Election and Local Computations on Edges (Extended Abstract)

Jérémie Chalopin
Yves Métivier

chalopin,metivier@labri.fr
LaBRI,
Université Bordeaux I, ENSEIRB,
351 cours de la Libération
33405 Talence, France

1 Introduction

The point of departure and the motivation for this paper are the results of Angluin [1] which has introduced a tool to analyze the election algorithm: the coverings, Yamashita and Kameda [21] and Mazurkiewicz [15] which have obtained characterizations of graphs in which election is possible under two different models of distributed computations. Our aim is twofold. First it is to obtain characterizations of graphs in which election is possible under intermediate models between the models of Yamashita-Kameda and of Mazurkiewicz. Our second aim is to understand the implications of the models for the borderline between positive and negative results for distributed computations. In this work, characterizations are obtained under three different models.

1.1 The Model

We consider networks of processors with arbitrary topology. A network is represented as a connected, undirected graph where vertices denote processors and edges denote direct communication links. Labels are attached to vertices and edges. The identities of the vertices, a distinguished vertex, the number of processors, the diameter of the graph or the topology are examples of labels attached to vertices; weights, marks that encode a spanning tree or the sense of direction are examples of labels attached to edges.

At each step of computation labels are modified on exactly one edge and its endvertices of the given graph, according to certain rules depending on the label of this edge and the labels of its endvertices only. Thus rules are of the form:

$$R : \overset{\mathbf{X}}{\circ} \overset{\mathbf{Y}}{\text{———}} \overset{\mathbf{Z}}{\circ} \quad \longrightarrow \quad \overset{\mathbf{X}'}{\circ} \overset{\mathbf{Y}'}{\text{———}} \overset{\mathbf{Z}'}{\circ}$$

Such local computations are called local computations on closed edges in this paper. The relabelling is performed until no more transformation is possible, i.e., until a normal form is obtained.

1.2 The Election Problem

The election problem is one of the paradigms of the theory of distributed computing. It was first posed by LeLann [10]. Considering a network of processors, the election problem is to arrive at a configuration where exactly one processor is in the state *elected* and all other processors are in the state *non-elected*. The elected vertex is used to make decisions, to centralize or to broadcast some information.

Known Results about the Election Problem. Graphs where election is possible were already studied, the algorithms usually involved some particular knowledge and some particular basic computation steps. Solving the problem for different knowledge has been investigated for some particular cases (see [2, 12, 19] for details) including : the network is known to be a tree, the network is known to be complete, the network is known to be a grid or a torus, the nodes have different identification numbers, the network is known to be a ring and has a known prime number of vertices. Characterizations of graphs where election is possible have been given under two models of computations.

- In [21], Yamashita and Kameda consider the following asynchronous model. In each step, a vertex, depending on its current label, either changes its label, sends a message via one of its ports, or receives a message via a port. The topology of the graph is assumed to be known. They proved that, knowing the topology or the size of the network, there exists an election algorithm for G if and only if the symmetricity of G is equal to 1 (where the symmetricity depends on the number of labelled trees isomorph to a certain tree associated to G) ([21], Theorem 1 p. 75).
- In [15], Mazurkiewicz considers the following asynchronous model. In each step, labels are modified on a subgraph consisting of a node and its neighbours, according to certain rules depending on this subgraph only. He proves that, given a graph G, there exists an election algorithm for G if and only if G is minimal for the covering relation (a graph H is a covering of a graph K if there exists a surjective morphism φ from H onto K which maps bijectively the neighbours of any vertex v onto the neighbours of $\varphi(v)$; a graph H is minimal if whenever H covers a graph K then H and K are isomorphic.).

1.3 The Main Results

We recall that at each step of computation, labels are modified on exactly two vertices linked by an edge and on this edge of the given graph, according to certain rules depending on the labels of this edge and on the labels of the two vertices only. Under this hypothesis, we give a characterization of graphs for which there exists an election algorithm. More precisely, we prove that, given a simple graph G (graph without self-loop or multiple edges) there exists an election algorithm for G if and only if G is minimal for the covering relation. Where the notion of covering is a generalization of the previous one. First we consider multigraphs: graphs having possibly multiple edges without self-loops.

For this class of graphs, a graph H is a covering of a graph K if there exists a surjective morphism φ from H onto K such that, for every vertex v, the restriction of φ to the set of edges incident to v is a bijection between this set of edges and the set of edges incident to $\varphi(v)$.

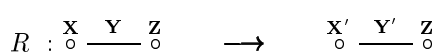
This condition is not equivalent to the condition of Mazurkiewicz. If we consider the ring with 4 vertices, denoted R_4, then it is minimal for the first notion of covering but it is not minimal for the generalization. Indeed, for the generalization it covers the graph H defined by 2 vertices having a double edge (see Fig.1).

Fig. 1. The graph R_4 covers the graph H.

Thus there exists an election algorithm for R_4 in the model of Mazurkiewicz and there does not exist an election algorithm for R_4 in the model studied in this paper.

In fact, the Mazurkiewicz algorithm is a distributed enumeration algorithm: it is a distributed algorithm such that the result of any computation, in a graph G minimal for the covering relation, is a labelling of the vertices that is a bijection from $V(G)$ to $\{1, 2, \ldots, |V(G)|\}$. For a given graph G, the election problem and the enumeration problem with termination detection are equivalent in the model of Mazurkiewicz; we prove that under the same hypothesis the two problems are also equivalent in the model studied in this paper. This property is no more true if we have no information on the graph like the size or the topology.

In the second part of this paper, we consider the following model of computation: at each step of computation labels are modified on exactly one edge and one endvertex of this edge of the given graph, according to certain rules depending on the label of this edge and the labels of its endvertices only (local computations on open edges). Thus the form of the rules is:



We prove that this model is equivalent to the model studied in the first part by using a simulation algorithm. Thus we obtain also a characterization of graphs where election is possible. This result is not immediate: for example, using the first model, it is easy to give a name to each edge of a given graph such that for a given vertex v, all the edges incident to v have a different name; if we do

not use the simulation algorithm this result is not trivial in the context of the second model. Finally, we extend the characterization concerning the election to the model where at each step of computation labels are modified on a subgraph consisting of a node and the incident edges, according to certain rules depending on the vertex, the incident edges and the endvertices (local computations on open star graphs). The end of the paper proves that models using labels on edges are strictly stronger than models without labels on edges.

1.4 Related Works and Results

In [21] the election problem is studied under other initial knowledges: the size of the graph, an upper bound of the number of vertices; in some cases multigraphs are necessary. In addition of the works of [1, 15, 21, 22] and [20, 23], one can cite the results of Boldi and Vigna who use directed graphs [4, 3, 5, 6]. They consider directed graphs coloured on their arcs. Each vertex changes its state depending on its previous state and on the states of its in-neighbours; activation of processors may be synchronous, asynchronous or interleaved. A generalization of coverings, called fibrations, is studied and properties which found applications in the distributed computing setting are emphasized. In [7, 16, 9, 8] the model of Mazurkiewicz is considered and a characterization of families of graphs in which election is possible is given; in [8] characterizations of recognizable classes of graphs by means of local computations are given.

2 Basic Notions and Notation

2.1 Graphs, Labelled Graphs and Coverings

The notations used here are essentially standard [18]. We consider finite, undirected, connected graphs without self-loop having possibly multiple edges. If $G = (V(G), E(G), \mathrm{Ends})$ is a graph, then $V(G)$ denotes the set of vertices, $E(G)$ denotes the set of edges and Ends denotes a map assigning to every edge two vertices: its ends. Two vertices u and v are said to be adjacent or neighbours if there exists an edge e such that $\mathrm{Ends}(e) = \{u, v\}$. In this paper, graphs may have several edges between the same two vertices; such edges are called multiple edges. A simple graph $G = (V(G), E(G))$ is a graph with no self-loop or multiple edges: $E(G)$ can be seen as a set of pairs of $V(G)$. Let e be an edge, if the vertex v belongs to $\mathrm{Ends}(e)$ then we say that e is incident to v. The set of all the edges of G incident with v is denoted $I_G(v)$. The set of neighbours of v in G, denoted $N_G(v)$, is the set of all vertices of G adjacent to v. For a vertex v, we denote by $B_G(v)$ the ball of radius 1 with center v, that is the graph with vertices $N_G(v) \cup \{v\}$ and edges $I_G(v)$. For an edge e, we denote $A_G(e)$ the single edge graph $(\mathrm{Ends}(e), \{e\})$; we call closed edge an edge with the two endvertices, if we consider the edge with only one endvertex it is an open edge.

Throughout the paper we will consider graphs where vertices and edges are labelled with labels from a recursive alphabet L. A graph labelled over L will be

denoted by (G, λ), where G is a graph and λ: $V(G) \cup E(G) \to L$ is the labelling function. The graph G is called the underlying graph and the mapping λ is a labelling of G. For a labelled graph (G, λ), $lab((G, \lambda))$ is the set of labels that occur in (G, λ). The class of labelled graphs over some fixed alphabet L will be denoted by $\mathcal{G}_L$. Let (G, λ) and (G', λ') be two labelled graphs. Then (G, λ) is a subgraph of (G', λ'), denoted by $(G, \lambda) \subseteq (G', \lambda')$, if G is a subgraph of G' and λ is the restriction of the labelling λ' to $V(G) \cup E(G)$.

Labelled graphs will be designated by bold letters like $\mathbf{G}$, $\mathbf{H}, \ldots$ If $\mathbf{G}$ is a labelled graph, then G denotes the underlying graph.

2.2 Coverings

We say that a graph G is a *covering* of a graph H via γ if γ is a surjective homomorphism from G onto H such that for every vertex v of $V(G)$ the restriction of γ to $I_G(v)$ is a bijection onto $I_H(\gamma(v))$. The covering is proper if G and H are not isomorphic.

The notion of covering extends to labelled graphs in an obvious way. The labelled graph (H, λ') is *covered* by (G, λ) via γ, if γ is a homomorphism from (G, λ) to (H, λ') such that for every vertex v of $V(G)$ the restriction of γ to $I_G(v)$ is a bijection onto $I_H(\gamma(v))$. Note that a graph covering is exactly a covering in the classical sense of algebraic topology, see [13].

Remark 1. We use a different definition for coverings than Angluin's one. In fact, if we consider only simple graphs these two definitions are equivalent. For Angluin, (H, λ') is *covered* by (G, λ) via γ, if γ is a homomorphism from (G, λ) to (H, λ') such that for every vertex v of $V(G)$ the restriction of γ to $N_G(v)$ is a bijection onto $N_H(\gamma(v))$. Given a simple graph G, for each vertex $u \in V(G)$, there is a natural bijection between $I_G(u)$ and $N_G(u)$ and therefore it is easy to see the equivalence.

We work with graphs that can have multiple edges and in this case the two definitions are not equivalent. Consider the graphs G and H from Fig. 2, if we consider the morphism φ defined from G to H by the letters a, b, α, β, we easily see that G is a covering of H. But if we use Angluin's definition of covering, G is not a covering of H since for each $u \in G$, $|N_G(u)| = 2$, whereas for each $v \in H$, $|N_H(v)| = 1$.

A graph $\mathbf{G}$ is called *minimal* if every covering from $\mathbf{G}$ to some $\mathbf{H}$ is a bijection. A simple graph $\mathbf{G}$ is called *$\mathcal{S}$-minimal* if every covering $\mathbf{G}$ to some simple graph $\mathbf{H}$ is a bijection. The graphs G' and H from Fig. 2 are minimal graphs, whereas G is a proper covering of H and therefore G is not minimal. Moreover, G or G' are not a proper covering of any simple graph: G and G' are $\mathcal{S}$-minimal.

We have the following basic property of coverings [17]:

Lemma 1. *For every covering γ from $\mathbf{G}$ to $\mathbf{H}$ there exists an integer q such that $card(\gamma^{-1}(v)) = q$, for all $v \in V(H)$.*

The integer q in the previous lemma is called the number of *sheets* of the covering. We also refer to γ as a *q-sheeted covering*.

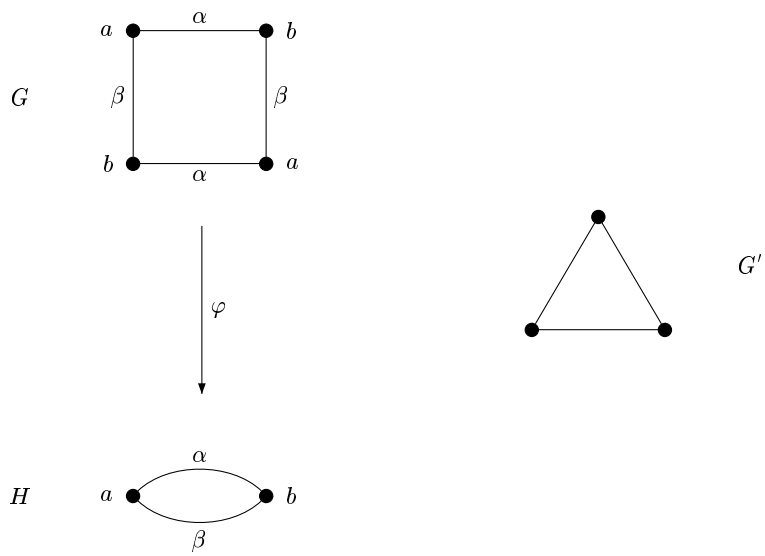


Fig. 2. First Examples.

Lemma 2. *Let* $\mathbf{G}$ *be a covering of* $\mathbf{H}$ *via* γ *and let* $e_1, e_2 \in E(G)$ *be such that* $e_1 \neq e_2$*. If* $\gamma(e_1) = \gamma(e_2)$ *then* $A_G(e_1) \cap A_G(e_2) = \emptyset$*, i.e.,* $Ends(e_1) \cap Ends(e_2) = \emptyset$*.*

3 Local Computations on Closed Edges

In this section we give the definition of local computations on closed edges and their relation with coverings. They model networks of processors of arbitrary topology. The network is represented as a connected, undirected graph where vertices denote processors and edges denote direct communication links. Labels (or states) are attached to vertices and edges. Local computations as considered here can be described in the following general framework. Let $\mathcal{G}_L$ be the class of L-labelled graphs and let $\mathcal{R} \subseteq \mathcal{G}_L \times \mathcal{G}_L$ be a binary relation on $\mathcal{G}_L$. Then $\mathcal{R}$ is called a *graph rewriting relation*. We assume that $\mathcal{R}$ is closed under isomorphism, i.e., if $\mathbf{G}\ \mathcal{R}\ \mathbf{G}'$ and $\mathbf{H} \simeq \mathbf{G}$ then $\mathbf{H}\ \mathcal{R}\ \mathbf{H}'$ for some labelled graph $\mathbf{H}' \simeq \mathbf{G}'$. In the remainder of the paper $\mathcal{R}^*$ stands for the reflexive-transitive closure of $\mathcal{R}$. The labelled graph $\mathbf{G}$ is *$\mathcal{R}$-irreducible* (or just irreducible if $\mathcal{R}$ is fixed) if there is no $\mathbf{G}'$ such that $\mathbf{G}\ \mathcal{R}\ \mathbf{G}'$. For $\mathbf{G} \in \mathcal{G}_L$, $\mathrm{Irred}_{\mathcal{R}}(\mathbf{G})$ denotes the set of $\mathcal{R}$-irreducible graphs obtained from $\mathbf{G}$ using $\mathcal{R}$, i.e., $\mathrm{Irred}_{\mathcal{R}}(\mathbf{G}) = \{\mathbf{H} \mid \mathbf{G}\,\mathcal{R}^*\,\mathbf{H}$ and $\mathbf{H}$ is $\mathcal{R}$-irreducible$\}$.

Definition 1. *Let* $\mathcal{R} \subseteq \mathcal{G}_L \times \mathcal{G}_L$ *be a graph rewriting relation.*

1. $\mathcal{R}$ *is a* relabelling relation *if whenever two labelled graphs are in relation then the underlying graphs are equal, i.e.:*

$$\mathbf{G}\ \mathcal{R}\ \mathbf{H} \textit{ implies that } G = H.$$

2. $\mathcal{R}$ *is* local *on closed edges if it can only modify an edge and its endvertices, i.e.,* $(G, \lambda)\ \mathcal{R}\ (G, \lambda')$ *implies that there exists an edge* $e \in E(G)$ *such that*

$$\lambda(x) = \lambda'(x) \textit{ for every } x \notin \textit{Ends}(e) \cup \{e\}.$$

The labelled single edge graph $(A_G(e), \lambda)$ *is a* support *of the relabelling relation.*

The next definition states that a local relabelling relation $\mathcal{R}$ is *locally generated* on closed edges if the applicability of any relabelling depends only on the single edge subgraphs.

Definition 2. *Let* $\mathcal{R}$ *be a relabelling relation. Then* $\mathcal{R}$ *is* locally generated *on closed edges if it is local on closed edges and the following is satisfied: For all labelled graphs* (G, λ), (G, λ'), (H, η), (H, η') *and all edges* $e \in E(G)$, $f \in E(H)$ *such that the* $A_G(e)$ *and* $A_H(f)$ *are isomorphic via* $\varphi\colon V(A_G(e)) \cup E(A_G(e)) \longrightarrow V(A_H(f)) \cup E(A_H(f))$, *the following three conditions:*

1. $\lambda(x) = \eta(\varphi(x))$ *and* $\lambda'(x) = \eta'(\varphi(x))$ *for all* $x \in V(A_G(e)) \cup E(A_G(e))$
2. $\lambda(x) = \lambda'(x)$, *for all* $x \notin V(A_G(e)) \cup E(A_G(e))$
3. $\eta(x) = \eta'(x)$, *for all* $x \notin V(A_H(f)) \cup E(A_H(f))$

imply that $(G, \lambda)\ \mathcal{R}\ (G, \lambda')$ *if and only if* $(H, \eta)\ \mathcal{R}\ (H, \eta')$.

By definition, local computations on closed edges on graphs are computations on graphs corresponding to locally generated relabelling relations on closed edges.

We now present the fundamental lemma connecting coverings and locally generated relabelling relations on closed edges [1]. It states that, whenever $\mathbf{G}$ is a covering of $\mathbf{H}$, every relabelling step in $\mathbf{H}$ can be lifted to a relabelling sequence in $\mathbf{G}$, which is compatible with the covering relation.

Lemma 3 (Lifting Lemma). *Let* $\mathcal{R}$ *be a locally generated relabelling relation on closed edges and let* $\mathbf{G}$ *be a covering of* $\mathbf{H}$ *via* γ. *If* $\mathbf{H}\ \mathcal{R}^*\ \mathbf{H}'$ *then there exists* $\mathbf{G}'$ *such that* $\mathbf{G}\ \mathcal{R}^*\ \mathbf{G}'$ *and* $\mathbf{G}'$ *is a covering of* $\mathbf{H}'$ *via* γ.

4 Election and Enumeration

The main result of this part is that for every graph $\mathbf{G}$, there exists an election algorithm using local computations on closed edges on $\mathbf{G}$ if and only if there exists an enumeration algorithm with termination detection using local computations on closed edges on $\mathbf{G}$.

4.1 Definitions

A distributed election algorithm on a graph $\mathbf{G}$ is a distributed algorithm such that the result of any computation is a labelling of the vertices such that exactly one vertex has the label *elected* and all other vertices have the label *non-elected*. The labels *elected* and *non-elected* are terminal, i.e., when they appear on a vertex they remain until the end of the computation. A distributed enumeration algorithm on a graph $\mathbf{G}$ is a distributed algorithm such that the result of any computation is a labelling of the vertices that is a bijection from $V(G)$ to $\{1, 2, \ldots, |V(G)|\}$. It is easy to see that if we have an enumeration algorithm on a graph $\mathbf{G}$ where vertices can detect whether the algorithm has terminated, we have an election algorithm on $\mathbf{G}$ by electing the vertex labelled by 1.

4.2 Impossibility results

Using the same method as in the Lifting Lemma [1], we obtain:

Proposition 1. *Let* $\mathbf{G}$ *be a labelled graph which is not minimal, there is no enumeration algorithm for* $\mathbf{G}$.

Consequently, there is no election algorithm for a graph $\mathbf{G}$, if $\mathbf{G}$ is not minimal. Otherwise, we could find an enumeration algorithm for $\mathbf{G}$, as it will be shown in the next section. Furthermore, we can prove that:

Proposition 2. *Given a graph* $\mathbf{G}$*, there is an algorithm using local computations on closed edges that solves the election problem on* $\mathbf{G}$ *if and only if there is an algorithm using local computations on closed edges that solves the enumeration problem with detection termination on* $\mathbf{G}$.

5 An Enumeration Algorithm

In this section, we describe an algorithm $\mathcal{M}$ using local computations on closed edges that solve the enumeration problem on a minimal graph $\mathbf{G}$. This algorithm uses some ideas developed in [15]. Each vertex v attempts to get its own number between 1 and $|V(G)|$. A vertex chooses a number and broadcasts it with its label and its labelled neighbourhood all over the network. If a vertex u discovers the existence of another vertex v with the same number, then it compares its *local view*, i.e., the labels and numbers of its neighbours, with the local view of v. If the label of u or the local view of u is "weaker", then u chooses another number and broadcasts it again with its local view. At the end of the computation, every vertex will have a unique number if the graph is covering-minimal.

5.1 Labels

Let $\mathbf{G} = (G, \lambda)$ and consider a vertex $v_0 \in G$, and the set $\{e_1, \ldots, e_d\}$ of edges that are incident to v_0.

For each edge $e \in E(G)$ such that $\text{Ends}(e) = \{v_1, v_2\}$, a number $p(e)$ will be associated to e such that for each $e' \in I_G(v_1) \cup I_G(v_2)$, $p(e) \neq p(e')$. The label of an edge e is the pair $(\lambda(e), p(e))$ and the initial labelling is $(\lambda(e), 0)$.

For each vertex $v \in V(G)$, the label of v is the pair $(\lambda(v), c(v))$ where $c(v)$ is a triple $(n(v), N(v),\ M(v))$ representing the following information obtained during the computation (formal definitions are given below):

- $n(v) \in \mathbb{N}$ is the *number* of the vertex v computed by the algorithm;
- $N(v) \in \mathcal{N}$ is the *local view* of v, and it is a set defined by:

$$\{(p(e), \lambda(e), n(v'), \lambda(v')) \mid e \in I_G(v),\ \text{Ends}(e) = \{v, v'\} \text{ and } p(e) \neq 0\};$$

- $M(v) \subseteq L \times \mathbb{N} \times \mathcal{N}$ is the *mailbox* of v and contains the whole information received by v at any step of the computation.

The initial labelling of any vertex v is $(\lambda(v), (0, \emptyset, \emptyset))$.

5.2 An Order on Local Views

The fundamental property of the algorithm is based on a total order on local views, as defined in [15], such that the local view of any vertex cannot decrease during the computation. We assume for the rest of this paper that the set of labels L is totally ordered by $<_L$. Consider a vertex v such that the local view $N(v)$ is the set $\{(p(e_1),\ \lambda(e_1),\ n(v_1),\ \lambda(v_1)),\ (p(e_2),\ \lambda(e_2),\ n(v_2),\ \lambda(v_2)),\ \ldots,\ (p(e_d),\ \lambda(e_d),\ n(v_d),\ \lambda(v_d))\}$, we assume that:

- $p(e_1) \geq p(e_2) \geq ... \geq p(e_d)$,
- if $p(e_i) = p(e_{i+1})$ then $\lambda(e_i) \geq_L \lambda(e_{i+1})$,
- if $p(e_i) = p(e_{i+1})$ and $\lambda(e_i) = \lambda(e_{i+1})$ then $n(v_i) \geq n(v_{i+1})$
- if $p(e_i) = p(e_{i+1})$, $\lambda(e_i) = \lambda(e_{i+1})$ and $n(v_i) = n(v_{i+1})$ then $\lambda(v_i) \geq_L \lambda(v_{i+1})$.

Let $\mathcal{N}_>$ be the set of all such ordered tuples. We define a total order $\prec$ on $\mathcal{N}_>$ by comparing the numbers, then the vertex labels and finally the edge labels. Formally, for two elements

$$((p_1, e_1, n_1, l_1), ..., (p_d, e_d, n_d, l_d)) \text{ and } ((p'_1, e'_1, n'_1, l'_1), ..., (p'_{d'}, e'_{d'}, n'_{d'}, l'_{d'})$$

of $\mathcal{N}_>$ we define

$$((p_1, e_1, n_1, l_1), ..., (p_d, e_d, n_d, l_d)) \prec ((p'_1, e'_1, n'_1, l'_1), ..., (p'_{d'}, e'_{d'}, n'_{d'}, l'_{d'})$$

if there exists i such that $(p_1, e_1, n_1, l_1) = (p'_1, e'_1, n'_1, l'_1), ..., (p_{i-1}, e_{i-1}, n_{i-1}, l_{i-1}) = (p'_{i-1}, e'_{i-1},\ n'_{i-1}, l'_{i-1})$ and such that one of the following holds

1. $p_i < p'_i$,
2. $p_i = p'_i$ and $e_i < e'_i$,
3. $p_i = p'_i$, $e_i = e'_i$ and $n_i < n'_i$,
4. $p_i = p'_i$, $e_i = e'_i$ and $n_i = n'_i$ and $l_i = l'_i$,
5. $i = d + 1$ and $d < d'$.

If $N(u) \prec N(v)$, then we say that the local view $N(v)$ of v is stronger than the one of u and that $N(u)$ is weaker than $N(v)$. The order $\prec$ is a total order on $\mathcal{N} = \mathcal{N}_> \cup \{\emptyset\}$, with, by definition, $\emptyset \prec N$ for every $N \in \mathcal{N}_>$.

5.3 Relabelling Rules

We now describe the five relabelling rules; the rules $\mathcal{M}_2$ and $\mathcal{M}_3$ are very close from the rules of the Mazurkiewicz algorithm. The first rule gives a name to each edge : two neighbours v and v' incident to a common edge e such that $p(e) = 0$ choose a value for $p(e)$ such that each node does not have two incident edges with the same label. This rule can only be applied once to each edge, since once an edge e has a number $p(e)$, this number does not change any more.

$\mathcal{M}_1$:

$$\begin{array}{ccc} (l_1, (n_1, N_1, M_1)) & (l_e, 0) & (l_2, (n_2, N_2, M_2)) \\ \circ & \text{———} & \circ \\ & \downarrow & \\ (l_1, (n_1, N_1', M_1')) & (l_e, p) & (l_2, (n_2, N_2', M_2')) \\ \circ & \text{———} & \circ \end{array}$$

with $p = 1 + \max\{p'; (p', l_e', n', l') \in N_1 \cup N_2\}$
$N_1' = N_1' \cup \{(p, l_e, 0, l_2)\}$
$N_2' = N_2' \cup \{(p, l_e, 0, l_1)\}$
$M_1' = M_1 \cup \{(l_1, n_1, N_1')\}$
$M_2' = M_2 \cup \{(l_2, n_2, N_2')\}$

The second rule enables two neighbours v and v' having different mailboxes to share the information they have about the labels present in the graphs.

$\mathcal{M}_2$:

$$\begin{array}{ccc} (l_1, (n_1, N_1, M_1)) & (l_e, p) & (l_2, (n_2, N_2, M_2)) \\ \circ & \text{———} & \circ \\ & \downarrow & \\ (l_1, (n_1, N_1, M')) & (l_e, p) & (l_2, (n_2, N_2, M')) \\ \circ & \text{———} & \circ \end{array}$$

if $p > 0$ and $M_1 \neq M_2$
with $M' = M_1 \cup M_2$

The third rule enables a vertex v to change its number if $n(v) = 0$ or if there exists a vertex v' such that $n(v) = n(v')$ and v has a weaker local view than v'.

$\mathcal{M}_3$:

$$\begin{array}{ccc} (l, (n, N, M)) & \longrightarrow & (l, (k, N, M')) \\ \circ & & \circ \end{array}$$

if $n = 0$ or $\exists (n, l_0, N_0) \in M$ such that $l <_L l_0$ or $l = l_0$ and $N \prec N_0$
with $k = 1 + \max\{n_1; (l_1, n_1, N_1) \in M\}$
$M' = M \cup \{(l, k, N)\}$

The fourth rule enables a node having a neighbour with exactly the same label to change its number. If this rule can be applied, it means that the two vertices have never exchange their number along this edge.

$\mathcal{M}_4$:

$$\begin{array}{ccc} (l,(n,N,M)) & & (l,(n,N,M)) \\ \circ & \xrightarrow{(l_e,p)} & \circ \\ (l,(k,N_1,M')) & & (l,(n,N_2,M')) \\ \circ & \xrightarrow{(l_e,p)} & \circ \end{array}$$

if $\quad p > 0$ and $n > 0$
with $k = 1 + \max\{n_1; (l_1, n_1, N_1) \in M\}$
$\quad N_1 = N \setminus \{(p, l_e, 0, l)\} \cup \{(p, l_e, n, l)\}$
$\quad N_2 = N \setminus \{(p, l_e, 0, l)\} \cup \{(p, l_e, k, l)\}$
$\quad M' = M \cup \{(l, k, N_1), (l, n, N_2)\}$

The fifth rule enables a vertex v to get information about the number of a neighbour v', either because v has no information about $n(v')$, or because $n(v')$ has changed since v got information about $n(v')$.

$\mathcal{M}_5$:

$$\begin{array}{ccc} (l_1,(n_1,N_1,M)) & & (l_2,(n_2,N_2,M)) \\ \circ & \xrightarrow{(l_e,p)} & \circ \\ (l_1,(n_1,N_1',M')) & & (l_2,(n_2,N_2',M')) \\ \circ & \xrightarrow{(l_e,p)} & \circ \end{array}$$

if $\quad p > 0, n_1 > 0, n_2 > 0, n_1 \neq n_2$
$\quad (p, l_e, i, l_2) \in N_1, (p, l_e, j, l_1) \in N_2$
$\quad$ and $i \neq n_2$ or $j \neq n_1$
with $N_1' = N_1 \setminus \{(p, l_e, i, l_2)\} \cup \{(p, l_e, n_2, l_2)\}$
$\quad N_2' = N_2 \setminus \{(p, l_e, j, l_1)\} \cup \{(p, l_e, n_1, l_1)\}$
$\quad M' = M \cup \{(l_1, n_1, N_1'), (l_2, n_2, N_2')\}$

For each run of this algorithm on a minimal graph $\mathbf{G}$ each vertex has a unique number. Finally:

Theorem 1. *For every graph* $\mathbf{G}$*, there exists an enumeration algorithm with termination detection on* $\mathbf{G}$ *and an election algorithm on* $\mathbf{G}$ *using local computations on closed edges if and only if* $\mathbf{G}$ *is a minimal graph.*

6 Two other Models of Local Computations

We consider now a different kind of local computations: we still consider locally generated relabelling relations, but during a relabelling step, the label of only one vertex and an incident edge can be modified, i.e., the form of the rules is :

$$R \ : \ \overset{\mathbf{X}}{\circ} \overset{\mathbf{Y}}{\text{———}} \overset{\mathbf{Z}}{\circ} \qquad \longrightarrow \qquad \overset{\mathbf{X}'}{\circ} \overset{\mathbf{Y}'}{\text{———}} \overset{\mathbf{Z}}{\circ}$$

To make a distinction between this model and the former one, we will say that model describe local computations on open edges. Since local computations on

open edges are also local computations on closed edges, each algorithm using local computations on open edges is also an algorithm using local computations on closed edges. We wonder if the power of computation of this new model is weaker or is the same as the precedent one. In fact, by a non trivial proof we have:

Proposition 3. *Given a problem P and a graph $\mathbf{G}$, there exists an algorithm using local computations on closed edges on $\mathbf{G}$ with termination detection if and only if there exists an algorithm using local computations on open edges that solves P on $\mathbf{G}$ with termination detection.*

We have already given a characterization of graphs in which we can solve the election problem and the enumeration problem with termination detection and we can therefore give the following corollary:

Corollary 1. *For every graph $\mathbf{G}$, there exists an enumeration algorithm with termination detection on $\mathbf{G}$ and an election algorithm on $\mathbf{G}$ using local computations on open edges if and only if $\mathbf{G}$ is a minimal graph.*

We now consider a model of local computations such that at each computation step, a vertex looks at the labels of its neighbours and its incident edges and modify its label and the labels of its incident edges. We say that at each step a star graph is relabelled and we talk about local computations on open star graphs. The relabelling rule are therefore triples (S, λ, λ') such that S is a star graph whose center is a node v_0 and λ, λ' are two labellings of S such that for every node $v \in V(G) \setminus \{v_0\}, \lambda(v) = \lambda'(v)$.

Theorem 2. *For every graph $\mathbf{G}$, there exists an enumeration algorithm with termination detection on $\mathbf{G}$ and an election algorithm on $\mathbf{G}$ using local computations on open star graphs if and only if $\mathbf{G}$ is a minimal graph.*

7 Is it Important to Have Labels on Edges ?

The power of the model of Mazurkiewicz does not change if we consider edges with or without labels.

In our models, we have considered labelled graphs such that the edges can have labels and this property has been used to describe the different algorithms we present. We wonder if the results remain true when we consider models where the edges cannot be labelled. We will present a minimal graph in which we cannot find an election algorithm using local computations on closed edges when the edges are not labelled and another minimal graph in which there does not exist any election algorithm using local computations on open star graphs if the edges cannot be labelled.

Local Computations on Closed Edges.

Consider the graph G described in Figure 3 which is a minimal graph and therefore we can solve the election problem with local computations on closed

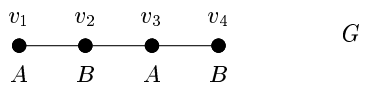


Fig. 3. A graph in which we cannot find an election algorithm using local computations on closed edges without labelling edges.

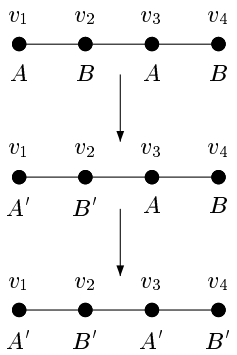
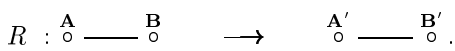


Fig. 4. Application of a relabelling rule

edges. Consider a noetherian relabelling relation $\mathcal{R}$ associated to an algorithm involving local computations on closed edges such that there is not any rule that labels the edges.

We prove by induction that there exist an execution of $\mathcal{R}$ such that the vertices v_1 and v_3 (resp. v_2 and v_4) have the same labels. Initially, the result is true and if at a step $i+1$, a rule R is applied, this rule has the following form:



As described in Figure 4, the rule R can be applied to the nodes v_1 and v_2 and then to the nodes v_3 and v_4: the property holds.

Local Computations on Open Star Graphs.

Consider the graph G described in Figure 5 which is a minimal graph and for which there exists an election algorithm using local computations on open star graphs. Suppose now that we can find an enumeration algorithm $\mathcal{A}$ using local computations on open star graphs such that the rules involved do not label the edges, i.e., the only label that changes in a relabelling step is the label of the center of the star graph involved.

Each time a rule is applied to v_1 or v_2, the same rule can also be applied to the other one and each time a rule is applied to v_3, v_4 or v_5, the same rule can be applied to the other ones. Therefore, we can find an execution of $\mathcal{A}$ such that the vertices v_1 and v_2 (resp. v_3, v_4 and v_5) have the same labels and consequently, we cannot find an election algorithm on G.

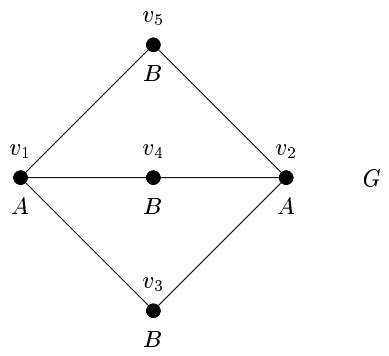


Fig. 5. A graph in which we cannot find an election algorithm using local computations on open star graphs without labelling edges.

References

1. D. Angluin. Local and global properties in networks of processors. In *Proceedings of the 12th Symposium on Theory of Computing*, pages 82–93, 1980.
2. H. Attiya and J. Welch. *Distributed computing: fundamentals, simulations, and advanced topics.* McGraw-Hill, 1998.
3. P. Boldi, B. Codenotti, P. Gemmell, S. Shammah, J. Simon, and S. Vigna. Symmetry breaking in anonymous networks: Characterizations. In *Proc. 4th Israeli Symposium on Theory of Computing and Systems*, pages 16–26. IEEE Press, 1996.
4. P. Boldi and S. Vigna. Computing anonymously with arbitrary knowledge. In *Proceedings of the 18th ACM Symposium on principles of distributed computing*, pages 181–188. ACM Press, 1999.
5. P. Boldi and S. Vigna. An effective characterization of computability in anonymous networks. In Jennifer L. Welch, editor, *Distributed Computing. 15th International Conference, DISC 2001*, volume 2180 of *Lecture Notes in Computer Science*, pages 33–47. Springer-Verlag, 2001.
6. P. Boldi and S. Vigna. Fibrations of graphs. *Discrete Math.*, 243:21–66, 2002.
7. E. Godard and Y. Métivier. A characterization of families of graphs in which election is possible (*ext. abstract*). In M. Nielsen and U. Engberg, editors, *Proc. of Foundations of Software Science and Computation Structures, FOSSACS'02*, number 2303 in LNCS, pages 159–171. Springer-Verlag, 2002.
8. E. Godard, Y. Métivier, and A. Muscholl. Characterization of Classes of Graphs Recognizable by Local Computations. *Theory of Computing Systems.* to appear.
9. E. Godard, Y. Métivier, and G. Tel. Election, termination and graph cartography. in preparation.
10. G. LeLann. Distributed systems: Towards a formal approach. In B. Gilchrist, editor, *Information processing'77*, pages 155–160. North-Holland, 1977.
11. L. Lamport and N. Lynch. Distributed computing: models and methods. *Handbook of theoretical computer science*, B:1157–1199, 1990.
12. N. A. Lynch. *Distributed algorithms.* Morgan Kaufman, 1996.
13. W. S. Massey. *A basic course in algebraic topology.* Springer-Verlag, 1991. Graduate texts in mathematics.

14. A. Mazurkiewicz. Trace theory. In W. Brauer et al., editor, *Petri nets, applications and relationship to other models of concurrency*, volume 255 of *Lecture notes in computer science*, pages 279–324. Spinger-Verlag, 1987.
15. A. Mazurkiewicz. Distributed enumeration. *Inf. Processing Letters*, 61:233–239, 1997.
16. Y. Métivier and G. Tel. Termination detection and universal graph reconstruction. In *SIROCCO 00 - 7th International Colloquium on Structural Information & Communication Complexity*, pages 237–251, 2000.
17. K. Reidemeister. *Einführung in die Kombinatorische Topologie.* Vieweg, Brunswick, 1932.
18. K. H. Rosen, editor. *Handbook of discrete and combinatorial mathematics.* CRC Press, 2000.
19. G. Tel. *Introduction to distributed algorithms.* Cambridge University Press, 2000.
20. M. Yamashita and T. Kameda. Computing functions on asynchronous anonymous networks. *Math. Systems Theory*, 29:331–356, 1996.
21. M. Yamashita and T. Kameda. Computing on anonymous networks: Part i - characterizing the solvable cases. *IEEE Transactions on parallel and distributed systems*, 7(1):69–89, 1996.
22. M. Yamashita and T. Kameda. Computing on anonymous networks: Part ii - decision and membership problems. *IEEE Transactions on parallel and distributed systems*, 7(1):90–96, 1996.
23. M. Yamashita and T. Kameda. Leader election problem on networks in which processor identity numbers are not distinct. *IEEE Transactions on parallel and distributed systems*, 10(9):878–887, 1999.