Distrib. Comput.
DOI 10.1007/s00446-012-0163-y

Election in partially anonymous networks with arbitrary knowledge in message passing systems

Jérémie Chalopin · Emmanuel Godard · Yves Métivier

Received: 2 November 2010 / Accepted: 27 December 2011

Abstract This paper attempts to find an answer to an open question of Angluin in her seminal paper (1980) about the election problem for families of graphs (Section 4, page 87). More precisely, we characterize families of (labelled) graphs which admit an election algorithm in the message passing model by using the notion of quasi-coverings which captures "*the existence of large enough area of one graph that looks locally like another graph*".

1 Introduction

The election problem is one of the paradigms of the theory of distributed computing. A distributed algorithm solves the election problem if it always terminates and in the final configuration exactly one process is marked as *elected* and all the other processes are labelled *non-elected*. Moreover, it is supposed that once a process becomes *elected* or *non-elected* then it remains in such a state until the end of the algorithm. Election algorithms constitute a building block of many other distributed algorithms: The elected vertex acts as coordinator, initiator, and more generally performs some special role (cf. [25] p. 262).

The election problem was first studied by LeLann [15] who gives a solution in rings where each process has a unique name. Solutions to this problem are studied under two classical assumptions (see ([22], Chapter 3) for details):

- each process is identified by a unique name (or identifier): its identity;
- processes have initially the same state (anonymous networks).

Several conditions were found to allow election algorithms, according the model, for each of these conditions specific algorithms are given. For example:

1. the network is a ring and each node has a unique name [15];
2. the network is known to be a tree or a complete graph [1];
3. the size of the network is known and the network is minimal for the covering relation [18].

If processes have initially unique identifiers, it is always possible to solve this problem by electing the process with the smallest identifier. Nevertheless, if we consider *anonymous* networks where processes do not have identifiers and execute the same algorithm, it is not always possible to solve the election problem. Angluin [1] has introduced the classical proof techniques used for showing the non-existence of an election algorithm based on coverings, which is a notion known from algebraic topology [16]. Finally, several characterizations of graphs for which there exists an election algorithm have been obtained [2,18,26].

J. Chalopin · E. Godard
Laboratoire d'Informatique Fondamentale de Marseille,
CNRS & Aix-Marseille Université, CMI,
39 rue Joliot-Curie, 13453, Marseille Cedex 13, France
e-mail: jeremie.chalopin@lif.univ-mrs.fr

E. Godard
e-mail: emmanuel.godard@lif.univ-mrs.fr

Y. Métivier (✉)
LaBRI UMR CNRS 5800, Université de Bordeaux,
351 cours dela Libération, 33405 Talence, France
e-mail: metivier@labri.fr

Published online: 22 February 2012

1.1 The problem and some motivations

The next question concerns the existence of an election algorithm which works for all the graphs of a given family of graphs. For example Angluin proves in [1] (by considering her model):

- there exists an election algorithm for the family of complete graphs (Theorem 4.1),
- there exists an election algorithm for the family of trees (Theorem 4.4),
- there is no election algorithm for the family of graphs containing all trees and the triangle (Theorem 4.6).

No characterization of families of graphs which admit an election algorithm exists and the aim of this paper is to present such a characterization. To obtain this characterization we use the notion of quasi-coverings. This notion captures the phenomenon which appears in the family formed by "trees and a triangle" and quoted by Angluin: "*the existence of a large enough area of one graph that looks locally like another graph*" ([1] p. 87, l. 13–17).

Furthermore, election algorithms for families of graphs, as explained by Yamashita and Kameda ([27] p. 878), work on dynamic or decentralized systems and tolerant to some transient faults; as examples we obtain by this way election algorithms under intermediate assumptions or knowledge such as:

- vertices have names which are not necessarily distinct: partially anonymous graphs,
- each vertex knows an upper bound of the size of the graph,
- each vertex knows an upper bound of the diameter of the graph,
- a combination of these hypothesises.

Distributed algorithms for families of graphs have implications for the design of more portable software (as explained by Santoro, [22, p.185]). Therefore it enables to build more maintainable systems because when the network is evolving, there is no need to update the software. Furthermore, for reasons of privacy or of security, some informations (size, names) may be hidden and an algorithm which works on families would be an answer in this case.

1.2 The model

Our model is the usual asynchronous message passing model [24,26]. A network is represented by a simple connected graph $G = (V(G), E(G))$ where vertices correspond to processes and edges to direct communication links. The state of each process is represented by a label $\lambda(v)$ associated to the corresponding vertex $v \in V(G)$; we denote by $\mathbf{G} = (G, \lambda)$ such a labelled graph.

We assume that each process can distinguish the different edges that are incident to it, i.e., for each $u \in V(G)$ there exists a bijection δ_u between the neighbours of u in G and $[1, \deg_G(u)]$. We will denote by δ the set of functions $\{\delta_u \mid u \in V(G)\}$. The numbers associated by each vertex to its neighbours are called *port-numbers* and δ is called a *port-numbering* of G. We will denote by $(\mathbf{G}, \delta)$ the labelled graph $\mathbf{G}$ with the port-numbering δ.

Each process v in the network represents an entity that is capable of performing computation steps, sending messages via some port and receiving any message via some port that was sent by the corresponding neighbour. We consider asynchronous systems, i.e., each computation may take an unpredictable (but finite) amount of time. Note that we consider only reliable systems: no fault can occur on processes or communication links. We also assume that the channels are FIFO, i.e., for each channel, the messages are delivered in the order they have been sent. In this model, a distributed algorithm is given by a local algorithm that all processes should execute (note that all the processes have the same algorithm). A local algorithm consists of a sequence of computation steps interspersed with instructions to send and to receive messages.

1.3 The main result and its construction

Let (G, λ) be a labelled graph with the port-numbering δ. We will denote by $(\mathrm{Dir}(\mathbf{G}), \delta)$ the symmetric labelled digraph $(\mathrm{Dir}(G), (\lambda, \delta))$ constructed in the following way. The vertices of $\mathrm{Dir}(G)$ are the vertices of G and they have the same labels in $\mathbf{G}$ and in $\mathrm{Dir}(\mathbf{G})$. Each edge $\{u, v\}$ of G is replaced in $(\mathrm{Dir}(\mathbf{G}), \delta)$ by two arcs $a_{(u,v)}, a_{(v,u)} \in A(\mathrm{Dir}(G))$ such that $s(a_{(u,v)}) = t(a_{(v,u)}) = u$, $t(a_{(u,v)}) = s(a_{(v,u)}) = v$, $\delta(a_{(u,v)}) = (\delta_u(v), \delta_v(u))$ and $\delta(a_{(v,u)}) = (\delta_v(u), \delta_u(v))$. Note that this digraph does not contain multiple arcs or loop. The object we use for our study is $(\mathrm{Dir}(G), (\lambda, \delta))$ and results are stated with symmetric labelled digraphs (directed graphs).

1.3.1 First step: an election algorithm for a labelled graph (already known)

Distributed tasks like election require the network to reach a *non-symmetric* state. A network state is symmetric if it contains different nodes that are in exactly the same situation; not only their local states, but also the states of their neighbors, of their neighbors' neighbors, etc. That is, there exists a "local similarity" between different nodes of infinite radius.

The replay argument shows that different nodes that are locally similar with infinite radius will exhibit the same behavior in some infinite computation. Thus, there is no algorithm

that guarantees that the symmetry ceases in all finite computations. Symmetry can be broken only by randomized protocols.

It is not difficult to see that local similarity of infinite radius may exist in finite graphs. It is precisely captured by the notion of covering used by Angluin and this is the mathematical tool to prove the existence of symmetries of infinite radius.

Networks in which symmetries exist are non minimal for the covering relation and impossibility of symmetry breaking can be shown for these graphs. In particular the election problem has no solution (Theorem 3.2). For a given graph, the characterization of Yamashita and Kameda [26] corresponds to minimal (di)graphs for the symmetric covering relation. Thus in the sequel we consider families of (symmetric) labelled (di)graphs minimal for the covering relation; for such labelled graphs there exists an election algorithm: [2,7,26]). In this work we use the election algorithm $\mathcal{M}$ (Sect. 3.3) given in [7].

1.3.2 Second step: an election algorithm for a family of labelled graphs

When we consider families of graphs, the existence of an election algorithm for this family is closely related to the problem of the termination detection of a distributed algorithm.

Termination detection requires that a node certifies, in a finite computation, that all nodes of the network have completed their computation. However, in a finite computation only information about a bounded region in the network can be gathered. The algorithm by Szymanski, Shy, and Prywes [23] does this for a region of pre-specified diameter; the assumption is necessary that a bound of the diameter of the entire network is known. This implies, that termination detection, unlike symmetry breaking, is possible in every graph, but provided some knowledge.

The termination detection algorithm by Szymanski et al. can be generalized in this way to work in a graph family $\mathcal{I}$. Nodes observe their neighborhood and determine in what graph H of $\mathcal{I}$ they are. Then they try to get a bound k on the radius to which a different graph of $\mathcal{I}$ can be locally similar to H, and then certify that all nodes within distance k are completed. The election algorithm for a family of graphs thus combines an universal election algorithm for a graph and a known termination detection algorithm.

Of course the approach fails if a graph $H \in \mathcal{I}$ is locally similar, with *unbounded* radius, to other graphs in $\mathcal{I}$. Local similarities of this type are made precise in the notion of *quasi-coverings* introduced in [21] and refined in [10]. Fortunately, the impossibility proof for the existence of an election can be extended to cover exactly those families of graphs that contain such unbounded-radius quasi-coverings.

Consequently, the election algorithm we give is the most general election algorithm possible, it is summarized by:

Theorem 1.1 *Let $\mathcal{I}$ be a recursive family of connected symmetric labelled digraphs. There exists an election algorithm for $\mathcal{I}$ if and only if every labelled digraphs of $\mathcal{I}$ is symmetric covering minimal, and there exists a computable function $\tau : \mathcal{I} \to \mathbb{N}$ such that for every labelled symmetric digraph $\mathbf{D}$ of $\mathcal{I}$, there is no quasi-covering of $\mathbf{D}$ of radius greater than $\tau(\mathbf{D})$ in $\mathcal{I}$, except $\mathbf{D}$ itself.*

Remark 1.2 This result is not specific to the asynchronous message passing model. As it is indicated in Sect. 5, it may be extended to the models of Angluin [1] and Mazurkiewicz [18], to the synchronous message passing model defined by Hoare [13] and more generally to the model of local computations on labelled edges [8].

We illustrate applications of this theorem through several examples (as Angluin's results cited above).

1.4 Related works: comparison and comments

Among studies related to our result one may cite [27]: authors consider the same model as in this paper (the asynchronous message passing model with port-numberings) and give a charaterization of networks which admit an election algorithm under the assumption that process identity numbers are not distinct, furthermore they assume that each process knows the size of the network.

In [2], the authors describe how to elect in anonymous networks within different communication models, assuming that each process knows the size of the network. In the case where the size of the network is not known, a "weak election" algorithm is presented.

In [4], Boldi and Vigna give an effective characterization of computability in anonymous networks. In this work, networks are directed graphs coloured on their arcs and each processor changes its state depending on its previous state and on the states of its in-neighbours. Processes start from the same state, use the same algorithm and know an upper bound of the size of the network.

We can note that the fundamental tool in [2,4,27] is the notion of view. The view from a vertex v of a labelled graph (G, λ) is an infinite labelled tree rooted in v obtained by considering all labelled walks in (G, λ) starting from v.

The characterization of graphs where election is possible obtained in [27] is formulated by using views whereas Boldi et al. [2] use fibrations. In both cases election algorithms are based on views and the election algorithms presented in [2,27] use messages with an exponential size. All executions are pseudo-synchronous and communication links behave like FIFO channels.

Techniques developed in this paper are inspired by the work of Mazurkiewicz [18]. He considers the asynchronous computation model where in one computation step labels of vertices are modified on a subgraph consisting of a node and its neighbours, according to rules depending on this subgraph only. Mazurkiewicz's characterization of the graphs where enumeration/election are possible is based on the notion of unambiguous graphs and may be formulated equivalently using coverings of simple graphs (see [11], p. 256). A graph G is a covering of another graph G' if there is a surjective homomorphism φ from G to G' which is locally bijective. He gives a nice and simple enumeration algorithm for the graphs that are minimal for the covering relation, i.e., which can cover only themselves. The fundamental tool is a total order attached to local views defined by a vertex and its neighbourhood. As consequence, our algorithms are totally asynchronous, messages are not necessarily FIFO and their sizes are polynomial.

In [10], the authors give a complete characterization of families of networks that admit an election algorithm in the model of Mazurkiewicz. Nevertheless, techniques used for quasi-coverings and the SSP algorithm in Sects. 4 and 5 are new.

More recently, [9,20] consider also intermediate cases in a ring. In the first paper authors treat the problem of election in the case of non unique labels and bounds on the size. More generally, the second paper consider computability of relations under the same assumptions.

Universal election algorithms are presented in [22] (Section 3.8); they assume that vertices have identities.

1.5 Summary

Section 2 reviews basic notions of digraphs, labelled digraphs and the model. Section 3 presents symmetric coverings and their links with the election problem. It presents also an election algorithm for a given labelled graph. Section 4 is devoted to the problem of an election algorithm for a family of labelled digraphs; it introduces quasi-coverings and gives a necessary condition for the existence of an election algorithm for a family of labelled digraphs. Section 5 presents an election algorithm for a family of labelled graphs and finally gives a characterization for the existence of a such algorithm. Section 6 gives some applications of the characterization.

2 Preliminaries

This part gives some notions on labelled digraphs we used in this paper and some precisions on the model.

2.1 Labelled digraphs

In the following, we will consider directed graphs (digraphs) with multiple arcs and self-loops. A *digraph* $D = (V(D), A(D), s_D, t_D)$ is defined by a set $V(D)$ of vertices, a set $A(D)$ of arcs and by two maps s_D and t_D that assign to each arc two elements of $V(D)$: a source and a target (in general, the subscripts will be omitted). If a is an arc, the arc a is said to be going out of $s(a)$ and coming into $t(a)$; we also say that $s(a)$ and $t(a)$ are incident to a. Let a be an arc, if $s(a) = u$ and $t(a) = v$ then v is an out-neighbour of u and u is an in-neighbour of v.

A *symmetric* digraph D is a digraph endowed with a symmetry, that is, an involution $Sym : A(D) \to A(D)$ such that for every $a \in A(D)$, $s(a) = t(Sym(a))$. In a symmetric digraph D, the degree of a vertex v is $\deg_D(v) = |\{a \mid s(a) = v\}| = |\{a \mid t(a) = v\}|$ and we denote by $N_D(v)$ the set of neighbours of v which is equal to the set of out-neighbours of v and to the set of in-neighbours of v.

Given two vertices $u, v \in V(D)$, a *path* π of *length* p from u to v in D is a sequence of arcs $a_1, a_2, \ldots a_p$ such that $s(a_1) = u$, $\forall i \in [1, p-1]$, $t(a_i) = s(a_{i+1})$ and $t(a_p) = v$. If for each $i \in [1, p-1]$, $a_{i+1} \neq Sym(a_i)$, π is *non-stuttering*. A digraph D is *strongly connected* if for all vertices $u, v \in V(D)$, there exists a path from u to v in D. In a symmetric digraph D, the *distance* between two vertices u and v, denoted $\mathrm{dist}_D(u, v)$ is the length of the shortest path from u to v in D.

A *homomorphism* γ between the digraph D and the digraph D' is a mapping $\gamma : V(D) \cup A(D) \to V(D') \cup A(D')$ such that for each arc $a \in A(D)$, $\gamma(s(a)) = s(\gamma(a))$ and $\gamma(t(a)) = t(\gamma(a))$. A homomorphism $\gamma : D \to D'$ is an *isomorphism* if γ is bijective, in this case, we note $D \simeq D'$.

Throughout the paper we will consider digraphs where the vertices and the arcs are labelled with labels from a recursive label set L. A digraph D labelled over L will be denoted by (D, λ), where $\lambda : V(D) \cup A(D) \to L$ is the labelling function. A mapping $\gamma : V(D) \cup A(D) \to V(D') \cup A(D')$ is a homomorphism from (D, λ) to (D', λ') if γ is a digraph homomorphism from D to D' which preserves the labelling, i.e., such that $\lambda'(\gamma(x)) = \lambda(x)$ for every $x \in V(D) \cup A(D)$. Labelled digraphs will be designated by bold letters like $\mathbf{D}, \mathbf{D}', \ldots$

In a symmetric digraph $\mathbf{D}$, we denote by $\mathbf{B_D}(v_0, r)$, the labelled ball of center $v_0 \in V(D)$ and of radius r that contains all vertices at distance at most r of v_0 and all arcs whose source or target is at distance at most $r - 1$ of v_0.

Given a set of labels L, we denote by $\mathcal{D}_L$ the set of all symmetric digraphs $\mathbf{D} = (D, \lambda)$ where for each $a \in A(D)$, there exist $p, q \in \mathbb{N}$ such that $\lambda(a) = (p, q)$ and $\lambda(Sym(a)) = (q, p)$ and for each $v \in V(D)$, $\lambda(v) \in L$ and $\{p \mid \exists a, \lambda(a) = (p, q) \text{ and } s(a) = v\} = [1, \deg_D(v)]$. In other words, $\mathcal{D}_L$ is the set of digraphs that locally look like some digraphs

obtained from a simple labelled graph $\mathbf{G}$ with a port-numbering whose labels belong to L.

2.2 More precisions on the model

We sometimes refer to the *synchronous* execution of an algorithm. Such an execution is a particular execution of the algorithm that can be divided in *rounds*. In each round, each process receives all the messages that have been sent to it by its neighbours in the previous round; then according to the information it gets, it can modify its state and send messages to its neighbours before entering the next round. Note that the synchronous execution of an algorithm is just a special execution of the algorithm and thus it belongs to the set of asynchronous executions of this algorithm.

Remark 2.1 Given a simple connected labelled graph $\mathbf{G} = (G, \lambda)$ with a port-numbering δ, let $\mathbf{D} = (\mathrm{Dir}(\mathbf{G}), \delta)$ be the corresponding labelled digraph $(\mathrm{Dir}(G), (\lambda, \delta))$. Let $\mathcal{A}$ be a distributed algorithm. we speak indifferently of an execution of $\mathcal{A}$ on $(\mathbf{G}, \delta)$ or on $\mathbf{D}$.

3 Symmetric coverings and the election problem for a labelled graph

This section presents a first tool: symmetric coverings, then it recalls the characterization of labelled graphs which admit an election and it presents the election algorithm and its main properties we use later.

3.1 Symmetric coverings

The notion of symmetric coverings is fundamental in this work; definitions and main properties are presented in [5]. This notion enables to express "similarity" between two digraphs.

A labelled digraph $\mathbf{D}$ is a *covering* of a labelled digraph $\mathbf{D}'$ via φ if φ is a homomorphism from $\mathbf{D}$ to $\mathbf{D}'$ such that each arc $a' \in A(D')$ and for each vertex $v \in \varphi^{-1}(t(a'))$ (resp. $v \in \varphi^{-1}(s(a'))$, there exists a unique arc $a \in A(D)$ such that $t(a) = v$ (resp. $s(a) = v$) and $\varphi(a) = a'$.

A symmetric labelled digraph $\mathbf{D}$ is a *symmetric covering* of a symmetric labelled digraph $\mathbf{D}'$ via φ if $\mathbf{D}$ is a covering of $\mathbf{D}'$ via φ and if for each arc $a \in A(D)$, $\varphi(Sym(a)) = Sym(\varphi(a))$. The homomorphism φ is a *symmetric covering projection* from $\mathbf{D}$ to $\mathbf{D}'$.

The following lemma shows the importance of symmetric coverings when we deal with anonymous networks. This is the counterpart of the lifting lemma that Angluin gives for coverings of simple graphs [1] and the proof can be found in [2,7].

Lemma 3.1 (Lifting Lemma [2]) *Let* $\mathbf{D}$ *and* $\mathbf{D}'$ *be two labelled symmetric digraphs of* $\mathcal{D}_L$. *If* $\mathbf{D}$ *is a symmetric covering of* $\mathbf{D}'$ *via* φ, *then any execution of an algorithm* $\mathcal{A}$ *on* $\mathbf{D}'$ *can be lifted up to an execution on* $\mathbf{D}$, *such that at the end of the execution, for any* $v \in V(D)$, v *is in the same state as* $\varphi(v)$.

A symmetric labelled digraph $\mathbf{D}$ is *symmetric covering minimal* if there does not exist any symmetric labelled digraph $\mathbf{D}'$ not isomorphic to $\mathbf{D}$ such that $\mathbf{D}$ is a symmetric covering of $\mathbf{D}'$.

3.2 Election in a labelled graph and symmetric coverings

First, we give a characterization of networks where election can be solved in the asynchronous message passing system.

Theorem 3.2 ([2,7]) *Given a simple labelled graph* $\mathbf{G} = (G, \lambda)$ *with a port-numbering* δ, *there exists an election (or a naming) algorithm for* $(\mathbf{G}, \delta)$ *if and only if* $(Dir(G), (\lambda, \delta))$ *is symmetric covering minimal.*

The necessary part of this theorem is a direct consequence of Lemma 3.1. The sufficient part needs the following algorithm (it will be used later).

3.3 An election algorithm for a symmetric covering minimal labelled digraph

The aim of a naming algorithm is to arrive at a final configuration where all processes have unique identities. Again this is an essential prerequisite to many other distributed algorithms which work correctly only under the assumption that all processes can be unambiguously identified. The enumeration problem is a variant of the naming problem. The aim of a distributed enumeration algorithm is to attribute to each network vertex a unique integer in such a way that this yields a bijection between the set $V(G)$ of vertices and $\{1, 2, \ldots, |V(G)|\}$.

In this section we describe an enumeration algorithm; by this way we obtain an election algorithm by considering that the vertex having the number $|V(G)|$ is elected (we assume that vertices know $|V(G)|$). This algorithm is presented in [7], it is inspired by the enumeration algorithm given by Mazurkiewicz in [18].

3.3.1 Informal description

We first give a general description of our algorithm, that will be denoted $\mathcal{M}$, when executed on a connected labelled simple graph $\mathbf{G}$ with a port-numbering δ.

During the execution of the algorithm, each vertex v attempts to get its own unique identity which is a number between 1 and $|V(G)|$. Once a vertex v has chosen a number $n(v)$, it sends it to each neighbour u with the port-number $\delta_v(u)$. When a vertex u receives a message from one

neighbour v, it stores the number $n(v)$ with the port-numbers $\delta_u(v)$ and $\delta_v(u)$. From all information it has gathered from its neighbours, each vertex can construct its *local view* (which is the set of numbers of its neighbours associated with the corresponding port-numbers). Then, a vertex broadcasts its number, its label and its mailbox (which contains a set of *local views*). If a vertex u discovers the existence of another vertex v with the same number then it should decide if it changes its identity. To this end it compares its local view with the local view of v. If the label of u or the local view of u is "weaker", then u picks another number — its new temporary identity — and broadcasts it again with its local view. At the end of the computation, if the digraph $(Dir(G), (\lambda, \delta))$ is symmetric covering minimal, then every vertex will have a unique number: the algorithm is a naming algorithm.

3.3.2 Labels

We consider a network $(\mathbf{G}, \delta)$ where $\mathbf{G} = (G, \lambda)$ is a simple labelled graph and where δ is a port-numbering of $\mathbf{G}$.

The function $\lambda : V(G) \to L$ is the initial labelling. We assume there exists a total order $<_L$ on L. We extend the order $<_L$ to $L \cup \{\perp\}$ (assuming that $\perp \notin L$) as follows: for all $\ell \in L$, $\perp < \ell$.

During the execution, the label of each v is a tuple $(\lambda(v), n(v), N(v), M(v))$ where:

- $\lambda(v) \in L$ is the initial label of v.
- $n(v) \in \mathbb{N}$ is the current *number* of v computed by the algorithm; initially $n(v) = 0$.
- $N(v) \in \mathcal{P}_{\text{fin}}(\mathbb{N} \times L \times \mathbb{N}^2)$[1] is the *local view* of v. At the end of the execution, if $(m, \ell, p, q) \in N(v)$, then v has a neighbour u whose number is m, whose label is ℓ and the arc from u to v is labelled (p, q). Initially $N(v) = \{(0, \perp, 0, q) \mid q \in [1, \deg_G(v)]\}$.
- $M(v)$ is a set, it is the *mailbox* of v; initially $M(v) = \emptyset$. An element of $M(v)$ has the following form: (m, ℓ, N) where $m \in \mathbb{N}$, $\ell \in L$ and N is a local view. It contains all information received by v during the execution of the algorithm. If $(m, \ell, N) \in M(v)$, it means that at some previous step of the execution, there was a vertex u such that $n(u) = m$, $\lambda(u) = \ell$ and $N(u) = N$.

3.3.3 Messages

Processes exchange messages of the form $< (n, \ell, M), p >$. If a vertex u sends a message $< (n, \ell, M), p >$ to one of its neighbour v, then the message contains following information: n is the current number $n(u)$ of u, ℓ is the label $\lambda(u)$ of u, M is the mailbox of u, and $p = \delta_u(v)$.

[1] For any set S, $\mathcal{P}_{\text{fin}}(S)$ denotes the set of finite subsets of S.

3.3.4 An order on local views

The interesting properties of the algorithm rely on a total order on local views.

Given two distinct sets $N_1, N_2 \in \mathcal{P}_{\text{fin}}(\mathbb{N} \times L \times \mathbb{N}^2)$, we define $N_1 \prec N_2$ if the maximum of the symmetric difference $N_1 \triangle N_2 = (N_1 \setminus N_2) \cup (N_2 \setminus N_1)$ for the lexicographic order belongs to N_2.

One also says that $(\ell, N) \prec (\ell', N')$ if either $\ell <_L \ell'$, or $\ell = \ell'$ and $N \prec N'$. We denote by $\preceq$ the reflexive closure of $\prec$.

3.4 The election algorithm $\mathcal{M}$

The algorithm for the vertex v_0 (see Algorithm 1) is expressed in an event-driven description (see Tel [24] p. 553). A vertex which executes one of the following actions is said active.

The action **I** can be executed by a process on wake-up only if it has not received any message. In this case, it chooses the number 1, updates its mailbox and informs its neighbours.

The action **R** describes the instructions the vertex v_0 has to follow when it receives a message $< (n_1, \ell_1, M_1), p_1 >$ from a neighbour via port q_1. First, it memorizes and it updates its mailbox by adding M_1 to it. Then it modifies its number if it is equal to 0 or if there exists $(n(v_0), \ell', N') \in M(v_0)$ such that $(\lambda(v_0), N(v_0)) \prec (\ell', N')$. Then, it updates its local view by removing elements which corresponds to the port q_1 (if they exist) and by adding (n_1, ℓ_1, p_1, q_1) to $N(v_0)$. Then, it adds its new state to its mailbox. Finally, if its mailbox has been modified by the execution of all these instructions, it sends its number and its mailbox to all its neighbours.

If the mailbox of v_0 is not modified by the execution of the action **R**, it means that the information v_0 has about its neighbour (i.e., its number) was correct, that all the elements of M_1 already belong to $M(v_0)$, and that for each $(n(v_0), \ell, N) \in M(v_0)$, $(\ell, N) \preceq (\lambda(v_0), N(v_0))$.

3.5 Some properties of algorithm $\mathcal{M}$

We consider an execution ρ of $\mathcal{M}$ on $(\mathbf{G}, \delta)$ and for each vertex $v \in V(G)$, we denote by $(\lambda(v), n_i(v), N_i(v), M_i(v))$ the state of v after the ith computation step of ρ on v. If the vertex v executes an action to go from the step i to the step $i + 1$, it is said active at step $i + 1$.

The following proposition summarizes some properties that are satisfied during the execution ρ on $(\mathbf{G}, \delta)$.

Proposition 3.3 ([6,7]) *Consider a vertex v and a step i.*

Then, $n_i(v) \leq n_{i+1}(v)$, $N_i(v) \preceq N_{i+1}(v)$, $M_i(v) \subseteq M_{i+1}(v)$.

For each $(m, \ell, N) \in M_i(v)$ and each $m' \in [1, m]$, $\exists (m', \ell', N') \in M_i(v), \exists v' \in V(G)$ such that $n_i(v') = m'$.

Algorithm 1: Algorithm $\mathcal{M}$.

I : $\{n(v_0) = 0$ and no message has arrived at $v_0\}$
begin
  $n(v_0) := 1$;
  $M(v_0) := \{(n(v_0), \lambda(v_0), \emptyset)\}$;
  for $i := 1$ ***to*** $\deg(v_0)$ **do**
    send $< (n(v_0), \lambda(v_0), M(v_0)), i >$ through i ;
end

R : {A message $< (n_1, \ell_1, M_1), p_1 >$ has arrived at v_0 through port q_1}
begin
  $M_{old} := M(v_0)$;
  $M(v_0) := M(v_0) \cup M_1$;
  if $n(v_0) = 0$ *or* $\exists (n(v_0), \ell', N') \in M(v_0)$ *such that* $(\lambda(v_0), N(v_0)) \prec (\ell', N')$ **then**
    $n(v_0) := 1 + \max\{n' \mid \exists (n', \ell', N') \in M(v_0)\}$;
  $N(v_0) := N(v_0) \setminus \{(n', \ell', p', q_1) \mid \exists (n', \ell', p', q_1) \in N(v_0)\} \cup \{(n_1, \ell_1, p_1, q_1)\}$;
  $M(v_0) := M(v_0) \cup \{(n(v_0), \lambda(v_0), N(v_0))\}$;
  if $M(v_0) \neq M_{old}$ **then**
    for $i := 1$ ***to*** $\deg(v_0)$ **do**
      send $< (n(v_0), \lambda(v_0), M(v_0)), i >$ through port i ;
end

Proof We suppose that some internal event is executed at step $i + 1$ by some vertex $v \in V(G)$. The property is obviously true for any vertex $w \in V(G) \setminus \{v\}$ and it is easy to see that $M_i(v) \subseteq M_{i+1}(v)$.

If $n_i(v) \neq n_{i+1}(v)$, then $n_{i+1}(v) = 1 + \max\{n_1 \mid (n', \ell', \mathcal{N}') \in M_i(v)\}$ and either $n_i(v) = 0 < n_{i+1}(v)$ or $(n_i(v), \lambda(v), N_i(v)) \in M_i(v)$ and therefore $n_i(v) < n_{i+1}(v)$.

If $N_i(v) \neq N_{i+1}(v)$, then v has received a message $< (n', n'_{old}, M'), p >$ through port q and $N_{i+1}(v) = N_i(v) \setminus \{(n'_{old}, p, q)\} \cup \{(n', p, q)\}$. Let v' be the neighbour of v such that $v_v(v') = q$; we know that $v_{v'}(v) = p$.

If $(n'_{old}, p, q) \notin N_i(v)$, then $\max N_{i+1}(v) \triangle N_i(v) = (n', p, q) \in N_{i+1}(v)$ and then $N_i(v) \prec N_{+1}(v)$.

If $(n'_{old}, p, q) \in N_i(v)$, then $n'_{old} \neq n'$. Let $j < i + 1$ be the computation step where v' has sent the message $< (n', n'_{old}, M'), p >$. We know that $n'_{old} \leq n' = n_j(v')$ and consequently, $\max N_{i+1}(v) \triangle N_i(v) = (n', p, q) \in N_{i+1}(v)$ and $N_i(v) \prec N_{+1}(v)$.

For the second part of the proposition: We first note that $(m, \ell, \mathcal{N})$ is added to $\bigcup_{v \in V(G)} M_i(v)$ at some step i only if there exists a vertex $v \in V(G)$ such that $n_i(v) = m$, $\lambda(v) = \ell$ and $N_i(v) = \mathcal{N}$.

Given a vertex $v \in V(G)$, a step i and an element $(m, \ell, \mathcal{N}) \in M_i(v)$, let $U = \{(u, j) \in V(G) \times \mathbb{N} \mid j \leq i, n_j(u) = m\}$ and $U' = \{(u, j) \in U \mid \forall (u', j') \in U, (\lambda(u'), N_{j'}(u')) \prec (\lambda(u), N_j(u))$ or $(\lambda(u'), N_{j'}(u')) = (\lambda(u), N_j(u))$ and $j' \leq j\}$. Since $(m, \ell, \mathcal{N}) \in M_i(v)$, U and U' are both non-empty and it is easy to see that there exists i_0 such that for each $(u, j) \in U'$, $j = i_0$.

If $i_0 < i$, let $(u, i_0) \in U'$; we know that $n_{i_0+1}(u) \neq n_{i_0}(u)$, but this is impossible, since by maximality of $(\lambda(u), N_{i_0}(u))$, u cannot have modified its number. Consequently, $i_0 = i$ and there exists $v' \in V(G)$ such that $n_i(v') = m$. This ends the proof. □

Consider the mailbox $M = M(v)$ of a vertex v during the execution of Algorithm $\mathcal{M}$ on a graph $(\mathbf{G}, \delta)$. We say that an element $(n, \ell, N) \in M$ is *maximal* in M if there does not exist $(n, \ell', N') \in M$ such that $(\ell, N) \prec (\ell', N')$. We denote by $S(M)$ the set of maximal elements of M. From Proposition 3.3, after each step of Algorithm $\mathcal{M}$, $(n(v), \lambda(v), N(v))$ is maximal in $M(v)$.

The set $S(M)$ is said *coherent* if it is non-empty and if for all $(n_1, \ell_1, N_1) \in S(M)$, for all $(n_2, \ell_2, p, q) \in N_1$, $p \neq 0, n_2 \neq 0$ and $\ell_2 \neq \perp$ and for all $(n'_2, \ell'_2, N'_2) \in S(M)$, there exists $(n'_2, \ell''_2, p', q') \in N_1$ if and only if $\ell'_2 = \ell''_2$ and $(n_1, \ell_1, q', p') \in N'_2$.

From [7], we know that once $n(v)$, $N(v)$ and $M(v)$ have reached their final values for all v, then $S(M(v))$ is coherent for any v. Thus, if $S(M(v))$ is not coherent, we know that $M(v)$ will be modified.

If the set $S(M)$ is coherent, one can construct a labelled symmetric digraph $\mathbf{D}_M = (D_M, \lambda_M)$ as follows. The set of vertices $V(D_M)$ is the set $\{n \mid \exists (n, \ell, N) \in S(M)\}$. For any $(n, \ell, N) \in S(M)$ and any $(n', \ell', p, q) \in N$, there exists an arc $a_{n,n',p,q} \in A(D_M)$ such that $t(a) = n$, $s(a) = n'$, $\lambda_M(a) = (p, q)$. Since $S(M)$ is coherent, we can define Sym by $Sym(a_{n,n',p,q}) = a_{n',n,q,p}$.

One can show that Algorithm $\mathcal{M}$ terminates and the final labelling verifies the following properties: $(Dir(G), (\lambda, \delta))$ is a symmetric covering of $\mathbf{D}_M$ (see Proposition 4.1 in [7]). Thus if $(Dir(G), (\lambda, \delta))$ is symmetric covering minimal then $\mathbf{D}_M$ is isomorphic to $(Dir(G), (\lambda, \delta))$ and therefore the set of numbers of the vertices is exactly $[1, |V(G)|]$: each vertex has a unique number. Moreover, the termination detection of the algorithm is possible on $\mathbf{G}$. Indeed, once a vertex gets the identity number $|V(G)|$ (which is known by each vertex), from Proposition 3.3, it knows that all the vertices have different identity numbers that will not change any more and it can conclude that the computation of numbers of vertices is over. In this case, one can also solve the election problem, since this vertex can take the label *elected* and broadcasts the information that a vertex has been elected. Finally, we obtain Theorem 3.2 presented above.

Remark 3.4 A natural question is: what happen if nodes know only a bound of the size of the graph? Does it exist a mechanism to detect the termination of $\mathcal{M}$. An extension of $\mathcal{M}$ and the associated properties in this direction do not seem obvious. The sequel of this paper gives a positive answer to this question.

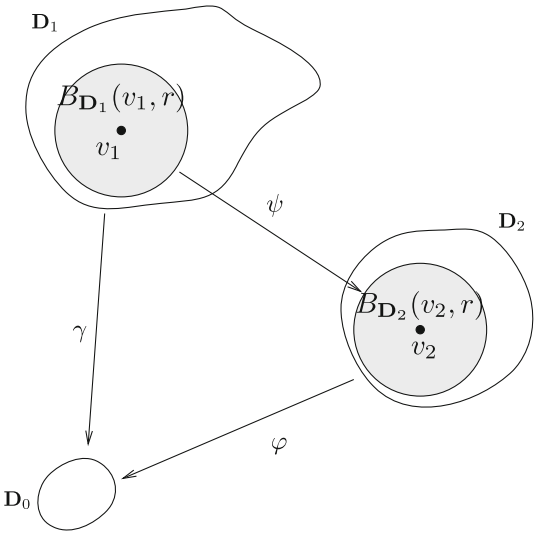

Fig. 1 Quasi-coverings diagram of Definition 4.1. The ball $B_{\mathbf{D}_1}(v_1, r)$ captures "*the existence of large enough area of one graph*" ($\mathbf{D}_1$) "*that looks locally like another graph*" ($\mathbf{D}_2$)

4 Quasi-coverings and the election problem for a family of labelled graphs

This section presents the second tool we use: quasi-coverings. This tool provides a necessary condition for the election in a family of labelled graphs presented at the end of the section.

4.1 Quasi-coverings

Quasi-coverings have been introduced to study the termination detection problem [21]. The idea behind quasi-coverings is to enable the simulation of local computations on a given graph in a restricted area of a larger graph, such that a replay technique can be used to prove impossibility results by contradiction. The restricted area where we can perform the simulation will shrink while the number of simulated steps increases. In [10], the definition of quasi-coverings have been slightly modified to express more easily this property as a Quasi-Lifting Lemma.

The next definition is an adaptation of this tool to labelled digraphs and is illustrated in Fig. 1.

Definition 4.1 Given two symmetric labelled digraphs $\mathbf{D}_0, \mathbf{D}_1$, an integer r, a vertex $v_1 \in V(D_1)$ and a homomorphism γ from $\mathbf{B}_{\mathbf{D}_1}(v_1, r)$ to $\mathbf{D}_0$, the digraph $\mathbf{D}_1$ is a *quasi-covering* of $\mathbf{D}_0$ of center v_1 and of radius r via γ if there exists a symmetric labelled digraph $\mathbf{D}_2$ that is a symmetric covering of $\mathbf{D}_0$ via a homomorphism φ and if there exist $v_2 \in V(D_2)$ and an isomorphism ψ from $\mathbf{B}_{\mathbf{D}_1}(v_1, r)$ to $\mathbf{B}_{\mathbf{D}_2}(v_2, r)$ such that for any $x \in V(B_{D_1}(v_1, r)) \cup A(B_{D_1}(v_1, r))$, $\gamma(x) = \varphi(\psi(x))$.

We define the *number of sheets* q of the quasi-covering to be the minimal cardinality of the sets of preimages of vertices of $\mathbf{D}_0$ which are in the ball: $q = \min_{v \in V(D_0)} |\{w \in \psi^{-1}(v) | B_{\mathbf{D}_1}(w, 1) \subset B_{\mathbf{D}_1}(v_1, r)\}|$.

Using the notation of the definition of a quasi-covering, we say that a quasi-covering is *strict* if $B_{\mathbf{D}_1}(v_1, r-1)$ is not equal to D_1. Note that any non-strict quasi-covering is a covering. We have [12]:

Lemma 4.2 *Let $\mathbf{D}_1$ be a strict quasi-covering of $\mathbf{D}_0$ of radius r via γ. Then, for any $q \in \mathbb{N}$, if $r \geq q|V(D_0)|$ then γ has at least q sheets.*

Remark 4.3 If a labelled digraph $\mathbf{D}_1$ is a symmetric covering of $\mathbf{D}_0$, then for any $v \in V(D_1)$ and for any $r \in \mathbb{N}$, $\mathbf{D}_1$ is a quasi-covering of $\mathbf{D}_0$, of center v and of radius r. Indeed, one just has to choose $\mathbf{D}_2 = \mathbf{D}_1$. In this case, we say that $\mathbf{D}_1$ is a quasi-covering of $\mathbf{D}_0$ of infinite radius. Reversely, if $\mathbf{D}_1$ is a quasi-covering of $\mathbf{D}_0$ of radius r strictly greater than the diameter of $\mathbf{D}_1$, then $\mathbf{D}_1$ is a covering of $\mathbf{D}_0$.

The following lemma makes precise the shrinking of the radius when k rounds of a synchronous execution are performed :

Lemma 4.4 (Quasi-Lifting Lemma) *Let $\mathbf{D}_1$ be a symmetric labelled digraph that is a quasi-covering of $\mathbf{D}_0$ of center v_1 and of radius r via γ. Let $k < r$ be a non negative integer. For any algorithm $\mathcal{A}$, let $\mathbf{D}'_0$ be the digraph obtained after k rounds of the synchronous execution of $\mathcal{A}$ on $\mathbf{D}_0$. Then $\mathbf{D}'_1$ obtained after a synchronous execution of $\mathcal{A}$ on $\mathbf{D}_1$ is a quasi-covering of $\mathbf{D}'_0$ of center v_1 and of radius $r-k$.*

Proof Consider an algorithm $\mathcal{A}$ and a symmetric labelled digraph $\mathbf{D}_1 = (D_1, \lambda_1)$ that is a quasi-covering of $\mathbf{D}_0 = (D_0, \lambda_0)$ of center v_1 and of radius r via γ. There exists a symmetric labelled digraph $\mathbf{D}_2 = (D_2, \lambda_2)$ that is a symmetric covering of $\mathbf{D}_0$ via a homomorphism φ and a vertex $v_2 \in V(D_2)$ such that $(B_{D_1}(v, r), \lambda_1)$ is isomorphic to $(B_{D_2}(v, r), \lambda_2)$ via an isomorphism ψ and for any $v \in B(v_1, r)$, $\gamma(v) = \varphi(\psi(v))$.

Let $\mathbf{D}'_0 = (D_0, \lambda'_0)$ (resp. $\mathbf{D}'_1 = (D_1, \lambda'_1)$, $\mathbf{D}'_2 = (D_2, \lambda'_2)$) be the labelled digraph where for each v, $\lambda'_0(v)$ (resp. $\lambda'_1(v)$, $\lambda'_2(v)$) is the state of v in D_0 (resp. D_1, D_2) after a computation round of $\mathcal{A}$ on $\mathbf{D}_0$ (resp. on $\mathbf{D}_1, \mathbf{D}_2$). To prove the lemma, it is sufficient to show that $\mathbf{D}'_1$ is a quasi-covering of $\mathbf{D}'_0$ of center v_1 and of radius $r-1$ via γ.

From Lemma 3.1, we know that $\mathbf{D}'_2$ is a covering of $\mathbf{D}'_0$. Moreover, for each $v \in V(B_{D_1}(v_1, r-1))$, $\lambda'_1(v) = \lambda'_2(\varphi(v))$ since $(B_{D_1}(v, 1), \lambda_1)$ is isomorphic to $(B_{D_2}(\varphi(v), 1), \lambda_2)$.

Consequently, $(B_{D_1}(v, r-1), \lambda'_1)$ is isomorphic to $(B_{D_2}(v, r-1), \lambda'_2)$ via ψ and thus $\mathbf{D}'_1$ is a quasi-covering of $\mathbf{D}'_0$ of center v_1 and of radius $r-1$ via γ. □

The following corollary is the counterpart of the lifting lemma for quasi-coverings.

Corollary 4.5 (Quasi-Lifting corollary) *Consider a symmetric labelled digraph* $\mathbf{D}_1$ *that is a quasi-covering of* $\mathbf{D}_0$ *of center* v_1 *and of radius* r *via* γ. *For any algorithm* $\mathcal{A}$, *after* r *rounds of the synchronous execution of an algorithm* $\mathcal{A}$ *on* $\mathbf{D}_1$, v_1 *is in the same state as* $\gamma(v_1)$ *after* r *rounds of the synchronous execution of* $\mathcal{A}$ *on* $\mathbf{D}_0$.

4.2 Election in a family of labelled graphs and quasi-coverings

Proposition 4.6 (Necessary condition) *Let* $\mathcal{I}$ *be a recursive family of connected symmetric covering minimal labelled digraphs such that there is an election algorithm for this family. Then there exists a computable function* $\tau : \mathcal{I} \to \mathbb{N}$ *such that for all labelled digraph* $\mathbf{D}$ *of* $\mathcal{I}$, *there is no quasi-covering of* $\mathbf{D}$, *distinct of* $\mathbf{D}$, *of radius greater than* $\tau(\mathbf{D})$ *in* $\mathcal{I}$.

Proof Let $\mathcal{A}$ denote an election algorithm on $\mathcal{I}$. For a labelled digraph $\mathbf{D} \in \mathcal{I}$, define $\tau(\mathbf{D}) = 2|V(\mathbf{D})| + n$ where n is the number of rounds of an entire synchronous execution of $\mathcal{A}$ on $\mathbf{D}$. Then τ has the desired property.

We prove this by contradiction. Let $\mathbf{D} \in \mathcal{I}$. Let $\mathbf{C}_i$ be the labelled graph obtained after the ith round used for the definition of $\tau(\mathbf{D})$. Let $\mathcal{C} = (\mathbf{C}_0 = \mathbf{D}, \mathbf{C}_1, \ldots, \mathbf{C}_n)$ such that no step of $\mathcal{A}$ can be applied on any vertex of $\mathbf{C}_n$, (at the end of the round n no message is sent by any vertex) and $\mathbf{C}_{i+1}$ is obtained from $\mathbf{C}_i$ by the execution of a round of $\mathcal{A}$. By hypothesis the label *elected* appears exactly once in $\mathbf{C}_n$.

Let $\mathbf{D}' \in \mathcal{I}$ be a quasi-covering of $\mathbf{D}$ of radius $\tau(\mathbf{D})$, distinct of D. By iteration of Lemma 4.4, we get $\mathbf{D}''$ such that $\mathbf{D}''$ is a quasi-covering of $\mathbf{C}_n$ of radius $\tau(D) - n = 2|V(D)|$. The labelled digraph $\mathbf{D}$ being symmetric covering minimal and distinct of $\mathbf{D}'$, the quasi-covering $\mathbf{D}''$ of $\mathbf{C}_n$ is strict. Hence, by Lemma 4.2, the label *elected* appears at least twice in $\mathbf{D}''$. A contradiction. □

5 An election algorithm for a family of labelled graphs

In this section, we present an algorithm we will use to obtain our sufficient conditions and finally a characterization of families of graphs which admit an election algorithm and Theorem 1.1 given in Introduction.

This algorithm is a combination of the election algorithm $\mathcal{M}$ for symmetric minimal labelled digraphs presented in Sect. 3.3 and a generalization of an algorithm of Szymanski, Shy and Prywes (the SSP algorithm for short) [23]. The SSP algorithm was originally used to detect the global termination of an algorithm with local termination provided the processes initially know a bound on the diameter of the graph.

The enumeration algorithm always terminates on any network $(\mathrm{Dir}(\mathbf{G}), \delta)$ and during the execution, each process v can reconstruct at some computation step i a symmetric labelled digraph $\mathbf{D}_i(v)$ such that $(\mathrm{Dir}(\mathbf{G}), \delta)$ is a quasi-covering of $\mathbf{D}_i(v)$. However, this algorithm does not enable v to compute the radius of this quasi-covering. We use a generalization of the SSP algorithm to enable each process to compute a lower bound on the radius of these quasi-coverings.

5.1 Termination detection mechanism

A vertex will detect if its state is final by using the necessary condition given by Proposition 4.6 thus we add to the label of each vertex two items:

- $c(v) \in \mathbb{Z}$ is a counter and initially $c(v) = -1$. In some sense, $c(v)$ represents the distance up to which all vertices have the same mailbox as v.
- $A(v) \in \mathcal{P}_{\mathrm{fin}}(\mathbb{N} \times \mathbb{N})$ encodes the information v has about the values of $c(u)$ for each neighbour u. Initially, $A(v) = \{(q, -1) \mid q \in [1, \deg_G(v)]\}$.

Thus now during the execution, the label of each v is a tuple $(\lambda(v), n(v), N(v), M(v), c(v), A(v))$.

A message sent by a vertex u via the port p to the vertex v has the following form $< (n, \ell, M, a), p >$ where n is the current number $n(u)$ of u, ℓ is the label $\lambda(u)$ of u, M is the mailbox of u, a is the value of the counter $c(u)$ and $p = \delta_u(v)$.

In the description of the algorithm we use the following predicate, denoted for a vertex v $\mathbf{QC}(v)$, and defined by:

$$\mathbf{QC}(v) = (S(M(v)) \text{ is coherent}) \text{ and } (c(v) \neq c_{old}(v)) \text{ and } (\mathbf{D}_{M(v)} \in \mathcal{I}) \text{ and } (c(v) \leq \tau(\mathbf{D}_{M(v)})).$$

5.2 Algorithm $\mathcal{M}_{Family}$

Our algorithm $\mathcal{M}_{Family}$ is described in Algorithm 2. We recall that the first rule **I** can be applied by a process v on wake-up only if it has not received any message: it takes the number 1, updates its mailbox and informs its neighbours. The second rule **R** describes the instructions a process v has to follow when it receives a message m from a neighbour. It updates its mailbox $M(v)$ and its local view $N(v)$ according to m. Then, if it discovers the existence of another vertex with the same number and a stronger local view, it takes a new number. If its mailbox has not changed, it updates $A(v)$ and increases $c(v)$ if possible. Finally, if $M(v)$ or $c(v)$ has been modified, it informs its neighbours.

Using the information stored in its mailbox, each process v will be able to reconstruct a labelled digraph $\mathbf{D}$ such that $(\mathrm{Dir}(\mathbf{G}), \delta)$ locally looks like $\mathbf{D}$ up to distance $c(v)$.

Algorithm 2: Algorithm $\mathcal{M}_{Family}$.

I : $\{n(v_0) = 0$ and no message has arrived at $v_0\}$
begin
  $n(v_0) := 1$;
  $M(v_0) := \{(n(v_0), \lambda(v_0), \emptyset)\}$;
  $c(v_0) := -1$;
  for $i := 1$ ***to*** $\deg(v_0)$ **do**
    send $< (n(v_0), \lambda(v_0), M(v_0), c(v_0)), i >$ through port i ;
end

R : {A message $< (n_1, \ell_1, M_1, a_1), p_1 >$ has arrived at v_0 through port q_1}
begin
  $M_{old} := M(v_0)$;
  $c_{old} := c(v_0)$;
  $M(v_0) := M(v_0) \cup M_1$;
  if $n(v_0) = 0$ *or* $\exists (n(v_0), \ell', N') \in M(v_0)$ *such that* $(\lambda(v_0), N(v_0)) \prec (\ell', N')$ **then**
    $n(v_0) := 1 + \max\{n' \mid \exists (n', \ell', N') \in M(v_0)\}$;
  $N(v_0) := N(v_0) \setminus \{(n', \ell', p', q_1) \mid \exists (n', \ell', p', q_1) \in N(v_0)\} \cup \{(n_1, \ell_1, p_1, q_1)\}$;
  $M(v_0) := M(v_0) \cup \{(n(v_0), \lambda(v_0), N(v_0))\}$;
  if $M(v_0) \neq M_{old}$ **then**
    $c(v_0) := -1$;
    $A(v_0) := \{(q', -1) \mid 1 \leq q' \leq deg(v_0)\}$;
  if $M(v_0) = M_1$ **then**
    $A(v_0) := A(v_0) \setminus \{(q_1, a') \mid \exists (q_1, a') \in A(v_0)\} \cup \{(q_1, a_1)\}$;
  if $(\forall (q', a') \in A(v_0), c(v_0) \leq a'$ *and* $\mathbf{QC}(v_0))$ **then**
  $c(v_0) := c(v_0) + 1$;
  if $M(v_0) \neq M_{old}$ *or* $c(v_0) \neq c_{old}$ **then**
    for $i := 1$ ***to*** $\deg(v_0)$ **do**
      send $< (n(v_0), \lambda(v_0), M(v_0), c(v_0)), i >$ through port i ;
end

5.2.1 Properties of algorithm $\mathcal{M}_{Family}$.

We consider a graph $\mathbf{G}$ with a port numbering δ and an execution ρ of Algorithm $\mathcal{M}_{Family}$ on $(\mathbf{G}, \delta)$.

For each vertex $v \in V(G)$, we note $(\lambda_i(v), n_i(v), N_i(v), M_i(v), c_i(v), A_i(v))$ the state of v after the ith computation step of ρ.

An interesting corollary of Proposition 3.3 is: there exists a step i_0 such that after this step for any v, the values of $\lambda(v)$, $n(v)$, $N(v)$ and $M(v)$ are not modified any more.

Proposition 5.1 *Consider a vertex v and a step i.*

If $M_i(v) = M_{i+1}(v)$ and if v is active at step $i+1$, then $c_i(v) \leq c_{i+1}(v) \leq c_i(v) + 1$ and $c_{i+1}(v) \geq \min\{a \mid \exists (q, a) \in A_{i+1}(v)\}$ if $\exists (q, a) \in A_{i+1}(v)$.

If $c_{i+1}(v) \geq 1$, for each $w \in N_G(v)$, there exists a step $j \leq i$ such that $c_j(w) \geq c_{i+1}(v) - 1$ and $M_j(w) = M_{i+1}(v)$.

Proof We prove the proposition by induction on i. Consider a vertex v that modifies its state at step $i+1$.

If v has applied rule **I** then it is easy to see that the claims hold.

Suppose now that v has applied rule **R** after receiving the message $m_1 =< (n_1, l_1, M_1, a_1), p_1 >$ via port q_1. Due to the algorithm, we have: $M_i(v) \subseteq M_{i+1}(v)$.

If $M_i(v) = M_{i+1}(v)$ and $c_i(v) \neq c_{i+1}(v)$ then $c_{i+1}(v) = c_i(v) + 1$.

Let $min_A = min\{a \mid \exists (q, a) \in A_{i+1}(v)\}$.

If $min_A \neq a_1$, then by induction $c_{i+1} \geq min_A$. If $M_{i+1}(v) \neq M_i(v)$ then $c_{i+1}(v) \geq -1 \geq min_A$. Suppose that $M_{i+1}(v) = M_i(v)$. If $(q_1, a_1) \in A_i(v)$ then $A_{i+1}(v) = A_i(v)$ and by induction $c_{i+1}(v) \geq min_A$. If $M_1 \neq M_{i+1}(v)$ then $A_{i+1}(v) = A_i(v)$ and $c_{i+1}(v) \geq min_A$.

Suppose now that $M_1 = M_{i+1}(v)$. If $a_1 = 0$ then v can increase $c(v)$ if it is equal to -1. Thus $c_{i+1}(v) \geq 0$ and consequently $c_{i+1}(v) \geq min_A$.

Otherwise, we may assume that $min_A = a_1 > 0$, $M_{i+1}(v) = M_i(v) = M_1$ and $(q_1, a_1) \notin A_i(v)$. Let $m_2 =< (n_2, \ell_2, M_2, a_2), p_1 >$ be the previous message received via port q_1. Since communication channels are FIFO, m_2 has been sent before m_1. Since $a_1 > 0$, $M_2 = M_1$ and thus, by induction, $a_2 = a_1 - 1$. Let $j \leq i$ be the step where v gets m_2. Since $M_2 \subseteq M_j(v) \subseteq M_i(v) = M_1 = M_2$, $M_j(v) = M_2$. Consequently, $(q, a_2) \in A_i(v)$. Since $a_1 = min_A$, $a_2 = a_1 - 1 = min\{a \mid \exists (q, a) \in A_i(v)\}$. If $c_i(v) \geq a_1$ then $c_{i+1}(v) \geq c_i(v) \geq min_A$. Otherwise, by induction, $c_i(v) = a_2 = a_1 - 1$, and, since $c_i(v) < a_1$ and $c_i(v) \leq min_A$, v increases $c_i(v)$. Thus $c_{i+1}(v) = 1 + c_i(v) = a_1 \geq min_A$.

Suppose that $c_{i+1}(v) \geq 1$ and consider a vertex $w \in N_G(v)$. There exists $(\delta_v(w), a) \in A_{i+1}(v)$ with $a \geq c_{i+1}(v) - 1 \geq 0$ and v gets a message $m =< (n, \ell, M, a), \delta_w(v) >$ from w at a step $i' \leq i+1$. Let $j < i+1$ be the step where w sent this message. Since $a \geq 0$, $M_{i'}(v) = M_{i+1}(v) = M_j(v)$ and $c_j(w) = a \geq c_{i+1}(v) - 1$. □

The following proposition shows that $\mathbf{D}_{M(v)}$ has some similarity with $(\mathrm{Dir}(\mathbf{G}), \delta)$.

Proposition 5.2 *If $S(M(v))$ is coherent, $(\mathrm{Dir}(\mathbf{G}), \delta)$ is a quasi-covering of $\mathbf{D}_{M(v)}$ of radius $c(v)$ and center v.*

We first prove a proposition that enables to present another definition of quasi-coverings that we will use in the proof of Proposition 5.2.

The proof of this proposition needs the definition of view we give now.

Definition 5.3 Consider a symmetric digraph $\mathbf{D} = (D, \lambda) \in \mathcal{D}_L$ and a vertex $v \in V(D)$. The *view* of v in $\mathbf{D}$ is an infinite rooted tree denoted by $\mathbf{T}_{\mathbf{D}}(v) = (T_D(v), \lambda')$ and defined as follows:

- $V(T_D(v))$ is the set of non-stuttering paths $\pi = a_1, \ldots, a_p$ in $\mathbf{D}$ with $s(a_1) = v$. For each path $\pi = a_1, \ldots, a_p$, $\lambda'(\pi) = \lambda(t(a_p))$.
- for each $\pi, \pi' \in V(T_D(v))$, there are two arcs $c_{\pi,\pi'}$, $c_{\pi',\pi} \in A(T_D(v))$ such that $Sym(c_{\pi,\pi'}) = c_{\pi',\pi}$ if and only if

$\pi' = \pi, a$. In this case, $\lambda'(c_{\pi,\pi'}) = \lambda(a)$ and $\lambda'(c_{\pi',\pi}) = \lambda(Sym(a))$.
- the root of $T_D(v)$ is the vertex corresponding to the empty path and its label is $\lambda(v)$.

Remark 5.4 For all vertices, u and v : $T_D(u)$ is isomorphic to $T_D(v)$. We denote by T_D this graph defined up to an isomorphism. It is the universal covering of D. It is useful to provide examples of quasi-coverings.

Consider the view $\mathbf{T_D}(v)$ of a vertex v in a digraph $\mathbf{D} \in \mathcal{D}_L$ and an arc a such that $s(a) = v$. We define $\mathbf{T_{D-a}}(v)$ be the infinite tree obtained from $\mathbf{T_D}(v)$ by removing the subtree rooted in the vertex corresponding to the path a.

Proposition 5.5 *Given two symmetric labelled digraphs $\mathbf{D}_0, \mathbf{D}_1$, an integer r, a vertex $v_1 \in V(D_1)$ and a homomorphism γ from $\mathbf{B_{D_1}}(v_1, r)$ to $\mathbf{D}_0, \mathbf{D}_1$ is a quasi-covering of $\mathbf{D}_0$ of center v_1 and of radius r via γ if and only if the following holds:*

(i) *for each arc $a \in A(B_{D_1}(v_1, r)), \gamma(Sym(a)) = Sym(\gamma(a))$,*
(ii) *for any $v \in \mathbf{B_{D_1}}(v_1, r)$, γ induces an injection between the incoming (resp.outgoing) arcs of v and the incoming (resp. outgoing) arcs of $\gamma(v)$,*
(iii) *for any $v \in \mathbf{B_{D_1}}(v_1, r-1)$, γ induces a surjection between the incoming (resp.outgoing) arcs of v and the incoming (resp. outgoing) arcs of $\gamma(v)$.*

Proof If $\mathbf{D}_1$ is a quasi-covering of $\mathbf{D}_0$ of center v_1 and of radius r via γ, then it is easy to see that γ satisfies these properties.

Reversely, we construct an infinite covering $\mathbf{D}_2 = (D_2, \lambda_2)$ of $\mathbf{D}_0 = (D_0, \lambda_0)$ as follows. First we take a copy $\mathbf{B}_1$ of $\mathbf{B_{D_1}}(v_1, r)$ and we note δ the isomorphism from $\mathbf{B_{D_1}}(v_1, r)$ to $\mathbf{B}_1$. Then, consider a vertex v such that $\text{dist}_{B_1}(v_1, v) = r$ and an arc $a_0 \in A(D_0)$ such that $v \in \gamma^{-1}(t(a_0))$. If there is no arc $a \in A(B_1)$ such that $t(a) = v$ and $\gamma(a) = a_0$, then we add a copy of $\mathbf{T_{D_0-a_0}}(s(a_0))$ to $\mathbf{D}_2$ and we note $v_2(a_0)$ the root of this tree. We add two arcs a_2, a_2' such that $t(a_2) = s(a_2') = v, s(a_2) = t(a_2') = v_2(a_0), \lambda_2(a_1) = \lambda_0(a_0), \lambda_2(a_2) = \lambda_0(Sym(a_0))$ and $Sym(a_2) = a_2'$.

The digraph $\mathbf{D}_2$ is the digraph obtained once these constructions have been done for all $v \in V(B_1)$ such that $\text{dist}_{B_1}(v, v_1) = r$. It is easy to see that $\mathbf{D}_2$ is a symmetric covering of $\mathbf{D}_0$ via some homomorphism φ and that for any $v \in V(B_{\mathbf{D}_1}(v, r)), \gamma(v) = \varphi(\delta(v))$. □

We now use Proposition 5.5 to prove Proposition 5.2.

Proof (of Proposition 5.2) Consider a step i and a process v such that $S(M_i(v))$ is coherent. If $c_i(v) = 0$, then we are done. Suppose now that $c_i(v) \geq 1$. From Proposition 5.1, for each $w \in V(Dir(G))$ such that $\text{dist}_{Dir(G)}(v, w) \leq c_i(v)$, there exists a step $j_w \leq i$ such that $c_{j_w}(w) \geq c_i(v) - \text{dist}_{Dir(G)}(v, w)$ and $M_{j_w}(w) = M_i(v)$.

Thus, for each $w \in V(B_{Dir(\mathbf{G})}(v, c_i(v)))$, j_w is defined and $c_{j_w}(w) \geq 0$ and for each $w \in V(B_{Dir(\mathbf{G})}(v, c_i(v) - 1)), c_{j_w}(w) \geq 1$. For each $w \in V(B_{Dir(\mathbf{G})}(v, c_i(v)))$, $(n_{j_w}(w), \lambda(v), N_{j_w}(w)) \in S(M_{j_w}(w)) = S(M_i(v))$ and thus $\deg_{G'}(w) = \deg_{D_{M_i(v)}}(n_{j_w}(w))$. Thus, we can define $\gamma(w) = n_{j_w}(w) \in V(D_{M_i(v)})$ and we have $\lambda_{M_i(v)}(\gamma(w)) = \lambda(w)$.

For each arc $a \in A(B_{Dir(\mathbf{G})}(v, c_i(v)))$, let $w = t(a), w' = s(a)$ (resp. $w = s(a), w' = t(a)$) and suppose without loss of generality that $\text{dist}_G(v, w) \leq c_i(v) - 1$. Let $m =< (n, \ell, M, a), p >$ be the last message received by w through port $\delta_w(w')$ before step j_w. Since $c_{j_w}(w) \geq 1, M = M_{j_w}(w) = M_{j_{w'}}(w'), n = n_{j_{w'}}(w'), p = \delta_{w'}(w)$ and $a \geq 0$. Thus, we can define $\gamma(a) = c_{n,n',p,q}$ (resp. $\gamma(a) = c_{n',n,q,p}$) where $n = n_{j_w}(w), n' = n_{j_{w'}}(w'), p = \delta_{w'}(w)$ and $q = \delta_w(w')$. It is easy to see that $\lambda_{M_i(v)}(\gamma(a)) = \lambda(a) = (p, q)$ (resp. $\lambda_{M_i(v)}(\gamma(a)) = \lambda(a) = (q, p)$) and that $Sym(\gamma(a)) = \gamma(Sym(a))$.

Consequently, γ is a homomorphism from $\mathbf{B}_{Dir(\mathbf{G})}(v, c_i(v))$ to $\mathbf{D}_{M_i(v)}$.

For each $w \in V(B_{Dir(\mathbf{G})}(v, c_i(v)))$, for all arcs a, a' such that $s(a) = s(a') = w$ (resp. $t(a) = t(a') = w$), $\lambda(a) \neq \lambda(a')$ and thus, $\gamma(a) \neq \gamma(a')$. For each $w \in V(B_{Dir(\mathbf{G})}(v, c_i(v) - 1))$, since $\deg_{G'}(w) = |N_{j_w}(w)| = \deg_{D_{M_i(v)}}(n_{j_w}(w))$, γ induces a bijection between the incoming (resp. outgoing) arcs of w in $Dir(\mathbf{G})$ and the incoming (resp. outgoing) arcs of $n_{j_w}(w)$ in $\mathbf{D}_{M_i(v)}$.

From Proposition 5.5, $Dir(\mathbf{G})$ is a quasi-covering of $\mathbf{D}_{M_i(v)}$ of center v and of radius $c_i(v)$ via γ. □

Remark 5.6 As can be seen in the proof, the value $\gamma(w)$ does not depend on the actual j_w. The quasi-covering γ is obtained from n, in the sense that the value of $\gamma(w)$ at $w \in B_{Dir(G)}(v, c(v))$ is equal to $n(w)$ at the time-step where $M(w) = M(v)$.

The algorithm $\mathcal{M}_{Family}$ is a combination of the algorithm $\mathcal{M}$ presented in Sect. 3.3 and a generalization of the algorithm of Szymanski, Shy and Prywes. The enumeration algorithm always terminates on any network $(\text{Dir}(\mathbf{G}), \delta)$. During the execution, each process v can reconstruct at some computation step i a symmetric labelled digraph $\mathbf{D}_i(v)$ such that $(\text{Dir}(\mathbf{G}), \delta)$ is a quasi-covering of $\mathbf{D}_i(v)$.

When the enumeration algorithm has terminated then for each vertex v, after the step t_v for the vertex v, there exists a step t_v' such that $c(v) \geq \tau(\mathbf{D}_{M(v)})$. This knowledge and the knowledge of the reconstructed graph $\mathbf{D}_{M(v)}$ (which belongs to $\mathcal{I}$) implies that each node knows that the enumeration algorithm has terminated and if its number is the maximal or not among numbers of the graph. Finally, each node can decide if it is elected or not.

The result of this subsection and of the previous are summarized by Theorem 1.1 states in Introduction.

5.3 Complexity analysis of $\mathcal{M}_{Family}$

We are interested in characterizing the complexity of $\mathcal{M}_{Family}$, thus we recall the time complexity of $\mathcal{M}$, the message complexity and the size of the messages and the size of the memory needed by each vertex [6,7].

As Tel [24] (p. 71), we define the time complexity by supposing that internal events need zero time units and that the transmission time (i.e., the time between sending and receiving a message) is at most one time unit. This corresponds to the number of rounds needed by a synchronous execution of the algorithm. Note that the correctness of $\mathcal{M}$ is independent of these assumptions.

Given a network $(\mathbf{G}, \delta)$, we denote by l the maximum size (in bits) of an initial label appearing on $\mathbf{G}$, and we denote by $|(\mathbf{G}, \delta)|$ the size of $|V(G)| + l$.

We can remark that the mailbox of each vertex contains a lot of useless information. Indeed, if some (p, ℓ, N) belongs to the mailbox $M(v)$ of a vertex v, one can remove from $M(v)$ all the elements $(p, \ell', N') \in M(v)$ such that $(\ell', N') \prec (\ell, N)$. We can thus replace the mailbox $M(v)$ of v by $\{(p, \ell, N) \in M(v) \mid \forall (p, \ell', N') \in M(v), (\ell', N') \preceq (\ell, N)\}$. In this way, the mailbox of each vertex contains at most $|V(G)|$ elements of the form (p, ℓ, N).

The communication channels have the FIFO property, one can reduce the size of the messages. Indeed, each time a vertex modifies its mailbox, it just has to send the elements it adds to its mailbox instead of sending the whole mailbox. In the complexity analysis we do, we suppose that the sizes of messages and mailboxes are reduced as we just explained. Moreover, we suppose that each process sends the elements $\{(p, p_{old}, N')\}$ of its mailbox one by one and it sends k messages if it has k elements to send to its neighbours.

The following proposition summarizes the complexity analysis of $\mathcal{M}$:

Proposition 5.7 ([6,7]) *Given a network $(\mathbf{G}, \delta)$ with n vertices, m edges, whose maximum degree is Δ and whose diameter is $Diam$, any execution of $\mathcal{M}$ on $(\mathbf{G}, \delta)$ needs $O(nDiam)$ time units and $O(m^2 n)$ messages of $O(\Delta(l + \log n))$ bits. Moreover, the memory needed by each process is $O(\Delta n(l + \log n))$ bits.*

Proof Consider a network $(\mathbf{G}, \delta)$ with n vertices, m edges, whose maximum degree is Δ and whose diameter is $Diam$. Consider an execution of $\mathcal{M}$ on $(\mathbf{G}, \delta)$. From Proposition 3.3, we know that each vertex modifies its number at most n times.

For each vertex v, since the numbers of v and of its neighbours can only increase, the couple $(n(v), N(v))$ can take at most $(\deg_G(v) + 1)n$ different values. Each time a vertex modifies its number or its local view, it can generate at most $O(m)$ messages, since a vertex whose mailbox already contains $(n(v), \lambda(v), N(v))$ will not broadcast this information to its neighbours. Consequently, any execution of $\mathcal{M}$ on $\mathbf{G}$ needs at most $O(m^2 n)$ messages. Moreover, since we suppose that all messages have the form $< (p, p_{old}, \{(n', \ell', N')\}), p >$ and since $N(v)$ contains at most Δ elements, each message has a size of $O(\Delta(l + \log n))$ bits.

We now consider the synchronous execution of our algorithm on $(\mathbf{G}, \delta)$. Each time a vertex v modify its number or its local view, all vertices of $(\mathbf{G}, \delta)$ know $(n(v), \lambda(v), N(v))$ in $Diam + 1$ time units. Thus if v takes the number $n(v)$ during a round i, for any vertex w that modifies its number during a round $j > i + Diam + 1, n_j(w) > n(v)$. Consequently, after $O(nDiam)$ time units each vertex has computed its final number $n(v)$. And thus, after $Diam + 1$ more rounds, the value of $(n(v), N(v), M(v))$ is not modified any more for any v.

For each vertex $v, n(v)$ can be encoded with $\log n$ bits and $N(v)$ can be encoded with $O(\Delta(l + \log n))$ bits. Since each vertex keeps only the information that is useful in its mailbox, there are at most n elements in $M(v)$ and each of these elements can be encoded with $O(\Delta(l + \log n))$ bits. Consequently, each vertex of G needs a memory of $O(\Delta n(l + \log n))$ bits to store its state.

□

We say that an algorithm $\mathcal{A}$ is said to be polynomial on a family $\mathcal{I}$ if there exists a polynomial p such that for each $(\mathbf{G}, \delta) \in \mathcal{I}$, for each execution of $\mathcal{A}$ on $(\mathbf{G}, \delta)$, the number of rounds, the number and the size of the messages are are bounded by $p(|(\mathbf{G}, \delta)|)$.

From the previous proposition we deduce:

Corollary 5.8 *There exists a polynomial algorithm solving leader election on $\mathcal{I}$ if and only if conditions of Theorem 1.1 hold and there exists a polynomial p such that for each $\mathbf{D}$ of $\mathcal{I}\tau(\mathbf{D}) \leq p(|\mathbf{D}|)$.*

Proof If there there exists a polynomial p such that for each $\mathbf{D}$ of $\mathcal{I}\tau(\mathbf{D}) \leq p(|\mathbf{D}|)$, from the previous proposition, we know that for each $\mathbf{D} \in \mathcal{I}$, each execution of $\mathcal{M}_{family}$ on $\mathbf{D}$ needs $O(nDiam + p(|\mathbf{D}|))$ time units and $O(m^2 n + mp(|\mathbf{D}|))$ of polynomial size are sent during this execution.

Conversely, suppose there exists a polynomial election algorithm $\mathcal{A}$ for $\mathcal{I}$ and let p be a polynomial such that for each $\mathbf{D}$ in $\mathcal{I}$, the synchronous execution of $\mathcal{A}$ on $\mathbf{D}$ elects a leader in less than $p(|\mathbf{D}|)$ rounds. Consider the

function τ defined as in the proof of Proposition 4.6: it is clear that for each $\mathbf{D} \in \mathcal{I}, \tau(\mathbf{D}) \leq p(|\mathbf{D}|) + O(|\mathbf{D}|)$.

□

5.4 About other models

The result obtained in this paper is not specific to the asynchronous message passing model; it may be extended to any model for which exist:

- covering-like and quasi-covering-like notions,
- an algorithm to distinguish a vertex, and
- a SSP-like algorithm.

By this way, Theorem 1.1 may be extended to the following models:

- the model defined by Angluin in [1] (inspired by Milne and Milner's model for distributed systems [19]) and, more generally, the model of local computations on labelled edges [8];
- the model defined by Mazurkiewicz [18];
- asynchronous systems where processes communicate with synchronous message passing (i.e., there is a synchronization between the process sending the message and the one receiving it, it has been defined by Hoare in [13]).

6 Some applications

This section illustrates Theorem 1.1 through some consequences. As we said in the introduction an immediate corollary of Theorem 1.1 can be formulated in the following way: to elect in a symmetric labelled digraph which is minimal for the covering relation we only need an upper bound of its size or of its diameter. This result is a non trivial extension of Yamashita and Kameda [26] or Mazurkiewicz [18] results in the anonymous case. Results obtained in [4,9,27] are now particular cases of Theorem 1.1 in the non-anonymous case: we need upper bounds on the multiplicity of a label or of the size of graphs.

6.1 The family of trees or the family of complete graphs

If we consider the model of Mazurkiewicz or an asynchronous system with synchronous message passing or Angluin's model no tree admit another tree as a quasi-covering of arbitrary large size. The same property is true for the family of complete graphs. Thus in these models there is an election algorithm for the family of trees or the family of complete graphs.

6.2 There is no election algorithm for the family of graphs containing all trees and the triangle

We consider the model of Angluin [1]. A characterization of graphs which admit an election algorithm by giving definition of coverings (for this model) and the corresponding election algorithm may be found in [8].

Let $\mathcal{F}$ be the family of graphs containing all trees and the triangle. Now we examine Angluin's question: Does it exist an election algorithm which works for this family of graphs?

The path graph P_n is the n-vertex graph with $n-1$ edges all on a single open path. The family $\mathcal{F}$ contains the path graphs $P_n (n \geq 1)$. Clearly the triangle admits quasi-coverings of arbitrary large radius in $\mathcal{F}$. Thus by Theorem 1.1 there is no election algorithm for this family (Theorem 4.6 in [1]).

More generally, there is no election for any family of graphs which contains all trees and a graph which is not a tree.

6.3 The family of prime rings

Election algorithms for rings are very studied under many assumptions concerning the model, the size of the ring or the intial states of vertices. Examples of election algorithms in rings with a prime size and impossibility if the size is composite may be found in [3,14,17]. A natural question is: does it exist an election algorithm which works for the family of rings with a prime size (for the given models). A prime ring admits prime rings as quasi-coverings of arbitrary large radius; thus from from Theorem 1.1 we deduce there is no election algorithm for this family.

Now we consider the point-to-point message passing model. If G is a ring then $Dir(G)$ is a symmetric covering of the digraph with one vertex and two loops; thus there is no election algorithm for an anonymous ring even if its size is prime in the message passing model. If we consider a labelled ring having a prime size and initially at least 2 vertices with different labels then the corresponding symmetric labelled digraph is minimal for the covering relation and there is an election algorithm [7]. One can wonder whether it is true for the family of such labelled rings. Proposition 4.6 implies directly a negative answer. The answer is positive if each vertex knows an upper bound of the size (or equivalently of the diameter).

The same considerations are true for tori.

6.4 The family of rings with one distinguished vertex

We consider the family of rings where each node is labelled by $\lambda(v)$, with the only constraint being that there exists a

unique node whose label is unique in the whole ring:

$$\mathcal{F}_D = \{(G, \lambda) \mid G \text{ is a ring}, \exists! v_0 \in V(G), \forall v \in V(G), \lambda(v) = \lambda(v_0) \Rightarrow v = v_0\}.$$

We can see the members of this family as rings with non-unique identities, except for one node. This node is "distinguished". We denote by $u(\mathbf{G})$ the label $\lambda(v_0)$ of $\mathbf{G}$ that is unique. Note that for two different rings $\mathbf{G}, \mathbf{G}'$ in $\mathcal{F}_D$, $u(\mathbf{G})$ may be different from $u(\mathbf{G}')$.

For every network $\mathbf{G}$ in $\mathcal{F}_D$, $Dir(\mathbf{G})$ is obviously symmetric covering minimal and there exists an election algorithm for each of them, for example the algorithm that elects the node labelled $u(\mathbf{G})$. However, Theorem 1.1 explains why there is no algorithm solving election in all graphs of $\mathcal{F}_D$. Indeed, for any network in $\mathcal{F}_D$, there exists quasi-coverings of arbitrary radius. We detailed how to construct them. Given a graph $\mathbf{G}$, select a label λ_0 that does not appear on $\mathbf{G}$. Then construct the following labelling for a (big) ring $\mathbf{K}$:

- choose an orientation for $\mathbf{G}$ and $\mathbf{K}$,
- choose a vertex v in $\mathbf{G}$ and another v' in $\mathbf{K}$,
- assign as label for v' the label $\lambda(v)$,
- take the neighbours (in the chosen orientations) of v and v', and repeat until there is only one vertex unlabelled in $\mathbf{K}$,
- assign label λ_0 to this last vertex.

By construction, $\mathbf{K} \in \mathcal{F}_D$ and is a quasi-covering of $\mathbf{G}$ of radius $\frac{Card(\mathbf{K})}{2} - 1$.

6.5 General graphs

We also get some new possibility results, in particular for symmetric covering minimal graphs with at least 1 and at most k distinguished vertices or for symmetric covering minimal graphs where a bound on the size is known. On the impossibility side, it is a corollary of Theorem 1.1 that there is no election algorithm for the family of all symmetric covering minimal graphs.

7 Conclusion

We presented a simple and comprehensive characterization of families of networks which admit an election algorithm. This characterization contains strictly and "unifies" known results on this question. It enables to find non trivial new ones (Sects. 6.4 and 6.5).

Our algorithm is based on an universal enumeration algorithm (inspired by Mazurkiewicz), a termination detection algorithm by Szymanski, Shy and Prywes, the notion of covering and the notion of quasi-covering which captures "*the existence of large enough area of one graph that looks locally like another graph*".

This work may be also considered as a theoretical contribution motivated by the design of portable software with more security or privacy.

Thanks to the techniques we use, the time complexity and the message complexity (size) are polynomial.

The natural extension of this work is the study of the characterization of the computability in networks through techniques and tools we have introduced.

Another important extension/application of techniques developed in this paper concerns snapshot computation and, more generally, detection of stable properties in networks in a totally distributed system with partially knowledge (in preparation).

References

1. Angluin, D.: Local and global properties in networks of processors. In: Proceedings of the 12th Symposium on Theory of Computing, pp. 82–93 (1980)
2. Boldi, P., Codenotti, B., Gemmell, P., Shammah, S., Simon, J., Vigna, S.: Symmetry breaking in anonymous networks: characterizations. In: Proceedings of the 4th Israeli Symposium on Theory of Computing and Systems, pp. 16–26. IEEE Press, New York (1996)
3. Burns, J.E., Pachl, J.: Uniform self-stabilizing rings. ACM Trans. Program. Lang. Syst. **11**, 330–344 (1989)
4. Boldi, P., Vigna, S.: An effective characterization of computability in anonymous networks. In: Welch, J.L. (ed.) Distributed Computing. 15th International Conference, DISC 2001, Lecture Notes in Computer Science, vol. 2180, pp. 33–47. Springer, Berlin (2001)
5. Boldi, P., Vigna, S.: Fibrations of graphs. Discret. Math. **243**, 21–66 (2002)
6. Chalopin, J.: Algorithmique distribuée, calculs locaux et homorphismes de graphes. PhD thesis, université Bordeaux 1 (2006)
7. Chalopin, J., Métivier, Y.: An efficient message passing election algorithm based on mazurkiewicz's algorithm. Fundam. Inform. **80**(1–3), 221–246 (2007)
8. Chalopin, J., Métivier, Y.: On the power of synchronization between two adjacent processes. Distrib. Comput. **23**, 177–196 (2010)
9. Dobrev, S., Pelc, A.: Leader election in rings with nonunique labels. Fundam. Inform. **59**(4), 333–347 (2004)
10. Godard, E., Métivier, Y.: A characterization of families of graphs in which election is possible (ext. abstract). In: Nielsen, M., Engberg, U. (eds.) Proceedings of Foundations of Software Science and Computation Structures, FOSSACS'02, number 2303 in LNCS, pp. 159–171. Springer, Berlin (2002)
11. Godard, E., Métivier, Y., Muscholl, A.: Characterization of classes of graphs recognizable by local computations. Theory Comput. Syst. **37**(2), 249–293 (2004)
12. Godard, E., Métivier, Y., Tel, G.: Termination detection of local computations. CoRR, abs/1001.2785 (2010)
13. Hoare, C.A.R.: Communicating sequential processes. Commun. ACM **21**(8), 666–677 (1978)
14. Huang, Shing-Tsaan: Leader election in uniform rings. ACM Trans. Program. Lang. Syst. **15**, 563–573 (1993)
15. LeLann, G.: Distributed systems: towards a formal approach. In: Gilchrist, B. (ed.) Information Processing'77, pp. 155–160. North-Holland, Amsterdam (1977)
16. Massey, W.S.: A basic course in algebraic topology. Springer, Berlin (1991). Graduate texts in mathematics

17. Mazurkiewicz, A.: Solvability of the asynchronous ranking problem. Inf. Process. Lett. **28**, 221–224 (1988)
18. Mazurkiewicz, A.: Distributed enumeration. Inf. Process. Lett. **61**(5), 233–239 (1997)
19. Milne, G., Milner, R.: Concurrent processes and their syntax. J. ACM **26**(2), 302–321 (1979)
20. Mavronicolas, M., Michael, L., Spirakis, P.G.: Computing on a partially eponymous ring. Theor. Comput. Sci. **410**(6-7), 595–613 (2009)
21. Métivier, Y., Muscholl, A., Wacrenier, P.-A.: About the local detection of termination of local computations in graphs. In: Krizanc, D., Widmayer, P. (eds.) SIROCCO 97—4th International Colloquium on Structural Information & Communication Complexity, Proceedings in Informatics, pp. 188–200. Carleton Scientific (1997)
22. Santoro, N.: Design and Analysis of Distributed Algorithm. Wiley, NewYork (2007)
23. Szymanski, B., Shy, Y., Prywes, N.: Synchronized distributed termination. IEEE Trans. Softw. Eng. **SE-11**(10), 1136–1140 (1985)
24. Tel, G.: Introduction to Distributed Algorithms. Cambridge University Press, Cambridge (2000)
25. Tanenbaum, A., van Steen, M.: Distributed Systems—Principles and Paradigms. Prentice Hall, Englewood Cliffs (2002)
26. Yamashita, M., Kameda, T.: Computing on anonymous networks: part 1—characterizing the solvable cases. IEEE Trans. Parallel Distrib. Syst. **7**(1), 69–89 (1996)
27. Yamashita, M., Kameda, T.: Leader election problem on networks in which processor identity numbers are not distinct. IEEE Trans. Parallel Distrib. Syst. **10**(9), 878–887 (1999)