Network Exploration by Silent and Oblivious Robots*

Jérémie Chalopin[1], Paola Flocchini[2], Bernard Mans[3], and Nicola Santoro[4]

[1] Laboratoire d'Informatique Fondamentale de Marseille
CNRS & Aix-Marseille Université, Marseille, France
jeremie.chalopin@lif.univ-mrs.fr
[2] SITE, University of Ottawa, Ottawa, Canada
flocchin@site.uottawa.ca
[3] Macquarie University, Sydney, Australia
bernard.mans@mq.edu.au
[4] School of Computer Science, Carleton University, Ottawa, Canada
santoro@scs.carleton.ca

Abstract. In this paper we investigate the basic problem of *Exploration* of a graph by a group of identical mobile computational entities, called robots, operating autonomously and asynchronously. In particular we are concerned with what graphs can be explored, and how, if the robots do not remember the past and have no explicit means of communication. This model of robots is used when the spatial universe in which the robots operate is *continuous* (e.g., a curve, a polygonal region, a plane, etc.). The case when the spatial universe is *discrete* (i.e., a graph) has been also studied but only for the classes of acyclic graphs and of simple cycles. In this paper we consider networks of arbitrary topology modeled as connected graphs with local orientation (locally distinct edge labels). We concentrate on class $\mathcal{H}_k$ of asymmetric configurations with k robots. Our results indicate that the explorability of graphs in this class depends on the number k of robots participating in the exploration. In particular, exploration is impossible for $k < 3$ robots. When there are only $k = 3$ robots, only a subset of $\mathcal{H}_3$ can be explored; we provide a complete characterization of the networks that can be explored. When there are $k = 4$ robots, we prove that all networks in $\mathcal{H}_4$ can be explored. Finally, we prove that for any odd $k > 4$ all networks in $\mathcal{H}_k$ can be explored by presenting a general algorithm. The determination of which networks can be explored when $k > 4$ is even, is still open but can be reduced to the existence of a gathering algorithm for $\mathcal{H}_k$.

1 Introduction

Consider a team (or swarm) of identical mobile robots located in a spatial universe. Each robot operates autonomously by cyclically executing three operations: `Look` - it observes the position of the other robots; `Compute` - based on

* This work was partially supported by ANR Project SHAMAN, by COST Action 295 DYNAMO, by NSERC, and by Dr. Flocchini's University Research Chair.

D.M. Thilikos (Ed.): WG 2010, LNCS 6410, pp. 208–219, 2010.

this input, it computes a destination (a neighbouring node) or it decides not to move, according to a predefined algorithm (the same for all robots); `Move`- it then moves to its computed destination. The robots are *silent*: they have no direct means of communication; *anonymous*: externally identical with no distinct identifiers that can be used in the execution of the algorithm; *asynchronous*: the time between each operation in the `Look-Compute-Move` cycle as well as between successive cycles is finite but arbitrary; *oblivious*: the robots have no memory of past actions and computations, and the computation is based solely on what determined in the current cycle.

This model of robots is used when the spatial universe in which the robots operate is *continuous*, e.g., a curve, a polygonal region, a plane, etc. In this setting, the computational power of such robots has been investigated with respect to a variety of problems, such as *Pattern Formation*, *Gathering*, *Flocking*, etc. (e.g. see [1, 3–5, 9, 12, 13]).

Recently investigations have also considered the case when the spatial universe is *discrete*, e.g., when the robots operate in a network [6–8, 10, 11]. The research has focused on two fundamental problems (extensively studied in the past in a variety of other models): *Gathering* (or *Rendezvous*), which requires all robots to move 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 and all robots reach a quiescent state within finite time. The results have however been limited to two classes of graphs: *rings* [6, 7, 10, 11] and *trees* [8] and assuming that the edges incident on a vertex are indistinguishable for the robots. No results exist to date for the exploration of arbitrary graphs by oblivious robots in the `Look-Compute-Move` model. Note that the latter assumption (unlabeled edges) is atypical; indeed in standard models of networks (anonymous or not) in distributed computing, the links incident on a node x have distinct labels, called *port numbers*.

The computational study of such weak robots is a difficult task due to the simultaneous presence of asynchrony, obliviousness, and lack of explicit communication. Lack of explicit communication means that synchronization, interaction, and communication of information among the robots can be achieved solely by means of observing the position of the other robots. However, because of asynchrony a robot r may observe the position of the robots at some time t; based on that observation, compute the destination at time $t' > t$, and move at an even later time $t'' > t'$; thus it might be possible that at time t'' some robots are in different positions from those previously perceived by r at time t, because in the meantime they performed their `Move` operations (possibly several times). In other words, robots may compute destinations and move based on significantly outdated perceptions, which adds to the difficulty of exploration. Moreover, since robots are oblivious, the task of deciding the global status of the exploration process, in particular termination detection, has to be performed without memory and without communication.

In this paper we continue the investigation of the computational power of such weak robots in the discrete setting, and consider the *exploration* problem

in anonymous networks, that is edge-labeled graphs (G, λ) where $G = (V, E)$ is a connected simple graph and $\lambda = \{\lambda_v : v \in V\}$ is the set of local port-numbering functions λ_v. Let Ψ be the placement function describing the position of the robots in the network, and let (G, λ, Ψ) denote the network with the placement of the robots. In particular, we are interested in determining which networks with what initial placements of the robots exploration is possible; that is, which (G, λ, Ψ) can be explored and how. The study of the specific classes of graphs investigated in absence of edge-labels, of trees and rings, becomes much simpler with edge-labels, and a complete characterization can be found in [2].

When considering networks with at most two robots, it is easy to see that exploration cannot be solved if the networks has at least three vertices. When the number of robots is larger and (G, λ, Ψ) is symmetric (i.e., there exists a non-trivial automorphism of (G, λ, Ψ) that preserves the labels and the placement of the robots), exploration cannot be generally solved: solvability depends on the number of equivalence classes of agents and the task is impossible when the number of equivalence classes is less than 2.

In this paper, we consider only initial configurations (G, λ, Ψ) that are asymmetric. We denote by $\mathcal{H}_k$ the class of configurations (G, λ, Ψ) with k robots such that there is no non-trivial automorphism that preserves the labels and the placement of the robots. Our goal is to determine which networks in this class can be explored and how.

Our results indicate that the explorability of networks in $\mathcal{H}_k$ depends on the number k of robots participating in the exploration. In particular, with one robot (resp. two robots), only a network with one vertex (resp. two vertices) can be explored. When there are only $k = 3$ robots, we prove that not all networks in $\mathcal{H}_3$ can be explored. More precisely, we do provide a complete characterization of the networks that can be explored with $k = 3$ robots, and present an algorithm that performs the exploration of those networks with three robots (Section 4). When there are $k = 4$ robots, we prove that all networks in $\mathcal{H}_4$ can be explored; the proof is constructive, and the algorithm is (unfortunately) quite involved (Section 5). Finally, we prove that for any odd $k > 4$ all networks in $\mathcal{H}_k$ can be explored by presenting a general algorithm. The determination of which networks can be explored when $k > 4$ is even, is still open, but it can be reduced to the existence of a gathering algorithm for $\mathcal{H}_k$.

Due to space limitations most proofs are omitted in this extended abstract and will be presented in the journal version of our paper.

2 Model and Basics

Let $G = (V, E)$ be an undirected connected simple graph where V is the set of vertices and E the set of edges, with $|V| = n$. The vertices of G are unlabeled. We denote by $N(v)$ the set of neighbors of v, and by $d(v)$, the degree of v. For each node v, there is a bijective function $\lambda_v : N(v) \to \{1, 2, \ldots d(v)\}$ which assigns unique labels to the edges incident to v. Each edge $uv \in E$ has two distinct labels $\lambda_u(v)$ and $\lambda_v(u)$. Let $\lambda = \{\lambda_v : v \in V\}$ be the global labeling function and let (G, λ) denote the resulting edge-labeled graph. The label $\Lambda(\pi)$ of a path

$\pi = (u_0, u_1, \ldots, u_k)$ is obtained by extending λ from edges to paths as follows: $\Lambda(\pi) = ((\lambda_{u_0}(u_1), \lambda_{u_1}(u_0)), \ldots, (\lambda_{u_{k-1}}(u_k), \lambda_{u_k}(u_{k-1}))$.

The shortest path from u to v is the minimal element we get when sorting the paths from u to v first by length, and then lexicographically using their labels. We say that a node u is closer from a node w than a node v if either $\mathsf{dist}(u, w) < \mathsf{dist}(v, w)$, or $\mathsf{dist}(u, w) = \mathsf{dist}(v, w)$ and the label $\Lambda(\pi_{uw})$ of a shortest path from u to w is lexicographically smaller (or "weaker") than the label $\Lambda(\pi_{vw})$ of any shortest path from v to w.

Operating in (G, λ) is a set $\mathcal{R}$ of k identical 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 [6–8, 10, 11], and hence any robot performing a `Look` operation sees all other robots at nodes and not on edges. We say that there is a *tower* on a node if the node is occupied by more than one robot. We call a robot *free* if it does not belong to a tower. Initially all robots are free; that is there is at most one robot in each node. During the `Look` operation, the robots can perceive if there is one or more robots in a given location; this ability, called *multiplicity detection* is a standard assumption in the continuous model.

Let $\Psi : \mathcal{R} \to V$ be the placement function returning the initial position of a given robot. Let (G, λ, Ψ) denote the edge-labeled graph with the initial placement of the robots. The vertices of (G, λ) (resp., (G, λ, Ψ)) 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 (resp., and placement) such that $\sigma(v) = v_0$. We say that (G, λ) (resp., (G, λ, Ψ)) is *symmetric* if there exists a non-trivial automorphism of (G, λ) (resp., (G, λ, Ψ)) that preserves the edge-labeling λ (resp., and the placement), *asymmetric* otherwise. Note that, in (G, λ) and (G, λ, Ψ) the equivalence classes can be ordered using the fact that each identifies a different "view" of the graph. We say that a robot a is closer from a node w than a robot b if $\Psi(a)$ is closer from w than $\Psi(b)$.

Given an initial placement Ψ of k robots in (G, λ), we say that a protocol $\mathcal{A}$ solves the exploration problem in (G, λ, Ψ) if in all possible executions of $\mathcal{A}$ by the k robots, every node of the graph is visited by at least one robot and all robots enter a quiescent state within finite time. We say that exploration of (G, λ, Ψ) with k robots is *impossible* if no protocol solves the exploration problem in (G, λ, Ψ). A network (G, λ) is explorable with k robots if there exists protocol $\mathcal{A}$ that solves the exploration problem in (G, λ, Ψ) for any placement of Ψ of k robots in (G, λ).

Lemma 1. *For any network (G, λ), it is impossible to solve exploration with one (resp. two) agents if $|V| > 1$ (resp. $|V| > 2$).*

In the rest of the paper we focus on *asymmetric configurations*, i.e. the set $\mathcal{H}_k$ of all (G, λ, Ψ) with k robots where (G, λ, Ψ) is asymmetric; and we assume multiplicity detection. Since exploration is impossible for networks in $\mathcal{H}_1$ and $\mathcal{H}_2$ we consider $k \geq 3$.

3 About Gathering and Asymmetry

3.1 Gathering Algorithms

In [2], we have studied the gathering problem: we want to gather all the robots at a single node. In the exploration algorithms we present below, we use the gathering algorithm of [2] as a subroutine. We describe it here briefly.

Consider a node $v \in V$; let $PATHS(v) = \{\Lambda(\pi) \mid \pi$ is the shortest path from v to $\Psi(a) \mid a \in \mathcal{R}\}$. 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. Let $SD(v) = \sum_{a \in \mathcal{A}} \mathsf{dist}(v, \Psi(a))$ denote the sum of the distances from a node v to all the robots and let $\mathsf{code}(v) = (SD(v), SORT(PATHS(v)))$.

Consider an equivalence class $[u]$ of (G, λ), a *Minimal Weber node* of $[u]$ is a node $u' \in [u]$ such that $\mathsf{code}(u') = \min\{code(u'') \mid u'' \in [u]\}$. Note that if (G, λ, Ψ) is asymmetric, then each class $[u]$ of (G, λ) has a unique Minimal Weber node, denoted by $MWN[u]$.

Given an equivalence class $[u]$ of (G, λ) (this class can be seen as a parameter of the algorithm), and starting from any asymmetric configuration (G, λ, Ψ) with k robots, the algorithm from [2] has the following properties.

- at each moment, at most one robot can move,
- if $k \geq 3$ is odd, then the only robot that is allowed to move is the closest from $u' = MWN[u]$ that is not on u'. After this move, we still have $MWN[u] = u'$, and $\mathsf{code}(u')$ has strictly decreased.
- if $k = 4$, the robot that is allowed to move is either the closest one from $u' = MWN[u]$ that is not on u', or the robot that is on u'. After the move, $MWN[u]$ can be modified, but $\mathsf{code}(MWN[u])$ has strictly decreased.

Thus, we have the following theorem.

Theorem 1 ([2]). *From any asymmetric configuration (G, λ, Ψ) with k agents, and for any equivalence class $[u]$ of (G, λ), there exists a gathering algorithm where the gathering node belongs to $[u]$ when $k \geq 3$ is odd, or when $k = 4$.*

3.2 Asymmetry

For Gathering, we know that if the initial configuration (G, λ, Ψ) is asymmetric, then it is impossible to solve gathering [2]. When considering exploration, we

don't have such strong results. However, we want the robots to be able to detect that they have explored the network. Since they are oblivious, they should be able to detect from the snapshot they compute if the configuration can be an initial configuration or not. To do so, the robots will create towers of two (or more) robots that cannot be part of the initial configuration. If in the initial configuration, all robots are in the same equivalence class, we know that no tower can be created [2] and thus, it implies that exploration is impossible. It means that when the degree of symmetry is too high, exploration is impossible. To avoid this problem, in this paper, we only consider initial configurations that are asymmetric. Note that, even if we do not ask the robots to stop once they have explored all the nodes (i.e. we consider perpetual exploration), there are still some symmetric networks that cannot be explored.

4 Exploration with $k = 3$ Robots

We prove that explorability of $(G, \lambda, \Psi) \in \mathcal{H}_3$ requires the presence of 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 several conditions. This class of graphs is denoted by $\mathcal{E}_3$.

Definition 1. *A graph $G \in \mathcal{E}_3$ if at least one of the two following conditions hold:*

(Case 1) *There exists an elementary path $P = (x_1, \ldots, x_m)$ 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 $P_1 = (x_1, \ldots, x_m)$ and $P_2 = (y_1, \ldots, y_\ell)$ and two neighbors u_0, v_0 both different from x_1 such that:*

(B1) $x_m = y_1$,

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

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

(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'.

Examples of graphs in $\mathcal{E}_3$ can be seen in Figure 1.

Remark 1. A graph satisfies the conditions of (Case 1) in Definition 1 if and only if there exists $u_0, v_0 \in V(G)$ such that either $G \setminus \{u_0\}$ or $G \setminus \{u_0, v_0\}$ has a Hamiltonian path.

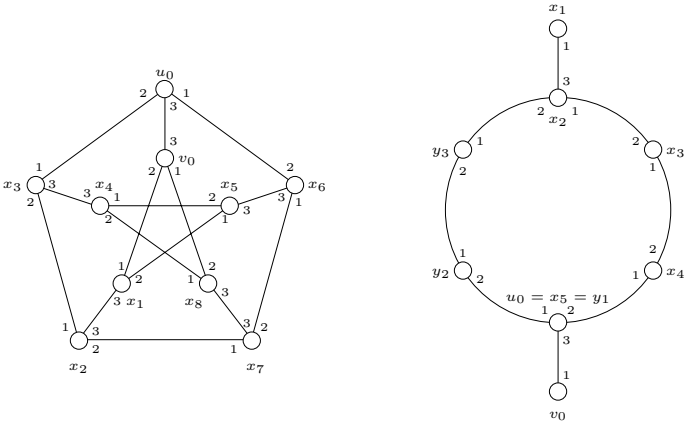

Fig. 1. On the left, a graph satisfying conditions of (Case 1) of Definition 1; on the right, a graph satisfying conditions of (Case 2) of Definition 1

Lemma 2. *If* $(G, \lambda, \Psi) \in \mathcal{H}_3 \setminus \mathcal{E}_3$ *exploration of* (G, λ, Ψ) *is impossible.*

We now briefly describe the exploration algorithm for $(G, \lambda, \Psi) \in \mathcal{E}_3$.

The algorithm is different depending on the characteristics of the graph. Suppose that the graph satisfies the conditions of (Case 1) in Definition 1 for some vertices u_0, v_0 and some elementary path $P = (x_1, \ldots, x_m)$. The idea is to first have the three robots place themselves on u_0, v_0, and x_1 (this procedure is not trivial; the details will appear in the complete version of this paper). Then the two robots that are on u_0, v_0 (up to isomorphism) create a small tower on u_0, and the third robot performs the exploration following 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 $P_1 = (x_1, \ldots, x_m)$ and $P_2 = (y_1, \ldots, y_\ell)$. Again one can show that we can have the three robots placed on u_0, v_0 and x_1. Then the two robots that are on u_0, v_0 (up to isomorphism) create a small tower on 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. From Lemma 2 and the Algorithm above we have:

Theorem 2. $(G, \lambda, \Psi) \in \mathcal{H}_3$ *can be explored if and only if* $G \in \mathcal{E}_3$.

5 Exploration with $k = 4$ Robots

We now turn to the case of $k = 4$ robots and we show that any $(G, \lambda, \Psi) \in \mathcal{H}_4$ can be explored. 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:

Lemma 3. *If a graph does not contain any pseudo-neck, it contains at least a neck.*

The general idea of the algorithms is the following. Two robots uniquely identify a pseudo-neck (or a neck if the pseudo-neck does not exist) by moving to the leaves of a pseudo-neck (or to the vertices of the neck). Once the pseudo-neck (or the neck) is occupied, the robots identify a unique spanning tree rooted in the pseudo-neck (or in the neck) and an ordering $f_1, \ldots f_m$ of its leaves. One of the robot that is not on the pseudo-neck (or the neck) moves to the first leaf f_1 of this spanning tree. Then, the two robots on the pseudo-neck (or the neck) create a tower. The two free robots occupy the first two leaves of the spanning tree, without creating symmetries and proceed with the exploration by 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 (a technique similar to the one used in [8] for asymmetric trees).

The algorithm will be discussed in details and analyzed in the rest of this section. Note that the algorithm will employ in some cases a preliminary step of "partial gathering". For such a step, we will adapt the gathering algorithm for asymmetric (G, λ, Ψ) described in [2].

5.1 Graphs with a Pseudo-neck

We now show how to create a tower on one of the nodes of the pseudo-neck and how to have a robot move on the first leaf of a uniquely identified spanning tree. If there is more than one pseudo-neck in the graph, ties are broken using the edge labels and we assume they are ordered from "weakest" to "strongest".

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 $f_1(T(v, v', u)), f_2(T(v, v', u)), \ldots, f_m(T(v, v', u))$ be an ordering of its leaves different from v, v'. We construct $T(v, v', u)$ in such a way that $f_1(T(v, v', u)) = f_1(T(v', v, u))$ (it is easy to see that this can always be done).

Algorithm Form-Tower-with-pseudo-neck is described by listing six possible cases. A robot checks the cases in this order and follows the first that applies to the observed configuration.

Algorithm. Form-Tower-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 $G \setminus \{v, v'\}$. 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 of [2] until the tower is formed, by considering as candidate gathering points nodes of degree 1 which have another node of degree 1 at distance 2.

Lemma 4. *Algorithm* Form-Tower-with-pseudo-neck *terminates and no movement can create a symmetry during the tower formation on a pseudo-neck with Algorithm* Form-Tower-with-pseudo-neck.

Algorithm Explore-with-pseudo-neck proceeds by letting the other two robots move sequentially. They move to $f_1(T(v, v', u))$ and $f_2(T(v, v', u))$; they then move on the spanning tree (which is invariant to their movement) 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. We can conclude that:

Lemma 5. *Let G have a pseudo-neck. Then any $(G, \lambda, \Psi) \in \mathcal{H}_4$ can be explored.*

5.2 Graphs with a Neck

We consider the case when a pseudo-neck does not exist; thus a neck exists. In this case the algorithm is more complex and we describe the process of tower creation and the one of exploration separately.

Placement on the Neck. 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 unlabeled blocks which intersect them, etc.

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

Algorithm. OCCUPY-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 of [2] 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. *There exists a 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 of [2] 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 $u_0 \in B_0$ 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 of [2] 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 of [2] 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 6. *In Algorithm* OCCUPY-NECK *two robots move on the two nodes of a neck without creating symmetries.*

Exploration. 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 $f_1, \ldots, f_m$ such that f_1 is as far as possible from u (using the labels on the paths from u to f to break the symmetry) in $G \setminus \{u, v\}$.

We say that a neck (n_1, n_2) is symmetric if there is an automorphism σ of G such that $\sigma(n_1) = n_2$, asymmetric otherwise. Exploration proceeds differently whether the neck is symmetric or not. (Note that a pseudo-neck is by definition never symmetric because of the edge labels). In case of symmetry, once two robots are on a neck, we have to carefully proceed to have a robot move to the first leaf. We now describe the two cases.

Draft of Algorithm EXPLORE-WITH-SYMMETRIC-NECK. Assume that the neck is symmetric. First of all note that, due to the port numbers, there is at most one automorphism σ of G such that $\sigma(n_1) = n_2$. 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. Without loss of generality, 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 towards f_1 (on a node denoted $\Psi'(a)$) in $G \setminus \{n_1, n_2\}$ if it does not creates a symmetry. Otherwise, let b move to $\Psi'(a)$ (if a symmetry appears when a move to $\Psi'(a)$, then one can show that $\Psi(b)$ is adjacent to $\Psi'(a)$). Once the neck and the first leaf are occupied, the exploration proceeds as in Section 5.1.

Lemma 7. *With Algorithm* Explore-with-symmetric-neck*, a robot reaches f_1 without creating any 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 without loss of generality 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. Again without loss of generality, 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 the symmetry). In this case, let a move towards f_1 (on a node denoted $\Psi'(a)$) in $G \setminus \{n_1, n_2\}$ if it does not creates a symmetry. Otherwise, let b move to $\Psi'(a)$ (if a symmetry appears when a move to $\Psi'(a)$, then one can show that $\Psi(b)$ is adjacent to $\Psi'(a)$). Once the neck and the first leaf are occupied, the exploration proceeds as in Section 5.1.

Lemma 8. *During the execution of Algorithm* Explore-with-asymmetric-neck*, a robot reaches f_1 without creating any symmetry.*

From Lemmas 6, 7 and 8, we have that any $(G, \lambda, \Psi) \in \mathcal{H}_4$ can be explored if G has a neck. Consequently, with Lemma 5, we have the following theorem.

Theorem 3. *Any $(G, \lambda, \Psi) \in \mathcal{H}_4$ can be explored.*

6 Exploration with $k > 4$ Odd

Let $k > 4$ be odd. In this case, the exploration algorithm is rather simple, and is drafted below.

The robots start by applying the gathering algorithm of [2] where the gathering point is chosen within an equivalence class $[v]$ of vertices where v is not an articulation point (it is easy to see that such a class can always be selected). The gathering algorithm is executed until $k-3$ robots have gathered in some vertex r creating a tower. At this point, the robots can select a spanning tree T - for example a Depth-First Spanning Tree - rooted in r and they can agree on one where r has a single neighbor in T. Let $f_1, \ldots, f_m$ be the ordered leaves of T encountered 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 the robot reaches r, we have a tower and two free robots one of which

is in f_1. At this point the two free robots recognize the situation that signals the beginning of the exploration and collaboratively explore the graph moving from leaf to leaf as in Section 5. Based on the asymmetry of the initial placement, on the uniqueness of the chosen spanning tree and on its preservation, and on the sequentiality of the moves, is not difficult to show that the algorithm terminates correctly and we have:

Theorem 4. *Let $k > 4$ be odd; then any $(G, \lambda, \Psi) \in \mathcal{H}_k$ can be explored.*

Note that if we are given an algorithm that solves gathering for even $k > 4$ robots in asymmetric configurations, then the previous algorithm can be used to explore all $(G, \lambda, \Psi) \in \mathcal{H}_k$.

References

1. Asahiro, Y., Fujita, S., Suzuki, I., Yamashita, M.: A self-stabilizing marching algorithm for a group of oblivious robots. In: Baker, T.P., Bui, A., Tixeuil, S. (eds.) OPODIS 2008. LNCS, vol. 5401, pp. 125–144. Springer, Heidelberg (2008)
2. Chalopin, J., Flocchini, P., Mans, B., Santoro, N.: Gathering and rendezvous by oblivious robots in arbitrary graphs, rings, and trees, Technical Report, University of Ottawa (2009)
3. Cieliebak, M., Flocchini, P., Prencipe, G., Santoro, N.: Solving the robots gathering problem. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 1181–1196. Springer, Heidelberg (2003)
4. Cohen, R., Peleg, D.: Convergence properties of the gravitational algorithm in asynchronous robot systems. SIAM J. Computing 34, 1516–1528 (2005)
5. Défago, X., Souissi, S.: Non-uniform circle formation algorithm for oblivious mobile robots with convergence toward uniformity. Theoretical Computer Science 396(1-3), 97–112 (2008)
6. Devismes, S., Petit, F., Tixeuil, S.: Optimal probabilistic ring exploration by semi-synchronous oblivious robots. In: Kutten, S., Žerovnik, J. (eds.) SIROCCO 2009. LNCS, vol. 5869, pp. 203–217. Springer, Heidelberg (2010)
7. Flocchini, P., Ilcinkas, D., Pelc, A., Santoro, N.: Computing without communicating: ring exploration by asynchronous oblivious robots. In: Tovar, E., Tsigas, P., Fouchal, H. (eds.) OPODIS 2007. LNCS, vol. 4878, pp. 105–118. Springer, Heidelberg (2007)
8. Flocchini, P., Ilcinkas, D., Pelc, A., Santoro, N.: Remembering without memory: Tree exploration by asynchronous oblivious robots. Theoretical Computer Science (2010) (to appear)
9. Flocchini, P., Prencipe, G., Santoro, N., Widmayer, P.: Arbitrary pattern formation by asynchronous anonymous oblivious robots. Theoretical Computer Science 407(1-3), 412–447 (2008)
10. Klasing, R., Kosowski, A., Navarra, A.: Taking advantage of symmetries: gathering of asynchronous oblivious robots on a ring. In: Baker, T.P., Bui, A., Tixeuil, S. (eds.) OPODIS 2008. LNCS, vol. 5401, pp. 446–462. Springer, Heidelberg (2008)
11. Klasing, R., Markou, E., Pelc, A.: Gathering asynchronous oblivious mobile robots in a ring. Theoretical Computer Science 390(1), 27–39 (2008)
12. Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots: formation of geometric patterns. SIAM J. Comput. 28, 1347–1363 (1999)
13. Yamashita, M., Suzuki, I.: Characterizing geometric patterns formable by oblivious anonymous mobile robots. Theoretical Computer Science (2010) (to appear)