, there exists always a robot such that its movement towards a minimal Weber node w (called* feasible movement*) preserves the existence of a unique minimal Weber node $w' \in [w]$ with $\mathsf{code}(w') < \mathsf{code}(w)$.*

In fact, the algorithms we have presented before works because the conjecture is true when (G, λ) is asymmetric, when there is an odd number of robots, or when there are 4 robots. More generally, if Conjecture 3.1 holds for some k, there exists a gathering algorithm for any asymmetric configuration with k robots. Thus, if true, Conjecture 3.1 would imply the following conjecture.

Conjecture 3.2 *There exists a gathering algorithm for any asymmetric (G, λ, ψ), regardless of $k \geq 3$.*

4 Exploration in Arbitrary Graphs

We now focus on the exploration problem. As for the gathering problem, we consider only asymmetric placements of the robots otherwise the problem cannot be solved. Interestingly, when only three robots are available not all graphs can be explored; moreover, the exploration algorithm is quite complicated. On the other hand, when k is odd strictly greater than three or $k = 4$ all graphs with asymmetric placement can be explored and the algorithms are given.

4.1 Exploring with $k = 3$

In the following we characterize the class of graphs where the problem is solvable and we design an exploration algorithm when the team is composed by three robots.

4.1.1 Necessary Condition

We now describe the necessary condition for explorability by three robots: there is a node (or an edge) in G whose removal creates either a graph with a Hamiltonian path, or a graph that is spanned by two intersecting elementary paths satisfying the conditions listed below.

Theorem 4.1 *A graph (G, λ) can be explored by three robots starting in any asymmetric configuration only if we are in one of the following cases:*

(Case 1) there exists an elementary path $P = (x_1, \ldots, x_k)$ and two neighbors u_0, v_0 both different from x_1 such that:

(A1) u_0 does not appear in P,

(A2) any vertex $v \notin \{u_0, v_0\}$ appears in P,

(A3) (G, λ, Ψ_0) is asymmetric where Ψ_0 maps a robot to x_1, one to u_0 and one to v_0,

(Case 2) There exists two elementary paths $P_1 = (x_1, \ldots, x_k)$ and $P_2 = (y_1, \ldots, y_\ell)$ and two neighbors u_0, v_0 both different from x_1 such that the following conditions are satisfied:

(B1) $x_k = y_1$,

(B2) either $u_0 = x_k$ or $u_0 \in N(x_k)$,

(B3) $u_0 \neq x_i$, for all $i \in [1, k-1]$,

(B4) any vertex $v \notin \{u_0, v_0\}$ appears either in P_1 or in P_2,

(B5) (G, λ, Ψ_0) is asymmetric where Ψ_0 maps a robot to x_1, one to u_0 and one to v_0,

(B6) for any consecutive vertices y, y' of P_2, and for any vertex x of P_1 adjacent to u_0, there is no automorphism of (G, λ) mapping x to y and u_0 to y'.

(B7) for any distinct vertices $y, y' \in P_2$, there is no automorphism of (G, λ) mapping y to y'.

Proof: Consider any initial configuration and an execution of an exploration algorithm $\mathcal{A}$. Note that as long as no tower is created, the situation cannot become symmetric during the execution. So, the situation always remains asymmetric and at some point, a tower is created: indeed, if no tower is created, then any configuration can be an initial configuration where no vertex has been explored. In the following, a *small* tower is a tower made of 2 robots, while a *big* tower is a tower made of 3 robots.

When a small tower with two robots a and b is created, if a can move without creating a symmetry then b can move too (they see exactly the same configuration). Suppose that there is a small tower of two robots a and b where the two robots can move on a vertex u. If the third robot c is not located on u, then since the execution is asynchronous, it is possible that only one robot, say a, moves towards u. In that case, we end up in a configuration without a tower, i.e., a potential initial configuration, and thus the whole graph must be re-explored. If the robot c is on u, if a can move towards u, then there is an execution where a and b move towards u, i.e., they create a big tower. Suppose now that a big tower is created and consider a synchronous execution where each time a robot of the big tower can move on an adjacent vertex, all three robots perform the move (they all see exactly the same configuration). Since the execution must terminate, we have that at some point the big tower is located on a vertex v where the robots do not move any more. Since the robots are oblivious, a tower of 3 robots located on a vertex u will always move to the same neighbor of u. Thus, if a vertex is visited twice by such a tower, the execution does not terminate. As a consequence, the sequence of vertices traversed by the tower is an elementary path.

We consider executions where, as long as no tower is created, at most one robot moves at each step. In the following, we distinguish two cases.

(Case 1) *starting from (G, λ, Ψ) no execution (satisfying the stated conditions) creates a big tower.* Consider the step of an execution where a small tower is created. As explained above, this tower does not move (because, either we will go back to an initial configuration, or we will create a big tower). Consider the graph (G, λ) just before the tower is created in some vertex u_0 by a robot in v_0 moving from v_0 to u_0 and let x_1 be the position of the third robot. Since the situation is asymmetric at this point, condition (A3) is satisfied. Moreover, this situation is a possible initial configuration. Since the tower does not move and since no tower of three robots is created, the third robot must go through every vertex of $V(G) \setminus \{u, v\}$ without traversing u. Since the robot

is oblivious, it cannot go twice through a same vertex. Consequently, the vertices traversed by the robots form an elementary path. Thus, Conditions (A1) and (A2) are satisfied.
(Case 2) *a big tower is created by some execution starting from* (G, λ, Ψ).
Suppose now that a big tower is created. A first part of the graph may be explored before the big tower is created (but after a small tower has been created), while the second part may be explored by the moving big tower. First note that, as explained above, we may assume that the small tower moves only to create the big tower. We consider an execution, where once in a big tower, the three robots always stay together (all moves are done synchronously by the three robots). Consider the step where a small tower is created (since at most one robot moves at each step while there is no tower, such a step always exists). Assume that a robot a on v_0 moves to a neighbor u_0 to create such a tower with a robot b located on u_0. Let x_1 be the position of the third robot c at this step. Consider now the sequence of configurations obtained until a big tower is created. Since we assume the tower in u_0 does not move, except to create a big tower, in each of these configurations, robot c moves. Since we cannot go twice trough the same configuration, robot c follows an elementary path $P = (x_1, \ldots, x_k)$ avoiding u_0 where x_k is a neighbor of u_0 and the big tower is created at the next step. The big tower can be made in two different ways, either robot c moves to u_0, or robots a and b move to x_k. In both cases, consider the execution where the three robots move together until they stop. This tower follows an elementary path $P' = (y_1, \ldots, y_\ell)$. When c moves to u_0, let $x_{k+1} = y_1 = u_0$, let $P_1 = (x_1, \ldots, x_k, x_{k+1})$ and let $P_2 = P' = (y_1, \ldots, y_\ell)$. In the second case, where a and b moves to x_k, let $P_1 = P(x_1, \ldots, x_k)$ and let $P_2 = P' = (y_1, \ldots, y_\ell)$ where $y_1 = x_k$. It is easy to see that in both cases, Conditions (B1), (B2), (B3) are satisfied. Moreover, since the configuration where a is in u_0, b is in v_0 and c is in x_1 may be an initial configuration, all other vertices must be explored after the tower of two robots is created in u_0: Condition $(B4)$ holds. Furthermore, the situation where a is in u_0, b is in v_0 and c is in x_1 appears in the execution and thus it cannot be symmetric, i.e., Condition (B5) holds. Suppose now that there exists an automorphism σ of (G, λ), a vertex $x_i \in P_1$ and two consecutive vertices $y_j, y_{j+1} \in P_2$ such that $\sigma(u_0) = y_j$ and $\sigma(x_i) = y_{j+1}$, or $\sigma(u_0) = y_{j+1}$ and $\sigma(x_i) = y_j$. Consider the step of the execution where the big tower is in y_j. At this point, following the algorithm, the robots should move to y_{j+1}. If $\sigma(u_0) = y_j$ and $\sigma(x_i) = y_{j+1}$, consider the execution where only one robot moves to y_j. If $\sigma(u_0) = y_{j+1}$ and $\sigma(x_i) = y_j$, consider the execution where exactly two robots move to y_j. In both cases, we arrive in a configuration (G, λ, Ψ_1) where there are two robots in $\sigma(u_0)$ and one robot in $\sigma(x_i)$. Let (G, λ, Ψ_2) be the configuration where there are two robots in u_0 and one in x_i. At this point we have constructed an execution where we go twice through the same configuration (up to isomorphism). This implies we can find an infinite execution of the algorithm where the robots never stop, i.e., the algorithm cannot solve exploration. Thus condition (B6) is satisfied. For the same reasons, we cannot have an automorphism mapping a vertex y_j of P_2 to another vertex $y_{j'}$ of P_2: again, it would mean that we can construct an execution where we go twice through the same configuration (up to isomorphism).

□

Remark 4.1 *Consider two vertices u, v of (G, λ). Since we are considering a graph with port-numbers, there is at most one vertex t such that there exists an automorphism σ such that $\sigma(u) = v$, $\sigma(v) = t$ and $\sigma(t) = u$. If such a t exists, then we call it the* troublemaker *of u and v and we denotes it by $tb(u, v)$.*

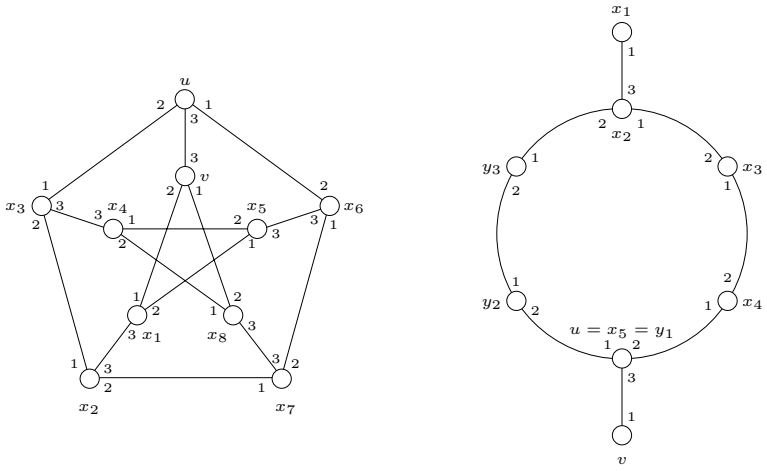

Figure 2: On the left, a graph satisfying conditions of (Case 1) of Theorem 4.1; on the right, a graph satisfying conditions of (Case 2) of Theorem 4.1

4.1.2 Sufficient Condition

We now constructively show that the necessary condition of Theorem 4.1 is also sufficient. We first describe a procedure SCATTER(u_0, v_0, x_1) that enables to scatter the robots on three vertices u, v, x such that there exists an automorphism σ of (G, λ) such that $\sigma(u_0) = u, \sigma(v_0) = v$ and $\sigma(x_1) = x$, provided that $\{u_0, v_0\}$ is an edge of G, that x_1 does not disconnect G and that (G, λ, Ψ_0) is asymmetric where Ψ_0 maps a robot to x_1, one to u_0 and one to v_0.

For any vertex $x \in [x_1]$, there is an automorphism σ of (G, λ) such that $\sigma(x_1) = x$; let $u(x) = \sigma(u_0)$, $v(x) = \sigma(v_0)$. Consider an asymmetric configuration (G, λ, Ψ) with three robots. For any $x \in [x_1]$, let a be the closest robot from x, let $b_1 \neq a$ (resp. b_2) be the closest robot from $u(x)$ (resp. $v(x)$) in $G \setminus \{x\}$ and let c_1 (resp. c_2) be the remaining one.

Remark 4.2 *Either c_1 can reach $v(x)$ in $G \setminus \{x, u(x)\}$, or c_2 can reach $u(x)$ in $G \setminus \{x, v(x)\}$.*

If c_1 can reach $v(x)$ in $G \setminus \{x, u(x)\}$, we will distinguish different cases in the algorithm:
- [(Case A)] *$tb(x, \Psi(c_1))$ does not exists, or b_1 can reach $u(x)$ in $G \setminus \{x, \Psi(c_1), tb(x, \Psi(c_1))\}$; and $tb(x, u(x))$ does not exists or c_1 can reach $v(x)$ in $G \setminus \{x, u(x), tb(x, u(x))\}$.*
- [(Case B)] *$tb(x, \Psi(c_1))$ does not exists, or b_1 can reach $u(x)$ in $G \setminus \{x, \Psi(c_1), tb(x, \Psi(c_1))\}$; and c_1 cannot reach $v(x)$ in $G \setminus \{x, u(x), tb(x, u(x))\}$.*
- [(Case C)] *b_1 cannot reach $u(x)$ in $G \setminus \{x, \Psi(c_1), tb(x, \Psi(c_1))\}$.*

If c_1 cannot reach $v(x)$ in $G \setminus \{x, u(x)\}$, then we know by Remark 4.2 that c_2 can reach $u(x)$ in $G \setminus \{x, v(x)\}$ and then, we will consider the same cases up to a permutation of $u(x)$ and $v(x)$ and a replacement of b_1 and c_1 respectively by b_2 and c_2. If we are in (Case B) (resp. (Case C)) and if by exchanging the roles of b_1 and c_1, we can be in (Case A) (resp. (Case A) or (Case B)), we will exchange their role.

Let $\pi(a)$ be the labeled shortest path from $\Psi(a)$ to x in G. In (Case A) and (Case B), let $\pi(b_1)$ be the one from b_1 to $u(x)$ in $G \setminus \{x, \Psi(c_1), tb(x, \Psi(c_1))\}$; in (Case C), let $\pi(b_1)$ be the one from b_1 to $u(x)$ in $G \setminus \{x\}$. In (Case A), let $\pi(c_1)$ be the labeled shortest path from

c_1 to $v(x)$ in $G \setminus \{x, u(x), tb(x, u(x))\}$ (or $G \setminus \{x, u(x), tb(x, u(x))\}$ if $tb(x, u(x))$ does not exists). In (Case B), let $\pi(c_1)$ be the one from c_1 to $tb(x, u(x))$ in $G \setminus \{x, u(x)\}$. In (Case C), let $\pi(c_1)$ be the one from c_1 to $v(x)$ in $G \setminus \{x, u(x)\}$. We define $\mathsf{route}'(x)(|\pi(a)| + |\pi(b_1)| + |\pi(c_1)|, |\pi(a)|, \Lambda(\pi(a)), |\pi(b_1)|, \Lambda(\pi(b_1)), |\pi(c_1)|, \Lambda(\pi(c_1)))$. In (Case A), $\mathsf{route}(x) = (A, \mathsf{route}'(x))$, in (Case B), $\mathsf{route}(x)(B, \mathsf{route}'(x))$, and in (Case C), $\mathsf{route}(x) = (C, \mathsf{route}'(x)))$. Given two vertices $x, x' \in [x_1]$, we use lexicographic order to compare $\mathsf{route}(x)$ and $\mathsf{route}(x')$. If c_1 cannot reach $v(x)$ in $G \setminus \{x, u(x)\}$, we do the same, but we exchange $u(x)$ and $v(x)$ and we replace b_1 and c_1 respectively by b_2 and c_2.

Algorithm SCATTER(u_0, v_0, x_1).

Case 1. [*There exists $x \in [x_1]$ such that $\pi(c_1)$ exists.*]
Let $x \in [x_1]$ such that $\pi(c_1)$ exists and such that $\mathsf{route}(x)$ is minimal.
- **Case 1.1.** [*There is a robot on x, a robot on $u(x)$ and a robot on $v(x)$*]
  Terminate.
- **Case 1.2.** [*There is a robot a on x, a robot b_1 on $u(x)$*]
  /* We know there exists a path from $\Psi(c_1)$ to $v(x)$ in $G \setminus \{x, u(x)\}$ and that we are in (Case A) or (Case B)*/
  - **Case 1.2.1.** [*$tb(x, u(x)$ does not exists, or there exists a path from $\Psi(c_1)$ to $v(x)$ in $G \setminus \{x, u(x), tb(x, u(x))\}$*] /*We are in (Case A)*/
    c_1 move towards $v(x)$ in $G \setminus \{x, u(x), tb(x, u(x))\}$.
  - **Case 1.2.2** [*$\Psi(c_1) \in N(tb(x, u(x)))$, and $z \notin \{\Psi(c_1), tb(\Psi(b_1), \Psi(c_1))\}$, where z the neighbor of x on the shortest path from x to $v(u(x))$ in $G \setminus \{u(x), u(u(x))\}$*]
    a move to z in $G \setminus \{u(x), u(u(x))\}$.
  - **Case 1.2.3** [$\Psi(c_1) \in N(tb(x, u(x)))$]
    b_1 move towards $v(tb(x, u(x)))$ in $G \setminus \{tb(x, u(x)), u(tb(x, u(x)))\}$.
  - **Case 1.2.4** [$\Psi(c_1) \notin N(tb(x, u(x)))$]
    c_1 move towards $tb(x, u(x))$ in $G \setminus \{x, u(x)\}$.
- **Case 1.3.** [*There is a robot a on x and b_1, the closest robot to $u(x)$ is not on $u(x)$*]
  - **Case 1.3.1** [*$\mathsf{route}(x) = (A, \mathsf{route}'(x))$ or $\mathsf{route}(x) = (B, \mathsf{route}'(x))$.*]
    b_1 moves towards $u(x)$ in $G \setminus \{x, \Psi(c_1), tb(x, \Psi(c_1))\}$.
  - **Case 1.3.2**[*$\mathsf{route}(x) = (C, \mathsf{route}'(x))$*]
    b_1 moves towards $u(x)$ in $G \setminus \{x\}$.
- **Case 1.4.** [*There is no robot on x*]
  a, the closest robot to x in G, moves towards x.

Case 2. The same, exchanging the role of $u(x)$ and $v(x)$ and replacing b_1 and c_1 resp. by b_2, c_2.

To examine the termination of the algorithm we first need some Lemmas.

Lemma 4.2 *When applying Case 1.2.1., Case 1.2.4., Case 1.3. or Case 1.4. during the execution of Algorithm* SCATTER *from an asymmetric configuration, no symmetry is created, no tower is created and $\mathsf{route}(x)$ strictly decreases.*

Proof:

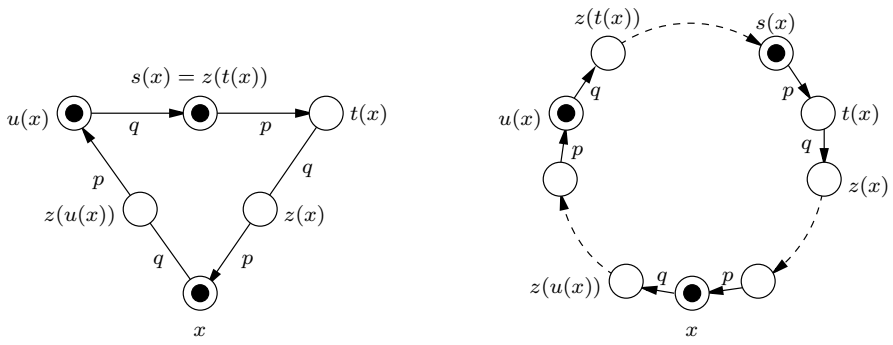

Figure 3: Different cases in Lemma 4.3

- *Case 1.4.*

  For the same reasons as in our Gathering algorithm, if the move of a towards x creates a symmetry, then x is not the element of $[x_1]$ minimizing $\mathsf{route}'(x)$. Indeed, since the distance from a to x has decreased while the other distances (that appears in $\mathsf{route}(x)$) have not been modified, $\mathsf{route}'(x)$ decreases when such a move is performed.

  Moreover, if before a moves, the first component of $\mathsf{route}(x)$ is A (resp. B), then it remains A (resp. B). Indeed, this component depends only from x, $\Psi(b_1)$ and $\Psi(c_1)$.

- *Case 1.3.1.*

  Since the first component of $\mathsf{route}(x)$ is A or B, b_1 avoids $tb(x, \Psi(c_1))$: no symmetry can be created.

- *Case 1.3.2.*

  Suppose that when b_1 moves towards $u(x)$, it creates a symmetry. Then it means that b_1 moves to $tb(x, \Psi(c_1))$.

  Let $t = \Psi(c_1)$ and $s = tb(x, t)$. Consequently, $t \in [x_1]$ and there exists σ such that $\sigma(t) = x$, $\sigma(x) = s$, $\sigma(s) = t$. Moreover, if we consider the configuration before b_1 moves, we have $\mathsf{dist}(c_1, t) = 0$, $\mathsf{dist}_{G\setminus\{t\}}(a, u(t))\mathsf{dist}_{G\setminus\{x\}}(s, u(x))\mathsf{dist}_{G\setminus\{x\}}(b_1, u(x)) - 1$.

  Since $\mathsf{dist}_{G\setminus\{G\setminus\{t,u(t)\}\}}(b_1, v(t)) \leq \mathsf{dist}_{G\setminus\{t,u(t)\}}(s, v(t)) + 1 = \mathsf{dist}(c_1, v(x)) + 1$, we have $\mathsf{route}(t)$ is weaker than $\mathsf{route}(x)$, which is impossible

- *Cases 1.2.1. and 1.2.4.*

  In Case 1.2.1. (resp. Case 1.2.4.), the first component of $\mathsf{route}(x)$ is C (resp. A) before and after c_1's move. Since c does not ends up in $tb(x, u(x))$, the next configuration is asymmetric.

Moreover, in all these cases, it is clear that $\mathsf{route}(x)$ decreases and that no tower is created. □

Lemma 4.3 *When a is on x, b_1 is on $u(x)$ and when c_1 is on a vertex in $N(tb(x, u(x)))$ and cannot reach $v(x)$ in $G \setminus \{x, u(x), tb(x, u(x))\}$, then either a can move towards $v(u(x))$, or a can move towards $v(tb(x, u(x)))$ without creating a tower or a symmetry. Moreover, $min\{\mathsf{route}(x)|x \in [x1]\}$ decreases strictly.*

Proof: Suppose that we are in Case 1.2.2. or 1.2.3., i.e., a is on x, b_1 is on $u(x)$ and $\Psi(c_1) \in N(tb(x,u(x))$. Let denote $tb(x,u(x))$ by $t(x)$ and let $s(x) = \Psi(c_1)$. Let $z(x)$ be the first vertex on a shortest path from $t(x)$ to $v(x)$. Let $p = \lambda_{s(x)}(t(x))$ and $q = \lambda_{t(x)}(z(x))$. Given $y \in [x_1]$ and an automorphism σ such that $\sigma(x) = y$, we denote $\sigma(s(x))$ (resp. $\sigma(t(x))$, $\sigma(z(x))$) by $s(y)$ (resp. $t(y)$, $z(y)$).

First suppose that $z(x) \notin [x_1]$ or that $s(x) \notin [x_1]$. In the first case, $z(u(x)), z(t(x)) \in [z(x)] \neq [x_1]$. If $z(u(x)) \neq s(x)$, a can safely move to $z(u(x))$ without creating a symmetry. In the new configuration, let consider $x' = u(x)$: b_1 is on $x' = u(x)$, $tb(x', \Psi(a))$ does not exists, c_1 can reach $u(x') = t(x)$ in $G\backslash\{x', \Psi(a)\}$ and a can reach $v(x')$ in $G\backslash\{x', u(x'), tb(x', u(x'))\} = G\backslash\{x, u(x), t(x)\}$. Consequently, the first component of $\mathsf{route}(x')$ is A while the first component of $\mathsf{route}(x)$ was B. Thus $\min\{\mathsf{route}(x) \mid x \in [x_1]\}$ has strictly decreased. If $s(x) = z(u(x))$, then $z(t(x)) \notin [x_1]$ and $z(t(x)) \neq s(x)$ and b_1 can safely move to $z(t(x))$ and for the same reasons as before $\min\{\mathsf{route}(x) \mid x \in [x_1]\}$ strictly decreases.

Suppose now that $s(x), z(x) \in [x_1]$. Since $s(x), t(x), z(x) \in [x_1]$, for any $y \in [x_1]$, there are two ports labeled by p and q at y and these two ports lead to vertices of $[x_1]$. Consider the path π_1 starting at $s(x)$ and obtained by taking alternatively ports p and q. We know that this path π_1 contains only vertices of $[x_1]$ and thus there exists p', q' such that for any consecutive vertices a, b in π_1, either $(\delta_a(b), \delta_b(a)) = (p, p')$ or $(\delta_a(b), \delta_b(a))(q, q')$. Consequently, this path ends up in $s(x)$. If the path π_1 avoids x and $u(x)$, then b_1 can use π_1 backwards to reach $z(x)$ in $G\backslash\{x, u(x), tb(x, u(x))\}$ and thus b_1 can also reach $v(x)$ in $G \setminus \{x, u(x), tb(x, u(x))\}$, i.e., we are in Case 1.2.1. We now suppose that x appears in π_1 and $u(x)$ does not, or that x appears before $u(x)$ in π_1. Let ℓ be the length of π_1 between $t(x)$ and x. Since the situation is symmetric, we can find a path π_2 (resp. π_3) of length ℓ, starting in x (resp. $u(x)$) obtained by alternating ports q and p. Let y be the vertex appearing just before x in π_1: in the following, we will distinguish two cases: either $\lambda_y(x) = p$, or $\lambda_y(x) = q$. If $\lambda_y(x) = p$ and $s(x) = z(t(x))$ (See Figure 3, left), a can move to $z(u(x))$. Suppose that the resulting configuration is symmetric. Since the neighbor of $z(u(x))$ by port p hosts a robot, then $t(x)$ must also hosts a robot, which is wrong. After such a move of a, b_1 is on $u(x)$, c_1 can reach $t(x) = u(u(x))$ in $G \setminus \{u(x), \Psi(a), tb(u(x), \Psi(a))\}$ and a can reach $v(x)$ in $G\setminus\{u(x), t(x), tb(u(x), t(x))\}$. Consequently, after the move of a, the first component of $\mathsf{route}(u(x))$ is A, while the first component of $\mathsf{route}(x)$ was B before. If $\lambda_y(x) = p$ and $s(x) \neq z(t(x))$ (See Figure 3, right), then b_1 can move to $z(t(x))$. Indeed, the distance between $z(t(x))$ and $s(x)$ along π_3 is $\ell - 2$, while the distance between $s(x)$ and x along π_1 is $\ell + 1$ (in both cases, we start by port p); consequently, the configuration is asymmetric. For the same reasons as before, we can see that after such a move, the first component of $\mathsf{route}(t(x))$ is A, while the first component of $\mathsf{route}(x)$ was B before b_1 moves. If $\lambda_y(x) = q$, then a can move to $z(u(x))$. Suppose that such a move creates a symmetry. Let denote the vertices of π_1 by $(s(x) = a_0, t(x) = a_1, z(x) = a_2, \ldots, a_{\ell-1}, a_\ell, a_{\ell+1} = x)$. Since the distance along π_2 between $z(u(x))$ and $u(x)$ is $\ell - 1$, the distance between $s(x)$ and $z(u(x))$ along π_1 must be $\ell - 1$. This means that $z(u(x)) = a_{\ell-1}$. However, since $\delta_y(x) = \delta_{a_\ell}(a_{\ell+1}) = q$, $\delta_{a_{\ell-2}}(z(u(x)) = q$ and thus $x = a_{\ell-2}$, which is impossible. For the same reasons as before, we can see that after such a move, the first component of $\mathsf{route}(u(x))$ is A, while the first component of $\mathsf{route}(x)$ was B before a moves. If $u(x)$ appears in π_1 and x does not, or if $u(x)$ appears before x in π_1, we can show by similar arguments, that either a or b_1 can move, resp., to $z(u(x))$ or $z(t(x))$. □

From Lemmas 4.2 and 4.3, we have that:

Theorem 4.4 *Any execution of Algorithm* SCATTER *correctly terminates.*

In order to use Procedure SCATTER in our algorithm, we need the following lemmas.

Lemma 4.5 *Given a graph (G, λ) satisfying the conditions of (Case 1) of Theorem 4.1 for some vertices u_0, v_0 and some path $P = (x_1, \ldots, x_k)$, we can assume that x_1 is not an articulation point of G.*

Proof: Suppose that x_1 is an articulation point. Since P is an elementary path and since $v_0 \in N(u_0)$, the connected components of $G \setminus \{x_1\}$ are $\{x_2, \ldots, x_k\}$ and $\{u_0, v_0\}$. Thus, $x_k \notin N(v_0)$ and consequently, $x_k \neq tb(u_0, v_0)$. Then replacing P by $P' = (x_k, \ldots, x_1)$, conditions of (Case 1) still hold and $x'_1 = x_k$ does not disconnect G. □

Lemma 4.6 *Given a graph G that does not satisfy the conditions of (Case 1) of Theorem 4.1, if the conditions of (Case 2) are satisfied by some vertices u_0, v_0 and some paths $P_1 = (x_1, \ldots, x_k)$ and P_2, we can assume that x_1 is not an articulation point of G.*

Proof: Suppose that x_1 is an articulation point. Since P_1 is an elementary path and since $x_k \in N(u_0) \cup \{u_0\}$, it implies that u_0 is in the same connected component as x_i, $i \in [2, k]$ in $G \setminus \{x_1\}$. Since $v_0 \neq x_1$, v_0 is also in the same connected component as u_0.

If $x_1 \neq y_j$ for all $j \in [1, \ell]$, all y_js are in the same component as u_0. Thus, there exists $p \in [1, \ell - 1]$ such that $y_p = x_1$.

Let $P'_1 = (y_\ell, y_{\ell-1}, \ldots, y_p, x_2, \ldots, x_k)$ and $P'_2 = (y_1, y_2, \ldots, y_{p-1})$. We show that conditions of (Case 2) still holds.

Since we are not in (Case 1), $k > 2$ and since $G \setminus \{x_1\}$ is disconnected, $p < \ell$ and for any $j \in [p + 1, \ell]$ and any $i \in [2, k]$, $y_j \neq x_i$. Consequently, P'_1 and P'_2 are elementary paths. It is easy to see that conditions $(B1, B2, B4, B7)$ still holds.

Moreover, condition $(B3)$ holds because if $u_0 = y_j$ for some $j \in [p + 1, \ell]$, $G \setminus \{x_1\}$ is not disconnected. If condition $(B5)$ is not satisfied, it implies that there is an automorphism σ mapping u_0 to v_0 and v_0 to y_j for some $j \in [p+1, \ell]$: thus, there is an edge between u_0 and y_j and $G \setminus \{x_1\}$ is not disconnected. Since u_0 has no neighbor in $\{y_j \mid j \in [p+1, \ell]\}$ and since condition $(B6)$ initially holds, condition $(B6)$ is still satisfied.

Therefore, we can always find vertices u_0, v_0 and paths P_1, P_2 satisfying conditions of (Case 2) such that $G \setminus \{x_1\}$ is connected. □

We are now ready to describe the algorithm that enables three asymmetrically placed robots to explore any graph satisfying conditions of Theorem 4.1.

First, suppose that conditions of (Case 1) of Theorem 4.1 are satisfied for some vertices u_0, v_0 and some elementary path $P = (x_1, \ldots, x_k)$. We know that $x_1 \neq tb(u_0, v_0)$ and from Lemma 4.5, we can assume that x_1 does not disconnect the graph. The idea is to first apply SCATTER(u_0, v_0, x_1). Then the two robots that are on u_0, v_0 (up to isomorphism) create a small tower on the node u_0, and then the third robot performs the exploration following the path P until it reaches the last node x_k of this path.

Suppose now that (Case 1) does not apply but (Case 2) does for some vertices u_0, v_0 and two elementary paths $P_1 = (x_1, \ldots, x_k)$ and $P_2 = (y_1, \ldots, y_\ell)$. From Lemma 4.6, we can assume that x_1 does not disconnect the graph. Using the algorithm SCATTER if necessary, we can assume that there is a robot on u_0, a robot on v_0 and a robot on x_1. Then the two robots that are on u_0, v_0 (up to isomorphism) create a small tower on the node u_0, and the third robot moves along the path P_1. Then, a big tower is created on y_1 and this big tower moves along P_2 until it reaches the last node

y_ℓ of this path. Note that the tower might break on the way due to asynchrony, it will however recompose itself along the path.

Algorithm EXPLORATION-3 (for robot a).

Case 1. [*There exists u_0, v_0 and a path P satisfying conditions of (Case 1) of Theorem 4.1*]

Case 1.1. [*There is a tower of two robots on u and a robot on $x_k(u)$*]
Terminate.

Case 1.2. [*There is a tower in $u \in [u_0]$ and $\Psi(a) = x_i(u)$*]
move to $x_{i+1}(u)$. /* in this case, $i < k$: we continue the exploration */

Case 1.3. [*there is a robot on $u \in [u_0]$, a robot on $x_1(u)$ and $\Psi(a) = v(u)$*]
move to u. /* in this case, we make a tower in u */

Case 1.4. [*there is no tower in the graph and there is no $u \in [u_0]$ such that there is a robot in u, a robot in $v(u)$ and a robot in $x_1(u)$*]
SCATTER(u_0, v_0, x_1).

Case 2. [*There exists u_0, v_0 and two paths P_1, P_2 satisfying conditions of (Case 2) of Theorem 4.1*]

Case 2.1. [*There is a tower of three robots on $y \in [y_\ell]$*]
Terminate.

Case 2.2. /* The robots are moving along P_2 together*/

- **Case 2.2.1** [*There is a tower of three robots on $\Psi(a) \in [y_j]$*]
  move to $\mathsf{next}(\Psi(a))$.
- **Case 2.2.2** [*There are two robots on $\Psi(a) = y \in [y_j]$ and a robot on $\mathsf{next}(y)$*]
  move to $\mathsf{next}(\Psi(a))$.
- **Case 2.2.3** [*$\Psi(a) = y \in [y_j]$ and there are two robots on $\mathsf{next}(y)$*]
  move to $\mathsf{next}(\Psi(a))$.

Case 2.3. /* We create a tower of three robots when $x_k \neq u_0$*/

- **Case 2.3.1** [*There are two robots on $\Psi(a) = u \in [u_0]$, a robot on $y_1(u) = x_k(u)$ and $u \neq x_k(u)$*]
  move to $x_k(u)$.
- **Case 2.3.2** *$\Psi(a) = u \in [u_0]$, there are two robots on $y_1(u) = x_k(u)$ and $u \neq x_k(u)$*
  move to $x_k(u)$.

Case 2.4. [*There are two robots on $u \in [u_0]$ and $\Psi(a)$ is on $x_i(u)$, $i < k$*]
move to $x_{i+1}(u)$. /* the robot a is moving along P_1*/

Case 2.5 [*There is a robot on $u \in [u_0]$, a robot on $x_1(u)$ and $\Psi(a) = v(u)$*]
move to u. /* we create a tower in u */

Case 2.6. [*There is no tower in the graph and there is no $u \in [u_0]$ such that there is a robot in u, a robot in $v(u)$ and a robot in $x_1(u)$*]
SCATTER(u_0, v_0, x_1).

Since we proved that Algorithm SCATTER is correct it is easy to check the correctness of the above exploration algorithm for a team of three robots.

Theorem 4.7 *Any execution of Algorithm* EXPLORATION-3 *correctly terminates.*

4.2 Exploring arbitrary graphs with four robots

We show that with four robots all graphs can be explored, as long as the initial placement of the robots is asymmetric. The algorithm is however slightly involved. We actually need to describe two versions depending on the existence of structures called *pseudo-neck* and *neck*. If there exist in G two vertices v, v' of degree 1 with a common neighbor u, we say that (v, v', u) is a *pseudo-neck*. A *neck* (n_1, n_2) in G consists of two neighbors $n_1, n_2 \in V(G)$ such that after removing them and their adjacent edges from G the resulting graph is connected. It is easy to see that:

Claim 1. *If a graph does not contain any pseudo-necks, it contains at least a neck.*

The general idea of the algorithms is a combination of the the cases of $k = 3$ and $k > 3$ odd. Here however we use a pseudo-neck (or a neck) to create the tower, and we identify a unique spanning tree in the rest of the graph. Two robots move to the pseudo-neck (or to the neck) while the third and fourth occupy the first two leaves of the spanning tree, without creating symmetries. The graph is then explored in a way similar to the one described in Section 4.3.

4.2.1 Graphs with a pseudo-neck

We start by describing the algorithm when the graph contains a pseudo-neck.

We explain how to make a tower on two nodes of the pseudo-neck and how to have a robot move on the first leaf. Note that a pseudo-neck is necessary asymmetric (since there is a port numbering and the port numbers of u break the symmetry). If there is more than one pseudo-neck in the graph, we assume they are ordered from weakest to strongest by an arbitrary predefined ordering and whenever we select a pseudo-neck (or an equivalence class of pseudo-necks) we do choose the weakest. Given a pseudo-neck (v, v', u), let $T(v, v', u)$ be a spanning tree obtained by a depth-first traversal starting in u. Let $f_1(T(v, v', u)), f_2(T(v, v', u)), \ldots, f_k(T(v, v', u))$ be an ordering of its leaves different from v, v'. We construct these trees in such a way that $f_1(T(v, v', u)) = f_1(T(v', v, u))$.

The algorithm (Gathering-with-pseudo-neck) is described by listing six possible cases. A robot checks the cases in order and follows the first that applies to the observed configuration.

Algorithm Gathering-with-pseudo-neck

- **Case 1.** *There is a pseudo-neck (v, v', u) with a robot in u, a robot in v and there is a robot in f_1.*
  If I am the robot in u: move to v creating a tower in v.

- **Case 2.** *There is a pseudo-neck (v, v', u) with a robot in v, a robot in v' and there is a robot in f_1.*
  If I am the robot in v': move to u.

- **Case 3.** *There is a pseudo-neck (v, v', u) with a robot in v, a robot in v' and there are no robots in $f_1(T(v, v', u)) = f_1(T(v', v, u))$.*
  Let a be the closest robot to $f_1(T(v, v', u))$ in $G \backslash \{v, v'\}$. If I am a: move toward $f_1(T(v, v', u))$.

- **Case 4.** *There is a pseudo-neck (v, v', u) with a robot in v, and robot a in u.*
  If I am a: move to v'.

- **Case 5.** *There is a pseudo-neck (v, v', u) with a robot in v, no robot in u nor in v'.*
  Among all the necks like that, consider the weakest such that $\min\{\mathsf{dist}(a, u) \mid \Psi(a) \neq v\}$ is minimal. Let a be the closest robot to u in G that is not in v. If I am a: move toward u.

- **Case 6.** *Any other situation.*
  Apply the gathering Algorithm by choosing a gathering point in the following class: nodes of degree 1 which have another node of degree 1 at distance 2.

Lemma 4.8 *No movement can create a symmetry during the tower formation on a neck with Algorithm* Gathering-with-pseudo-neck.

Proof: Case 1. The tower is created which means that we are in an asymmetric configuration
Case 2. If the movement creates a symmetry, it means that $f_1(T(v, v', u))$ is of degree 1 and that it has a neighbor x that has another neighbor y such that $(f_1(T(v, v', u), y, x)$ is a neck. Since the situation is symmetric, $f_1(T((f_1(T(v, v', u), y, x)) = u$ and then, we would be in Case 1 and not in Case 2.
Case 3. Suppose there is an automorphism τ after the movement of a. Wlog, assume $\tau(f_1(T(v, v', u))) = v$ and let $y = \tau(v')$. It means that a was on the unique neighbor $x\tau(u)$ of $\tau(v) = f_1(T(v, v', u))$ before the move and that $(\tau(v), y, x)$ is a neck. Since the situation is symmetric, $f_1(T(\tau(v), y, x)) = v$ and thus, $f_1(T(y, \tau(v), x)) = v$: we would be in Case 1.
Case 4. Since we are not in any of the previous case, there is no robot in $f_1(T(v, v', u))$ or in v'. Consider now the movement of a to v'. If it creates a symmetry, it means there exists a pseudo-neck (y, y', x) with a robot in y and a robot in y', i.e., we would be in a previous case.
Case 5. Due to our choice of the neck and since we are not in Case 1 nor Case 4, it cannot create a symmetry.

□

It is easy to see that this algorithm terminates and thus we have the following theorem.

Theorem 4.9 *Algorithm* Gathering-with-pseudo-neck *terminates correctly.*

4.2.2 Graphs with a neck

We now consider the case when a pseudo-neck does not exist and thus a neck exists.

Graphs with a neck: tower creation. We first introduce some notation. A *block* in a graph is an inclusion-maximal 2-vertex-connected component (possibly reduced to one edge). Two blocks of G are either disjoint or share a single vertex, that is an articulation point. Any graph G admits a block-decomposition in the form of a rooted tree T: each vertex of T is a block of G, pick any block B_1 as a root of T, label it, and make it adjacent in T to all blocks intersecting it, then label that blocks and make them adjacent to all nonlabeled blocks which intersect them, etc.

We describe how to have two robots move on a neck and create a tower. We identify four situations and for each we sketch the algorithm followed by the robots.

Algorithm Robots-on-neck

- **Case 1.** *G is 2-connected.*
  In this case any edge is a neck. Consider a class $[u_0]$ of vertices of G and execute the gathering algorithm until there is a robot a in one vertex $u \in [u_0]$ and a robot b in a vertex v incident to u. Then a and b are on a neck.

- **Case 2.** *Any leaf B in the block-decomposition of G is of size* 2.
  Note that, since there does not exist any pseudo-neck, it implies that there exists a vertex u_0 of degree 1 adjacent to a vertex v_0 of degree 2. In that case, execute the gathering algorithm until there is a robot a in one vertex $u \in [u_0]$ and a robot b in a vertex v incident to u. Then a and b are on a neck.

- **Case 3.** *There exists a leaf B_0 in the block-decomposition of G, whose articulation point is w_0 such that there exists a vertex $u_0 \in B$ at distance* 2 *from w_0 in G.*
  Note that any edge $(u_0, v) \in E(G)$ is a neck. In this case, execute the gathering algorithm until there is a robot a in one vertex $u \in [u_0]$ and a robot b in a vertex v incident to u. Then a and b are on a neck.

- **Case 4.** *In any leaf B of the block-decomposition, the articulation point is adjacent to all vertices of B.*
  In this case, consider a leaf B_0 of the block-decomposition of size at least 3. Let w_0 be its articulation point and let u_0 be a vertex of B_0 distinct from u_0. Execute the gathering algorithm until there is a robot a in a vertex $u \in [u_0]$ and a robot b in a vertex v incident to u. If $v \in [w_0]$, b can safely move to a neighbor $t \notin [w_0]$ of u.

Lemma 4.10 *In Algorithm* Robots-on-neck *two robots move on the two nodes of a neck without creating symmetries.*

Proof: Cases 1,2,3 are asy to see. Consider now Case 4. If $v \notin [w_0]$, then a and b are on a neck. If $v \in [w_0]$, let t be a neighbor of u distinct from v. Since $u \in [u_0]$, u belongs to a leaf B of the block-decomposition isomorphic to B_0 and thus, v and t are incident. Thus b can move from v to t without creating a symmetry. Indeed, since v is the only articulation point incident to u and t, since an articulation point is mapped to an articulation point by an automorphism, and since there is a port-numbering, v and t are not in the same class. Thus, if it creates a symmetry, it implies that there is another robot c in $u' \in [u_0]$ and a robot d on a naighbor $t' \in [t_0]$ of u', i.e., there was already two robots on a neck. □

Graphs with a neck: rest of exploration. Once two robots are on a neck, we now describe how to have another robot move to the first leaf. The subsequent exploration proceeds like in Algorithm Exploration of Section 4.3.

Given a vertex u, let $T(u)$ be a spanning tree of G obtained by doing a depth-first traversal (DFT) starting from u. If (u, v) is a neck, let $T(u, v)$ be a spanning tree obtained by a DFT starting from u and using (u, v) as a first edge, i.e., $T(u, v) \setminus \{u\}$ is a DFT of $G \setminus \{u\}$ starting in v. Given a spanning tree $T(u, v)$ of G, consider an ordering of its leafs $f_1, \ldots, f_k$ such that $\mathsf{dist} f_1$ is as far as possible from u (using the labels on the paths from u to f to break the symmetry) in $G \setminus \{u, v\}$. Given a neck (n_1, n_2) with some robots on n_1 and n_2, we now have two cases: symmetric and asymmetric neck.

Draft of Algorithm Explore-with-symmetric-neck. Assume that the neck is symmetric, i.e., there is an automorphism σ of G such that $\sigma(n_1) = n_2$ (note that there is at most one such σ due to port numbers). Let $T = T(n_1, n_2)$ and $T' = T(n_2, n_1)$. Let a and b be two robots that are not on the neck. Wlog, assume that $\mathsf{dist}_{G \setminus \{n_1, n_2\}}(\Psi(a), f_1) \leq \min \{\mathsf{dist}_{G \setminus \{n_1, n_2\}}(\Psi(a), f'_1),$

$\mathsf{dist}_{G\setminus\{n_1,n_2\}}(\Psi(b), f_1')$, $\mathsf{dist}_{G\setminus\{n_1,n_2\}}(\Psi(b), f_1)\}$ and that if there is an equality, the label of the shortest path between $\Psi(a)$ and f_1 is "weaker" than the label of the other shortest paths (since the situation is asymmetric, even if σ is an automorphism). In this case, let a move in $\Psi'(a)$ towards f_1 in $G \setminus \{n_1, n_2\}$ if it does not creates a symmetry. Otherwise, let b move to $\Psi'(b) = \Psi'(a)$ towards f_1 otherwise.

Lemma 4.11 *With Algorithm* Explore-with-symmetric-neck*, a robot reaches f_1 without creating any symmetry.*

Proof: Suppose that $\mathsf{dist}_{G\setminus\{n_1,n_2\}}(\Psi(a), f_1) < \mathsf{dist}_{G\setminus\{n_1,n_2\}}(\Psi(b), f_1')$. Then, after a moves, $\mathsf{dist}(\Psi'(a), \Psi(b)) \geq 2$. Thus, it cannot creates a symmetry such that $\Psi'(a)$ is mapped to $\Psi'(b)$ and n_1 to n_2 (because that would be σ, but then we should have $\mathsf{dist}_{G\setminus\{n_1,n_2\}}(\Psi'(a), f_1)\mathsf{dist}_{G\setminus\{n_1,n_2\}}(\Psi(b), f_1')$, which is impossible). Note that $\Psi'(a)$ cannot either be mapped by an automorphism to n_1 (resp. n_2), since it would imply that $\mathsf{dist}(\Psi'(a), \Psi(b))\mathsf{dist}(n_1, n_2) = 1$. Consequently, if $\mathsf{dist}_{G\setminus\{n_1,n_2\}}(\Psi(a), f_1) < \mathsf{dist}_{G\setminus\{n_1,n_2\}}(\Psi(b), f_1')$, a can always move towards f_1.

Suppose that $\mathsf{dist}_{G\setminus\{n_1,n_2\}}(\Psi(a), f_1)\mathsf{dist}_{G\setminus\{n_1,n_2\}}(\Psi(b), f_1')$ and that when a moves in $\Psi'(a)$, it creates a symmetry. It cannot creates a symmetry such that $\Psi'(a)$ is mapped to $\Psi(b)$ and n_1 to n_2 (because that would be σ, but then we should have $\mathsf{dist}_{G\setminus\{n_1,n_2\}}(\Psi'(a), f_1)\mathsf{dist}_{G\setminus\{n_1,n_2\}}(\Psi(b), f_1')$, which is impossible). Thus, there is automorphism τ such that $\tau(\Psi'(a)) = n_1$ (resp. $\tau(\Psi'(a)) = n_2$) and $\tau(\Psi(b)) = n_2$ (resp. $\tau(\Psi(b))n_1$). Since $\sigma(n_1) = n_2$, it implies that $\lambda_{n_1}(n_2) = \lambda_{n_2}(n_1) = \ell$ and thus, $\Psi'(a)$ and $\Psi(b)$ are neighbors and $\lambda_{\Psi'(a)}(\Psi(b)) = \lambda_{\Psi(b)}(\Psi'(a)) = \ell$.

Consequently, it implies that $\lambda_{\Psi'(a)}(\Psi(a)) \neq \ell$, and thus b can move in $\Psi'(a)$ towards f_1 without creating a symmetry. □

Draft of Algorithm Explore-with-asymmetric-neck. Assume now that the neck is asymmetric, i.e., there is no automorphism σ of G such that $\sigma(n_1) = n_2$. Using an arbitrary predefined order, we assume wlog that n_1 is "weaker" than n_2. Let $T = T(n_1, n_2)$ and let a and b be two robots that are not on the neck. Wlog, assume that $\mathsf{dist}_{G\setminus\{n_1,n_2\}}(\Psi(a), f_1) \leq \mathsf{dist}_{G\setminus\{n_1,n_2\}}(\Psi(b), f_1)$ (using labels to break symmetry). In this case, let a move in $\Psi'(a)$ towards f_1 in $G \setminus \{n_1, n_2\}$ if it does not creates a symmetry. Otherwise let b move to $\Psi'(b) = \Psi'(a)$ towards f_1 otherwise.

Lemma 4.12 *With Algorithm* Explore-with-asymmetric-neck*, a robot reaches f_1 without creating any symmetry.*

Then the overall exploration algorithm is the following:

Algorithm Gathering-with-neck.

Robots-on-neck
If neck is symmetric **then**
  Explore-with-symmetric-neck
else
  Explore-with-asymmetric-neck

Theorem 4.13 *Algorithm* Gathering-with-neck *terminates correctly.*

4.3 Exploring with $k > 3$ robots

When the set of robots R is such that $|R| = 2l + 1 > 3$, the algorithm (Exploration-many(R)) is considerably simpler and we draft it in the following.

We start by applying the gathering algorithm of Section 3 after choosing an equivalence class $[v]$ of vertices where v is not an articulation point as possible gathering points. The algorithm is executed until $k - 3$ robots have gathered in some vertex r. We call a robot "free" if it does not belong to a tower. We now define a unique spanning tree T (for example a Depth-First Spanning Tree) rooted in r such that r has only one neighbor in T. Let $f_1, \ldots, f_k$ be the ordered leaves of T met in some predefined traversal of T (for example depth-first). We let the robots move sequentially with the following idea: the closest robot to f_1 (among the three remaining free robots) moves to f_1. The closest robot (not on f_1 nor on r) now moves to r. Once it reaches r, we have a tower and two free robots one of which is in f_1. At this point the two free robots collaboratively explore the graph by moving on the spanning tree placing on consecutive leaves f_i and f_{i+1}, and having the robot on the smaller leaf f_i move to f_{i+2} until the last leaf is reached.

Finally we have:

Theorem 4.14 *Any execution of Algorithm* Exploration-many(R) *correctly terminates for any odd k.*

5 Special topologies: rings and trees

The tree and the ring are the only topologies studied in the literature in this weak model, with the additional restriction of not having labels on the edges [7, 9, 10, 14, 15]. In the following, we assume the standard assumption of local orientation and we sketch our results.

5.1 Gathering and exploration in rings

Consider a ring $\mathcal{R}$ of n nodes, $u_0, u_1, \ldots, u_{n-1}$ where u_i is connected[1] to both u_{i-1} and u_{i+1}. The indices are used for notation purposes; in fact, the nodes are anonymous (i.e., identical). The edges of the ring are labeled with a local orientation λ. Let $r_0, \ldots, r_{k-1}$ denote the k robots in clockwise order starting from an arbitrary one. Note that the ring is not necessarily oriented; we consider the robots in a clockwise order for ease of description only. Let $d_j(t)$ denote the distance between robot r_j and robot r_{j+1} (where the operations on indices of robots are modulo k).

We now introduce some more notation to describe the gathering algorithm for $k > 2$. Given a robot r_j, we denote by $\delta^{+j}(t)$ the string of distances given by $\delta^{+j}(t) =< d_j(t)\, d_{j+1}(t)\ \ldots d_{j+k-1}(t) >$, and by $\delta^{-j}(t)$ the sequence $\delta^{-j}(t) =< d_j(t)\ d_{j-1}(t)\ \ldots\ d_{j-(k-1)}(t) >$. Let $\Delta^+(t) = \{\delta^{+j}(t) : 0 \leq j < k-1\}$ and $\Delta^-(t) = \{\delta^{-j}(t) : 0 \leq j < k-1\}$. Let $\Lambda^{+j}(t)$ denote the sequence of labels from r_j in clockwise order $\Lambda^{+j}(t) = < \lambda_{u_j}(u_j, u_{j+1}),\ \lambda_{u_{j+1}}(u_j, u_{j+1}), \ldots, \lambda_{u_{j-k}}(u_{j-k}, u_{j-(k-1)}) >$, and $\Lambda^{+j}(t)$ in counter-clockwise order: $\Lambda^{-j}(t) =< \lambda_{u_j}(u_j, u_{j-1}), \lambda_{u_{j-1}}(u_{j-1}, u_{j-2}) \ldots >$. Moreover, let $\Lambda^j(t) = \min\{\Lambda^{+j}(t), \Lambda^{-j}(t)\}$. We will denote by $\delta_{min}(t)$ the lexicographically minimum sequence in $\Delta^+(t) \cup \Delta^-(t)$. Note that there can be at most two robots r_i and r_j such that $min\{\Lambda^{+j}(t), \Lambda^{-j}(t)\} = min\{\Lambda^{+i}(t), \Lambda^{-i}(t)\} = \delta_{min}(t)$

[1] Here and in the following, all operations on the indices are modulo n.

Given a robot r_i let NEXT(r_i) denote r_{i+1} if $\delta_{min}(t) = \delta^{+j}(t)$, r_{i-1} if $\delta_{min}(t) = \delta^{-j}(t)$. With an abuse of notation, given a tower t we will denote by NEXT(t) the robot following the tower in the direction of the minimum string departing from t.

The idea of the algorithm is the following. When a robot r observes the environment, it identifies the robots from which the lexicographically minimum string of distances starts. It is easy to see that there are at most two such robots. If two of them are identified, ties are broken by considering the one from which the lexicographically minimum string of labels starts. This robot s is unique. If r follows s in the minimus string of distances and if its movement towards s would preserve the asymmetry of $(\mathcal{R}, \lambda, \Psi)$ then s moves towards r. Otherwise r moves towards s.

GATHERING IN THE RING
If there is a tower t and I am NEXT(t) **then**
move towards t;
else
R = set of robots such that $\min_j\{\delta^{+j}(t), \delta^{-j}(t)\} = \delta_{min}(t)$;
If $|R| = 1$ **then** let $R = \{r\}$;
If $|R| = 2$ **then** let $R = \{r_i, r_j\}$;
let $m = \min_f\{\Lambda^f(t) : f \in \{i, j\}\}$;
If I am r_m**then**
move towards NEXT(r_m);

Lemma 5.1 *Let R be the set of robots $\{r_j\}$ such that $\min_j\{\delta^{+j}(t), \delta^{-j}(t)\} = \delta_{min}(t)$. We have that $|R| \leq 2$. Moreover, if $R = \{r_i, r_j\}$ and $m = \min_f\{\Lambda^f(t) : f \in \{i, j\}\}$, then r_m is unique.*

Lemma 5.2 *Algorithm* GATHERING IN THE RING *creates a tower in finite time.*

Proof: By Lemma 5.1, when a robot r observes at time t, it identifies a unique robot $s = r_j$.

Let t be a time when some robots can move after their observations. Let $S(t)$ be the set of robots that observes at time t and let $M(t)$ the set of the ones that can move after the observation. By definition of the algorithm we have that $|M(t)| = 1$: only s can move And the only possible movement for s is to move towards $r =$ NEXT(s). None of the other robots can move.

We want to show that after a movement, $(\mathcal{R}, \lambda, \Psi)$ is still asymmetric, the distance between r and s has decreased, and the unique robot identified by the algorithm in the next observation will be still be s.

Let $s = r_j$ and, wlg $r = r_{j+1}$. Before the movement we have that: $\delta_{min}(t) =< d_j, d_{j+1}, \ldots, d_{j+k} >$. Notice that, by definition of lexicographical minimum, $d_{j+k} \geq d_{j+1}$ otherwise $< d_j, d_{j+k}, d_{j+k-1} \ldots, d_{j+1} >$ would be smaller. After the movement of $s = r_j$ towards r_{j+1} we have a new string of distances from s: $< d_j - 1, d_{j+1}, d_{j+2}, \ldots, d_{j+k} + 1 >$. Clearly the distance between s and r has decreased. Moreover, since $d_{j+k} + 1 > d_{j+1}$, we have that $< d_j - 1, d_{j+1}, d_{j+2}, \ldots, d_{j+k} + 1 > <_{lex} < d_j - 1, d_{j+k} + 1 d_{j+k-1}, \ldots, d_{j+1} >$. Since $d_j - 1$ was not present as a distance at time t we can conclude that $< d_j - 1, d_{j+1}, d_{j+2}, \ldots, d_{j+k} + 1 >$ is the new lexicographically minimum string and is unique.

□

After the tower is created, by definition of the algorithm the robots join the tower one at a time, and we have:

Theorem 5.3 *Algorithm* GATHERING IN THE RING *terminates correctly when executed on an asymmetric* $(\mathcal{T}, \lambda, \Psi)$ *for* $k > 2$.

Consider now the exploration problem. For $k = 2$ exploration is impossible, we then consider first the case of three robots. Using an algorithm similar to the gathering algorithm the three robots are placed on 3 consecutive nodes. Two of them form a tower choosing as a location for the tower node x such that the label of the edge incident on x in the direction of the third robot is smaller than the one in the opposite direction. Such a tower indicates a direction for the ring that can be agreed with the third robot. Once the tower is formed, the third robot explores the ring and termination occurs when the single robot is next to the tower in the direction of the larger of the two labels. When there are more than three robots, the idea is to perform gathering until a tower containing $k - 2$ robots is formed. The presence of the tower provides an orientation to the ring (say clockwise) that can be agreed by all the robots. At this point the two remaining robots move to the two consecutive nodes adjacent to the tower in clockwise direction; only now the one closest to the tower joins it and the other explores the ring. Termination occurs when there is a tower and a single robot next to it in counter-clockwise direction (which indicates that exploration has terminated).

We can then conclude that:

Theorem 5.4 *Exploration in the ring can be performed in any asymmetric* $(\mathcal{R}, \lambda, \Psi)$ *for* $k > 2$.

5.2 Gathering and Exploration in trees

Let $\mathcal{T}$ denote a tree and consider an asymmetric $(\mathcal{T}, \lambda, \Psi)$. Consider the following simplified gathering algorithm. If the tree has a unique center, let it be the gathering point and the robots can move there independently (just making sure not to create other towers). If instead the tree has two neighboring centers, one of the two is selected exploiting the asymmetry of the robots placement. We apply Algorithm GATHERING where the initial set of robots R for the routine GATHERINGPOINT contains the two centers. It is easy to see that:

Theorem 5.5 *The* GATHERING *algorithm terminates correctly when executed on an asymmetric* $(\mathcal{T}, \lambda, \Psi)$ *for* $k > 2$.

Consider now the exploration problem. Notice that Theorem 4.1 indicates quite a restrictive class of trees where exploration might be performed by three robots. It is easy to see that such trees can indeed be explored. As for the case of $k > 3$, a simplified version of the general algorithms described in sections 4.1, 4.1, 4.3 can be applied for any $k > 2$ (because gathering can be done for any $k > 2$). We can then conclude that:

Theorem 5.6 *For* $k = 4$*, exploration can be performed in any asymmetric* $(\mathcal{T}, \lambda, \Psi)$ *satisfying the condition of Theorem 4.1.*

Theorem 5.7 *Exploration can be performed in any asymmetric* $(\mathcal{T}, \lambda, \Psi)$ *for* $k \geq 4$.

References

[1] N. Agmon, D. Peleg, Fault-tolerant gathering algorithms for autonomous mobile robots, *SIAM J. Comput.* 36, 56-82, 2006.

[2] M. Cieliebak, P. Flocchini, G. Prencipe, N. Santoro, Solving the robots gathering problem, *30th Int. Colloquium on Automata, Languages and Programming* (ICALP), LNCS 2719, 1181-1196, 2003.

[3] R. Cohen and D. Peleg, Convergence properties of the gravitational algorithm in asynchronous robot systems, *SIAM J. Computing*, 34:1516–1528, 2005.

[4] R. Cohen, D. Peleg, Robot convergence via center-of-gravity algorithms, *11th Int. Colloquium on Structural Information and Communication Complexity* (SIROCCO), LNCS 3104, 79-88, 2004.

[5] J. Czyzowicz, L. Gasieniec, A. Pelc, Gathering few fat mobile robots in the plane, *10th Int. Conference on Principles of Distributed Systems* (OPODIS), LNCS 4288, 744-753, 2006.

[6] X. Défago and A. Konagaya. Circle formation for oblivious anonymous mobile robots with no common sense of orientation. 2^{nd} *ACM Workshop on Principles of Mobile Computing*, 97–104, 2002.

[7] S Devismes, F Petit, S Tixeuil. Optimal probabilistic ring exploration by asynchronous oblivious robots, CoRR abs/0902.1834, 2009.

[8] A. Efrima, D. Peleg, Distributed Models and Algorithms for Mobile Robot Systems, *33rd Conf. on Current Trends in Theory and Practice of Computer Science* (SOFSEM), 70-87, 2007.

[9] P. Flocchini, D. Ilcinkas, A. Pelc, N. Santoro, Computing without communicating: ring exploration by asynchronous oblivious robots, *11th Int. Conf. on Principles of Distributed Systems* (OPODIS), 105-118, 2007.

[10] P. Flocchini, D. Ilcinkas, A. Pelc, N. Santoro. Remembering without memory: Tree exploration by asynchronous oblivious robots. *15th Int. Coll. on Structural Information and Comm. Complexity* (SIROCCO), 33-47, 2008.

[11] P. Flocchini, G. Prencipe, N. Santoro, P. Widmayer. Arbitrary pattern formation by asynchronous anonymous oblivious robots. *Theoretical Computer Science* 407 (1-3), 412-447, 2008.

[12] P. Flocchini, G. Prencipe, N. Santoro, P. Widmayer, Gathering of asynchronous robots with limited visibility. *Theoretical Computer Science* 337 (1-3) ,147-168, 2005.

[13] Y. Katayama, Y. Tomida, H. Imazu, N. Inuzuka, and K. Wada. Dynamic compass models and gathering algorithms for autonomous mobile robots. *14th Int. Colloquium on Structural Information and Communication Complexity* (SIROCCO), 274–288, 2007.

[14] R. Klasing, A. Kosowski, A. Navarra. Taking advantage of symmetries: gathering of asynchronous oblivious robots on a ring. *12th Int. Conf. on Principles of Distributed Systems* (OPODIS), 446-462, 2008.

[15] R. Klasing, E. Markou, A. Pelc, Gathering asynchronous oblivious mobile robots in a ring, *Theoretical Computer Science* 390 (1), 27-39, 2008.

[16] G. Prencipe, On the feasibility of gathering by autonomous mobile robots. *12th Int. Colloquium on Structural Information and Communication Complexity* (SIROCCO), LNCS 3499, 246-261, 2005.

[17] G. Prencipe, N. Santoro. Distributed algorithms for mobile robots. *5th IFIP Int. Conference on Theoretical Computer Science* (TCS), 47-62, 2006.

[18] S. Souissi, X. Défago and M. Yamashita. Gathering asynchronous mobile robots with inaccurate compasses. *10th Int. Conf. on Principles of Distributed Systems* (OPODIS), 4305, 333–349, 2006.

[19] I. Suzuki, M. Yamashita, Distributed anonymous mobile robots: formation of geometric patterns. *SIAM J. Comput.* 28 (1999) 1347-1363.