Anonymous Graph Exploration with Binoculars

Jérémie Chalopin, Emmanuel Godard, Antoine Naudin

Anonymous Graph Exploration with Binoculars*

Jérémie Chalopin, Emmanuel Godard and Antoine Naudin

LIF, Université Aix-Marseille and CNRS, FRANCE

Abstract. We investigate the exploration of networks by a mobile agent. It is long known that, without global information about the graph, it is not possible to make the agent halts after the exploration except if the graph is a tree. We therefore endow the agent with *binoculars*, a sensing device that can show the local structure of the environment at a constant distance of the agent current location.
We show that, with binoculars, it is possible to explore and halt in a large class of non-tree networks. We give a complete characterization of the class of networks that can be explored using binoculars using standard notions of discrete topology. This class is much larger than the class of trees: it contains in particular chordal graphs, plane triangulations and triangulations of the projective plane. Our characterization is constructive, we present an Exploration algorithm that is universal; this algorithm explores any network explorable with binoculars, and never halts in non-explorable networks.

1 Introduction

Mobile agents are computational units that can progress autonomously from place to place within an environment, interacting with the environment at each node that it is located on. Such software robots (sometimes called bots, or agents) are already prevalent in the Internet, and are used for performing a variety of tasks such as collecting information or negotiating a business deal. More generally, when the data is physically dispersed, it can be sometimes beneficial to move the computation to the data, instead of moving all the data to the entity performing the computation. The paradigm of mobile agent computing / distributed robotics is based on this idea. As underlined in [8], the use of mobile agents has been advocated for numerous reasons such as robustness against network disruptions, improving the latency and reducing network load, providing more autonomy and reducing the design complexity, and so on (see e.g. [17]).

For many distributed problems with mobile agents, exploring, that is visiting every location of the whole environment, is an important prerequisite. In its thorough exposition about Exploration by mobile agents [8], Das presents numerous variations of the problem. In particular, it can be noted that, given some global information about the environment (like its size or a bound on the diameter), it is always possible to explore, even in environments where there is no local information that enables to know, arriving on a node, whether it has

* This work was partially supported by ANR project MACARON (ANR-13-JS02-0002)

already been visited (e.g. anonymous networks). If no global information is given to the agent, then the only way to perform a network traversal is to use a *unlimited* traversal (e.g. with a classical BFS or Universal Exploration Sequences [1,15,19] with increasing parameters). This infinite process is sometimes called *Perpetual Exploration* when the agent visits infinitely many times every node. Perpetual Exploration has application mainly to security and safety when the mobile agents are a way to regularly check that the environment is safe. But it is important to note that in the case where no global information is available, it is impossible to always detect when the Exploration has been completed. This is problematic when one would like to use the Exploration algorithm composed with another distributed algorithm.

In this note, we focus on Exploration with termination. It is known that in general anonymous networks, the only topology that enables to stop after the exploration is the tree-topology. From standard covering and lifting techniques, it is possible to see that exploring with termination a (small) cycle would lead to halt before a complete exploration in huge cycles. Would it be possible to explore, with full stop, non-tree topologies without global information? We show here that it is possible to explore a larger set of topologies while only providing the agent with some local information.

The information that is provided can be informally described as giving *binoculars* to the agent. This constant range sensor enables the agent to see the relationship between its neighbours. Using binoculars is a quite natural enhancement for mobile robots. In some sense, we are trading some a priori global information (that might be difficult to maintain efficiently) for some local information that the agent can *autonomously* and *dynamically* acquire. We give here a complete characterization of which networks can be explored with binoculars.

2 Exploration with Binoculars

2.1 The Model

Mobile Agents. We use a standard model of mobile agents.A mobile agent is a computational unit evolving in an undirected simple graph $G = (V, E)$ from vertex to vertex along the edges. A vertex can have some labels attached to it. There is no global guarantee on the labels, in particular vertices have no identity (anonymous/homonymous setting), i.e., local labels are not guaranteed to be unique. The vertices are endowed with a port numbering function available to the agent in order to let it navigate within the graph. Let v be a vertex, we denote by $\delta_v : V \to \mathbb{N}$, the injective port numbering function giving a locally unique identifier to the different adjacent nodes of v. We denote by $\delta_v(w)$ the port number of v leading to the vertex w, i.e., corresponding to the edge vw at v. We denote by (G, δ) the graph G endowed with a port numbering $\delta = \{\delta_v\}_{v \in V(G)}$.

When exploring a network, we would like to achieve it for any port numbering. So we consider the set of every graph endowed with a port numbering function, called $\mathcal{G}^\delta$. By abuse of notation, since the port numbering is usually fixed, we denote by G a graph $(G, \delta) \in \mathcal{G}^\delta$.

The behaviour of an agent is cyclic: it obtains local information (local label and port numbers), computes some values, and moves to its next location according to its previous computation. We also assume that the agent can backtrack, that is the agent knows via which port number it accessed its current location. We do not assume that the starting point of the agent (that is called the *homebase*) is marked. All nodes are a priori indistinguishable except from the degree and the label. We assume that the mobile agent is a Turing machine (with unbounded local memory). Moreover we assume that an agent accesses its memory and computes instructions instantaneously. An execution ρ of an algorithm $\mathcal{A}$ for a mobile agent is composed by a (possibly infinite) sequence of edge traversals (or *moves*) by the agent. The length $|\rho|$ of an execution ρ is the total number of moves. The complexity measure we are interested in is the number of moves performed by the agent during the execution of the algorithm.

Binoculars. Our agent can use "binoculars" of range 1, that is, it can "see" the graph (with the labels and the port numbers) that is induced by its current location and the adjacent nodes. In order to reuse standard techniques and algorithms, we will actually assume that the nodes of the graph we are exploring are labelled by these induced balls. It is straightforward to see that in a graph with such a *binoculars labelling* of the nodes, an agent with binoculars has the same computational power as an agent without binoculars (the "binoculars" primitive gives only access to more information, it does not enable more moves).

2.2 The Exploration Problem

We consider the Exploration Problem with Binoculars for a mobile agent. An algorithm $\mathcal{A}$ is an Exploration algorithm if for any graph $G = (V, E)$ with binoculars labelling, for any port numbering δ_G, starting from any arbitrary vertex $v_0 \in V$,

- either the agent visits every vertex at least once and terminates;
- either the agent never halts. [1]

In other words, if the agent halts, then we know that every vertex has been visited. The intuition in this definition is to model the absence of global knowledge while maintaining safety of composition. Since we have no access to global information, we might not be able to visit every node on some networks, but, in this case, we do not allow the algorithm to appear as correct by terminating. This allows to safely compose an Exploration algorithm with another algorithm without additional global information.

We say that a graph G is *explorable* if there exists an Exploration algorithm that halts on G starting from any point. An algorithm $\mathcal{A}$ explores $\mathcal{F}$ if it is an Exploration algorithm such that for all $G \in \mathcal{F}$, $\mathcal{A}$ explores and halts. (Note that

[1] a seemingly stronger definition could require that the agent performs perpetual exploration in this case. It is easy to see that this is actually equivalent for computability considerations since it is always possible to compose in parallel (see below) a perpetual BFS to any never halting algorithm.

since $\mathcal{A}$ is an Exploration algorithm, for any $G \notin \mathcal{F}$, $\mathcal{A}$ either never halts, or $\mathcal{A}$ explores G.)

In the context of distributed computability, a very natural question is to characterize the maximal sets of explorable networks. It is not immediate that there is a maximum set of explorable networks. Indeed, it could be possible that two graphs are explorable, but not explorable with the same algorithm. However, we note that explorability is monotone. That is if $\mathcal{F}_1$ and $\mathcal{F}_2$ are both explorable then $\mathcal{F}_1 \cup \mathcal{F}_2$ is also explorable. Consider $\mathcal{A}_1$ that explores $\mathcal{F}_1$ and $\mathcal{A}_2$ that explores $\mathcal{F}_2$ then the parallel composition of both algorithms (the agent performs one step of $\mathcal{A}_1$ then backtracks to perform one step of $\mathcal{A}_2$ then backtracks, etc ...; and when one of $\mathcal{A}_1$ or $\mathcal{A}_2$ terminates, the composed algorithm terminates) explores $\mathcal{F}_1 \cup \mathcal{F}_2$ since these two algorithms guarantee to have always explored the full graph when they terminate on any network. So there is actually a maximum set of explorable graphs.

2.3 Our Results

We give here a complete characterization of which networks can be explored with binoculars. We first give a necessary condition for a graph to be explorable with binoculars using the standard lifting technique. Using the same technique, we give a lower bound on the move complexity to explore a given explorable graph. Then we show that the Exploration problem admits a universal algorithm, that is, there exists an algorithm that halts after visiting all vertices on all explorable graphs. This algorithm, together with the necessary condition, proves that the explorable graphs are exactly the graphs whose clique complexes admit a finite universal cover (these are standard notions of discrete topology, see Section 3). This class is larger than the class of tree networks that are explorable without binoculars. It contains graphs whose clique complex is simply connected (like chordal graphs or planar triangulations), but also triangulations of the projective plane. Finally, we show that the move complexity of any universal exploration algorithm cannot be upper bounded by any computable function of the size of the network.

Related works. To the best of our knowledge, using binoculars has never been considered for mobile agent on graphs. In the classical "Look-Compute-Move" model [16, Chap. 5.6], vision is usually global and only coordination problems, like Rendezvous or Gathering, have been considered, even when the vision is limited to the immediate neighbourhood (e.g. in [10]). When the agent can only see the label and the degree of its current location, it is well-known that any Exploration algorithm can only halts on trees and a standard DFS algorithm enables to explore any tree in $O(n)$ moves. Gasieniec et al. [2] show that an agent can explore any tree and stop on its starting position using only $O(\log n)$ bit of memory matching a lower bound proved in [9]. For general anonymous graphs, Exploration with halt has mostly been investigated assuming some global bounds are known, in the goal of optimizing the move complexity. It can be done in $O(\Delta^n)$ moves using a DFS traversal while knowing the size n when the

maximum degree is Δ. This can be reduced to $O(n^3\Delta^2 \log n)$ using Universal Exploration Sequences [1,15] that are sequences of port numbers that an agent can follow and be assured to visit any vertex of any graph of size at most n and maximum degree at most Δ. Reingold [19] showed that universal exploration sequences can be constructed in logarithmic space.

Trading global knowledge for structural local information by designing specific port numberings, or specific node labels that enable easy or fast exploration of anonymous graphs have been proposed in [7,11,14]. Note that using binoculars is a local information that can be locally maintained contrary to the schemes proposed by these papers where the local labels are dependent of the full graph structure. See also [8] for a detailed discussion about Exploration using other mobile agent models (with pebbles for examples).

3 Definitions and Notations

3.1 Graphs

We always assume simple and connected graphs. Let G be a graph, we denote $V(G)$ (resp. $E(G)$) the set of vertices (resp. edges). If two vertices $u, v \in V(G)$ are adjacent in G, the edge between u and v is denoted by uv.

A *path* p of *length* k in a graph G is a sequence of vertices $(v_0, \ldots, v_k)$ such that $v_iv_{i+1} \in E(G)$ for every $0 \leq i < k$. A path is *simple* if for any $i \neq j$, $v_i \neq v_j$. A *cycle* c of length k is a path $(v_0, \ldots, v_k)$ such that $v_0 = v_k$. A cycle $(v_0, \ldots, v_k)$ is simple if it is the empty path (i.e., $k = 0$) or if the path $(v_0, \ldots, v_{k-1})$ is simple. A *loop* c of length k is a sequence of vertices $(v_0, \ldots, v_k)$ such that $v_0 = v_k$ and $v_i = v_{i+1}$ or $v_iv_{i+1} \in E(G)$, for every $0 \leq i < k$; the length of a loop is denoted by $|c|$. On a graph endowed with a port numbering, a path $p = (v_0, \ldots, v_k)$ is labelled by $\lambda(p) = (\delta_{v_0}(v_1), \delta_{v_1}(v_2), \ldots, \delta_{v_{k-1}}(v_k))$.

The distance between two vertices v and v' in a graph G is denoted by $d_G(v, v')$. It is the length of a shortest path between v and v' in G. Let $N_G(v, k)$ be the set of vertices at distance at most k from v in G. We denote by $N_G(v)$, the vertices at distance at most 1 from v. We define $B_G(v, k)$ to be the subgraph of G induced by the set of vertices $N_G(v, k)$.

Binoculars labelling. In the following, we always assume that every vertex v of G has a label $\nu(v)$ corresponding to the *binoculars labelling* of v. This binoculars label $\nu(v)$ is a graph isomorphic to $B_G(v, 1)$ with its port numbering.

Coverings. We now present the formal definition of graph homomorphisms that capture the relation between graphs that locally look the same in our model. A map $\varphi : V(G) \to V(H)$ from a graph G to a graph H is a *homomorphism* from G to H if for every edge $uv \in E(G)$, $\varphi(u)\varphi(v) \in E(H)$. A homomorphism φ from G to H is a *graph covering* if for every $v \in V(G)$, $\varphi_{|N_G(v)}$ is a bijection between $N_G(v)$ and $N_H(\varphi(v))$.

These standard definitions extend naturally to labelled graphs: for any functions $label$ defined on $V(G)$ and $label'$ defined on $V(H)$ and for any port numberings δ of G and δ' of H, $\varphi : V(G) \to V(H)$ is a homomorphism (resp. a graph covering) from $(G, \delta, label)$ to $(H, \delta', label')$ if $\varphi : G \to H$ is a homomorphism (resp. a graph covering) such that $label'(\varphi(u)) = label(u)$ for every $u \in V(G)$ and $\delta_u(v) = \delta'_{\varphi(u)}(\varphi(v))$ for every edge $uv \in E(G)$.

3.2 Simplicial Complexes

Definitions in this section are standard notions from discrete topology [18]. Given a set V, a *simplex* s of dimension $n \in \mathbb{N}$ is a subset of V of size $n+1$. A *simplicial complex* K is a collection of simplices such that for every simplex $s \in K$, $s' \subseteq s$ implies $s' \in K$. A simplicial complex K is k-dimensional if the largest dimension of a simplex of K is k.

A graph G can be seen as a 1-dimensional simplicial complex where $V(G)$ is the set of 0-dimensional simplices and $E(G)$ is the set of 1-dimensional simplices.

Given a simplicial complex K, the 0-dimensional simplices of K are the *vertices* of K and the 1-dimensional simplices of K are the *edges* of K. For a simplicial complex K, we denote by $V(K)$ (resp. $E(K)$) the set of vertices (resp. of edges) of K, and the 1-*skeleton* of K is the graph $G(K) = (V(K), E(K))$. A simplicial complex is said to be connected if its 1-skeleton is connected. We consider only connected complexes.

The *star* $St(v, K)$ of a vertex v in a simplicial complex K is the subcomplex defined by taking the collection of simplices of K containing v and their subsimplices.

It also possible to have a notion of covering for simplicial complexes. A *simplicial map* $\varphi : K \to K'$ is a map $\varphi : V(K) \to V(K')$ such that for any simplex $s = \{v_1, \ldots, v_k\}$ in K, $\varphi(s) = \{\varphi(v_1), \ldots, \varphi(v_k)\}$ is a simplex in K'.

Definition 3.1. *A simplicial map $\varphi : K \to K'$ is a* simplicial covering *if for every vertex $v \in V(K)$, $\varphi|_{St(v,K)}$ is a bijection between $St(v, K)$ and $St(\varphi(v), K')$.*

Examples of simplicial coverings are presented at the end of this section.For any simplicial complex K, the following proposition shows that there always exists a "maximal" cover of K that is called *the universal cover* of K.

Proposition 3.2 (Universal Cover). *For any simplicial complex K, there exists a possibly infinite complex (unique up to isomorphism) denoted $\widehat{K}$ and a simplicial covering $\mu : \widehat{K} \to K$ such that, for any complex K', for any simplicial covering $\varphi : K' \to K$, there exists a simplicial covering $\gamma : \widehat{K} \to K'$ and $\varphi \circ \gamma = \mu$.*

Given a graph $G = (V, E)$, the *clique complex* of G, denoted $\mathcal{K}(G)$ is the simplicial complex formed by the cliques of G. Note that for any graph G, the 1-skeleton of $\mathcal{K}(G)$ is G. Examples of simplicial coverings and clique complexes are presented in Figure 1.

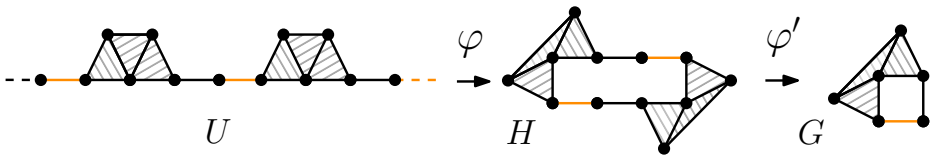

Fig. 1: $\mathcal{K}(H)$ is a (simplicial) cover of $\mathcal{K}(G)$. $\mathcal{K}(U)$ is an infinite graph that is a simplicial cover of both $\mathcal{K}(H)$ and $\mathcal{K}(G)$.

Given two graphs G, G', a map $\varphi : V(G) \to V(G')$ is a simplicial map from $\mathcal{K}(G)$ to $\mathcal{K}(G')$ if and only if for each edge $uv \in E(G)$, either $\varphi(u) = \varphi(v)$ or $\varphi(u)\varphi(v) \in E(G')$. Note that if $\varphi : \mathcal{K}(G) \to \mathcal{K}(G')$ is a simplicial covering, then φ is also a graph covering from G to G'. Note however that the converse does not hold. Indeed, let C_3 and C_6 be two cycles of respective lengths 3 and 6. There is a graph covering from C_6 to C_3 but there is no simplicial covering from C_6 to C_3 since every vertex of $\mathcal{K}(C_3)$ belongs to a 2-dimensional simplex while no vertex of $\mathcal{K}(C_6)$ does.

However, when we consider graphs labelled with their binoculars labelling, the two notions are equivalent. Note that in the previous example with C_6 and C_3, there is no graph covering from C_6 to C_3 that preserves the binoculars labels.

Proposition 3.3. *Let G and H be two graphs labelled with their binoculars labelling and consider a homomorphism $\varphi : G \to H$. The map φ is a graph covering from G to H if and only if φ is a simplicial covering from $\mathcal{K}(G)$ to $\mathcal{K}(H)$.*

From standard distributed computability results [3,4,5,20], it is known that the structure of graph coverings explains what can be computed or not. So in order to investigate the structure induced by coverings of graphs with binoculars labelling, we will investigate the structure of simplicial coverings of simplicial complexes.

In the following, we will only consider simplicial coverings, and for sake of simplicity, we will name them "coverings".

Homotopy. We say that two loops $c = (v_0, v_1, \ldots, v_{i-1}, v_i, v_{i+1}, \ldots, v_k)$ and $c' = (v_0, v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_k)$ in a complex K are related by an *elementary homotopy* if one of the following conditions holds: $v_i = v_{i+1}$, $v_{i-1} = v_{i+1}$, or $v_{i-1}v_iv_{i+1}$ is a triangle of K (i.e., $v_{i-1}v_{i+1}$ is an edge of K when K is a clique complex).

Note that being related by an elementary homotopy is a reflexive relation (we can either increase or decrease the length of the loop). We say that two loops c and c' are homotopic equivalent if there is a sequence of loops $c_1, \ldots, c_k$ such that $c_1 = c$, $c_k = c'$, and for every $1 \leq i < k$, c_i is related to c_{i+1} by an elementary homotopy. A loop is k-contractible (for $k \in \mathbb{N}$) if it can be reduced to a vertex by a sequence of k elementary homotopies. A loop is contractible if there exists $k \in \mathbb{N}$ such that it is k-contractible.

Remark that the number of elementary homotopies required to contract a loop is not necessarily monotone nor bounded by the number of vertices in the graph. For instance, you might have to enlarge a cycle before contracting it (think about the top cycle of a "sockwind-like surface").

Simple Connectivity. A *simply connected* complex is a complex where every loop can be reduced to a vertex by a sequence of elementary homotopies. These complexes have lots of interesting combinatorial and topological properties.

Proposition 3.4 ([18]). *Let K be a connected complex, then K is isomorphic to its universal cover $\widehat{K}$ if and only if it is simply connected.*

In fact, in order to check the simple connectivity of a simplex K, it is enough to check that all its simple cycles are contractible. The proof is straightforward.

Proposition 3.5. *A complex K is simply connected if and only if every simple cycle is contractible.*

Complexes with Finite Universal Cover. We define $\mathcal{FC} = \{G \mid$ the universal cover of $\mathcal{K}(G)$ is finite $\}$ and $\mathcal{IC} = \{G \mid G$ is finite and the universal cover of $\mathcal{K}(G)$ is infinite $\}$. Note that $\mathcal{FC}$ admits one interesting sub-class $\mathcal{SC} = \{G \mid G$ is finite and $\mathcal{K}(G)$ is simply connected$\}$.

.

4 First Impossibility Result and Lower Bound

First, in Lemma 4.1, we propose a Lifting Lemma for coverings of clique complexes. This lemma shows that every execution on a graph G can be lifted up to every graph G' such that $\mathcal{K}(G')$ is a cover of $\mathcal{K}(G)$, and in particular, to the 1-skeleton $\widehat{G}$ of the universal cover of $\mathcal{K}(G)$.

Consider an algorithm $\mathcal{A}$ and an execution of $\mathcal{A}$ performed by a mobile agent with binoculars starting on a vertex v in a network G. For any $i \in \mathbb{N}$, we denote respectively the position of the agent and its state (i.e., the content of its memory) at step i by $\mathtt{pos}_i(\mathcal{A}, G, v)$ and $\mathtt{mem}_i(\mathcal{A}, G, v)$. By standard techniques ([3,4,5,20]), we have the following lemma.

Lemma 4.1 (Lifting Lemma). *Consider two graphs G and G' such that there exists a covering $\varphi : \mathcal{K}(G') \to \mathcal{K}(G)$. For any algorithm $\mathcal{A}$ and for any vertices $v \in V(G)$ and $v' \in V(G')$ such that $\varphi(v') = v$, for any step $i \in \mathbb{N}$, $\mathtt{mem}_i(\mathcal{A}, G', v') = \mathtt{mem}_i(\mathcal{A}, G, v)$ and $\varphi(\mathtt{pos}_i(\mathcal{A}, G', v')) = \mathtt{pos}_i(\mathcal{A}, G, v)$.*

Using the Lifting Lemma above, we are now able to prove a first result about explorable graphs and the move complexity of their exploration.

Proposition 4.2. *Any explorable graph G belongs to $\mathcal{FC}$, and any Exploration algorithm exploring G performs at least $|V(\widehat{G})| - 1$ moves, where $\widehat{G}$ is the 1-skeleton of the universal cover of the clique complex $\mathcal{K}(G)$.*

Proof. Suppose it is not the case and assume there exists an exploration algorithm $\mathcal{A}$ that explores a graph $G \in \mathcal{IC}$ when it starts from a vertex $v_0 \in V(G)$. Let r be the number of steps performed by $\mathcal{A}$ on G when it starts on v_0.

Let $\widehat{G}$ be the 1-skeleton of the universal cover of $\mathcal{K}(G)$. Consider a covering $\varphi : \mathcal{K}(\widehat{G}) \to \mathcal{K}(G)$ and consider a vertex $\widehat{v}_0 \in V(\widehat{G})$ such that $\varphi(\widehat{v}_0) = v_0$. By Lemma 4.1, when executed on $\widehat{G}$, $\mathcal{A}$ stops after r steps. Consider the graph $H = B_{\widehat{G}}(\widehat{v}_0, r+1)$. Since $G \in \mathcal{IC}$, $\widehat{G}$ is infinite and $|V(H)| > r+1$. When executed on H starting in $\widehat{v}_0$, $\mathcal{A}$ behaves as in $\widehat{G}$ during at least r steps since the r first moves can only depend of $B_H(\widehat{v}, r) = B_{\widehat{G}}(\widehat{v}, r)$. Consequently $\mathcal{A}$ stops after r steps when executed on H starting in $\widehat{v}_0$. Since $|V(H)| > r+1$, $\mathcal{A}$ stops before it has visited all nodes of H and thus $\mathcal{A}$ is not an Exploration algorithm, a contradiction.

The move complexity bound is obtained from the Lifting Lemma applied to any covering $\varphi : \mathcal{K}(\widehat{G}) \to \mathcal{K}(G)$. Assume we have an Exploration algorithm $\mathcal{A}$ halting on G at some step q. If $|V(\widehat{G})| > q+1$ then $\mathcal{A}$ halts on $\widehat{G}$ and has not visited all vertices of $\widehat{G}$ since at most one vertex can be visited in a step (plus the homebase). A contradiction. □

Note that this is the same lifting technique that shows that, without binoculars, tree networks are the only explorable networks without global knowledge.

5 Exploration of $\mathcal{FC}$

We propose in this section an Exploration algorithm for the family $\mathcal{FC}$ in order to prove that this family is the maximum set of explorable networks.

The goal of Algorithm 1 is to visit, in a BFS fashion, a ball centered on the homebase of the agent until the radius of the ball is sufficiently large to ensure that G is explored. Once such a radius is reached, the agent stops. To detect when the radius is sufficiently large, we use the view of the homebase (more details below) to search for a simply connected graph which locally looks like the explored ball.

The view of a vertex is a standard notion in anonymous networks [4,20]. The *view* of a vertex v in a labelled graph $(G, label)$ is a possibly infinite tree composed by paths starting from v in G. From [20], the view $\mathcal{T}_G(v)$ of a vertex v in G is the labelled rooted tree built recursively as follows. The root of $\mathcal{T}_G(v)$, denoted by x_0, corresponds to v and is labelled by $label(x_0) = label(v)$. For every vertex v_i adjacent to v, we add a node x_i in $V(\mathcal{T}_G(v))$ with $label(x_i) = label(v_i)$ and we add an edge x_0x_1 in $E(\mathcal{T}_G(v))$ with $\delta_{x_0}(x_i) = \delta_v(v_i)$ and $\delta_{x_i}(x_0) = \delta_{v_i}(v)$. To finish the construction, every node x_i adjacent to x_0 is identified with the root of the tree $\mathcal{T}_G(v_i)$. We denote by $\mathcal{T}_G(v,k)$, the view $\mathcal{T}_G(v)$ truncated at depth k. If the context permits it, we denote it by $\mathcal{T}(v,k)$. Given an integer $k \in \mathbb{N}$, we define an equivalence relation on vertices using the views truncated at depth k: $v \sim_k w$ if $\mathcal{T}_G(v,k) = \mathcal{T}_G(w,k)$.

Note that in the following, we will consider the case where for each node v, $label(v)$ is equal to $\nu(v)$, the graph that is obtained using binoculars from v.

Algorithm 1: $\mathcal{FC}$-Exploration algorithm

$k := 0$;
repeat
  Increment k ;
  Compute $\mathcal{T}(v_0, 2k)$;
  Find a complex H (if it exists) such that:
    – $|V(H)| < k$, and
    – $\exists \widetilde{v}_0 \in V(H)$ such that $\widetilde{v}_0 \sim_{2k} v_0$, and
    – every simple cycle of $\mathcal{K}(H)$ is k-contractible;
until *H is defined*;
Stop the exploration;

5.1 Presentation of the Algorithm

Consider a graph G and let $v_0 \in V(G)$ be the homebase of the agent in G. Let k be an integer initialized to 1. Algorithm 1 is divided in phases. At the beginning of a phase, the agent follows all paths of length at most $2k$ originating from v_0 in order to compute the view $\mathcal{T}(v_0, 2k)$ of v_0.

At the end of the phase, the agent backtracks to its homebase, and enumerates all graphs of size at most k until it finds a graph H such that all simple cycles of $\mathcal{K}(H)$ are k-contractible and such that there exists a vertex $\widetilde{v}_0 \in V(H)$ that has the same view at distance $2k$ as v_0, i.e., $\mathcal{T}_H(\widetilde{v}_0, 2k) = \mathcal{T}_G(v_0, 2k)$.

If such an H exists then the algorithm stops. Otherwise, k is incremented and the agent starts another phase.

Deciding the k-contractibility of a given cycle is computable (by considering all possible sequences of elementary homotopies of length at most k). Since the total number of simple cycles of a graph is finite, Algorithm 1 can be implemented on a Turing machine.

5.2 Correction of the algorithm

In order to prove the correction of this algorithm, we prove that when the first graph H satisfying every condition of Algorithm 1 is found, then $\mathcal{K}(H)$ is actually the universal cover of $\mathcal{K}(G)$ (Corollary 5.2). Intuitively, this is because it is not possible to find a *simply connected* complex that looks locally the same as a *strict subpart* of another complex.

Remember that given a path p in a complex G, $\lambda(p)$ denotes the sequence of outgoing port numbers followed by p in G. We denote by $\textsc{dest}_G(v_0, \lambda(p))$, the vertex in G reached by the path starting in v_0 and labelled by $\lambda(p)$. We show (Proposition 5.1) that if we fix a vertex $\widetilde{v}_0 \in V(H)$ such that $\widetilde{v}_0 \sim_{2k} v_0$, we can define unambiguously a map φ from $V(H)$ to $V(G)$ as follows: for any $\widetilde{u} \in V(H)$, let p be any path from $\widetilde{v}_0$ to $\widetilde{u}$ in H and let $u = \varphi(\widetilde{u})$ be the vertex reached from v_0 in G by the path labelled by $\lambda(p)$.

Proposition 5.1. *Consider a graph G such that Algorithm 1 stops on G when it starts in v_0. Let $k \in \mathbb{N}$ and let H be the graph computed by the algorithm before it stops. Consider any vertex $\widetilde{v}_0 \in V(H)$ such that $v_0 \sim_{2k} \widetilde{v}_0$.*

For any vertex $\widetilde{u} \in V(H)$, for any two paths $\widetilde{q}, \widetilde{q}'$ from $\widetilde{v}_0$ to $\widetilde{u}$ in H, $\textsc{dest}_G(v_0, \lambda(\widetilde{q})) = \textsc{dest}_G(v_0, \lambda(\widetilde{q}'))$.

The proof is rather technical and involves careful inductions inside the space of homotopies. It is omitted here for lack of space, the complete proof is presented in the full version [6]. Showing that φ is a covering, we get the following corollary.

Corollary 5.2. *Consider a graph G such that Algorithm 1 stops on G when it starts in $v_0 \in V(G)$ and let H be the graph computed by the algorithm before it stops. The clique complex $\mathcal{K}(H)$ is the universal cover of $\mathcal{K}(G)$.*

Proof. By the definition of Algorithm 1, the complex $\mathcal{K}(H)$ is simply connected. Consequently, we just have to show that $\mathcal{K}(H)$ is a cover of $\mathcal{K}(G)$.

Consider any vertex $\widetilde{v}_0 \in V(H)$ such that $v_0 \sim_{2k} \widetilde{v}_0$. For any vertex $\widetilde{u} \in V(H)$, consider any path $\widetilde{p}_{\widetilde{u}}$ from $\widetilde{v}_0$ to $\widetilde{u}$ and let $\varphi(\widetilde{u}) = \textsc{dest}_G(v_0, \lambda(\widetilde{p}_{\widetilde{u}}))$. From Proposition 5.1, $\varphi(\widetilde{u})$ is independent from our choice of $\widetilde{p}_{\widetilde{u}}$. Since $v_0 \sim_{2k} \widetilde{v}_0$ and since $|V(H)| \leq k$, for any $\widetilde{u} \in V(H)$, $\nu(\varphi(\widetilde{u})) = \nu(\widetilde{u})$. Consequently, for any $\widetilde{u} \in V(H)$ and for any neighbour $\widetilde{w} \in N_H(\widetilde{u})$, there exists a unique $w \in N_G(\varphi(\widetilde{u}))$ such that $\lambda(\widetilde{u}, \widetilde{w}) = \lambda(\varphi(\widetilde{u}), w)$. Conversely, for any $w \in N_G(\varphi(\widetilde{u}))$, there exists a unique $\widetilde{w} \in N_H(\widetilde{u})$ such that $\lambda(\widetilde{u}, \widetilde{w}) = \lambda(\varphi(\widetilde{u}), w)$. In both cases, let $\widetilde{p}_{\widetilde{w}} = \widetilde{p}_{\widetilde{u}} \cdot (\widetilde{u}, \widetilde{w})$; this is a path from $\widetilde{v}_0$ to $\widetilde{w}$. From Proposition 5.1, $\varphi(\widetilde{w}) = \textsc{dest}_G(v_0, \lambda(\widetilde{p}_{\widetilde{w}})) = \textsc{dest}_G(u, \lambda(\widetilde{u}, \widetilde{w})) = w$. Consequently, φ is a graph covering from H to G, and by definition of H, φ also preserves the binoculars labelling. Therefore, the complex $\mathcal{K}(H)$ is a cover of the complex $\mathcal{K}(G)$. □

To finish to prove that Algorithm 1 is an Exploration algorithm for $\mathcal{FC}$, we remark that, when considering connected complexes (or graphs), coverings are always surjective. Consequently, G has been explored when the algorithm stops.

Theorem 5.3. *Algorithm 1 is an Exploration algorithm for $\mathcal{FC}$.*

Proof. From Corollary 5.2, we know that if Algorithm 1 stops, then the clique complex $\mathcal{K}(H)$ of the graph H computed by the algorithm is a cover of $\mathcal{K}(G)$. Moreover, since $|V(G)| \leq |V(H)| \leq k$ and since the agent has constructed $\mathcal{T}_G(v, 2k)$, it has visited all vertices of G.

We just have to prove that Algorithm 1 always halts on any graph $G \in \mathcal{FC}$. Consider any graph $G \in \mathcal{FC}$ and let $\widehat{G}$ be the 1-skeleton of the universal cover of $\mathcal{K}(G)$. Since $G \in \mathcal{FC}$, $\widehat{G}$ is finite and there exists $k' \in \mathbb{N}$ such that every simple cycle of $\widehat{G}$ is k'-contractible. Let $k = \max(|V(\widehat{G})|, k')$. At phase k, since $\mathcal{K}(\widehat{G})$ is the universal cover of $\mathcal{K}(G)$, there exists $\widetilde{v}_0 \in V(\widehat{G})$ such that $\mathcal{T}_G(v_0) = \mathcal{T}_{\widehat{G}}(\widetilde{v}_0)$. Consequently, $\mathcal{T}_G(v_0, 2k) = \mathcal{T}_{\widehat{G}}(\widetilde{v}_0, 2k)$, $|V(\widehat{G})| \leq k$, and every simple cycle of $\mathcal{K}(\widehat{G})$ is k-contractible. Therefore, at iteration k, the halting condition of Algorithm 1 is satisfied. □

From Proposition 4.2 and Theorem 5.3 above, we get the following corollary.

Corollary 5.4. *The family $\mathcal{FC}$ is the maximum set of Explorable networks.*

6 Complexity of the Exploration Problem

In the previous section, we did not provide any bound on the number of moves performed by an agent executing our universal exploration algorithm. In this section, we study the complexity of the problem and we show that there does not exist any exploration algorithm for all graphs in $\mathcal{FC}$ such that one can bound the number of moves performed by the agent by a computable function.

The first reason that such a bound cannot exist is rather simple: if the 1-skeleton $\widehat{G}$ of the universal cover of the clique complex of G is finite, then by Lemma 4.1, when executed on G, any exploration algorithm has to perform at least $|V(\widehat{G})|-1$ steps before it halts. In other words, one can only hope to bound the number of moves performed by an exploration algorithm on a graph G by a function of the size of $\widehat{G}$.

However, in the following theorem, we show that even if we consider only graphs with simply connected clique complexes (i.e., they are isomorphic to their universal covers), there is no Exploration algorithm for this class of graph such that one can bound its complexity by a computable function. Our proof relies on a result of Haken [12] that show that it is undecidable to detect whether a finite simplicial complex is simply connected or not.

Theorem 6.1. *Consider any algorithm $\mathcal{A}$ that explores every finite graph $G \in \mathcal{SC}$. For any computable function $\mathtt{t} : \mathbb{N} \to \mathbb{N}$, there exists a graph $G \in \mathcal{SC}$ such that when executed on G, $\mathcal{A}$ executes strictly more than $\mathtt{t}(|V(G)|)$ steps.*

Proof. Suppose this is not true and consider an algorithm $\mathcal{A}$ and a computable function $\mathtt{t} : \mathbb{N} \to \mathbb{N}$ such that for any graph $G \in \mathcal{SC}$, $\mathcal{A}$ visits all the vertices of G and stops in at most $\mathtt{t}(|V(G)|)$ steps. We show that in this case, it is possible to algorithmically decide whether the clique complex of any given graph G is simply connected or not. However, this problem is undecidable [12] and thus we get a contradiction[2].

Algorithm 2 is an algorithm that takes as an input a graph G and then simulates $\mathcal{A}$ on G for $\mathtt{t}(|V(G)|)$ steps. If $\mathcal{A}$ does not stop within these $\mathtt{t}(|V(G)|)$ steps, then by our assumption on $\mathcal{A}$, we know that $G \notin \mathcal{SC}$ and the algorithm returns NO. If $\mathcal{A}$ stops within these $\mathtt{t}(|V(G)|)$ steps, then we check whether there exists a graph H such that $|V(G)| < |V(H)| \leq \mathtt{t}(|V(G)|)$ and such that the clique complex $\mathcal{K}(H)$ is a cover of $\mathcal{K}(G)$. If such an H exists, then $G \notin \mathcal{SC}$ and the algorithms returns NO. If we do not find such an H, the algorithm returns YES.

In order to show Algorithm 2 decides simple connectivity, it is sufficient to show that when the algorithm returns YES on a graph G, the clique complex $\mathcal{K}(G)$ is simply connected. Suppose it is not the case and let $\widehat{G}$ be the 1-skeleton of the universal cover of the clique complex $\mathcal{K}(G)$. Consider a covering φ from

[2] Note that the original result of Haken [12] does not assume that the simplicial complexes are clique complexes. However, for any simplicial complex K, the barycentric subdivision K' of K is a clique complex that is simply connected if and only if K is simply connected (see [13]).

Algorithm 2: An algorithm to check simple connectivity

Input: a graph G

Simulate $\mathcal{A}$ starting from an arbitrary starting vertex v_0 during $\mathtt{t}(|V(G)|)$ steps ;
if *$\mathcal{A}$ halts within $\mathtt{t}(|V(G)|)$ steps* **then**
  if *there exists a graph H such that $|V(G)| < |V(H)| \leq \mathtt{t}(|V(G)|)$ and such that the clique complex $\mathcal{K}(H)$ is a cover of the clique complex $\mathcal{K}(G)$* **then**
    return NO; // $\mathcal{K}(G)$ is not simply connected
  else
    return YES; // $\mathcal{K}(G)$ is simply connected
else return NO; // $\mathcal{K}(G)$ is not simply connected;

$\mathcal{K}(\widehat{G})$ to $\mathcal{K}(G)$ and let $\widehat{v}_0 \in V(\widehat{G})$ be any vertex such that $\varphi(\widehat{v}_0) = v_0$. By Lemma 4.1, when executed on $\widehat{G}$ starting in $\widehat{v}_0$, $\mathcal{A}$ stops after at most $\mathtt{t}(|V(G)|)$ steps.

If $\widehat{G}$ is finite, then $\widehat{G} \in \mathcal{SC}$ and by our assumption on $\mathcal{A}$, when executed on $\widehat{G}$, $\mathcal{A}$ must explore all vertices of $\widehat{G}$ before it halts. Consequently, $\mathcal{K}(\widehat{G})$ is a covering of $\mathcal{K}(G)$ with at most $\mathtt{t}(|V(G)|)$ vertices. Since $\mathcal{K}(G)$ is not simply connected, necessarily $|V(G)| < |V(\widehat{G})|$ and in this case, the algorithm returns NO and we are done.

Assume now that $\widehat{G}$ is infinite. Let $r = \mathtt{t}(|V(G)|)$ and let $B = B_{\widehat{G}}(\widehat{v}_0, r)$. Note that when $\mathcal{A}$ is executed on $\widehat{G}$ starting in $\widehat{v}_0$, any node visited by $\mathcal{A}$ belongs to B. Given two vertices, $\widehat{u}, \widehat{v} \in V(\widehat{G})$, we say that $\widehat{u} \equiv_B \widehat{v}$ if there exists a path from $\widehat{u}$ to $\widehat{v}$ in $\widehat{G} \setminus B$. It is easy to see that $\equiv_B$ is an equivalence relation, and that every vertex of B is the only vertex in its equivalence class. For a vertex $\widehat{u} \in V(\widehat{G})$, we denote its equivalence class by $[\widehat{u}]$. Let H be the graph defined by $V(H) = \{[\widehat{u}] \mid \widehat{u} \in V(\widehat{G})\}$ and $E(H) = \{[\widehat{u}][\widehat{v}] \mid \exists \widehat{u}' \in [\widehat{u}], \widehat{v}' \in [\widehat{v}], \widehat{u}'\widehat{v}' \in E(\widehat{G})\}$.

We now show that the clique complex $\mathcal{K}(H)$ is simply connected. Let $\varphi : V(\widehat{G}) \to V(H)$ be the map defined by $\varphi(\widehat{u}) = [\widehat{u}]$. By the definition of H, for any edge $\widehat{u}\widehat{v} \in E(\widehat{G})$, either $[\widehat{u}] = [\widehat{v}]$, or $[\widehat{u}][\widehat{v}] \in E(H)$. Consequently, φ is a simplicial map. Consider a loop $c_0 = (u_1, u_2, \ldots, u_p)$ in H. By the definition of H, there exists a loop $\widehat{c}_0 = (\widehat{u}_{1,1}, \ldots, \widehat{u}_{1,\ell_1}, \widehat{u}_{2,1}, \ldots, \widehat{u}_{2,\ell_2}, \ldots, \widehat{u}_{p,1}, \ldots, \widehat{u}_{p,\ell_p})$ in G such that for each $1 \leq i \leq p$ and each $1 \leq j \leq \ell_i$, $\varphi(\widehat{u}_{i,j}) = u_i$. Note that $\varphi(\widehat{c}_0) = (\varphi(\widehat{u}_{1,1}) = u_1, \ldots, \varphi(\widehat{u}_{1,\ell_1}) = u_1, \varphi(\widehat{u}_{2,1}) = u_2, \ldots, \varphi(\widehat{u}_{2,\ell_2}) = u_2, \ldots, \varphi(\widehat{u}_{p,1}) = u_p, \ldots, \varphi(\widehat{u}_{p,\ell_p}) = u_p)$ is homotopic to c_0.

Since $\mathcal{K}(\widehat{G})$ is simply connected, $\widehat{c}_0$ is contractible and thus there exists a sequence $\widehat{c}_0, \widehat{c}_1, \ldots, \widehat{c}_p$ such that $|\widehat{c}_p| = 1$ and there is an elementary homotopy between $\widehat{c}_{i-1}$ and $\widehat{c}_i$ for every $1 \leq i \leq p$. Since φ is a simplicial map, for every $1 \leq i \leq p$, there is an elementary homotopy between $\varphi(\widehat{c}_{i-1})$ and $\varphi(\widehat{c}_i)$. Consequently, $\varphi(\widehat{c}_0)$ is contractible and thus c_0 is also a contractible loop of H. Therefore, H is simply connected.

Since G is finite, the degree of every vertex of $\widehat{G}$ is bounded by $|V(G)|$ and consequently, the number of equivalence classes for the relation $\equiv_B$ is finite. Consequently, the graph H is finite and thus $H \in \mathcal{SC}$. Moreover, since for every

$\widehat{u} \in B$, $[\widehat{u}] = \{\widehat{u}\}$, the ball $B_H([\widehat{v}_0], r)$ is isomorphic to B. Consequently, when $\mathcal{A}$ is executed on H starting in $[\widehat{v}_0]$, $\mathcal{A}$ stops after at most r steps before it has visited all vertices of H, contradicting our assumption on $\mathcal{A}$. □

7 Conclusion

Enhancing a mobile agent with binoculars, we have shown that, even without any global information it is possible to explore and halt in the class of graphs whose clique complex have a finite universal cover. This class is maximal and is the counterpart of tree networks in the classical case without binoculars. Note that, contrary to the classical case, where the detection of unvisited nodes is somehow trivial (any node that is visited while not backtracking is new, and the end of discovery of new nodes is immediate at leaves), here we had to introduced tools from discrete topology in order to be able to detect when it is no more possible to encounter "new" nodes.

The class where we are able to explore is fairly large and has been proved maximal when using binoculars of range 1. When considering binoculars of range k, clique complexes are no longer the right tool to use, but we believe we can obtain a similar characterization of explorable graphs by considering other cell complexes associated with the graph. Note that for triangle-free networks, enhancing the agent with binoculars of range 1 does not change the class of explorable networks. More generally, from the proof techniques in Section 4, it can also be shown that providing only local information (e.g. using binoculars of range k) cannot be enough to explore all graphs (e.g. graphs with large girth).

While providing binoculars is a natural enhancement, it appears here that explorability increases at the cost of a huge increase in complexity: the number of moves, as a function of the size of the graph, increase faster that any computable function. This cannot be expected to be reduced for all explorable graphs for fundamental Turing computability reasons. But preliminary results show that it is possible to explore with binoculars with a linear move complexity in a class that is way larger that the tree networks. So the fact that the full class of explorable networks is not explorable efficiently should not hide the fact that the improvement is real for large classes of graphs. One of the interesting open problem is to describe the class of networks for which explorability is increased while still having reasonable move complexity, like networks that are explorable in linear time.

Note that our Exploration algorithm can actually compute the universal cover of the graph, and therefore yields a Map Construction algorithm if we know that the underlying graph has a simply connected clique complex. However, note that there is no algorithm that can construct the map for all graphs of $\mathcal{FC}$. Indeed, there exist graphs in $\mathcal{FC}$ that are not simply connected (e.g. triangulations of the projective plane) and by the Lifting Lemma, they are indistinguishable from their universal cover. Note that without binoculars, the class of trees is not only the class of graphs that are explorable without information, but also the class of graphs where we can reconstruct the map without information. Here, adding

binoculars, not only enables to explore more networks but also give a model with a richer computability structure : some problems (like Exploration and Map Construction) are no longer equivalent.

References

1. Aleliunas, R., Karp, R.M., Lipton, R., Lovász, L., Rackoff, C.: Random walks, universal traversal sequences, and the complexity of maze problems. In: FOCS 1979. pp. 218–223 (1979)
2. Ambühl, C., Gąsieniec, L., Pelc, A., Radzik, T., Zhang, X.: Tree exploration with logarithmic memory. ACM Transactions on Algorithms 7(2), 17:1–17:21 (2011)
3. Angluin, D.: Local and global properties in networks of processors. In: STOC 1980. pp. 82–93 (1980)
4. Boldi, P., Vigna, S.: An effective characterization of computability in anonymous networks. In: DISC 2001. LNCS, vol. 2180, pp. 33–47 (2001)
5. Chalopin, J., Godard, E., Métivier, Y.: Election in partially anonymous networks with arbitrary knowledge in message passing systems. Distributed Computing 25(4), 297–311 (2012)
6. Chalopin, J., Godard, E., Naudin, A.: Anonymous graph exploration with binoculars. Tech. rep. (2015), http://arxiv.org/abs/1505.00599
7. Cohen, R., Fraigniaud, P., Ilcinkas, D., Korman, A., Peleg, D.: Label-guided graph exploration by a finite automaton. In: ICALP 2005. LNCS, vol. 3580, pp. 335–346 (2005)
8. Das, S.: Mobile agents in distributed computing: Network exploration. Bulletin of the EATCS 109, 54–69 (Aug 2013)
9. Diks, K., Fraigniaud, P., Kranakis, E., Pelc, A.: Tree exploration with little memory. J. Algorithms 51(1), 38–63 (2004)
10. Guilbault, S., Pelc, A.: Gathering asynchronous oblivious agents with local vision in regular bipartite graphs. Theor. Comput. Sci. 509, 86–96 (2013)
11. Gąsieniec, L., Radzik, T.: Memory efficient anonymous graph exploration. In: WG 2008. LNCS, vol. 5344, pp. 14–29 (2008)
12. Haken, W.: Connections between topological and group theoretical decision problems. In: Word Problems Decision Problems and the Burnside Problem in Group Theory, Studies in Logic and the Foundations of Mathematics, vol. 71, pp. 427–441. North-Holland (1973)
13. Hatcher, A.: Algebraic topology. Cambridge University Press (2002)
14. Ilcinkas, D.: Setting port numbers for fast graph exploration. Theor. Comput. Sci. 401(1–3), 236–242 (2008)
15. Koucký, M.: Universal traversal sequences with backtracking. J. Comput. Syst. Sci. 65(4), 717–726 (2002)
16. Kranakis, E., Krizanc, D., Markou, E.: The Mobile Agent Rendezvous Problem in the Ring. Synthesis lectures on distributed computing theory, Morgan & Claypool Publishers (2010)
17. Lange, D.B., Oshima, M.: Seven good reasons for mobile agents. Commun. ACM 42(3), 88–89 (Mar 1999)
18. Lyndon, R., Schupp, P.: Combinatorial Group Theory. Ergebnisse der Mathematik und ihrer Grenzgebiete, Springer-Verlag (1977)
19. Reingold, O.: Undirected connectivity in log-space. J. ACM 55(4) (2008)
20. Yamashita, M., Kameda, T.: Computing on anonymous networks: Part I - Characterizing the solvable cases. IEEE Trans. Parallel Distrib. Syst. 7(1), 69–89 (1996)