

Election in the Qualitative World

J er mie Chalopin

chalopin@labri.fr

LaBRI, Universit  Bordeaux 1

351 cours de la Lib ration

33405 Talence, France

Abstract. In [3], Barri re et al. consider a *qualitative* model of distributed computing, where the labels of the entities are distinct but mutually incomparable. They study the leader election problem in a distributed mobile environment and they wonder whether there exists an algorithm such that for each distributed mobile environment, it either states that the problem cannot be solved in this environment, or it successfully elects a leader. In this paper, we give a positive answer to this question. We also give a characterization of the distributed mobile environments where the election problem can be solved.

1 Introduction

Consider an intercontinental highway network linking different cities in different countries. In each city, the directions to the other cities are written in the language that is locally spoken. Consider now a set of different drivers coming from different countries. Initially, each driver starts in his town and all the drivers want to meet at a single place. The only mean they have to communicate is to leave messages in each city they reach, but each driver can only speak his mother tongue: he can see that another driver left some message, but he cannot understand it. Moreover, each driver can consistently distinguish the different directions in each city, but the drivers cannot agree on an alphabetical order on these directions: a French driver would not be able to figure out how to order Chinese words in the Chinese way, for example. We wonder whether there exists a procedure that enables them to meet at a single point in a finite time.

In distributed computing, the links incident to each process are usually labelled by distinct numbers in order to allow each process (or each mobile agent) to consistently distinguish its neighbours; this labelling is usually called a port-numbering. In fact, these numbers allow not only to distinguish the links, but also to order them. Many distributed algorithms assume also that all the processes can be unambiguously identified, and therefore the processes are given numbers. Again, one can see that this allows to order the different processes according to their labels. This usual setting is a *quantitative* model, since each label can be seen as a number.

Nevertheless, as in the example presented above, one may be able to distinguish labels without being able to order them. In this paper, we consider

distributed mobile environments where mobile agents are scattered all over a network. All the agents have distinct *colors* (their labels), which are mutually incomparable: each agent can just check whether two colors are equal or not. The links incident to each vertex are also given distinct incomparable colors. This model is close to the one introduced by Barrière et al. in [3]; it is *qualitative*, in the sense that there is no a priori order between the labels. As in [3], we study the impact of the lack of a total order on the set of labels in a distributed mobile environment. In this way, we investigate the leader election problem, that is a classical problem to highlight the differences between various models of distributed computing.

In usual models, there is always an implicit order over the set of labels, since for each agent, each information is just a sequence of bits. Nevertheless, consider an algorithm designed to be executed by mobile agents over a network. If the agents have been implemented by different companies, and if the specifications of the algorithm do not specify how the integers must be represented, some agents can for instance store numbers with most significant bit first whereas other agents store numbers with least significant bit first; in this case, the agents would not agree on the meaning of the sequence 01101. Moreover, it is always interesting to deal with algorithms that need less specifications, since they are generally more robust, and easier to implement in different models of distributed computing.

The Model. In this paper, an agent is an entity which executes an algorithm: it can move from place to place (with some data and its algorithm) through communication links, it can make local computations on a place (a place provides tools for local computations: data, memories and process) and leave messages on a place. In our model, the environment is represented by a simple undirected connected graph $G = (V(G), E(G))$ and a set \mathcal{E} of mobile agents is scattered over G . Communications between agents is achieved through writing messages on *whiteboards*, where agents can read, write, and erase messages. There is one whiteboard on each vertex of G , and access to a whiteboard is in mutual exclusion. Initially, all the whiteboards are empty. Let $p : \mathcal{E} \rightarrow V(G)$ be the injection describing the initial placement of the agents in G . The vertex $p(r)$ is called the *homebase* of the agent $r \in \mathcal{E}$. We will denote such a distributed mobile environment by (G, \mathcal{E}, p) .

We consider a set of colors C and a function $color : \mathcal{E} \rightarrow C$ that associates to each agent a unique color. There is no *a priori* order on the set of colors: each agent can give its own order on the set of colors, but the agents do not agree on a particular order. Each agent can understand a message it has written, but it cannot understand a message written by another agent, it can just know the color of the message. We also suppose that initially, the homebases are marked: they contain a marker that enables each agent to know that a place is a homebase and to detect the color of this homebase. In each place, the incident links are labelled by different colors that enable each agent to consistently distinguish the neighbours of the place: for each vertex u , there exists an injective function δ_u that associates a color from a new set C' (i.e., $C' \cap C = \emptyset$) to each edge incident to u . The set $\delta = \{\delta_u : u \in V(G)\}$ constitutes the *ports-labelling* of G . Thanks

to this labelling δ , each agent can make a distinction between the incident edges of each vertex. Such a distributed colored mobile environment will be denoted by $(G, \delta, \mathcal{E}, p, color)$.

The agents are asynchronous, in the sense that every action they perform (computing, moving, etc.) takes a finite but otherwise unpredictable amount of time. Moreover, we suppose that an agent has not an initial knowledge of the network topology, neither of its size nor of the number of agents in the system. The actions an agent a located at a node v can perform depends on the current state of a , the current state of the whiteboard at v , and the color of the port through which a entered v . According to these informations, a can decide to write a message on the whiteboard of v , to leave v (through a port whose color may result from some computation), or to stay at v (for example, to wait that another agent leaves a message on the whiteboard).

This model is more restrictive than the one presented in [3], since in the model of Barrière et al. the agents cannot agree on an order on the set of colors, but they fully understand the symbols written by the other agents. However, the necessary condition presented in our model is the same as the one presented in [3]: the results presented in this paper remain true in the model of [3].

The Election Problem. The election problem is one of the paradigms of the theory of distributed computing. In the distributed mobile setting, the aim of a leader election algorithm is to distinguish one agent among the others. All the agents execute the same protocol, i.e., the only initial difference between two agents is their colors. At the end of the execution of the algorithm, there is exactly one agent in the state *elected*, whereas all the other agents enter the state *non-elected*. Moreover, it is supposed that once an agent enters in the state *elected* or *non-elected*, it remains in such a state until the end of the computation. Another important problem in this setting is the rendez-vous problem. The aim of a rendez-vous algorithm is to reach a configuration where all the mobile agents gather in the same vertex of the graph. These two problems are equivalent, since once an agent has been elected, if all the agents agree on the label *elected*, all the agents can gather in the homebase of the elected agent. Reversely, once all the agents have gathered in some place, the first agent that writes on the whiteboard of this place is elected, whereas all the others become non-elected. There exists a large variety of results for these problems in the mobile agent setting assuming different properties of the environment [2, 4, 5, 10, 11]. The election problem has also been extensively studied in the distributed setting, and particularly in anonymous networks, where the processes do not have distinct labels [1, 6, 9, 13].

Consider a graph G and a set of agents \mathcal{E} scattered over the network according to a function p . We say that we can solve the election problem on (G, \mathcal{E}, p) if the problem can be solved on $(G, \delta, \mathcal{E}, p, color)$ for all ports-labellings δ and all agent-coloring functions $color$. This implies that an election algorithm in the distributed mobile environment (G, \mathcal{E}, p) must not use some particularity of the ports-labelling or make any assumption on the set of colors (for example, if one know that there is always a red agent, one can design an algorithm that elects

the red agent). Note that, as for anonymous networks in the distributed setting [9, 13], the protocols must not depend on the ports-labelling. Indeed, the role of the ports-labelling is just to enable an agent to make a distinction between the different neighbours of a vertex.

As in [3], we say that an algorithm \mathcal{A} is an *effective* election algorithm if for each distributed mobile environment (G, \mathcal{E}, p) , each ports-labelling δ and each coloring function $color$, for all the executions of \mathcal{A} on $(G, \delta, \mathcal{E}, p, color)$, either all the agents detect that the election problem cannot be solved in (G, \mathcal{E}, p) , or the agents successfully elect one of them. In particular, note that such an algorithm does not need any initial knowledge about the topology, the size, the diameter of the network or about the number of agents.

Main Results. In this work, we give a characterization (Theorem 1) of distributed mobile environments, where the election problem can be solved.

In [3], Barrière et al. wonder whether there exists an effective algorithm for the qualitative world. The algorithm we describe gives a positive answer to this question (Theorem 2).

To obtain a necessary condition (Proposition 2), we use *well-balanced* automorphisms that have been introduced by Bougé in [8].

Then, we show that this necessary condition is also sufficient: we use some links between fibrations and automorphisms presented in [7] to describe an effective algorithm in Section 4.2 that solves the election problem when the necessary condition is satisfied.

2 Preliminaries

Labelled Digraphs. In the following, we will consider directed graphs (digraphs) with multiple arcs and self-loops. A digraph $D = (V(D), A(D), s_D, t_D)$ is defined by a set $V(D)$ of vertices, a set $A(D)$ of arcs and by two maps s_D and t_D that assign to each arc two elements of $V(D)$: a source and a target (in general, the subscripts will be omitted); if a is an arc, the arc a is said to be going out of $s(a)$ and coming into $t(a)$. We say that $s(a)$ is a predecessor of $t(a)$ and that $t(a)$ is a successor of $s(a)$. A digraph D is strongly connected if for all vertices $u, v \in V(D)$, there exists a sequence of arcs a_1, a_2, \dots, a_p such that $s(a_1) = u, \forall i \in [1, p-1], t(a_i) = s(a_{i+1})$ and $t(a_p) = v$. In the following, we will only consider strongly connected digraphs. A *symmetric* digraph D is a digraph endowed with a symmetry, that is, an involution $Sym : A(D) \rightarrow A(D)$ such that for every $a \in A(D), s(a) = t(Sym(a))$.

A digraph homomorphism γ between the digraph D and the digraph D' is a mapping $\gamma : V(D) \cup A(D) \rightarrow V(D') \cup A(D')$ such that if u, v are vertices of D and a is an arc such that $u = s(a)$ and $v = t(a)$ then $\gamma(u) = s(\gamma(a))$ and $\gamma(v) = t(\gamma(a))$. We say that γ is an isomorphism if γ is bijective and γ^{-1} is a homomorphism, too.

Throughout the paper we will consider digraphs where the vertices and the arcs are labelled with labels from a recursive label set L . A digraph G labelled over L will be denoted by (D, λ) , where $\lambda : V(D) \cup A(D) \rightarrow L$ is the labelling

function. The digraph D is called the underlying digraph and the mapping λ is a labelling of D . A mapping $\gamma: V(D) \cup A(D) \rightarrow V(D') \cup A(D')$ is a homomorphism from (D, λ) to (D', λ') if γ is a digraph homomorphism from D to D' which preserves the labelling, i.e., such that $\lambda'(\gamma(x)) = \lambda(x)$ for every $x \in V(D) \cup A(D)$. Labelled digraphs will be designated by bold letters like $\mathbf{D}, \mathbf{G}, \dots$. If \mathbf{D} is a labelled digraph, then D denotes the underlying digraph.

Let $G = (V(G), E(G))$ be a connected simple graph. The symmetric strongly connected digraph associated to G and denoted by $Dir(G)$ is (V, A) defined by: there is an arc a_1 from v_1 to v_2 and an arc a_2 from v_2 to v_1 in A if $\{v_1, v_2\} \in E(G)$ and $Sym(a_1) = a_2$. Note that this digraph does not contain multiple arcs or self-loops. Given a mobile environment (G, \mathcal{E}, p) , we define the labelling function χ_p of the vertices by $\chi_p(v) = 1$ if there exists an agent a such that $p(a) = v$, and $\chi_p(v) = 0$ otherwise. A distributed mobile environment (G, \mathcal{E}, p) can therefore be represented by the labelled digraph $(Dir(G), \chi_p)$.

For any set S , $|S|$ denotes the cardinality of S . For any integer q , we denote by $[1, q]$ the set of integers $\{1, 2, \dots, q\}$.

Fibrations and Coverings. The notions of fibrations and coverings are fundamental in this work; definitions and main properties are presented in [7].

A *fibration* between the digraphs D and D' is a homomorphism φ from D to D' such that for each arc a' of $A(D')$ and for each vertex v of $V(D)$ such that $\varphi(v) = v' = t(a')$ there exists a unique arc a in $A(D)$ such that $t(a) = v$ and $\varphi(a) = a'$. The arc a is called the lifting of a' at v , D is called the total digraph and D' the base of φ . We shall also say that D is fibred (over D'). The fibre over a vertex v' (resp. an arc a') of D' is the set $\varphi^{-1}(v')$ of vertices of D (resp. the set $\varphi^{-1}(a')$ of arcs of D).

An *opfibration* between the digraphs D and D' is a homomorphism φ from D to D' such that for each arc a' of $A(D')$ and for each vertex v of $V(D)$ such that $\varphi(v) = v' = s(a')$ there exists a unique arc a in $A(D)$ such that $s(a) = v$ and $\varphi(a) = a'$.

A *covering projection* is a fibration that is also an opfibration. If a covering projection $\varphi: D \rightarrow D'$ exists, D is said to be a *covering* of D' via φ . A symmetric digraph D is a *symmetric covering* of a symmetric digraph D' via a homomorphism φ if D is a covering of D' via φ such that $\forall a \in A(D), \varphi(Sym(a)) = Sym(\varphi(a))$. A digraph D is *symmetric-covering-minimal* if there does not exist any digraph D' not isomorphic to D such that D is a symmetric covering of D' .

Given two strongly connected digraphs D and D' , an interesting property satisfied by any covering projection φ from D to D' is that there exists $q \in \mathbb{N}$ such that $\forall x' \in V(D') \cup A(D'), |\varphi^{-1}(x')| = q$.

The notions of fibrations and coverings extend to labelled digraphs in a natural way: the homomorphisms must preserve the labelling. Examples of fibrations and coverings are given in Figure 1.

Fibrations, Coverings and Automorphisms. We now describe some properties of the relations that exist between fibrations and the automorphisms of a digraph. These results are described and proved in [7].

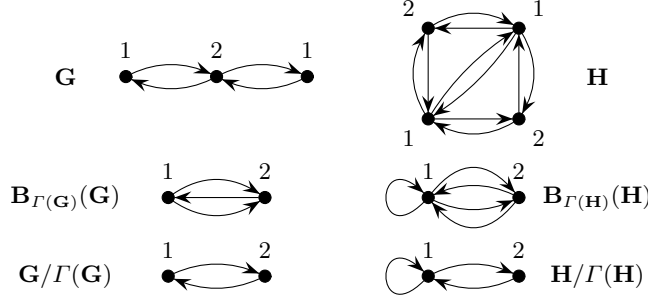


Fig. 1. The digraph \mathbf{G} is fibred over $\mathbf{B}_{\Gamma(\mathbf{G})}(\mathbf{G})$ via the homomorphism $\varphi_{\mathbf{G}}$ that maps each vertex of \mathbf{G} labelled i to the unique vertex labelled i of $\mathbf{B}_{\Gamma(\mathbf{G})}(\mathbf{G})$. The digraph \mathbf{H} is a covering of $\mathbf{B}_{\Gamma(\mathbf{H})}(\mathbf{H})$ via the homomorphism $\varphi_{\mathbf{H}}$ defined in the same way. The digraph $\mathbf{G}/\Gamma(\mathbf{G})$ (resp. $\mathbf{H}/\Gamma(\mathbf{H})$) is the digraph whose vertices and arcs correspond to equivalence classes of vertices and arcs of \mathbf{G} (resp. \mathbf{H}) under the action of $\Gamma(\mathbf{G})$ (resp. $\Gamma(\mathbf{H})$).

An automorphism σ of a digraph \mathbf{G} is an isomorphism from the digraph \mathbf{G} onto itself. Consider a subgroup Γ of the group $\Gamma(\mathbf{G}) = \text{Aut}(\mathbf{G})$ of the automorphisms of a digraph $\mathbf{G} = (G, \lambda)$; we will denote by Id the identity automorphism of \mathbf{G} . The action of this group on \mathbf{G} induces an equivalence relation over the vertices and the arcs of \mathbf{G} : for each $x, x' \in V(G) \cup A(G)$, $x \sim_{\Gamma} x'$ if there exists $\sigma \in \Gamma$ such that $\sigma(x) = x'$. The equivalence class of x is called the orbit of x and is denoted by $[x]_{\Gamma}$. Recall that an automorphism of (G, λ) must preserve the labelling, and therefore for all elements $x_1, x_2 \in [x]_{\Gamma}$, $\lambda(x_1) = \lambda(x_2)$. If $\Gamma = \Gamma(\mathbf{G})$, we will note $x \sim x'$ (resp. $[x]$) for $x \sim_{\Gamma} x'$ (resp. $[x]_{\Gamma}$).

Remark 1. For all vertices $v, v' \in V(G)$, if $v \sim_{\Gamma} v'$, then there is a bijection between the incoming arcs of v and the incoming arcs of v' .

We will now describe two kinds of constructions. The first one allows us to build a digraph $\mathbf{B}_{\Gamma}(\mathbf{G})$ from a digraph \mathbf{G} such that \mathbf{G} is fibred over $\mathbf{B}_{\Gamma}(\mathbf{G})$. The second one allows to build the quotient-graph \mathbf{G}/Γ . Examples are presented in Figure 1 where $\Gamma = \Gamma(\mathbf{G})$.

From the relation \sim_{Γ} , we construct the directed graph $B_{\Gamma}(\mathbf{G})$ defined as follows: $V(B_{\Gamma}(\mathbf{G}))$ is the set of the equivalence classes of $V(G)$ under the action of Γ and there are as many arcs from $[v]_{\Gamma}$ to $[w]_{\Gamma}$ as each vertex in $[w]_{\Gamma}$ has predecessors in $[v]_{\Gamma}$. Due to Remark 1, this does not depend on the choice of the element of $[w]_{\Gamma}$. We define the labelling ν of $B_{\Gamma}(\mathbf{G})$ by $\nu([v]_{\Gamma}) = \lambda(v)$ for each $v \in V(G)$. We label the arcs from $[v]$ to $[w]$ with the labels of the arcs from the elements of $[v]$ to w in \mathbf{G} . By Remark 1, there exists a fibration φ from \mathbf{G} to $(B_{\Gamma}(\mathbf{G}), \nu)$.

We consider also the quotient-graph \mathbf{G}/Γ whose vertices and arcs are the equivalence classes of the vertices and the arcs of G under the action of Γ and whose labelling μ is defined by $\mu([x]_{\Gamma}) = \lambda(x)$ for each $x \in V(G) \cup A(G)$. There exists a natural surjective homomorphism from $(B_{\Gamma}(\mathbf{G}), \nu)$ to \mathbf{G}/Γ which is the

identity on the vertices and which maps an arc a to $[a]_\Gamma$ (a can be seen as an arc of G).

We say that a subgroup Γ of $\Gamma(\mathbf{G})$ acts *freely* on \mathbf{G} if for each $x, y \in V(G) \cup A(G)$, there is at most one $\sigma \in \Gamma$ such that $\sigma(x) = y$. Equivalently, Γ acts freely on \mathbf{G} if and only if for each $\sigma \in \Gamma \setminus \{Id\}$, σ has no fixpoint.

In the following, we will use a particular class of automorphisms: the class of *well-balanced* automorphisms. These automorphisms have been introduced by Bougé in [8] to study the importance of the guards in CSP through the symmetric election problem. In [12], Palamidessi uses also well-balanced automorphisms to study the same problem in order to give a hierarchy between different subsets of the π -calculus. An automorphism σ of a digraph \mathbf{G} is *well-balanced* if there exists an integer q such that for each vertex or arc x of \mathbf{G} , $|\{\sigma^k(x) \mid k \in \mathbb{N}\}| = q$. Equivalently, σ is well-balanced if and only if the subgroup Γ_σ generated by σ acts freely on \mathbf{G} .

The group Γ contains only well-balanced automorphisms if and only if Γ acts freely on \mathbf{G} . Thanks to this equivalence and the results of Boldi and Vigna [7], we have the following property.

Proposition 1. *Given any strongly connected digraph \mathbf{G} , the quotient projection $\Gamma : \mathbf{G} \rightarrow \mathbf{G}/\Gamma$ is a covering projection if and only if for each $\sigma \in \Gamma$, σ is well-balanced.*

3 Impossibility Result

The following proposition gives a necessary condition that the distributed mobile environment (G, \mathcal{E}, p) must verify if there exists an election algorithm for (G, \mathcal{E}, p) . This necessary condition is equivalent to the one presented in [3].

Proposition 2. *Consider a graph G and an initial placement of the agents p . If there exists a non-trivial well-balanced automorphism σ of the digraph $\mathbf{G}' = (Dir(G), \chi_p)$, then there is no election algorithm over the graph G with the initial placement of the agents p .*

Using known results in distributed computing [6, 9, 13], we can show that in the anonymous setting, i.e., when the agents can understand each other but do not have distinct labels, there exists an election algorithm for an environment (G, \mathcal{E}, p) if and only if the labelled digraph $\mathbf{G}' = (Dir(G), \chi_p)$ is symmetric-covering-minimal. Moreover, from Proposition 1, we know that if the symmetric digraph $\mathbf{G}' = (Dir(G), \chi_p)$ admits a non-trivial well-balanced automorphism σ , then \mathbf{G}' is a symmetric covering of $\mathbf{G}'/\Gamma_\sigma$ that is not isomorphic to \mathbf{G}' .

Consequently, an interesting corollary of Proposition 2 is that if the election problem cannot be solved on (G, \mathcal{E}, p) in the qualitative setting, then it cannot be solved on (G, \mathcal{E}, p) in the anonymous setting. On the other hand, we will show in the following that this necessary condition is also sufficient. Note that there exist symmetric digraphs that are not symmetric-covering-minimal and that does not admit any non-trivial well-balanced automorphism. It means that one can solve the election problem in strictly more environments in the qualitative setting than in the anonymous one.

4 An Effective Election Algorithm

4.1 How to order the equivalence classes?

We use the same ideas as Barrière et al. [3] to define a total order between the different equivalent classes. The idea is to construct an ordering on the unlabelled digraphs of size n ; we extend it to digraphs labelled by elements of a totally ordered set.

Consider a labelled digraph $\mathbf{G} = (G, \lambda)$ without multiple arcs where λ is a labelling function from $V(G) \cup A(G)$ to a totally ordered set L with a minimal element \perp . We suppose that $\forall x \in V(G) \cup A(G), \lambda(x) \in L \setminus \{\perp\}$.

Let $n = |V(G)|$ and consider an enumeration function num of the vertices (i.e., num is a one-to-one mapping from $V(G)$ onto $[1, n]$). We say that num is an *increasing* enumeration of the vertices if for all vertices $v, v' \in V(G)$, if $num(v) \leq num(v')$, then $\lambda(v) \leq_L \lambda(v')$. Given an increasing enumeration num , we define the adjacency matrix M_{num} as follows: for all vertices v, v' , $M_{num}[num(v), num(v')] = \ell$, if there is an arc from v to v' labelled by ℓ , and $M_{num}[num(v), num(v')] = \perp$ otherwise. To this matrix, we associate the word $w(M_{num})$ obtained by the concatenation of the n rows of M_{num} .

To each vertex $v \in V(G)$ (resp. arc $a \in A(G)$), we choose num such that $(num(v), w(M_{num}))$ (resp. $(num(s(a)), num(t(a)), w(M_{num}))$) is minimum for the lexicographic order and associate this value, denoted by $\pi(v)$ (resp. $\pi(a)$), to v (resp. a). Note that there exists an automorphism σ of \mathbf{G} such that $\sigma(x) = x'$ if and only if $\pi(x) = \pi(x')$. Consequently, this induces a total ordering of the equivalence classes of vertices and arcs: we will write $[x'] \prec [x]$ if $\pi(x)$ is greater than $\pi(x')$ in the lexicographic order.

Remark 2. In the following, we will show that all the agents agree on a total order of the classes and all the agents use the same order. Actually, as it was already explained in [3], even if the agents cannot agree on an a priori order over the set of colors, they can agree on an order on the different classes, provided that all the agents have the same representation of the graph (up to isomorphism).

In fact, we suppose that each agent has its own totally ordered set isomorphic to (\mathbb{N}, \leq) and each agent can use its own way to compute its order: the algorithm does not make any assumption on the way the order is implemented by each agent.

4.2 An Election Algorithm

In this subsection, we describe our effective election algorithm. In a first phase all the agents reconstruct the digraph $(Dir(G), \chi_p)$ and check that the election problem can be solved on (G, \mathcal{E}, p) . Then, using its knowledge of the graph, each agent constructs the equivalence classes induced by $\Gamma((Dir(G), \chi_p))$. During successive rounds, using the order between the different classes defined above, some agents become passive and get the label *non-elected*, whereas the active agents mark some vertices and some arcs of the digraph to obtain a new labelling μ of the digraph on which all the active agents agree. At the end of the computation,

the automorphism group of $(Dir(G), \mu)$ consists only of the identity and each vertex has a unique label. At this point, there is exactly one active agent that is elected. A high level description of the algorithm is presented in Algorithm 1.

Algorithm 1: The Election Algorithm

```

Every agent builds a map of the graph;
Synchronization;
if there exists a non-trivial well-balanced automorphism of  $\mathbf{G}' = (Dir(G), \chi_p)$ 
then
    | Every agent knows that it is impossible to solve the election problem;
else
    | Every agent marks as many vertices as possible;
    | Synchronization;
    | /* Initially, all the agents are active */
    | repeat
    |     | The active agents compute the classes of all the vertices and all the arcs;
    |     | The active agents give different numbers to different classes of vertices
    |     | and arcs;
    |     | if All the active agents are not in the same class then
    |     |     | Select active agents;
    |     |     | The passive agents take the label non-elected;
    |     |     | The active agents mark the homebases of the passive agents;
    |     |     | Synchronization;
    |     | else if  $\mathbf{G}'$  is not a covering of  $\mathbf{B}_{\Gamma(\mathbf{G}')}(\mathbf{G}')$  then
    |     |     | The active agents mark a class of vertices;
    |     |     | Synchronization;
    |     | else if  $\mathbf{G}'$  is not a covering of  $\mathbf{G}'/\Gamma(\mathbf{G}')$  then
    |     |     | The active agents mark a class of arcs;
    |     |     | Synchronization;
    |     | else
    |     |     | /* In this case, there is exactly one active agent */
    |     |     | The active agent takes the label elected;
    | until An agent is elected;

```

A Synchronization Procedure. In the algorithm we describe below, we distinguish different rounds. An important point is that an active agent does not enter in a new round if another active agent has not finished the previous one. To be able to avoid this kind of situation, we synchronize the active agents.

Each agent can consistently distinguish its homebase; therefore, we can construct an algorithm such that no agent needs to write anything on its homebase. Moreover, we suppose that each agent has already built its own map of the graph and does not need to write anything on any whiteboard in order to perform a traversal of the graph.

In the following, the active agents will do some traversals of the network and they will store the colors of the marks that appear on each vertex to construct what we will call a *colored map* of the network. The marks that appear in a colored map of an active agent will correspond to marks that have been put by other active agents during the round (but it will not necessary contain all

the marks the active agents should put during this round). In the algorithm described below, each active agent can know from a colored map if any other active agent has marked all the vertices it should have marked during the round. Furthermore, in each round of the algorithm, each agent will mark at least one vertex (which is not its homebase).

In the synchronization procedure described below, some active agents will have to wait on some particular vertices for other agents to put (resp. remove) some marks. Each time an agent arrives on a place where it has to wait for a mark to be put (resp. removed), it can immediately continue to execute the procedure if this mark is present (resp. not present).

To synchronize the agents, we proceed as follows. During each round, each active agent r executes the following instructions.

- (1) The agent r marks some vertices (but not its homebase) according to the computation rules of the round.
- (2) The agent r does a traversal of the network and stores all the colors of the marks that appear on each vertex to construct a colored map of the network.
- (3) If there exists another active agent r' that has not finished Step (1) (the agent r can detect it from the colored map it has of the network), then the agent r goes to the homebase of r' and waits until the agent r' puts a mark on its homebase. Then the agent r does a traversal of the network and stores all the colors of the marks that appear on each vertex in order to update its colored map of the network.
- (4) The agent r puts a mark on its homebase.
- (5) The agent r does a traversal of the network and each time it arrives on the homebase of another agent r' , it waits until the agent r' marks its homebase.
- (6) The agent r does a traversal of the network and it removes the marks it puts during Step (1), but not the mark on its homebase.
- (7) The agent r does a traversal of the network. Each time it arrives on a vertex that has been marked by another active agent r' during this round (but that is not the homebase of r'), it waits until the agent r' removes its mark.
- (8) The agent r removes the mark it puts on its homebase.
- (9) The agent r does a traversal of the network. Each time it arrives on the homebase of an active agent r' , it waits until the agent r' removes its mark on its homebase.

We can note that the synchronization procedure enables also to erase all the marks that have been put on the vertices during the round, i.e., when one agent has finished Step (9) of a round, then all the marks that have been left by the active agents during this round have been erased. The following proposition ensures that the procedure is indeed a synchronization procedure.

Proposition 3. *Each time an agent starts executing Step (1) of the $i + 1$ th round, then each active agent knows what vertices have been marked by the other active agents during Step (1) of the i th round and all the marks that have been put during the i th round have been removed. Moreover, the synchronization procedure avoids any deadlock.*

Initialization. During the first phase of the algorithm, each agent reconstructs the graph with the position and the colors of the different homebases. Using the whiteboards, each agent performs a depth first traversal of the graph.

Since each agent can distinguish all the homebases, we suppose that during this traversal, the agents do not write anything on the whiteboard of any of the homebases. Once an agent has reconstructed the whole graph, it performs a traversal of the network using the information it has stored to erase what it has written on the whiteboards. At this point, each agent puts a mark on the homebase of another agent.

During this first phase, no agent has written anything on its homebase. Furthermore, an agent has finished to perform this phase if and only if it has marked the homebase of another agent and this can be checked from a colored map of the network. Moreover, at the end of this phase, each agent has reconstructed a map of the network and it knows the position of all the homebases. We can therefore use the synchronization procedure defined above at this point.

If the digraph $(Dir(G), \chi_p)$ admits a well-balanced automorphism σ different from Id , then each agent detects it and declares that the election problem is unsolvable in this environment. We will now suppose that $(Dir(G), \chi_p)$ does not admit such an automorphism.

Once the graph is known by all the agents, each agent tries to mark as many vertices of the networks as possible. It does a traversal of the network and each time it arrives on a vertex that is not a homebase, it performs one of the two following actions. Either the whiteboard is blank and it puts a mark with its color on the whiteboard, or there is already a mark on the whiteboard and it stores the color of the mark. Once an agent has finished this traversal, it puts a mark on the homebase of another agent. Again we use the synchronization procedure at this point. Then each agent is aware of the different vertices marked by the other agents during this round.

At the end of this phase, each agent reconstructs a graph where all the vertices are colored (they belong to the agent that has this color) and it knows the position and the color of the homebases of all the other agents.

How can the agents increase their territory? During the different phases of the algorithm, some agents become passive whereas the others continue to execute the protocol in order to elect one of them. In our algorithm, in order to break the symmetry between the agents, all the vertices must belong to one active agent, and all the active agents must agree on which agent a vertex belongs to. During the initialization, each vertex is marked by one agent and we say that it belongs to this agent. Once an agent becomes passive, the vertices that were belonging to this agent must be given to another agent.

Once a selection between agents is done, the agents that become passive take the label *non-elected* and become passive until the end of the algorithm, whereas the others try to mark the homebases of these agents that have just become passive. Each active agent knows what are the colors of the other active agents. From its representation of the graph, each active agent can reach the homebases of the passive agents.

The first agent that reaches such a homebase during this round puts a mark with its color on the vertex. The other agents (there is already a mark on the homebase when they reach it) store the color of the agent that owns this vertex (i.e., the color of the mark). Again, at the end of its traversal of the graph, each agent puts a mark on the homebase of another active agent. Therefore, each active agent can detect from a colored map if another active agent has finished this phase. Then the active agents apply the synchronization procedure.

If an agent has marked the homebase of a passive agent, then all the vertices that were belonging to this passive agent belong now to this active agent. For each vertex of the graph, all the active agents agree on the color of the agent that owns this vertex.

How to refine the labelling μ ? During the execution of the algorithm, the agents mark vertices and arcs to break the symmetry that may exist in the network. In this way, at each round, numbers will be associated to some vertices and arcs and we will obtain a labelling of the graph μ . Initially, all the homebases have the label 1 whereas all the other vertices have the label 0 and all the arcs are labelled 0.

At the beginning of each round, from its representation $(Dir(G), \mu)$ of the graph, each agent computes the value $\pi(v)$ (resp. $\pi(a)$) for each vertex $v \in V(Dir(G))$ (resp. for each arc $a \in A(Dir(G))$). We say that two agents are equivalent if their homebases are in the same equivalence class, and we use the order \prec on the homebases of the agents to order the classes of agents.

Since all the agents agree on the order to compare the equivalence classes, we can use the following procedure. If there exist two vertices v, v' such that $\mu(v) = \mu(v')$ and $\pi(v) \neq \pi(v')$, then let m be the lowest number such that there exist v, v' with $\mu(v) = \mu(v') = m$ and $[v] \prec [v']$. Suppose that there exist exactly j classes $\{[v_i] | i \in [1, j]\}$ such that $\mu(v_i) = m$ and $[v_1] \prec [v_2] \prec \dots \prec [v_j]$. For each vertex $v \in [v_i]$ with $i < j$, we define $\mu'(v) = q + i$, where q is the greatest label that appears on a vertex in (G, μ) . The labels of the other vertices are not changed.

We apply the same method to arcs using the order we have on the classes of arcs, i.e., the lexicographic order over the $\pi(a)$. Thanks to this procedure, two arcs that are not in the same class are given distinct labels. We repeat this procedure, until all the vertices (resp. all the arcs) that have the same label are in the same equivalence class.

If some active agents do not own the same number of vertices in a given class. We consider now a configuration such that two vertices (resp. arcs) in different classes have different numbers. Consider a class of agents $[r]$ and a class of vertices $[v]$. We define $NotBalanced([r], [v])$ to be false if all the agents of $[r]$ own the same number of vertices in $[v]$, and true otherwise. If there exist $[r], [v]$ such that $NotBalanced([r], [v])$ is true, then we apply the following technique to split some class of vertices.

Consider the minimum class $[r]$ of agents, according to \prec , such that there exists a class $[v]$ of vertices satisfying $NotBalanced([r], [v])$. Consider the minimum class of vertices $[v]$ such that $NotBalanced([r], [v])$ is true. In this case, we

give different numbers to the homebases of the agents that do not own the same number of vertices in $[v]$. We subdivide the class $[r]$ into a partition R_1, \dots, R_j such that the agents in R_i own strictly more vertices in $[v]$ than the agents in $R_{i'}$ when $i < i'$. Using the same technique as before, we give different numbers to the homebases of the agents that are not in the same R_i and then obtain a new representation of the digraph $(Dir(G), \mu')$. Then, the agents try to refine again this new labelling.

How to split the arc classes thanks to the colors of their ends? We will say that an arc a belongs to an agent r , if r owns $s(a)$ and $t(a)$. Otherwise, the arc is such that $s(a)$ belongs to an agent r_1 and $t(a)$ to a distinct agent r_2 . We will say that this arc is shared by r_1 and r_2 . If there exists a class of arcs $[a]$ such that some arcs of $[a]$ belong to some agents, whereas the other arcs are shared by distinct agents, then we apply the following technique.

Consider a class of arcs a such that for each class $[a'] \prec [a]$, either $[a']$ contains only arcs that belong to some agents or $[a']$ contains only arcs shared by different agents. We suppose also that $[a]$ contains arcs that belong to some agents and arcs that are shared. All the arcs in $[a]$ that are shared by distinct agents are relabelled $q + 1$, where q is the greatest label that appears on an arc in $(Dir(G), \mu)$. Then, the agents try to refine again this new labelling.

If some active agents are in different classes. At this point, if the active agents are not in the same equivalence class, we are able to select some agents. Consider all the equivalence classes of active agents that contains a minimal number of agents. Among these classes, we select the class $[r]$ such that $\pi(v)$ is minimal, where v is the homebase of r . The agents that do not belong to this class take the label *non-elected* and become passive. The agents of the class $[r]$ remain active and try to increase their territory as explained above. Then they try to refine again the labelling μ .

If $\mathbf{G}' = (Dir(\mathbf{G}), \mu)$ is not a covering of $\mathbf{B}_{\Gamma(\mathbf{G}')}(\mathbf{G}')$. There exist some configurations where it is impossible to select some agents just by using the representation the agents have of the graph, because there is too much symmetry in the graph. Nevertheless, we now explain how active agents can break these symmetries by marking some vertices or arcs.

All the active agents agree on the graph $\mathbf{G}' = (Dir(G), \mu)$. All these agents consider the automorphism group $\Gamma(\mathbf{G}')$ and construct the graphs $\mathbf{B}_{\Gamma(\mathbf{G}')}(\mathbf{G}')$ and $\mathbf{G}'/\Gamma(\mathbf{G}')$. We already know that \mathbf{G}' is fibred over $\mathbf{B}_{\Gamma(\mathbf{G}')}(\mathbf{G}')$. If \mathbf{G}' is not a covering of $\mathbf{B}_{\Gamma(\mathbf{G}')}(\mathbf{G}')$, it implies that there exist two classes of vertices $[v]$ and $[v']$ such that $|[v]| \neq |[v']|$. Let $[r]$ be the class of the homebases of the active agents.

Consider a class $[v]$ such that for each class $[v'] \prec [v]$, $|[v']| = |[r]|$ and $|[v]| \neq |[r]|$. We already know that each active agent owns the same number of vertices in $[v]$ and therefore $|[r]|$ divides $|[v]|$. Each active agent then marks a vertex it owns that is in $[v]$. An agent has finished this round if and only if it has marked exactly one vertex in $[v]$: it can be detected from a colored map of the graph. Then, the agents synchronize.

At the end of this round, all the agents give the number $q + 1$ to the vertices that have just been marked, where q is the greatest label that appears on a vertex in $(Dir(G), \mu)$. Using this new labelling μ' , the active agents try to refine the labelling μ' , as explained above.

If $\mathbf{G}' = (Dir(G), \mu)$ is not a covering of $\mathbf{G}'/\Gamma(\mathbf{G}')$. We suppose now that \mathbf{G}' is a covering of $\mathbf{B}_{\Gamma(\mathbf{G}')}(\mathbf{G}')$ but not of $\mathbf{G}'/\Gamma(\mathbf{G}')$. It means that all the equivalence classes of vertices have the same size s , but there exists an equivalence class of arcs $[a]$ such that $|[a]| > s$. Instead of marking vertices, we mark arcs in this round.

Each class $[a]$ of arcs of \mathbf{G}' corresponds to exactly one arc in $\mathbf{G}'/\Gamma(\mathbf{G}')$. Consider the class of arcs $[a]$ such that for each class $[a'] \prec [a]$, $|[a']| = s$ but $|[a]| > s$. We already know that each active agent owns exactly one vertex in $[s(a)]$ and one vertex in $[t(a)]$. Since $|[a]| > s$ and since two arcs in the same class are either both owned by an agent or both shared by distinct agents, we know that each arc in $[a]$ is shared.

To select arcs from $[a]$, each agent r just chooses one arc a_r in $[a]$ such that $s(a_r)$ belongs to r and then puts a mark with its color on $t(a_r)$. An agent has finished this round if and only if it has marked exactly one vertex: it can be detected from a colored map of the graph. Then, the agents synchronize. Once an agent knows what vertices have been marked by the other agents, it knows what are the arcs that have been marked.

At the end of this round, all the agents give the number $q + 1$ to the arcs that have just been marked, where q is the greatest label that appears on an arc in $(Dir(G), \mu)$. Using this new labelling μ' , the active agents try to refine again the labelling μ' , as explained above.

If \mathbf{G}' is a covering of $\mathbf{G}'/\Gamma(\mathbf{G}')$. At this point, $\mathbf{G}' = (Dir(G), \mu)$ is a covering of $\mathbf{G}'/\Gamma(\mathbf{G}')$. From Proposition 1, it implies that $\Gamma(\mathbf{G}')$ contains only well-balanced automorphisms, and since we already know that there is no well-balanced isomorphism of $(Dir(G), \chi_p)$ different from Id , we have $\Gamma(\mathbf{G}') = \{Id\}$. Consequently, there is exactly one active agent, since the set of active agents is an equivalence class of the relation induced by $\Gamma(\mathbf{G}')$ and this agent takes the label *elected*.

4.3 The Characterization

In Section 3, we have shown that if the graph $(Dir(G), \chi_p)$ admits a well-balanced automorphism, then it is impossible to solve the election problem on (G, \mathcal{E}, p) . The algorithm described in Section 4.2 is an algorithm that answers that it is impossible to solve the problem if the graph $(Dir(G), \chi_p)$ admits a well-balanced automorphism, and otherwise it successfully elects an agent: it is an effective algorithm. We have therefore proved the following theorems.

Theorem 1. *There exists an election algorithm for a distributed mobile environment (G, \mathcal{E}, p) if and only if $(Dir(G), \chi_p)$ does not admit a non-trivial well-balanced automorphism.*

Theorem 2. *Algorithm 1 is an effective election algorithm in the qualitative world.*

The traditional complexity measures for mobile agents are the number of agents moves and the amount of time of a synchronous execution of the algorithm, where in each round, each active agent traverses an edge.

In a distributed mobile environment (G, \mathcal{E}, p) with $|V(G)| = n$, $|E(G)| = m$ and $|\mathcal{E}| = k$, when executing Algorithm 1, the agents detect with $O(mk)$ moves in time $O(m)$ if the election problem can be solved; if it is possible, they successfully elects a leader with $O(mn \log k)$ moves in time $O(mn)$.

References

1. D. Angluin. Local and global properties in networks of processors. In *Proceedings of the 12th Symposium on Theory of Computing, STOC'80*, pages 82–93, 1980.
2. B. Awerbuch, M. Betke, R. Rivest, and M. Singh. Piecemeal graph exploration by a mobile robot (extended abstract). In *Proc. of the 8th annual conference on Computational Learning Theory, COLT'95*, pages 321–328. ACM Press, 1995.
3. L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Can we elect if we cannot compare? In *Proc. of the 15th annual ACM Symposium on Parallel Algorithms and Architectures, SPAA'03*, pages 324–332. ACM Press, 2003.
4. L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Rendezvous and election of mobile agents: Impact of sense of direction. *Theory of Computing Systems*, to appear.
5. M. Bender and D. Slonim. The power of team exploration: Two robots can learn unlabeled directed graphs. In *Proc. of the 35th annual Symposium on Foundations of Computer Science, FOCS'94*, pages 75–85, 1994.
6. P. Boldi, B. Codenotti, P. Gemmel, S. Shammah, J. Simon, and S. Vigna. Symmetry breaking in anonymous networks: Characterizations. In *Proc. 4th Israeli Symposium on Theory of Computing and Systems*, pages 16–26. IEEE Press, 1996.
7. P. Boldi and S. Vigna. Fibrations of graphs. *Discrete Math.*, 243:21–66, 2002.
8. L. Bougé. On the existence of symmetric algorithms to find leaders in networks of communicating sequential processes. *Acta Informatica*, 25(2):179–201, 1988.
9. J. Chalopin and Y. Métivier. A bridge between the asynchronous message passing model and local computations in graphs (*extended abstract*). In *Proc. of Mathematical Foundations of Computer Science, MFCS'05*, volume 3618 of *LNCS*, pages 212–223, 2005.
10. S. Das, P. Flocchini, A. Nayak, and N. Santoro. Distributed exploration of an unknown graph. In *Proc. of the 12th international colloquium on Structural Information and Communication Complexity, SIROCCO'05*, volume 3499 of *LNCS*, pages 99–114, 2005.
11. A. Dessmark, P. Fraigniaud, and A. Pelc. Deterministic rendezvous in graphs. In *Proc. of the 11th annual European Symposium on Algorithms, ESA'03*, volume 2832 of *LNCS*, pages 184–195, 2003.
12. C. Palamidessi. Comparing the expressive power of the synchronous and the asynchronous π -calculus. *Mathematical Structures in Computer Science*, 13(5):685–719, 2003.
13. M. Yamashita and T. Kameda. Computing on anonymous networks: Part i - characterizing the solvable cases. *IEEE Transactions on parallel and distributed systems*, 7(1):69–89, 1996.