Can we elect if we cannot compare?[*]

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

Abstract

The objective of this paper is to study the computational power of the *qualitative* model, where computing entities (e.g., processing nodes, mobile agents, etc.) 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 cases where there is no a priori agreement on a common encoding of the labels. We investigate the qualitative model through the problem of deterministic leader election in a distributed mobile environment. All known deterministic 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. We first give a necessary condition for election being possible in the qualitative model. We then describe a protocol that performs election of one agent among any set of agents in any network, under restrictive conditions on the network and on the initial positions of the agents. Our main result is the design of an election protocol that is proved to be *effectual* 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. 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.

Keywords: Mobile Agents, Election, Anonymous Networks, Cayley graphs.

[*]A preliminary version of this paper appeared in the proceedings of the 15th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), San-Diego, 2003.

[†]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, 91405 Orsay, France. Email: pierre@lri.fr. Additional support from the project PairAPair of the ACI "Masse de données", from Carleton University, and from 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.

1 Introduction

1.1 Qualitative Model

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: even if labels are distinct, what about if, after phase 2 of the protocol, the whiteboard displays names written in Latin, Arabic, Hebrew and Greek alphabets, or displayed with Chinese or Japanese characters. In this case, the protocol may fail because it does not only assume the existence of a set of pairwise distinct labels, but it also assumes *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, if this encoding is one-to-one and everybody agree on it. However, such an encoding might not exist. This is currently the case with names written using totally different systems of characters. (Note that even in the computer worlds, numbers may be encoded differently by different manufactories). This example illustrates a situation in which a protocol which seems obviously correct actually relies on the assumption that labels are distinct <u>and</u> comparable. Electing the chair of an international organization 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 (say "room 1402", or "the room at the North-East corner"), 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, then the problem becomes difficult because the representatives do not even know where to gather. There is a theoretical framework capturing such an absence of node-labeling: *anonymous networks*. However, in anonymous networks, the links incident to each node are traditionally 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., [3, 4, 10, 18, 23, 33]): "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 $A\kappa\rho o\pi o\lambda\eta\sigma$ (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 geometric figures, algebraic symbols (e.g., the elements of a group), or colors[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 the *quantitative* model, where labels are integer values. We investigate the qualitative model through the problem of deterministic leader election among mobile agents. 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.

Incomparable labels are not used in practice, as our computational world is a quantitative one. Our investigation is actually mostly motivated by a natural scientific curiosity, for tackling the question of computing without comparability. However, our investigation can still have a significant practical impact for quantitative worlds. In particular, a setting in which our investigation can find applications is 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 [17], which might be unknown *a priori*.

1.2 Computing in Qualitative Worlds

An environment supporting mobile agents 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 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 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., 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. Each agent is capable to distinguish these symbols, and to produce its own encoding of them. 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.

We consider the standard model of computing with mobile entities. That is 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 by assuming a fair mutual exclusion mechanism. 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

[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, and may even be not applicable if colors are not pure.

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 $p : A \rightarrow V(G)$ 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 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 $c : A \rightarrow C$ be the one-to-one function assigning colors 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, on its color, 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 (i.e., the colored signs written on the whiteboard), and the degree of x. According to these information, a may decide to access x's whiteboard, to leave x, or to stay at x, for instance waiting for the arrival of another agent.

Protocol design

Protocol design in the qualitative world significantly differs from the protocol design in the quantitative world, because a protocol in the qualitative world cannot apply the operations $<$ and $>$ to the IDs of the agents, but can only test whether two colors are equal or different. The designer of a protocol in the qualitative world must proceed without assuming comparability. He or she must design the protocol knowing that the agents will run in a qualitative world, that is will be provided with colors from a set C, but the designer does not need to know C, and thus cannot assume or compute any a priori total ordering on the set C of colors used by the agents.

1.3 An Exemplary Problem: Election

The agents are aiming at electing one of them as a leader[2]. 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., protocols that can be run independently of the network, its size, the initial position of the agents,

[2]Once a leader is elected, many other computational tasks become straightforward. Such is the case for the gathering or rendezvous problem.

and their number. Thus, 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. More precisely, 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 also comparable (i.e., from a 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 *effectual* 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 [33], we define the following:

Definition 1.1 *Given an election protocol $\mathcal{P}$, $\mathcal{I}(\mathcal{P})$ is 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. *$\mathcal{I}$ is the set of pairs (G, p) which belong to $\mathcal{I}(\mathcal{P})$ for at least one election protocol $\mathcal{P}$, i.e., $\mathcal{I} = \cup_{\mathcal{P}}\mathcal{I}(\mathcal{P})$. A generic election protocol $\mathcal{P}_0$ is effectual if it elects a leader for every input in $\mathcal{I}$, i.e., $\mathcal{I}(\mathcal{P}_0) = \mathcal{I}$.*

By definition, for any protocol $\mathcal{P}$, $\mathcal{I}(P) \subseteq \mathcal{I}$. The existence of an effectual protocol, i.e., a protocol $\mathcal{P}_0$ such that $\mathcal{I} \subseteq \mathcal{I}(\mathcal{P}_0)$, is not obvious.

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. effectual 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 an effectual election protocol cannot exist if both the agents and the network are anonymous. Indeed, assume for the purpose of contradiction that such an effectual protocol $\mathcal{P}$ does exist. Then consider the two following 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, agent a cannot distinguish whether it is alone in G_1 or with a companion b 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 effectual, a contradiction.

We therefore rephrase our problem as follows: does it exist an effectual election algorithm in a qualitative graph world (i.e., when agents are given distinct but non-necessarily comparable labels)?

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 different when the network itself is very symmetric. In this paper, we focus our attention on highly symmetric networks, namely on Cayley graphs.

Definition 1.2 *The Cayley graph Cay(Γ, S) is described by a group Γ and a generating set S, with $S = S^{-1}$. The nodes of Cay(Γ, S) are the elements of Γ, and there is an edge $\{a, b\}$ if and only if $b^{-1}a \in S$.*

Cayley graphs form a rich class of graphs, including most of the usual models for structured interconnection networks [22, 24], e.g., complete graphs, cycles, hypercubes, multi-dimensional toroidal meshes, Cube-Connected-Cycles, wrapped Butterflies, Star-graphs, circulant graphs, etc. For instance, the n-node cycle is the Cayley graph

$$C_n = \mathrm{Cay}(\mathbb{Z}_n, \{+1, -1\})$$

and the d-dimensional hypercube is the Cayley graph

$$Q_d = \mathrm{Cay}(\mathbb{Z}_2 \times \mathbb{Z}_2 \times \ldots \times \mathbb{Z}_2, \{(1,0,0,\ldots,0), (0,1,0,\ldots,0), \ldots, (0,0,0,\ldots,1)\}).$$

In the former case, the group operation is the addition modulo n, and in the latter the bitwise xor on d-dimensional binary vectors. 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 [6]) that almost all vertex-transitive graphs are Cayley graphs. Finally, Cayley graphs can be used to build explicit expanders [32].

1.4 Our results

We first give a necessary condition for election being possible in the qualitative model.

We then describe a protocol, called Elect, that performs election of one agent among any set of agents in any network, under restrictive conditions on the network and on the initial positions of the agents.

Our main result is the design of an election protocol resulting from a slight modification of Elect, that is proved to be *effectual* 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.

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

	Universal	effectual	
Agents		Arbitrary	Cayley
Anonymous	No	No	No
Qualitative	**No**	**?**	**Yes**
Quantitative	Yes	Yes	Yes

Table 1: Summary of the results of this paper: election in anonymous networks.

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, i.e., the network is anonymous.

- If the agent are anonymous, then we have seen in Section 1.3 that rings are counter examples for the existence of a universal election protocol in arbitrary networks. In fact, since rings are Cayley graphs, they are also counter examples for the existence of a universal election protocol in Cayley graphs.
- If the agents are given comparable labels (i.e., if agents belong to the quantitative world), then we have seen in Section 1.3 that there is a universal election protocol. A universal election protocol being effectual, this implies that there is an effectual election protocol in the quantitative graph world, for arbitrary graphs, and therefore for Cayley graphs in particular.
- Our contribution concerns the qualitative graph world where agents are given distinct but otherwise uncomparable labels. K_2 is a counter example for the existence of a universal election protocol in the qualitative graph world. We prove that there exists an effectual algorithm for the class of Cayley graph.

We were not able to prove or disprove the existence of an effectual election protocol for arbitrary graphs in the qualitative graph world. We discuss the possible existence of such a protocol in Section 5.

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.

1.5 Related Works

This investigation is part of an ongoing research effort on understanding the algorithmic limits of computing with mobile agents. The study of the election problem in environments other than the traditional distributed ones is currently being carried out. Such is the case for radio networks, ad hoc networks, etc. See for instance [12, 21, 26, 29]. However, these investigations all assume comparable labels.

Among other problems being attacked in the distributed mobile framework, network exploration in all its variants has definitely played a central role. In [1, 5, 9, 13, 15, 16, 20, 19, 31], the problem of traversing a graph by a finite automaton, or a finite family of cooperative finite automata (using pebbles or not) is considered. The objective is either to derive feasibility results, or to bound the time required for the exploration, or to bound the local memory of the agents. In the map drawing problem [8], not only all edges must be traversed by the (group of cooperative) agent(s) but a map of the graph must also be returned. In the rendezvous (or gathering) problem [2, 14, 25], all agents must gather at the same place. All protocols described in the papers mentioned above assume agents with comparable labels.

There are some problems for which agents are not assumed to be given comparable labels. This is for instance the case of graph searching. In this problem, agents move along the edges of the graph in order to surround an intruder (see, e.g., [28]). However, all graph searching protocols derived so far are centralized, and agents are rather considered as token that are placed, moved, and removed from the graph by an external player (with the exception of [7] which however assumes a way to break symmetry among the agent). On the contrary, all protocols described in this paper are fully decentralized.

2 A Necessary Condition for Election

In this paper, all graphs are assumed to be connected. Given an input (G, p) of the election problem, the placement function p defines a bi-coloring of the nodes of the graph G, say *black* nodes correspond to home-bases, and *white* nodes to those nodes that are initially not occupied by agents. Hence, thoughout all the paper, we will often consider bi-colored graphs $G = (V, E)$, i.e., graphs whose nodes are either black or white (not to be confused with the colors of the agents).

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.

Definition 2.1 *Two nodes x and y of a bi-colored graph (G, p) are* equivalent*, denoted by $x \sim y$, if there is a color-preserving 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.

Let $G = (V, E)$ be an edge-labeled graph. 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.

Definition 2.2 *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.

Since all morphisms must preserve the black or white colors of the nodes determined by the initial placement p of the agents, a label-preserving morphism in (G, p) preserves both edge-labels and node-colors. As a consequence

$$x \sim_{lab} y \Rightarrow x \sim y$$

where $\sim$ is defined in Definition 2.1.

Lemma 2.1 *All equivalence classes of $\sim_{lab}$ have the same size.*

Proof. Assume, for the purpose of contradiction, that not all equivalence classes have the same size, and let C and C' be two equivalence classes with $|C| > |C'|$. One can choose C and C' such that there is at least one edge between them. Let $C = \{x_0, x_1, \ldots, x_n\}$ and, for $i > 0$, let ϕ_i be a label-preserving automorphism such that $\phi_i(x_0) = x_i$. Let e be an edge leading from x_0 to some node y_0 in C', and let $\alpha = \ell_{x_0}(e)$. Since $x_i \sim_{lab} x_0$, if x_0 has an incident edge labeled α, then x_i has also an incident edge labeled α. Moreover, for $i \geq 0$, if y_i is the neighboring node of x_i reached after traversing edge labeled α from x_i, then $y_i \sim_{lab} y_0$. Indeed, on one hand,

$$\alpha = \ell_{x_0}(\{x_0, y_0\}) = \ell_{\phi_i(x_0)}(\{\phi_i(x_0), \phi_i(y_0)\}) = \ell_{x_i}(\{x_i, \phi_i(y_0)\}),$$

and, on the other hand,

$$\alpha = \ell_{x_i}(\{x_i, y_i\}).$$

Therefore $\phi_i(y_0) = y_i$ because the labels of the edges incident to x_i are pairwise distinct. Therefore $y_i \sim_{lab} y_0$ for all $i > 0$, i.e., for any $i \geq 0$, $y_i \in C'$. Moreover, all y_i's are pairwise distinct because, for every $i \geq 1$,

$$\ell_{y_i}(\{y_i, x_i\}) = \ell_{\phi_i^{-1}(y_i)}(\{\phi_i^{-1}(y_i), \phi_i^{-1}(x_i)\}) = \ell_{y_0}(\{y_0, x_0\}).$$

Thus, if $y_i = y_j$ for $i \neq j$ then the two edges $\{y_i, x_i\}$ and $\{y_i, x_j\}$ would have the same label at y_i. Since all y_is are pairwise distinct and all belong to C, we would get $|C'| \geq |C|$, a contradiction. ■

Theorem 2.1 *If there exists an edge-labeling of (G, p) such that the label-equivalence classes (i.e., the classes for $\sim_{lab}$) have size > 1 then election in (G, p) is impossible.*

Proof. Recall that the *view* [33] 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 [30], 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 $\mathcal{V}^{(n-1)}(y)$. (Boldi and Vigna [10] proved that one can actually consider only views up to diameter.) First we show that

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

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 $\psi : \mathcal{V}^{(n-1)}(x) \to \mathcal{V}^{(n-1)}(y)$ be defined as follows. Any node u of $\mathcal{V}^{(n-1)}(x)$ is uniquely defined by a sequence of labels $i_1, \ldots, i_k$, $k \leq n-1$,

```
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

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. Thus the sequence of labels $i_1, \ldots, i_k$ defines a path from y to some node v in $\mathcal{V}^{(n-1)}(y)$. We define $\psi(u) = v$. One can easily check that ψ is a label-preserving isomorphism from $\mathcal{V}^{(n-1)}(x)$ to $\mathcal{V}^{(n-1)}(y)$.

All equivalence classes of G for $\sim_{view}$ have the same size $\sigma_\ell(G)$, which depends on the edge-labeling ℓ of G (see [33]). 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 generic 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 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 [33] 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 [33]) 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 $(G, p) \notin \mathcal{I}$, i.e., election in impossible for this setting. ■

Observe here a major difference between quantitative and qualitative computing. In the former,

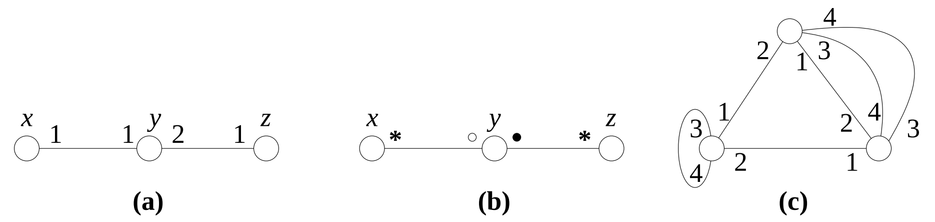

Figure 2: Quantitative labeling vs. qualitative labeling

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 $\{x, y, z\}$ (cf. Figure 2(a)) with

$$\ell_x(\{x,y\}) = 1, \ell_y(\{x,y\}) = 1, \ell_y(\{y,z\}) = 2, \text{ and } \ell_z(\{y,z\}) = 1.$$

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 [33]). The situation is more complex in the qualitative graph world. Indeed, labels are from a non totally ordered set C, unknown to the designer. This does not allow the designer 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 $\{x, y, z\}$ (cf. Figure 2(b)). Assume moreover, according to the qualitative model, that

$$\ell_x(\{x,y\}) = *, \ell_y(\{x,y\}) = \circ, \ell_y(\{y,z\}) = \bullet \text{ and } \ell_z(\{y,z\}) = *.$$

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.

Note also that the reciprocal of Equation 1 does not hold. For instance, consider the following graph $G = (V, E)$ with $V = \{x, y, z\}$, and $E = E_{ring} \cup E_{mess}$ (cf. Figure 2(c)). $G' = (V, E_{ring})$ is a ring of three vertices whose edges are labeled 1 in the clockwise direction, and 2 in the counterclockwise direction. $G'' = (V, E_{mess})$ is a graph of three vertices with two edges e_1 and e_2 between x and y, and a loop f around z. The edge-labels of G'' are as follows: $\ell_x(e_1) = \ell_y(e_2) = 3$, $\ell_x(e_2) = \ell_y(e_1) = 4$, and the two extremities of f are labeled 3 and 4. One can check that all nodes have the same view, although z is obviously not label-equivalent to x nor y, and hence there are three classes of size 1 for $\sim_{lab}$. (This example can easily be extended to a simple graph without loops nor double edges.)

3 Protocol Elect

3.1 A Common Ordering

Let $d(\cdot,\cdot)$ be the distance function in a graph. Our election protocol uses the following notion.

Definition 3.1 *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 $\mathcal{S}(u)$.

The fact that graphs can be canonically ordered is well known (cf., e.g., [27]). To prove the following lemma, we need to take care of some technicalities.

Lemma 3.1 *There is a deterministic algorithm that computes, for any bi-colored graph (G, p), a total ordering $\prec$ for its equivalence classes.*

Proof. We first prove that two nodes of the bi-colored network (G, p) are equivalent (cf., Def. 2.1) if and only if their surroundings are isomorphic. Assume u and v are equivalent in G, i.e., there exists a color-preserving 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 of 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

```
Begin
    MAP-DRAWING;
    COMPUTE & ORDER classes $C_1, \dots, C_\ell, C_{\ell+1}, \dots, C_k$;
    $D := C_1$;
    $i := 2$;
    SYNCHRONIZE($D$);
    While ($i \leq \ell$ and $|D| > 1$) do   /* stage agent-agent */
        $D \leftarrow$ AGENT-REDUCE($D, C_i$);
        $i \leftarrow i + 1$;
        SYNCHRONIZE($D$);
    While ($i \leq k$ and $|D| > 1$) do   /* stage agent-node */
        $D \leftarrow$ NODE-REDUCE($D, C_i$);
        If $D$ has been modified, then SYNCHRONIZE($D$);
        $i \leftarrow i + 1$;
    If $|D| = 1$ then the unique agent in $D$ is the leader
    else election fails;
End.
```

Figure 3: Protocol ELECT

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. ■

Remark. Lemma 3.1 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.

3.2 Description of Protocol ELECT

We now describe our election protocol, called ELECT. An agent is successively asleep, awake, active, possibly passive, and eventually leader or defeated. Agents communicate via the node whiteboards by writing and reading (colored) messages.

Protocol ELECT is sketched on Figure 3. It 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, called MAP-DRAWING, 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 (i.e, the colors of the agent initially placed at every home-base). For that purpose, marking the whiteboards, each agent performs a DFS traversal of G. Note that the distincness of the agents' colors is required for the agents to draw a map of the graph G. Without such a distincness, this task would be impossible. 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 executes COMPUTE & ORDER. I.e., it locally 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 locally 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 $C_1 \prec C_2 \prec \dots \prec C_\ell$ and $C_{\ell+1} \prec C_{\ell+2} \prec \dots \prec C_k$ where $\prec$ is the total order 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.

There are at most $k-1$ other phases, called reduction phases, where k is the number of equivalence classes. Each reduction phase executes one of the two variants of the same subroutine. These variants are called AGENT-REDUCE and NODE-REDUCE. Given two sets A and B of agents, the goal of Subroutine AGENT-REDUCE is to reduce the number of agents to $\gcd(|A|, |B|)$. Given a set A of agents, and a set B of nodes, the goal of Subroutine NODE-REDUCE is to reduce the number of agents to $\gcd(|A|, |B|)$. The descriptions of subroutines AGENT-REDUCE and NODE-REDUCE are in Section 3.3. 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", corresponding to the two while-loops in Figure 3. 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 $k - \ell$. Between phases, active agents synchronize by traversing the network and letting appropriate colored signs on the whiteboards. This is the role of procedure SYNCHRONIZE.

Stage agent-agent: If $\ell > 1$, then agents in C_1 and C_2 become active. The algorithm is aiming at reducing the number of active agents to $d_1 = \gcd(|C_1|, |C_2|)$. For that purpose, active agents apply the sub-routine AGENT-REDUCE. 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 at reducing 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 AGENT-REDUCE, 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 AGENT-REDUCE. 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.

Stage agent-node: In this stage, the phases perform very similarly as for the stage agent-agent. Let $D_{\ell-1}$ be the set of $d_{\ell-1}$ active agents resulting from the stage agent-agent (or the set C_1 of active agents if $\ell = 1$, in which case, we set $d_0 = |C_1|$). If $d_{\ell-1} > 1$, then phase ℓ aims at reducing the number of active agents to $d_\ell = \gcd(d_{\ell-1}, |C_{\ell+1}|)$. This is performed by application of the sub-routine NODE-REDUCE which returns a set D_ℓ of active agents, $|D_\ell| = d_\ell$. If $d_\ell > 1$, then yet another phase is initiated, aiming at reducing the number of active agents to $d_{\ell+1} = \gcd(d_\ell, |C_{\ell+2}|)$. This phase is again performed by application of NODE-REDUCE. An so on, until either some $d_i = 1$, or $k - 1$ phases have been performed in total. 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.

3.3 Subroutines Agent-Reduce and Node-Reduce

We introduce few new states for the agents. An active agent may be alternatively *searching* or *waiting*. A waiting agent is a priori *unmatched*, and can be *matched* by a searching agent.

3.3.1 Subroutine Agent-Reduce

Subroutine Agent-Reduce takes as input a set D of active agents, and a set C of asleep agents. It is described on Figure 4. Initially, Agents in D traverse the graph to wakeup the agents in C. Let $S = \text{argmin}\{|C|, |D|\}$ and $W = \text{argmax}\{|C|, |D|\}$. Agents in W are waiting agents, and agents in S are searching agents. Note that $|W| \geq |S|$.

Agent-Reduce performs in a certain number of rounds. Before entering every round, the searching agent synchronize by traversing the network and letting appropriate colored signs on the white-boards. At every round, waiting agents stay in their home-base, and wait to be visited by all the searching agents. The searching agents traverse the graph to visit the waiting agents. If a searching agent matches with a waiting agent, the latter becomes matched. When a matched waiting agent has been visited by all the searching agents, it becomes passive. 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 Agent-Reduce, 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 the sets S' and W' of searching and waiting agents for the next round are:

$$\begin{cases} S' = S \text{ and } W' = W \setminus P & \text{if } |W| - |S| \geq |S|; \\ S' = W \setminus P \text{ and } W' = S & \text{otherwise.} \end{cases}$$

Note that $|W'| \geq |S'|$. Agent-Reduce stops when $|S| = |W|$, and then it returns the set S. Agents in W become passive.

3.3.2 Subroutine Node-Reduce

The subroutine Node-Reduce performs roughly the same as Agent-Reduce. The major difference between the two subroutines is that Agent-Reduce involves matchings between two agents whereas Node-Reduce involves matchings between agents and selected nodes. Initially, all nodes in the input class C_i are selected. We describe hereafter how matchings between α agents and β selected 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[3]. Each of the β nodes can be

[3]Agent a acquires a node simply means that a is able to access the whiteboard, and write "acquired" on it.

```
Input: a set $D$ of active agents, and a set $C$ of asleep agents;
Begin
    $S \leftarrow$ argmin$\{|C|, |D|\}$ and $W \leftarrow$ argmax$\{|C|, |D|\}$;
    While $|S| < |W|$ do
        Synchronize($S$);
        Agents in $S$ match with a subset $P$ of agents in $W$;
        If $|W| - |S| \geq |S|$ then $W \leftarrow W \setminus P$
        else $W' \leftarrow S$, $S \leftarrow W \setminus P$, $W \leftarrow W'$;
    Return $S$;
End.
```

Figure 4: Subroutine Agent-Reduce

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 Node-Reduce returns the current set of active agents.

3.4 Properties of Protocol Elect

The main properties of Protocol Elect are summarized by the following:

Theorem 3.1 *Protocol* Elect *elects a leader among any set of agents placed by p in a network $G = (V, E)$, provided that* $\gcd(|C_1|, \ldots, |C_k|) = 1$ *where the C_i's are the equivalence classes of (G, p). Moreover, if* $\gcd(|C_1|, \ldots, |C_k|) > 1$ *then* Elect *lets the agents know about the failure of the election. The total number of moves and whiteboard accesses performed by r agents is $O(r|E|)$.*

Proof. Given two sets C and D of agents, Agent-Reduce completes with $\gcd(|C|, |D|)$ active agents. Indeed, by construction of the sets S's and W's, the sequence of pairs $(|S|, |W|)$ is the sequence of pairs of integers obtained by computing $\gcd(|C|, |D|)$ using Euclid's algorithm. Similarly, given a set D of agents, and a set C of nodes, Node-Reduce completes with $\gcd(|C|, |D|)$ active agents. Indeed, if $b = qa + r$, $0 < r \leq a$, then $\gcd(a, b) = \gcd(a, r)$. Given an input (G, p) of the election problem with $\gcd(|C_1|, \ldots, |C_k|) = 1$, protocol Elect returns one agent in state leader and the other agents in state defeated because an invariant of the protocol is that, after execution of the while-loop for index i, $|D| = \gcd(|C_1|, \ldots, |C_i|)$.

Every agent performs a constant number of whiteboard accesses each time it visits a node. Therefore, the number of whiteboard accesses is at most big-O of the number of moves. During Map-Drawing, every agent performs a traversal of the network, resulting in $O(r|E|)$ moves. Between two consecutive phases the Stage agent-agent, active agents perform a traversal to synchronize. It results in $O(\sum_{i=1}^{\ell-1} d_i|E|)$ moves, hence in $O(r|E|)$ moves since $d_i \leq |C_{i+1}|$ for $i = 1, \ldots, \ell - 1$, and $r = \sum_{i=1}^{\ell} |C_i|$. Between two consecutive phases of Stage agent-node, active agents perform a traversal to synchronize only if the number of active agents have been modified. In this case, the number of active agents has been at least halved. It results in $O(d_{\ell-1}|E|)$ moves for synchronisation between two phases of Stage agent-node, that is in $O(r|E|)$ moves.

Now, we count the number of moves inside Agent-Reduce and Node-Reduce. In Agent-Reduce, the total number of traversals for one matching and one synchronization is at most twice the total number of matched waiting agents. Indeed, only searching agents move. Every other traversal results in a matching between the searching agents and some waiting agents (the other traversal is for synchronization). These two traversals are charged to the agents matched at this round. More precisely, if Agent-Reduce is called with C and D, and results in D', the number of traversals is at most $2(|D| + |C| - |D'|)$. Therefore, by summing up the costs of every call, the total number of moves due to matching or synchronization traversals during the executions of all calls to Agent-Reduce is $O(\sum_i |C_i||E|) = O(r|E|)$. Finally, if a round is performed during Node-Reduce, then the number of active agents is at least halved every two rounds. Indeed, Cases 1 and 2 described in Section 3.3.2 alternate every two rounds because ϱ replaces $\max\{\alpha, \beta\}$ for the next round. Therefore, the total number of moves due to matching or synchronization traversals during all the executions of Node-Reduce is also $O(r|E|)$. Hence, the total number of moves and whiteboard accesses is $O(r|E|)$, which completes the proof. $\blacksquare$

Remark. In this paper, we aim at establishing feasibility thresholds for the election problem, as a function of the computational ability of the agents, and of the knowledge given to these agents. As a consequence, we are less concerned by the time it may take to solve that problem. We agree though that, if time is an issue, in particular as far as practical applications are concerned, then Protocol Elect presents two major drawbacks, despite the fact that the total number of moves and whiteboard accesses is kept reasonably small. Firstly, computing the equivalence classes $C_1, \ldots, C_k$ can be quite time-consuming since graph-isomorphism is not known to be in P, in NPC, or in between these two classes. Secondly, computing the ordering $\prec$ also involves time-consuming computations. An interesting and challenging problem is to devise a computationally effective alternative to Protocol Elect.

4 Effectual Election in Cayley networks

To get an effectual protocol for the class of Cayley graphs, we slightly modify Elect 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 equivalence classes for *translations* instead of equivalence classes for arbitrary automorphisms. Recall that a translation is an automorphism of $\mathrm{Cay}(\Gamma, S)$ of the form

$$\begin{aligned} \phi_\gamma : \quad & \Gamma \mapsto \Gamma \\ & a \to \gamma a \end{aligned}$$

for any $\gamma \in \Gamma$. 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 maps x to y is $\gamma = yx^{-1}$. However, the black and white coloring of the nodes creates different classes. Two nodes that are equivalent for 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 the translation $a \mapsto a + (n/2-2) \bmod n$ maps 0 to $n/2-2$, and these two nodes are of different colors (since one is an home-base, whereas the other is not). We prove the following:

Theorem 4.1 *Protocol* Elect *using translation is an effectual election protocol 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 first 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 $(\gamma x)^{-1}(\gamma y) = x^{-1}y$. This is because generators in S act on the right whereas translations act on the left.

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 any 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. I.e., marked edges are mapped to marked edges of same labels, and unmarked edges to unmarked 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'. Indeed, let $e = \{x, y\}$, $x \in C$, $y \in C'$, and $\ell_x(e) = s$. Thus $s = x^{-1}y$. Let $e' = \{x', y'\}$, $x' \in C$, and $\ell_{x'}(e') = s$. We get $y' = x's = x'x^{-1}y$. Thus, if $x' = \gamma x$ for some $\gamma \in \Gamma$, we get $y' = \gamma x x^{-1} y = \gamma y$, and therefore y' is translation-equivalent to y, hence $y' \in C'$.

Since $|C| \neq |C'|$, by the invariant property, 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 Cayley edge-labeling, all pseudo label-equivalence classes remain pseudo-equivalence classes after updating the labeling, with the exception of C' that 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 either y has no marked edge labeled s^{-1}, or the edges labeled s^{-1} at x and y have their other extremities in two different classes.

Another invariant of our process is the following: at any time, the gcd of the sizes of the current pseudo label-equivalence classes is equal to d. Indeed:

1. each iteration transforms a pair (C, C') into a triple $(C, Cs, C' \setminus Cs)$ with $|Cs| = |C|$;
2. all other classes remain of same size; and
3. $\gcd(|C|, |Cs|, |C' \setminus Cs|) = \gcd(|C|, |C'| - |C|) = \gcd(|C|, |C'|)$. (This latter equality follows from 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.

Hence, election is impossible if $\gcd(|C_1|, \ldots, |C_k|) > 1$. On the other hand, if $\gcd(|C_1|, \ldots, |C_k|) = 1$, then, from Theorem 3.1, Elect elects a leader. Therefore, protocol Elect is effectual for the class of Cayley graphs. ■

We have designed an election protocol that is proved to be effectual for the class of Cayley graphs in the qualitative graph world. However, there are symmetric graphs for which protocol Elect is not effectual. In particular, the Petersen graph (see Figure 5(a)) allows to construct a counter-example of Elect's effectualness. Consider two agents whose initial positions are marked in black on Figure 5(a). The equivalence classes are indicated on Figure 5(b). There are three classes, marked black, gray, 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 gray nodes are exchanged by ϕ. The dotted arrows indicate an automorphism ψ exchanging white nodes. ψ also exchanges two gray nodes, while it lets two other gray nodes fixed, as well as the two black nodes. Combining ϕ and ψ with the horizontal symmetry makes all white nodes equivalent, and all gray nodes equivalent. Finally, gray nodes are adjacent to black nodes, and hence form a class distinct from the class of white nodes.

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 the neighboring nodes of your home-base, 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;

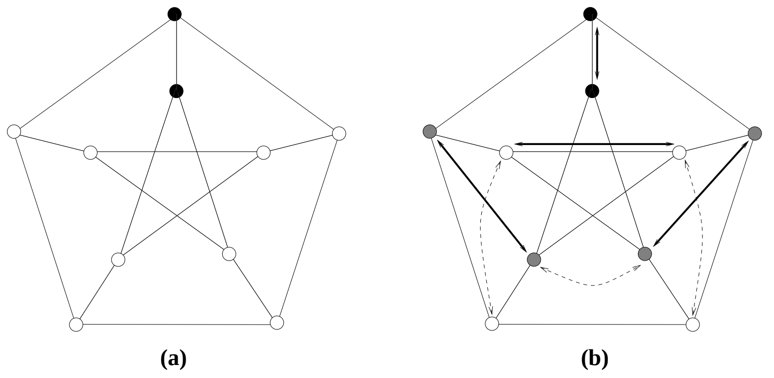

Figure 5: The Petersen graph

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 effectual.

This counter example of Elect's effectualness 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 5(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, $G \equiv \mathrm{Cay}(\Gamma, S)/H$ where $\Gamma = \mathrm{Aut}(G)$, $H = \mathrm{stab}(u_0) = \{\phi \in \Gamma \ / \ \phi(u_0) = u_0\}$, 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.

5 Concluding Remarks

We have shown that, even without comparability, election is still solvable: for all instances but those for which $\gcd(|C_1|, \ldots, |C_k|) > 1$, distinctness is a necessary and sufficient condition for leader election. It is not surprising that algorithms in the qualitative graph world strongly depend on structural properties of graphs. Indeed, qualitative computing cannot benefit of any help from labeling, and hence should count only on asymmetries. The equivalence classes of the network form structures that an effectual algorithm in the qualitative model should take into account, and this is exactly what protocol Elect does. However, we were unable to prove or disprove whether it is the ultimate structure that an algorithm can take benefit of, and hence to prove or disprove the existence of an effectual election protocol in the qualitative graph world. We hence state explicitly the following question:

Open problem 1: Does it exist an effectual election protocol for arbitrary graphs in the qualitative model?

Since the writing of this paper, a positive answer to this question has been recently announced in [11].

More generally, the main goal of this paper was to establish feasibility thresholds for problems arising in the distributed mobile framework, as a function of the computational ability of the agents, and of the knowledge given to these agents. Very intriguing questions of a general nature are the following. What (natural) problems are not solvable without comparability, or solvable in a very restricted context only? What (natural) problems are (almost always) solvable in the qualitative graph world without additional constraints? For such problems, what is the degradation of the performances in comparison with those observed in the quantitative graph world? These questions can obviously be asked not only in the mobile computational setting, but also in the standard (i.e. static) distributed one. Notice that, in this latter setting, the relationship between computability and comparability had never before been examined. Hence, we conclude by the following general informal problem:

Open problem 2: What is the impact of incomparability on computability?

References

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

[2] S. Alpern and S. Gal. The theory of search games and rendezvous. Int. Series in Operations research and Management Science, number 55, Kluwer Academic Publishers, 2002.

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

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

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

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

[7] L. Barrière, P. Flocchini, P. Fraigniaud and N. Santoro. Capture of an Intruder by Mobile Agents In 14th ACM Symposium on Parallel Algorithms and Architectures (SPAA), pages 200-209, 2002.

[8] 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), pages 269–278, 1998.

[9] M. Bender and D. Slonim. The power of team exploration: Two robots can learn unlabeled directed graphs. In 35th IEEE Symp. on Foundations of Computer Science (FOCS), pages 75–85, 1994.

[10] P. Boldi and S. Vigna, Computing anonymously with arbitrary knowledge. In 18th ACM Symp. on Principles of Distributed Computing (PODC), pages 181-188, 1999.

[11] J. Chalopin. Election in the Qualitative World. In 13th Colloquium on Structural Information and Communication Complexity (SIROCCO), July 3 - 5, 2006, Chester, UK.

[12] Y. Chen and J. Welch. Self-stabilizing mutual exclusion using tokens in mobile ad hoc networks. In 6th ACM Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIALM), Pages 34-42, 2002.

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

[14] A. Dessmark, P. Fraigniaud and A. Pelc. Deterministic rendezvous in graphs. In 11th Annual European Symposium on Algorithms (ESA), LNCS 2832, pages 184-195, 2003.

[15] K. Diks, P. Fraigniaud, E. Kranakis, and A. Pelc, Tree exploration with little memory. In 13th ACM-SIAM Symp. on Discrete Algorithms (SODA), pages 588–597, 2002.

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

[17] 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.

[18] 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), pages 101-112, 2000.

[19] P. Fraigniaud and D. Ilcinkas. Digraphs Exploration with Little Memory In 21st Symposium on Theoretical Aspects of Computer Science (STACS), LNCS 2996, pages 246-257, 2004

[20] P. Fraigniaud, L. Gasieniec, D. Kowalski, and A. Pelc. Collective Tree Exploration. In 6th Latin American Theoretical Informatics Symposium (LATIN), Buenos-Aires, 2004

[21] 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), pages 251–260, 1999.

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

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

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

[25] W. Lim and S. Alpern. Minimax rendezvous on the line. SIAM J. on Control and Optimization 34(5), pp. 1650-1665, 1996.

[26] 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.

[27] B. McKay. Practical Graph Isomorphism. Congressus Numerantium, Vol. 30, pp. 45-87, 1981.

[28] N. Megiddo, S. Hakimi, M. Garey, D. Johnson and C. Papadimitriou. The complexity of searching a graph. Journal of the ACM 35(1):18–44, 1988.

[29] 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.

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

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

[32] E. Rozenman, A. Shalev, and A. Wigderson. A new family of Cayley expanders. In 36th ACM Symposium on Theory of Computing (STOC), pages 445-454, 2004.

[33] 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.

[34] M. Yamashita and T. Kameda. Leader election problem on networks in which processor identity numbers are not distinct. IEEE Transaction on Parallel and Distributed Systems 9(10):878-887, 1999.