Can We Elect If We Cannot Compare?

Lali Barrière[*] Paola Flocchini[†] Pierre Fraigniaud[‡] Nicola Santoro[§]

ABSTRACT

The aim of this paper is to study the computational power of the *qualitative* model, where entities are given distinct labels which are however mutually incomparable; this model is opposed to the *quantitative* model, where labels are integers. The qualitative model captures, for example, the case when the labels are written in different alphabets (e.g., Cyrillic, Latin) and there is no a priori agreement on a common encoding. We investigate the qualitative model through the problem of leader election in a distributed mobile environment. All known leader election protocols assume that the initial input values are distinct and pairwise comparable. While *distinctness* of the input values is clearly required, the *comparability* assumption is questionable. Our concern is whether it is possible to remove this comparability assumption. To focus solely on this concern, we consider the problem in its weakest setting: anonymous highly symmetric networks (i.e., Cayley graphs). In this way, to break the symmetry (and thus elect a leader) among the incomparable mobile agents, we can not rely on the existence of distinguished node labels nor on any topological asymmetry of the network. We describe a generic election protocol which is effective for all anonymous Cayley graphs; i.e., it solves the election problem if the problem is solvable, otherwise it determines that the problem is not solvable. For arbitrary networks, our protocol is conditionally effective; that is, it performs election of one agent among any set of agents in any network, under some weak conditions on the network and on the initial positions of the agents. Our work is a first step toward a better understanding of the inherent differences between "quantitative computing" where parameters are taken from a total order, and "qualitative computing" where parameters are taken from a partial order.

Categories & Subject Descriptors: F.0, C.2.4.
General Terms: Algorithms, Theory.
Keywords: Mobile Agents, Distributed Computing, Election, Anonymous Networks, Cayley graphs, Rendez-Vous.

1. INTRODUCTION

Assume that one wants to elect the chair of an international organization, chosen among the representatives of all nations belonging to this organization. A possible protocol for such an election could be: (1) every representative goes to the meeting room; (2) representatives aiming to be elected write their names on the whiteboard; (3) the representative whose name arrives first in alphabetic order is elected. Beside the philosophical or sociological aspects, this protocol has a major drawback: it may fail. There are at least two reasons for the failure of the protocol. The first reason is obvious: there might be different representatives with the same name. In this case, the protocol fails because it assumes the existence of a set of *labels* (here the names of the representatives) such that all entities participating to the election have pairwise *distinct* labels.

The second reason for failure is a bit more subtle: what about if, after phase 2 of the protocol, the whiteboard displays names written in Latin, Arabic, Hebrew and Greek alphabets, and even names displayed with Chinese or Japanese characters. In this case, all labels may be distinct, but still the protocol may fail because it does not only assume the existence of a set of pairwise distinct labels, but it also assume *comparability* between these labels. Of course, as far as computer science is concerned, there exists a natural set of comparable labels: the binary strings. Encoding labels with binary strings would provide a set of labels supporting distinctness and comparability, only if this encoding is *one-to-one*. However, such an encoding might not exist. This is currently the case with names written using totally different systems of characters. This informal example illustrates a situation in which a protocol which seems obviously correct actually relies on the assumption that labels are distinct *and* comparable. Electing the chair of an international orga-

[*]Departament de Matemàtica Aplicada IV, Universita Politècnica de Catalunya, Barcelona, Spain. Email: lali@mat.upc.es. Additional support from the French-Spanish research program PICASSO, Carleton University, and University of Ottawa.

[†]School of Information Technology and Engineering, University of Ottawa, Ottawa, Canada. Email: flocchin@site.uottawa.ca.

[‡]CNRS, Laboratoire de Recherche en Informatique, Université Paris Sud, Orsay, France. Email: pierre@lri.fr. Additional support from the French-Spanish research program PICASSO, Carleton University, and University of Ottawa.

[§]School of Computer Science, Carleton University, Ottawa, Canada. Email: santoro@scs.carleton.ca. Additional support from the Natural Sciences and Engineering Research Council (NSERC), NATO, and Nortel Networks.

SPAA'03, June 7–9, 2003, San Diego, California, USA.

nization may actually require a more sophisticated protocol since comparability of the labels may not be assumed. Actually, if there is an agreed-upon meeting room, then another simpler election protocol could be: the first representative who writes his or her name on the whiteboard of the meeting room is elected. However, if there is no such an agreed-upon meeting room, e.g., if the building is *anonymous*, then the problem becomes difficult because the representatives do not even know where to gather.

So, let us turn our attention to computing in anonymous networks. In such networks, nodes are unlabeled. However, the links incident to each node are given distinct local labels between 1 and d where d is the degree of the node. The argument supporting the need for such an edge-labeling is often stated as (see, e.g., [2, 3, 8, 13, 16, 22]): "if links would be unlabeled, then one could not distinguish them". This affirmation is correct. However, distinctness is one thing, and comparability is another. For instance, the streets out-going from the $A\kappa\rho o\pi o\lambda\iota\sigma$ (Akropolis), are labeled with labels from a totally ordered set, but this total order remains unknown to a tourist not aware of the Greek alphabet. Still, any tourist in Athens can distinguish among those streets and can give *relative* or *local* comparable labels to the streets, e.g., "the second to the right". However, this labeling is not absolute nor it is fixed in advance. Hence, although it makes sense to enforce distinctness of link-labels at each node of an anonymous network, there is no formal reason to assume these labels to be mutually comparable. Instead, one could simply assume a local edge-labeling using distinct symbols such as colors or geometric figures[1].

The aim of this paper is to study the computational power of the *qualitative* model, where entities are given distinct labels, called *colors*, which are however mutually incomparable: two colors can be distinguished but no other order relationship derived. This model is opposed to a the *quantitative* model, where labels are values. We investigate the qualitative model through the problem of leader election in a distributed mobile environment. Indeed, all leader election protocols we are aware of assume either that the initial labels are pairwise comparable, or that there exists an encoding of the labels providing pairwise comparable codes. That is, all protocols assume the existence of a total order on the set of labels (this total order being either inherent to the label set, or provided by an appropriate encoding of the labels). Our concern is whether it is possible to get rid of the comparability assumption, to eventually solve the election problem in the international organization mentioned before.

Incomparable labels are not used in practice, as our computational world is a quantitative one. However, our investigation can still have a significant practical impact for quantitative worlds. Consider for instance an environment where input values are both distinct and comparable but there is no *a priori* agreement among the agents on the comparability criteria; e.g., some agents might prefer the decreasing ordering while others the increasing one, while others might have yet a different ordering criteria. It is for example known that the properties of distributed optimization algorithms for the coordination of multi-agent systems depend on the underlying preference structures of the agents [12], which might be unknown *a priori*. Another setting in which our investigation can find applications is when every agent does not want to let its full name known to the other agents, or when agents want to keep their name private. Still leader election should work in this context.

[1] There is an underlying total order on the colors, induced by their wavelength. However, ordering colors using this total order is not an obvious task for most people. B.t.w., which color is smaller: brown or magenta?

1.1 Quantitative computing

At an abstract level, a *distributed mobile environment* can be described as a collection $\mathcal{E}$ of autonomous mobile entities located in a spatial universe $\mathcal{U}$. The entities have computing capabilities, exhibit the same behavior (i.e., execute the same protocol), and can move in $\mathcal{U}$ (their movement is constrained by the nature of $\mathcal{U}$). Depending on the context, the entities are sometimes called mobile *agents*, other times *robots*. In the following we shall use these terms interchangeably. In the *mobile computing* setting, two main models have been considered: the *geometric world* where the universe is assumed to be the set $\mathbb{R}^2$ of points in the plane, and the entities to be autonomous mobile robots, and the *graph world* where the universe is a simple graph G, and the entities software agents in a network. This paper will focus on this latter model, i.e., graph world. More precisely, we consider anonymous networks, i.e., simple connected undirected graphs $G = (V, E)$ whose n nodes are unlabeled. The $d = \deg(x)$ incident edges of node x are labeled by d pairwise distinct symbols. These symbols are chosen from an *a priori* unknown set of symbols. Every edge hence receives two labels, one for each of its extremities. We denote $\ell_x(e)$ the label at x of the edge e incident to x.

Communication between agents is achieved through writing of signs on whiteboards, i.e., local storages where agents can read, write (and erase) signs. There is one whiteboard per node, and access to a whiteboard is done in mutual exclusion. The concept of whiteboard is an abstraction for the use of local memory space available at each processor of a network. The agents execute the same protocol, and can move from node to node along the links of G. The initial position of an agent is called its *home-base*. For the sake of clarity, we assume that there cannot be more than one agent initially placed at a node. However, all our results extend to the case where more than one agents can occupy a single node when the protocol starts. Let A be a set of $r \leq n$ mobile agents located at different nodes of G. Let $p : A \to V(G)$ be the injection describing the placement of the agents in G. Hence, given $a \in A$, $p(a)$ is the home-base of agent a. The agents are said *placed* by p in G.

Let C be a set of r mutually incomparable elements, called *colors*. I.e., for any $x, y \in C$ it can only be determined whether they are equal or different. Every agent $a \in A$ is assigned a distinct color. Let $c : A \to C$ be the bijection assigning colors to the agents. Again, C is *a priori* unknown to the agents. Each agent a initially knows its color $c(a)$ only. In a qualitative graph world colored by C, the basic unit of information is the *colored sign*, i.e., a string of bits with a color. The home-base of $a \in A$ is marked with a sign of color $c(a)$; the sign is the same for all home-bases, only the colors of the signs differ. Agents exchange information through the whiteboards only. Again, an agent can write on the whiteboards signs colored by its own color. It can read colored signs from the whiteboards, and it is able to distinguish colors and to produce its own encoding of these colors (i.e., if it sees red signs on two different whiteboards, it will know that they are of the same 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. The actions of an agent a at a node x depends on the current state of a, and on local information available at x, such as the label of the input port through which a entered x, the content of x's whiteboard, and the degree of x. According to these information, a may decide to access x's whiteboard, to leave x (through an edge whose port label may result from some computation), or to stay at x, for instance waiting for the arrival of another agent.

1.2 The election problem

The agents aim to elect one of them as a leader. Here we don't assume that each agent is endowed with a numerical input value, but consider the case where the input value is the agent's color (thus distinguishable but not comparable). In particular, when two agents meet, the nature of the colors alone is not sufficient to break the symmetry. We focus on *generic* protocols, i.e., initially, the agents are unaware of G's topology, nor of its size. They are also unaware of their numbers; i.e., G, n, and r are all *a priori* unknown to the agents. Among generic solutions, we would like to design *universal* protocols, that is deterministic solutions which solve the problem in every network.

DEFINITION 1.1. *A generic election protocol $\mathcal{P}$ is* universal *if, for any network $G = (V, E)$, and any initial placement p of the agents, $\mathcal{P}$ elects one agent as a leader.*

If agents are labeled with *distinct* elements that are *comparable* (i.e., from an *a priori* known totally ordered set), then there is a universal election protocol. This universal election protocol implements the strategy of the protocol described at the very beginning of the paper, without an agreement on the meeting room. It performs in two phases. During phase 1, every agent performs a traversal of the graph to collect all agent labels. During phase 2, every agent elects the agent of maximum label as the leader. However, universality is a demanding concept if agents are unlabeled or labeled with incomparable labels. For instance, in the complete network of two nodes K_2, with one agent sitting at each node, there is no way to break the symmetry. Therefore, we rather focus our attention on *effective* protocols, that is deterministic solutions that solve the problem if the problem is solvable, and otherwise determine that it is not solvable. More precisely, following the guidelines of [22], given an election protocol $\mathcal{P}$, we denote by $\mathcal{I}(\mathcal{P})$ the set of input pairs (G, p) for which $\mathcal{P}$ successfully elects a leader among agents placed by p in G, no matter the edge-labeling is. Let $\mathcal{D}$ be the set of pairs (G, p) which belong to $\mathcal{I}(\mathcal{P})$ for at least an election protocol $\mathcal{P}$.

DEFINITION 1.2. *A generic election protocol is* effective *if it elects a leader for every input in $\mathcal{D}$.*

Note that, as for quantitative computing, the protocols should be independent of the edge-labeling of the networks. Indeed, the role of edge-labeling in anonymous networks is just to allow the agents to make a distinction between the edges that are incident to a same node. Specific edge-labeling may introduce, or break symmetries in the graph. Effective protocols are not allowed to use such asymmetries. In other words, they must complete even if the edge-labeling has been maliciously chosen by an adversary. The impact of specific kinds of edge-labeling (e.g., compass-compatible, sense-of-direction, etc.) for the protocol design is beyond the scope of this paper.

Observe that effective election protocol cannot exist if both the agents and the network are anonymous. Indeed, assume for the purpose of contradiction that such an effective protocol $\mathcal{P}$ does exist. Then consider the two inputs (G_1, p_1) and (G_2, p_2). G_1 is the ring of 3 nodes, and there is one agent. G_2 is the ring of 6 nodes, and two agents are initially placed at distance 3 from each other. Consider a synchronous scheduler: in each execution, all agents perform their instructions and moves simultaneously. Moreover, even if the agents have no notion of left and right, the scheduler only allows executions in which two agents in the same state, and which have to move, perform their move in the same direction in the ring. An agent a executing $\mathcal{P}$ behaves the same in G_1 and G_2. That is, a cannot distinguish whether it is alone in G_1 or with a companion in G_2, although election is possible for the former instance (the protocol is: "I am the leader"), and not for the latter (the scheduler described above implies that both agents will always remain in the same state, and hence none of them can become the leader). Hence, $\mathcal{P}$ is not effective, a contradiction. Now, this counter example does not apply in the qualitative graph world, because the two instances (G_1, p_1) and (G_2, p_2) can be discriminated by the agents thanks to their distinct colors. We therefore rephrase our problem as follows: does it exist an effective election algorithm in a qualitative graph world?

The impossibility of distinguishing nodes in an anonymous network yields symmetry in computations and restricts the computational power of the agents. Breaking the symmetries can be achieved using the asymmetries of the network. For instance, election is trivial in a star: all agents go to the central node, and the first node which writes on the whiteboard is elected (recall that the access to a whiteboard is done in mutual exclusion). However the problem becomes difficult when the network itself is very symmetric. We therefore focus our attention on highly symmetric networks, namely on Cayley graphs. Cayley graphs form a rich class of graphs, including most of the usual models for structured interconnection networks [15, 17], e.g., complete graphs, cycles, hypercubes, multi-dimensional toroidal meshes, Cube-Connected-Cycles, wrapped Butterflies, Star-graphs, circulant graphs, etc. The Cayley graph $\mathrm{Cay}(\Gamma, S)$ is described by a group Γ and a generating set S, with $S = S^{-1}$. The nodes of $\mathrm{Cay}(\Gamma, S)$ are the elements of Γ, and there is an edge $\{a, b\}$ if and only if $b^{-1}a \in S$. For instance, the n-node cycle C_n is the Cayley graph $(\mathbb{Z}_n, \{+1, -1\})$, and the d-dimensional hypercube Q_d is the Cayley graph $(\mathbb{Z}_2 \times \mathbb{Z}_2 \times \cdots \times \mathbb{Z}_2, \{(1, 0, 0, \ldots, 0), (0, 1, 0, \ldots, 0), \ldots, (0, 0, 0, \ldots, 1)\})$. In the former case, the group operation is the addition modulo n, and in the latter the bitwise xor. All Cayley graphs are vertex-transitive graphs (i.e., graphs that "look the same" from every vertex). The converse does not hold. However, any vertex-transitive graph G is a quotient of a Cayley graph. Moreover, it is conjectured (see [5]) that almost all vertex-transitive graphs are Cayley graphs.

1.3 Our results

We first give a necessary condition for an election protocol to be effective in anonymous networks. Then we de-

Agents	Universal	Effective	
		Arbitrary	Cayley
Anonymous	No	No	No
Qualitative	No	?	Yes
Quantitative	Yes	Yes	Yes

Table 1: Summary of the results of this paper.

scribe a generic election protocol which is effective for all anonymous Cayley graphs. Actually, our election protocol applies not only to Cayley graphs. For instance, it is also effective for the class of trees. Actually, it can be run on any network, in which case, it is conditionally effective; that is, it elects one agent under some weak conditions on the network and on the initial positions of the agents. These results are partially summarized in Table 1. In this table, the row "anonymous" corresponds to the case where the agents are unlabeled; the row "qualitative" corresponds to the case where the agents are given distinct but incomparable labels; and the row "quantitative" corresponds to the case where the agents are given comparable labels. In all three cases, the nodes are unlabeled.

Notice that, once a leader is elected, many other computational tasks become straightforward. Such is the case for the *gathering* or *rendez-vous* problem. Our work is a first step toward a better understanding of the inherent differences between "quantitative computing" where parameters are taken from a total order, and "qualitative computing" where parameters can be incomparable (i.e., taken from a partial order), though distinguishable.

Remark: Even if our election protocol can be proved effective for Cayley graphs only (which is yet a large class of graphs), it applies to every graph, and is not limited to run only on the family of Cayley graphs.

1.4 Related Works

This investigation is part of an ongoing research effort on understanding the algorithmic limits of computing with mobile agents. Among the problems being attacked, *network exploration* in all its variants (e.g., search, map drawing, map verification, etc.) has definitely played a central role (e.g., see [1, 4, 7, 9, 10, 11, 21]). As for the election problem, its study in environments other than the traditional distributed ones is currently being carried out. Such is the case for radio networks, ad hoc networks, etc. (e.g., see [14, 18, 19]). However, they all assume comparable labels.

2. A NECESSARY CONDITION FOR DETERMINISTIC ELECTION

We consider bi-colored graphs $G = (V, E)$, i.e., graphs whose nodes are either black or white. The placement function p defines the coloring. Black nodes correspond to home-bases, and white nodes to those nodes that are initially not occupied by agents. Now, recall that an *isomorphism* from G_1 to G_2 is a one-to-one and onto mapping $\phi : V(G_1) \to V(G_2)$ satisfying: $\{x, y\} \in E(G_1) \Leftrightarrow \{\phi(x), \phi(y)\} \in E(G_2)$. If $G_1 = G_2$, then isomorphisms are called *automorphisms*. Finally, two nodes x and y of G are *equivalent* if there is an automorphism ϕ such that $\phi(x) = y$. All morphisms considered in this paper must preserve the coloring of nodes, thus they are always *color-preserving* morphisms, and two equivalent nodes have always the same color, i.e., black or white (not to be confused with the colors of the agents' home-bases). Assume an edge-labeled graph $G = (V, E)$ is given. A *label-preserving* automorphism is an automorphism ϕ satisfying: For every $x \in V$, $\ell_{\phi(x)}(\{\phi(x), \phi(y)\}) = \ell_x(\{x, y\})$ for every neighbor y of x. We say that two nodes x and y are *label-equivalent* if there is a label-preserving automorphism ϕ such that $\phi(x) = y$. We denote by $x \sim_{lab} y$ the fact that x and y are label-equivalent. (Again, all morphisms must preserve the color of the nodes. Hence a label-preserving morphism preserves both edge-labels and node-colors.) One can check that all equivalence classes of $\sim_{lab}$ have the same size.

THEOREM 2.1. *If there exists an edge-labeling of (G, p) such that the label-equivalence classes have size > 1 then election in (G, p) is impossible.*

PROOF. Recall that the *view* [22] of an edge-labeled graph G from a node v is the infinite labeled rooted tree $\mathcal{V}(v)$ defined as the union of the edge-labeled paths originated at v in G. This notion can be trivially extended to bi-colored graphs, and then the views are bi-colored trees. The view can be alternatively recursively defined as follows. $\mathcal{V}^{(0)}(v)$ is v, and for $k \geq 1$, $\mathcal{V}^{(k)}(v)$ is the tree rooted at v and having $\deg(v)$ subtrees $\mathcal{V}^{(k-1)}(w_i)$, one for each neighbor w_i of v in G. The edge from the root of $\mathcal{V}^{(k)}(v)$ to the root of $\mathcal{V}^{(k-1)}(w_i)$ has the same two labels as the edge $\{v, w_i\}$ in G. We denote by $x \sim_{view} y$ the fact that x and y have the same view, i.e., there is a label-preserving isomorphism between $\mathcal{V}(x)$ and $\mathcal{V}(y)$. From Norris [20], in an n-node graph, $x \sim_{view} y$ if and only if there is a label-preserving isomorphism between $\mathcal{V}^{(n-1)}(x)$ and $\mathcal{V}^{(n-1)}(y)$. First we show that

$$x \sim_{lab} y \quad \Rightarrow \quad x \sim_{view} y \tag{1}$$

for any edge-labeled bi-colored graph G, and any two nodes x and y. For that purpose, let ϕ be a label-preserving automorphism of G such that $\phi(x) = y$. Let $\psi : \mathcal{V}^{(n-1)}(x) \to \mathcal{V}^{(n-1)}(y)$ defined as follows. Any node u of $\mathcal{V}^{(n-1)}(x)$ is uniquely defined by a sequence of labels $i_1, \ldots, i_k$, $k \leq n - 1$, in the sense that starting from the root of $\mathcal{V}^{(n-1)}(x)$, and traversing downward edges of successive labels $i_1, \ldots, i_k$ leads to u by a shortest path in $\mathcal{V}^{(n-1)}(x)$. Since ϕ is label-preserving, starting from x (resp., y) and traversing edges of successive labels $i_1, \ldots, i_k$ in G constructs a path P_x (resp., P_y) of G. P_x and P_y belongs to $\mathcal{V}^{(n-1)}(x)$, and $\mathcal{V}^{(n-1)}(y)$, respectively. Let v be the extremity of P_y in $\mathcal{V}^{(n-1)}(y)$. We define $\psi(u) = v$. Obviously, ψ is a label-preserving isomorphism from $\mathcal{V}^{(n-1)}(x)$ to $\mathcal{V}^{(n-1)}(y)$.

It is shown in [22] that all equivalence classes of G for $\sim_{view}$ have the same size $\sigma_\ell(G)$ which depends on the edge-labeling ℓ of G. As a direct consequence of Equation 1, we get that $\sigma_\ell(G)$ is at least as large as the size of the equivalence classes of $\sim_{lab}$ for the edge-labeling ℓ.

Now, we prove that there is a universal transformation from an election protocol for mobile agents in an anonymous network G to a distributed election protocol for the anonymous processor-network G. Here is the transformation. All processors execute the same program. The memory of a processor is its whiteboard W. A message is an agent, and is of the form (P, M) where P is the program of the agent, and M is the memory content of the agent. A processor executes the

Repeat:

- Wait for reception of a message;
- Upon reception of a message (P, M), execute P with data M and W;
- If the execution leads to a move through the edge labeled i, then send a message (P, M') through the edge labeled i, where M' is the memory content of the agent when it leaves the node;

Figure 1: Transformation of a protocol for mobile agents into a protocol for processor-network

instructions specified on Figure 1. Initially, when an agent wakes up, the corresponding processor (i.e., the home-base of the agent) wakes up and starts executing the program from the second instruction, as if it would have received a message. Obviously, by this transformation, a mobile election protocol in the anonymous network G is transformed into a distributed election protocol for the processor network G.

Assume now that there exists an edge-labeling ℓ of G such that all label-equivalence classes of (G, p) have size d. Therefore, $\sigma_\ell(G)$ is at least as large as d. Yamashita and Kameda [22] defined the *symmetricity* of a network H as: $\sigma(H) = \max\{\sigma_\ell(H), \ell \text{ edge-labeling of } H\}$, and showed (Lemma 7 in [22]) that if election is possible in an anonymous processor network then the symmetricity of that network is 1. Yamashita and Kameda's theory applies in our context because their impossibility result is valid even with complete knowledge, that is all nodes are given the knowledge of the network topology, and their location in the network. This is our setting since agents are able to traverse the network and to build a map of it. By our transformation, we get that if there is an election protocol for mobile agents in G then there is a distributed election protocol for the processor-network G. Therefore, if $d > 1$ then $(G, p) \notin \mathcal{D}$. $\Box$

Remark. Observe here a major difference between quantitative and qualitative computing. In the former, labels are chosen from the set of integers. As a consequence, one can fix *a priori* an arbitrary ordering of the views, and this ordering gives a way to elect a leader, provided that the symmetricity of the graph is 1. For instance, consider the path $\{x, y, z\}$ with $\ell_x(\{x, y\}) = 1$, $\ell_y(\{x, y\}) = 1$, $\ell_y(\{y, z\}) = 2$, and $\ell_z(\{y, z\}) = 1$. All the views are different, and each view can be represented by, e.g., an array of integers. It results from this observation that an order on the views can be obtained from an order on the arrays of integers. Actually, the condition of Theorem 2.1 is also a sufficient condition in quantitative computing (see [22]). The situation is more complex in the qualitative graph world. Indeed, labels are chosen from an *a priori* unknown partially ordered set. This does not allow to fix an *a priori* common ordering of the views, and hence the design of an election protocol becomes more difficult, even when the symmetricity of the graph is 1. For instance, consider the same example as before, i.e., the path $\{x, y, z\}$. Assume moreover, according to the qualitative model, that $\ell_x(\{x, y\}) = *$, $\ell_y(\{x, y\}) = \circ$, $\ell_y(\{y, z\}) = \bullet$, and $\ell_y(\{y, z\}) = *$. All the views are different, as in the example for the quantitative graph world. However, since there is no *a priori* order on the symbols $*, \circ$, and $\bullet$, one cannot order the views. Another way to realize the difficulty of ordering the views is to act as an agent would do. For instance, agent a_x starting from x and traversing the graph towards z will see the sequence $*, \circ, \bullet, *$, that could be coded $1, 2, 3, 1$ by applying the rule consisting to code i the ith symbol met so far. Similarly, an agent a_z starting from z and traversing the graph toward x will see the sequence $*, \bullet, \circ, *$, that will also be coded $1, 2, 3, 1$ by a_z applying the same rule as a_x. The two agents a_x and a_z have therefore the same view of the network, once the views are coded by integers. So election cannot be performed by just sorting the views.

3. AN ELECTION PROTOCOL FOR THE QUALITATIVE GRAPH WORLD

Let $d(\cdot, \cdot)$ be the distance function in a graph. Our election protocol uses the following notion. The *surrounding* of a node u in a bi-colored network $G = (V, E)$ is the directed graph $\mathcal{S}(u) = (V, X)$ defined on the same set of nodes as G, with the same node-coloring, and such that $(x, y) \in X \Leftrightarrow (\ \{x, y\} \in E \text{ and } d(u, x) \leq d(u, y)\ )$. In other words, the out-neighbors of a node x in $\mathcal{S}(u)$ are all the neighbors y of x in G such that $d(u, y) = d(u, x)$ or $d(u, y) = d(u, x) + 1$. Note that u is the unique node with in-degree 0 in $\mathcal{S}(u)$. We prove the following:

LEMMA 3.1. *There is a deterministic algorithm which computes, for any bi-colored graph, a total ordering $\prec$ for its equivalence classes.*

PROOF. We first prove that two nodes of a bi-colored network G are equivalent if and only if their surroundings are isomorphic. Assume u and v are equivalent in G, i.e., there exists an automorphism ϕ of G mapping u to v. ϕ trivially extends to an isomorphism between $\mathcal{S}(u)$ and $\mathcal{S}(v)$. Conversely, let ψ be an isomorphism between $\mathcal{S}(u)$ and $\mathcal{S}(v)$. Since u and v are the unique nodes with in-degree 0 in $\mathcal{S}(u)$ and $\mathcal{S}(v)$ respectively, ψ maps u to v. ψ trivially extends to an automorphism of G, and hence u and v are equivalent in G.

Next we prove that there is a deterministic algorithm which computes a total ordering of the directed graphs. Let G be any n-node digraph, and let M be an adjacency matrix of G, i.e., nodes are arbitrarily labeled from 1 to n, and there is a 1-entry at row i and column j if and only if node j is an out-neighbor of node i, and there is a 0-entry otherwise. Let $w(M)$ be the boolean word on length n^2 obtained by concatenating the n rows of M: row 1, then row 2, etc. Every different labeling of the nodes from 1 to n corresponds to a permutation π of the rows and the columns of M. The resulting matrix is denoted by $\pi(M)$. Let $w(G)$ be the boolean word of length n^2 defined as the minimum, taken over all permutations π of n symbols, of $w(\pi(M))$. Obviously, $w(G) = w(G')$ if and only if G and G' are isomorphic. Therefore, the lexicographic order on the boolean words of length n^2 produces a total ordering of the n-node directed graphs. Digraphs of different order are just ordered according to their number of vertices.

We extend this construction to bi-colored digraphs to show that there is a deterministic algorithm which computes a total ordering of the bi-colored digraphs. For that purpose, we define a *hair* of a graph G as a maximal path $x_0, x_1, \ldots, x_k$ in G such that $\deg(x_i) = 2$ for every i, $0 < i < k$, and $\deg(x_k) = 1$. We first order the bi-colored digraphs by their number of vertices, and then by the maximum length of their hairs (this maximum length is zero if G has no hair). The bi-colored digraphs of the same order, and having hairs of the same maximum length are ordered as follows. Let G and G' be two non color-isomorphic bi-colored digraphs of hairs of maximum length k. G and G' are transformed into two non isomorphic uni-colored digraphs $\widehat{G}$ and $\widehat{G'}$ by replacing every black node by a white node attached to a path of white nodes of length $k+1$. The digraphs G and G' are then ordered according to the ordering of $\widehat{G}$ and $\widehat{G'}$ expressed before.

Therefore, two non-equivalent nodes u and v have two non-isomorphic surroundings $\mathcal{S}(u)$ and $\mathcal{S}(v)$. One of the two surroundings is smaller than the other according to the ordering of bi-colored graphs. □

We now sketch the description of our election protocol, called ELECT. More details can be found in [6]. Every agent uses a set of public variables that partially indicate its state. An agent is successively asleep, awake, active, possibly passive, and eventually leader or defeated. An active agent may be alternatively searching or waiting. A waiting agent is *a priori* unmatched, and can be matched by a searching agent. The meaning of this terminology will hopefully appear clear later. Agents communicate via the node whiteboards, so it is understood that the values of the public variables of an agent are readable on the whiteboard of its home-base.

Protocol ELECT proceeds in successive phases. The number of phases depends on the structure of the network, and on the initial positions of the agents. An initial phase allows each agent placed by p in a network G to draw a map of G, including the positions and the colors of the home-bases. For that purpose, marking the whiteboards and using the edge-labels, each agent performs a DFS traversal of G. During its traversal, if an agent meets a sleeping agent, then it wakes up this agent. Once the map of G has been drawn, every agent computes the surroundings of all nodes in (G,p), including those that contain home-bases of agents, and those that are initially empty. Based on these surroundings, every agent computes the equivalence classes $C_1,\dots,C_k$ of the nodes; let $C_1,\dots,C_\ell$ be those that include the home-bases of the agents. We assume $C_1 \prec C_2 \prec \cdots \prec C_\ell$ and $C_{\ell+1} \prec C_{\ell+2} \prec \cdots \prec C_k$ where $\prec$ is defined in Lemma 3.1. Observe that all agents agree on the classes C_i's, i.e., agree on which node belongs to which class, simply because they have a map of the same graph. Moreover, all agents agree on the order $\prec$ of these classes. This is a key point: although colors do not give an *a priori* order to the agents, agents are able to group themselves in subsets, and to assign a total order to these sets based on the topology of the network G and on the initial placement p of the agents.

There are at most $k-1$ other phases, called reduction phases. Each reduction phase executes one of the two variants of the same subroutine. These variants are called REDUCE1 and REDUCE2. Given two sets A and B of active agents, the goal of Subroutine REDUCE1 is to reduce the number of agents to $\gcd(|A|,|B|)$. Given a set A of active agents, and a set B of selected nodes, the goal of Subroutine REDUCE2 is to reduce the number of active agents to $\gcd(|A|,|B|)$. Basically, both routines execute Euclid's Algorithm. Their descriptions are sketched in Section 3.1. A new phase begins if the number of current active agents is larger than 1. Phases are grouped into two stages, one called "agent-agent", and the other called "agent-node". The number of phases performed in the agent-agent stage is $\ell-1$, while the number of phases performed in the stage agent-node is $k-\ell$.

Let us describe the stage agent-agent. If $\ell > 1$, then agents in C_1 and C_2 become *active*. We aim to reduce the number of active agents to $d_1 = \gcd(|C_1|,|C_2|)$. For that purpose, active agents apply the sub-routine REDUCE1. It results in a set D_1 of active agents, with $|D_1| = d_1$. If $d_1 > 1$, then a new phase is performed, aiming to reduce the number of active agents to $d_2 = \gcd(|D_1|,|C_3|)$. For that purpose, agents in D_1 start activating the agents of C_3 by visiting them. An agent in C_3 becomes active when it has been visited by all agents in D_1. Phase 2 then begins, and the selection of $d_2 = \gcd(d_1,|C_3|)$ active agents is again performed by application of the routine REDUCE1, which results in a set D_2 of active agents, $|D_2| = d_2$. If $d_2 > 1$, then yet another phase is performed. Agents in D_2 activate agents in C_4, and agents in D_2 and C_4 enter phase 3, i.e., execute the routine REDUCE1. An so on, until either some $d_i = 1$, or $\ell-1$ phases have been performed. After phase $\ell-1$, the number of active agents is reduced to $d_{\ell-1} = \gcd(|C_1|,|C_2|,\dots,|C_\ell|)$, and the stage agent-agent is completed.

In the stage agent-node, the phases perform very similarly. We are given the set of active agents resulting from the stage agent-agent, or the set C_1 of active agents if $\ell = 1$. In this latter case, we set $d_0 = |C_1|$. If $d_{\ell-1} > 1$, then phase ℓ aims to reduce the number of active agents to $d_\ell = \gcd(d_{\ell-1},|C_{\ell+1}|)$. This is performed by application of the sub-routine REDUCE2 which returns a set D_ℓ of active agents, $|D_\ell| = d_\ell$. If $d_\ell > 1$, then yet another phase is initiated, aiming to reduce the number of active agents to $d_{\ell+1} = \gcd(d_\ell,|C_{\ell+2}|)$. This phase is again performed by application of REDUCE2. An so on, until either some $d_i = 1$, or k phases have been performed in total (including the initial phase). Then, stage agent-node is completed.

It results from this sequence of phases that the number of active agents is reduced to $d_{k-1} = \gcd(|C_1|,|C_2|,\dots,|C_k|)$. If $d_{k-1} = 1$ then the unique remaining active agent is elected as leader. In that case, the leader traverses the network to let all other agents know about the color of the leader, and these other agents become defeated. If $d_{k-1} > 1$, then the remaining active agents traverse the network to inform the other agents of the failure of the election, and all agents become passive. We obtain the following result:

THEOREM 3.1. *Protocol* ELECT *elects a leader among any set of r agents placed by p in a network $G=(V,E)$, provided that* $\gcd(|C_1|,\dots,|C_k|) = 1$ *where the C_i's are the equivalence classes of (G,p). Moreover, if* $\gcd(|C_1|,\dots,|C_k|) > 1$ *then* ELECT *lets the agents know about the failure of the election.*

Remark. One can check that the total number of moves and whiteboard accesses performed by the agents is $O(r|E|)$.

3.1 Sketch of procedures Reduce1 and Reduce2

In REDUCE1, agents are split in *searching* and *waiting* agents. (Initially, one of the two input sets forms the set of searching agents, and the other the set of waiting agents.) Waiting agents stay in their home-base, and wait to be visited by all the searching agents. If a searching agent matches with it, it becomes *matched.* When a matched waiting agent has been visited by all the searching agents, it becomes *passive.* The searching agents performs a traversal of the network looking for waiting agents. The first time a searching agent visits the home-base of an unmatched waiting agent, it matches with it. A passive agent, i.e., a matched waiting agent, remains at its home-base. (It will remain idle until it will be shoulder tapped at the very end of the protocol ELECT.)

At every round of REDUCE1, the set of searching and waiting agents changes according to the following rule. If we denote by S, W, and P the current sets of searching, waiting, and passive agents, then, according to Euclid's algorithm, the sets of searching and waiting agents for the next round are respectively S and $W \setminus P$ if $|W| - |S| \geq |S|$, and $W \setminus P$ and S otherwise. REDUCE1 stops when $|S| = |W|$, and then returns the set S.

The subroutine REDUCE2 performs roughly the same as REDUCE1. The major difference between them is that REDUCE1 involves matchings between two agents whereas REDUCE2 involves matchings between one agent and one node. We describe hereafter how matchings between α agents and β nodes can be performed, $\alpha \neq \beta$. Two cases are considered:

- Case 1: $\alpha > \beta$. Then let q and ϱ such that $\alpha = q \cdot \beta + \varrho$, $0 < \varrho \leq \beta$. Each of the α agents traverses the network in order to "acquire" one of the β nodes. Each of the β nodes can be acquired by q agents. I.e., as soon as an agent a that has not yet acquired any node visits a node b that has not yet been acquired by q other agents, a acquires b. Agents that have acquired a node become passive, and the remaining ϱ agents remain active.
- Case 2: $\alpha < \beta$. Then let q and ϱ such that $\beta = q \cdot \alpha + \varrho$, $0 < \varrho \leq \alpha$. Each of the α agents traverses the network in order to acquire q of the β nodes. A node can be acquired by one agent only. Only the ϱ nodes that are not acquired remain selected.

These operations are repeated until the number of active agents is equal to the number of selected nodes, and then REDUCE2 returns the current set of active agents.

4. EFFECTIVENESS OF PROTOCOL ELECT IN CAYLEY NETWORKS

To make protocol ELECT effective for the class of Cayley graphs, we slightly modify it as follows: during the first phase, once the map of the network G has been computed by the agents, they test whether G is a Cayley graph (it is time-consuming, but decidable). If G is Cayley, then the agents carry on ELECT using the equivalence classes under *translations* instead of the usual equivalence classes. Recall that a translation is an automorphism of $\mathrm{Cay}(\Gamma, S)$ of the form $a \mapsto \gamma a$ with $\gamma \in \Gamma$. Importantly, note that, since every agent executes the same protocol, if the network is Cayley, then agents select isomorphic groups, and hence all nodes agree on the translation-classes. Note also that the translations act transitively on $\mathrm{Cay}(\Gamma, S)$, and the translation which sends x to y is $\gamma = yx^{-1}$. However, the black and white coloring of the nodes creates different classes. Two nodes that are equivalent under translation are called translation-equivalent. Two equivalent nodes may not be translation-equivalent. For instance, consider the cycle $\mathrm{Cay}(\mathbb{Z}_n, \{+1, -1\})$, n even, with two agents placed at nodes 0 and $n/2$. Nodes 1 and $n/2 - 1$ are equivalent (the automorphism is a symmetry), but are not translation-equivalent because $a \mapsto a + (n/2 - 2) \bmod n$ sends 0 to $n/2 - 2$, and these two nodes are of different colors. We prove the following:

THEOREM 4.1. *Protocol* ELECT *is effective for the class of Cayley graphs.*

PROOF. Let $G = \mathrm{Cay}(\Gamma, S)$ be a Cayley graph, and p be a placement of agents in G. Let $C_1, \ldots, C_k$ be the translation-equivalence classes of (G, p).

Assume that $\gcd(|C_1|, \ldots, |C_k|) = d > 1$, and let us show that election is then impossible. The elements of the generating set S induces a natural edge-labeling of $\mathrm{Cay}(\Gamma, S)$: $\ell_x(x, y) = x^{-1}y \in S$. We claim that the label-equivalence classes of ℓ have size d. Note that, for any $\gamma \in \Gamma$, the translation γ preserves the labeling since $(\gamma x)^{-1}(\gamma y) = x^{-1}y$. We proceed iteratively, by refining the translation-equivalence classes to create label-equivalence classes. At each iteration, we mark with two elements in S the two extremities of some set of yet unmarked edges. Initially, all edges are unmarked. At the end of the process, all edges are marked, and the mark at the extremity x of an edge e is the label $\ell_x(e)$. At a current state of the process, we say that two nodes are *pseudo* label-equivalent if they are label-equivalent relatively to the currently marked edges. Initially, all nodes of a same translation-equivalence class are pseudo label-equivalent since no label is taken into account. An invariant of our process is that, if an edge between two nodes of two pseudo label-equivalence classes C and C' has been marked, then $|C| = |C'|$. A new iteration starts if not all the pseudo label-equivalence classes have the same size. If so, take two pseudo label-equivalence classes C and C', of different size, and linked by some edges. (Since S is a generating set, $\mathrm{Cay}(\Gamma, S)$ is connected, and hence such C and C' do exist.) Assume, w.l.o.g., that $|C| < |C'|$. If the edge labeled s at $c \in C$ has its other extremity in C', then, by translation, the edge labeled s at any node of C has its other extremity in C'. Since $|C| \neq |C'|$, let s be an element of S corresponding to a not yet marked edge between C and C'. Let $Cs = \{cs, c \in C\} \subset C'$. Mark with s and s^{-1} the extremities of all edges corresponding to the generator s between C and Cs. Since translations preserve the edge-labeling, all pseudo label-equivalence classes remain so, with the exception of C', which is split into two classes Cs and $C' \setminus Cs$. Indeed, no translation can send $x = cs \in Cs$ to $y \in C' \setminus Cs$ while preserving the current labeling because the edges labeled s^{-1} at x and y have their other extremities in two different classes. Observe another invariant of our process: at any time, the gcd of the sizes of the current pseudo label-equivalence classes is equal to d. Indeed, each iteration transforms a pair (C, C') into a triple $(C, Cs, C' \setminus Cs)$ with $|Cs| = |C|$, all other classes remain of same size, and $\gcd(|C|, |Cs|, |C' \setminus Cs|) = \gcd(|C|, |C'| - |C|) = \gcd(|C|, |C'|)$, the latter equality following Euclid's Theorem. After a certain number of iterations, we end up with pseudo label-equivalence classes of same size. From the previous observation, this size is d, the gcd of the original translation-equivalence classes. If some edges remain unmarked, mark them with the corresponding element of S. Again, since translations preserve the edge-labeling all resulting pseudo label-equivalence classes remain the same, and become label-equivalence classes, all of size d. Theorem 2.1 states that election is then impossible.

If $\gcd(|C_1|, \ldots, |C_k|) = 1$, then, from Theorem 3.1, ELECT elects a leader.

Therefore, protocol ELECT is effective for the class of Cayley graphs. □

5. CONCLUDING REMARKS

We have designed the election protocol ELECT that is proved to be effective for the class of Cayley graphs in the qualitative graph world. However, there are symmetric graphs for which protocol ELECT is not effective. In particular, the Petersen graph (see Figure 2(a)) allows to construct a counter-example of ELECT's effectiveness. Consider two agents whose initial positions are marked in black on Figure 2(a). The equivalence classes are indicated on Figure 2(b). There are three classes, marked black, Grey, and white on the figure, and denoted by C_b, C_g, and C_w respectively. The bold arrows indicate an automorphism ϕ exchanging the two black nodes (i.e., home-bases). Also, ϕ exchanges two white nodes while it lets the two others fixed. Two pairs of Grey nodes are exchanged by ϕ. The dotted arrows indicate an automorphism ψ exchanging white nodes. ψ also exchanges two Grey nodes, while it lets two other Grey nodes fixed, as well as the two black nodes. Combining ϕ and ψ with the horizontal symmetry makes all white nodes equivalent, and all Grey nodes equivalent. Finally, Grey nodes are adjacent to black nodes, and hence form a class distinct from the class of white nodes.

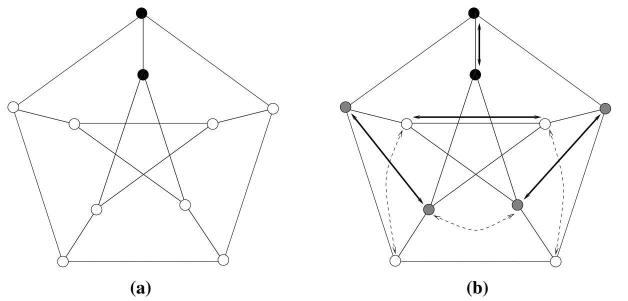

Figure 2: The Petersen graph

We have $\gcd(|C_b|, |C_g|, |C_w|) = 2$, and hence protocol ELECT does not elect a leader. However, there is a protocol, *specific* of this setting, which allows to elect a leader. Here are the actions to be performed by each of the two agents for election:

1. Wake up the other agent (if not yet awaken);
2. Go to one of your neighboring nodes, distinct from the home-base of the other agent, and mark the whiteboard of that node; (Note that the two home-bases have distinct sets of neighbors.)
3. Visit the neighboring nodes of the other agent to check which one was marked by it;
4. Try to acquire the unique common neighbor x of the two marked nodes;
5. If you acquired x then you are the leader, otherwise you are defeated.

Obviously, this sequence of actions allows to elect a leader, and hence protocol ELECT is not effective.

This counter example of ELECT's effectiveness demonstrates that we cannot always relate the sizes of the equivalence classes $C_1, \ldots, C_k$ with the size d of the label-equivalence classes. It may happen that $\gcd(|C_1|, \ldots, |C_k|) > 1$ while $d = 1$. An example is provided by the Petersen graph with two agents as described in Figure 2(a). Any edge-labeling will result in label-equivalence classes of size 1, whereas $\gcd(|C_b|, |C_g|, |C_w|) = 2$. It is interesting to note that the Petersen graph is not Cayley but vertex-transitive. Now, from Sabidussi characterization theorem, any vertex-transitive graph G is a quotient of a Cayley graph. More precisely, $G \equiv \text{Cay}(\Gamma, S)/H$ where $\Gamma = \text{Aut}(G)$, $H = \text{stab}(u_0) = \{\phi \in \Gamma \ / \ \phi(u_0) = u_0\}$, and $S = \{\phi \in \Gamma \ / \ d(\phi(u_0), u_0) = 1\}$. In other words, $G = (V, E)$ where $V = \{\phi H, \phi \in \Gamma\}$, and $\{\phi H, \phi' H\} \in E$ if and only if there are $h, h' \in H$ and $\sigma \in S$ such that $\phi' h' = \phi h \sigma$. The quotient operation seems therefore enough to destroy some of the properties of translations in Cayley graphs, enough to invalidate a generalization of Theorem 4.1 to vertex-transitive networks.

We were unable to prove or disprove the existence of an effective election protocol for arbitrary networks. We hence state explicitly the following question:

Open problem: Does it exist an effective election protocol for arbitrary graphs in the qualitative world?

6. REFERENCES

[1] S. Albers and M. Henzinger. Exploring unknown environments. In 29th ACM Symp. on Theory of Computing (STOC '97), pages 416-425, 1997.

[2] D. Angluin. Local and global properties in networks of processors. In 12th ACM Symp. on Theory of Computing (STOC '80), 82-93, 1980.

[3] H. Attiya, M. Snir, and M. Warmuth, Computing on an anonymous ring. Journal of the ACM 35:845–875, 1988.

[4] B. Awerbuch, M. Betke, and M. Singh, Piecemeal graph learning by a mobile robot. Information and Computation, 152:155–172, 1999.

[5] L. Babai, Automorphism groups, isomorphism, reconstruction. Handbook of Combinatorics, Chapter 7, 1994.

[6] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro, Distributed mobile computing with incomparable labels. Technical Report LRI-1309, Laboratoire de recherche en Informatique, Université Paris-Sud, France, 2002.

[7] M. Bender, A. Fernandez, D. Ron, A. Sahai, and S. Vadhan, The power of a pebble: Exploring and mapping directed graphs. In 30th ACM Symp. on Theory of Computing (STOC '98), pages 269–278, 1998.

[8] P. Boldi and S. Vigna, Computing anonymously with arbitrary knowledge. In 18th ACM Symp. on Principles of Distributed Computing (PODC '99), 1999.

[9] X. Deng and C. Papadimitriou, Exploring an unknown graph. Journal of Graph Theory, 32:265-297, 1999.

[10] K. Diks, P. Fraigniaud, E. Kranakis, and A. Pelc, Tree exploration with little memory. In 13th ACM-SIAM

Symp. on Discrete Algorithms (SODA '02), pages 588–597, 2002.

[11] C. Duncan, S. Kobourov, and V. Kumar, Optimal constrained graph exploration. In 12th ACM-SIAM Symp. on Discrete Algorithms (SODA '01) pages 807-814, 2001.

[12] P. Faratin and B. Van de Walle, Agent preference relations: strict, indifferent, and incomparable. In 1st ACM joint conference on Autonomous Agents and Multiple Agents Systems (AAMAS), pages 1317–1324, 2002.

[13] P. Fraigniaud, A. Pelc, D. Peleg, and S. Pérennes, Assigning labels in unknown anonymous networks. In 19th ACM Symp. on Principles of Distributed Computing (PODC '00), pages 101-112, 2000.

[14] K. Hatzis, G. Pentaris, P. Spirakis, V. Tampakas, and R. Tan, Fundamental control algorithms in mobile networks. In 11th ACM Symp. on Parallel Algorithms and Architectures (SPAA '99), pages 251–260, 1999.

[15] M.-C. Heydemann, Cayley graphs and interconnection networks. In Graph Symmetry, G. Hahn and G. Sabidussi (eds.), pages 167-224, Kluwer Academic Publishers, 1997.

[16] E. Kranakis, Symmetry and computability in anonymous networks: A brief survey. In 3rd Colloquium on Structural Information and Communication Complexity (SIROCCO '97), pages 1-16, Carleton Scientific, 1997.

[17] T. Leighton, Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes. Morgan Kaufmann, 1992.

[18] N. Malpani, J. Welch, and N. Vaidya, Leader election algorithms for mobile ad hoc networks. In 4th Int. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM), pages 96-103, 2000.

[19] K. Nakano and S. Olariu, Leader election protocols for radio networks. Chapter in Handbook of Wireless Networks and Mobile Computing, Wiley, pages 219-242, 2002.

[20] N. Norris. Universal covers of graphs: isomorphism to depth $n-1$ implies isomorphism to all depths. Discrete Applied Mathematics, 56:61–74, 1995.

[21] P. Panaite and A. Pelc, Exploring unknown undirected graphs. Journal of Algorithms, 33:281–295, 1999.

[22] M. Yamashita and T. Kameda, Computing on anonymous networks, part I: characterizing the solvable cases. IEEE Transaction on Parallel and Distributed Computing, 7(1):69–89, 1996.