Local Computations on Closed Unlabelled Edges: the Election Problem and the Naming Problem (Extended Abstract)

Jérémie Chalopin

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

1 Introduction

The different local computations mechanisms are very useful for delimiting the borderline between positive and negative results in distributed computations. Indeed, they enable to study the importance of the synchronization level and to understand how important is the initial knowledge. A high level of synchronization involved in one atomic computation step makes a model powerful but reduces the degree of parallelism. Charron-Bost et al. [5] study the difference between synchronous and asynchronous message passing models. The model studied in this paper involves more synchronization than the message passing model: an elementary computation step modifies the states of two neighbours in the network, depending only on their current states. The information the processors initially have can be global information about the network, such as the size, the diameter or the topology of the network. The initial knowledge can also be local: each node can initially know its own degree for example. Another example of local knowledge is the existence of a port numbering: each processor locally gives numbers to its incident edges and in this way, it can consistently distinguish its neighbours. In Angluin's model [1], it is assumed that a port numbering exists, whereas it is not the case in our model. In fact, we obtain a model with a strictly lower power of computation by relaxing the hypothesis on the existence of a port numbering.

The Model. A network of processors will be represented as a simple connected undirected graph. As usual the vertices represent processors and edges direct communication links. The state of each processor is represented by the label $\lambda(v)$ of the corresponding vertex.

An elementary computation step will be represented by relabelling rules of the form given schematically in Figure 1. If, in a graph G, there are two neighbours labelled X and Y then applying this rule we replace X (resp. Y) by a new label X' (resp. Y'). The labels of all other graph vertices are irrelevant for such

a computation step and remain unchanged. The computations using uniquely this type of relabelling rules are called in this paper *local computations on closed unlabelled edges*. Thus an algorithm in our model is simply given by some (possibly infinite but always recursive) set of rules of the type presented in Figure 1. A run of the algorithm consists in applying the relabelling rules specified by the algorithm until no rule is applicable, which terminates the execution. The relabelling rules are applied asynchronously and in any order, which means that given the initial labelling usually many different runs are possible.

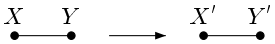


Fig. 1. Graphical form of a rule for local computations on closed unlabelled edges.

Election, Naming and Enumeration. The election problem is one of the paradigms of the theory of distributed computing. It was first posed by LeLann [7]. A distributed algorithm solves the election problem if it always terminates and in the final configuration exactly one processor is marked as *elected* and all the other processors are *non-elected*. Moreover, it is supposed that once a processor becomes *elected* or *non-elected* then it remains in such a state until the end of the algorithm. The naming problem is another important problem in the theory of distributed computing. The aim of a naming algorithm is to arrive at a final configuration where all processors have unique identities. The enumeration problem is a variant of the naming problem whose aim is to give to each node a unique number between 1 and the size of the graph. These problems are important since they constitute basic initial steps of many other distributed algorithms.

Related Works. Graphs where election is possible were already studied for different types of basic computation steps and particular types of network topology (tree, grid, torus, ring with a known prime number of vertices, etc.), see [11].

Yamashita and Kameda [12] characterize the graphs for which there exists an election algorithm in the message passing model and they study the importance of the port numbering in [13].

Mazurkiewicz [8] considers an asynchronous computation model where in one computation step, labels are modified on a subgraph consisting of a node and its neighbours, according to certain rules depending on this subgraph only. His characterization of the graphs where enumeration and election are possible can be formulated using coverings [6]. In this model, the port numbering does not give a more powerful model, since in each computation step, a node can always distinguish its neighbours.

Chalopin and Métivier [3] consider three different asynchronous models that are defined by the rules presented in Figure 2. Note that, contrary to the model we examine in the present paper, all these models allow edge labelling. In fact, allowing to label the edges is equivalent to the existence of a port numbering,

since in these models, it is always possible for a processor to consistently identify its neighbours. Consequently, the first model studied in [3] is equivalent to the model of Angluin [1]. It turns out that for all models of Figure 2 the election and naming problems can be solved on a graph G if and only if G is not a covering of any graph H not isomorphic to G, where H can have multiple edges but no self-loop. Mazurkiewicz [9] has also studied the first model described in Figure 2 and he gives an equivalent characterization thanks to equivalence relations over the vertices and the edges of the graph.

In the conclusion of [13], Yamashita and Kameda underline the importance of the edge labelling and it is a natural question to wonder if the edge labelling, or equivalently the existence of a port numbering, modify the power of the different models of Figure 2. Boldi et al. [2] consider a model where the network is a directed multigraph. When a processor is activated, it changes its state depending on its previous state and on the states of its ingoing neighbours. They characterize the graphs that admits an election algorithm using fibrations, that are generalization of coverings to directed graphs. Chalopin et al. [4] consider a model where an elementary computation step modifies the state of one vertex depending on its current state and the state of one of its neighbours; as in the model studied here, the edges are not labelled. In this model, naming and election are not equivalent and characterizations are given using submersions that are locally surjective morphisms. The comparison between the characterization given in [2], in [4] and in [3] shows that for the second and the third model of Figure 2, it gives strictly more powerful models to allow the edges to be labelled. In this paper, we complete the study of the importance of the port numbering: the characterization we give of the graphs for which the naming and the election problems can be solved for the model of Figure 1 is very different of the characterization given in [3]. Moreover, we can remark that the three models of Figure 2 that are equivalent when the edges can be labelled are no longer equivalent when this hypothesis is relaxed.

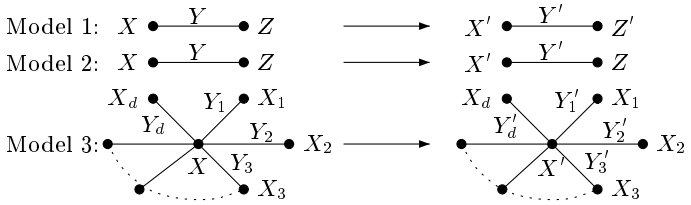


Fig. 2. Elementary relabelling steps for the models studied in [3].

Main Results. We introduce in Section 2 the notion of pseudo-covering, that is a generalization of coverings. We prove in Section 3 that naming and election can be solved on a graph G if and only if G is minimal for the pseudo-covering relation (Theorem 1).

The problems are solved constructively: we encode an enumeration algorithm with explicit termination by local computations on closed unlabelled edges that

work correctly for all graphs where these problems are solvable. This algorithm uses some ideas from Mazurkiewicz'algorithm [8]. However, in the models considered in [2, 3, 8, 9, 13], a node can consistently distinguish its neighbours whereas it is impossible in the model studied here. Each execution of our algorithm on a graph G computes a labelling that induces a graph H such that G is a pseudo-covering of H. Consequently, there exists an integer k such that each label of the final labelling of G appears exactly k times in the graph; it is not the case for the model studied in [4]. In our solution, stamps are associated to synchronizations between neighbours. These associated stamps solve the problem, but they introduce a non-trivial difficulty in the proof of the termination of the algorithm: we must prove that stamps are bounded.

Imposed space limitations do not allow to present all the proofs in the paper.

2 Preliminaries

Graphs. The notations used here are essentially standard [10]. We consider finite, undirected, connected graphs $G = (V(G), E(G))$ with vertices $V(G)$ and edges $E(G)$ without multiple edges or self-loop. Two vertices u and v are said to be adjacent or neighbours if $\{u, v\}$ is an edge of G and $N_G(v)$ will stand for the set of neighbours of v. An edge e is incident to a vertex v if $v \in e$ and $I_G(v)$ will stand for the set of all the edges of G incident to v.

A homomorphism between graphs G and H is a mapping $\gamma\colon V(G) \to V(H)$ such that if $\{u, v\} \in E(G)$ then $\{\gamma(u), \gamma(v)\} \in E(H)$. We say that γ is an isomorphism if γ is bijective and γ^{-1} is a homomorphism.

A graph H is a subgraph of G, noted $H \subseteq G$, if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. A subgraph H of G is called a partial graph of G if G and H have the same set of vertices.

A matching F of a graph G is a subset of $E(G)$ such that for every $e, e' \in F$, $e \cap e' = \emptyset$: F is a set of disjoint edges of G. A matching F of G is perfect if every vertex $v \in V(G)$ is the endvertex of exactly one edge $e \in F$.

Throughout the paper we will consider graphs where vertices are labelled with labels from a recursive label set L. A graph labelled over L is a couple $\mathbf{G} = (G, \lambda)$, where G is the underlying non labelled graph and $\lambda\colon V(G) \to L$ is the (vertex) labelling function. Let H be a subgraph of G and λ_H the restriction of the labelling $\lambda\colon V(G) \to L$ to $V(H)$. Then the labelled graph $\mathbf{H} = (H, \lambda_H)$ is called a subgraph of $\mathbf{G} = (G, \lambda)$; we note this fact by $\mathbf{H} \subseteq \mathbf{G}$. A homomorphism of labelled graphs is just a homomorphism that preserves the labelling.

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

Coverings and Pseudo-Coverings. 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. A graph G is called *covering-minimal* if every covering from G to some H is a bijection.

A graph G is a *pseudo-covering* of H via a morphism φ modulo a graph G' if G' is a partial graph of G that is a covering of H via the restriction $\varphi_{|G'}$ of φ to G'. The pseudo-covering is proper if G and H are not isomorphic. A graph G is said *pseudo-covering-minimal* if there does not exist a graph H such that G is a proper pseudo-covering of H. An example of pseudo-covering is given in Figure 3. Naturally, coverings and pseudo-coverings of labelled graphs are just coverings and pseudo-coverings of underlying graphs such that the associated morphisms preserve the labelling.

If $\mathbf{G}$ is a pseudo-covering of a graph $\mathbf{H}$ via φ modulo $\mathbf{G'}$, then for every edge $f = \{w_1, w_2\} \in E(H)$, $\varphi^{-1}_{|G'}(f)$ is a perfect matching of $\varphi^{-1}(\{w_1, w_2\})$. Consequently, there exists an integer q such that for every vertex $v \in V(H)$, $|\varphi^{-1}(v)| = q$.

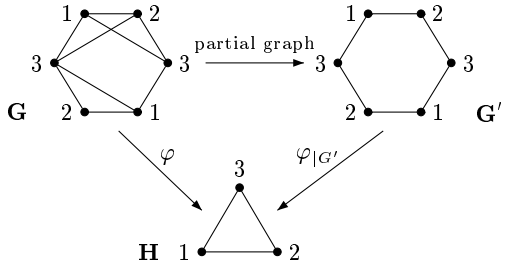


Fig. 3. The graph $\mathbf{G}$ is a pseudo-covering of $\mathbf{H}$ via the mapping φ modulo $\mathbf{G'}$ where φ maps each vertex of $\mathbf{G}$ labelled i to the unique vertex of $\mathbf{H}$ with the same label i. This pseudo-covering is proper and the graph $\mathbf{H}$ is pseudo-covering-minimal.

Local Computations on Closed Unlabelled Edges. For any set $\mathcal{R}$ of edge local relabelling rules of the type described in Figure 1 we shall write $\mathbf{G}\ \mathcal{R}\ \mathbf{G'}$ if $\mathbf{G'}$ can be obtained from $\mathbf{G}$ by applying a rule of $\mathcal{R}$ on some edge of $\mathbf{G}$. Obviously, $\mathbf{G}$ and $\mathbf{G'}$ have the same underlying graph G, only the labelling changes for the endvertices of exactly one (active) edge. Thus, slightly abusing the notation, $\mathcal{R}$ will stand both for a set of rules and the induced relabelling relation over labelled graphs. The reflexive transitive closure of such a relabelling relation is noted $\mathcal{R}^*$. The relation $\mathcal{R}$ is called *noetherian* on a graph $\mathbf{G}$ if there is no infinite relabelling sequence $\mathbf{G}_0\ \mathcal{R}\ \mathbf{G}_1\ \mathcal{R}\ \ldots$, with $\mathbf{G}_0 = \mathbf{G}$. The relation $\mathcal{R}$ is noetherian on a set of graphs if it is noetherian on each graph of the set. Finally, the relation $\mathcal{R}$ is called noetherian if it is noetherian on each graph. Clearly noetherian relations code always terminating algorithms.

The following lemma is a counterpart of the lifting lemma of Angluin [1] adapted to pseudo-coverings; it exhibits a strong link between pseudo-coverings and local computations on closed unlabelled edges. An immediate corollary is that there does not exist any algorithm using local computations on closed unlabelled edges that solves the election problem or the naming problem on a graph $\mathbf{G}$ that is not pseudo-covering-minimal.

Lemma 1 (Lifting Lemma). *Let $\mathcal{R}$ be a relabelling relation encoding an algorithm using local computations on closed unlabelled edges and let $\mathbf{G}_0$ be a*

pseudo-covering of $\mathbf{H}_0$. *If* $\mathbf{H}_0 \mathcal{R}^* \mathbf{H}_1$ *then there exists* $\mathbf{G}_1$ *such that* $\mathbf{G}_0 \mathcal{R}^* \mathbf{G}_1$ *and* $\mathbf{G}_1$ *is a pseudo-covering of* $\mathbf{H}_1$.

3 An Enumeration Algorithm

In this section, we describe a Mazurkiewicz-like algorithm $\mathcal{M}$ using local computations on closed unlabelled edges that solves the enumeration problem on a pseudo-covering-minimal graph $\mathbf{G}$.

Each vertex v attempts to get its own number between 1 and $|V(G)|$. A vertex chooses a number and exchanges its number with its neighbours. If during a computation step, two neighbours exchange their numbers, a stamp o is given to the operation such that two operations involving the same vertex have different stamps. Each node broadcasts its number, its label and its *local view* (the numbers of its neighbours and the stamps of the operations of exchange associated to each neighbour) 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 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 pseudo-covering-minimal.

The main difference with Mazurkiewicz'algorithm is the existence of the stamps o. The algorithm we will describe below computes a graph $\mathbf{H}$ such that $\mathbf{G}$ is a pseudo-covering of $\mathbf{H}$. To define a pseudo-covering, we need to define a morphism and a subset of $E(G)$. As in Mazurkiewicz'algorithm, the numbers of the nodes will be used to define the morphism φ whereas the stamps o will be used to select the edges of G.

Labels. We consider a labelled graph $\mathbf{G} = (G, \lambda)$. For each vertex $v \in V(G)$, the label of v is the pair $(\lambda(v), c(v))$ where $\lambda(v)$ is the initial label of v whereas $c(v)$ is a triple $(n(v), N(v), M(v))$ representing the following information obtained during the computation:

- $n(v) \in \mathbb{N}$ is the *number* of the vertex v computed by the algorithm;
- $N(v) \in \mathcal{N}$ is the *local view* of v. If the node v has a neighbour v', some relabelling rules will allow v and v' to add $n(v')$ in $N(v)$ and $n(v)$ in $N(v')$. Each time this operation is done between two neighbours a stamp o is given to the operation and $(n(v'), o)$ is added to $N(v)$ (resp. $(n(v), o)$ is added to $N(v')$). Consequently, $N(v)$ is a finite set of pairs (n, o);
- $M(v) \subseteq \mathbb{N} \times L \times \mathcal{N}$ is the *mailbox* of v and contains the whole information received by v at any step of the computation, i.e., the numbers, the labels and the local views of the nodes of the network.

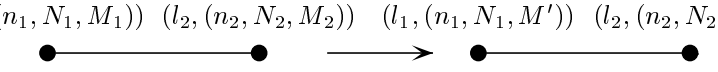
The fundamental property of the algorithm is based on a total order on local views, as defined in [8], such that the local view of any vertex cannot decrease during the computation. Consider a vertex v such that the local view $N(v) \in \mathcal{N}$ is the set $\{(n_1, o_1), (n_2, o_2), \ldots, (n_d, o_d)\}$, we assume that $n_1 > n_2 > ... > n_d$

and we say that the d-tuple $((n_1, o_1), (n_2, o_2), \ldots, (n_d, o_d))$ is the ordered representation of $N(v)$. We define a total order $\prec$ on such ordered tuples using the alphabetical order; it induces naturally a total order on $\mathcal{N}$. We assume that the set of labels L is totally ordered by $<_L$ and we extend $\prec$ on $L \times \mathcal{N}$.

The Relabeling Rules. We now describe the relabelling rules of the algorithm; the first rule $\mathcal{M}_0$ is a special rule that extends the initial label $\lambda(v)$ of each vertex to $(\lambda(v), (0, \emptyset, \emptyset))$. The rules $\mathcal{M}_1$ and $\mathcal{M}_2$ are very close to the rules of Mazurkiewicz's algorithm.

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

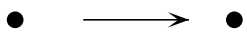
$\mathcal{M}_1$:



If $M_1 \neq M_2$ then $M' := M_1 \cup M_2$.

The second 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 label or a weaker local view than v'.

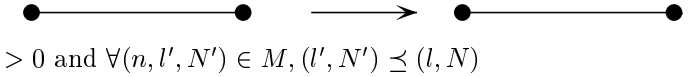
$\mathcal{M}_2$:



If $n = 0$ or $\exists (n, l', N') \in M$ such that $(l, N) \prec (l', N')$
then $k := 1 + \max\{n' \mid \exists (n', l', N') \in M\}$ and $M' := M \cup \{(k, l, N)\}$.

The third rule enables a node having a neighbour with exactly the same label to change its number. If this rule is applied, the number of each node is inserted in the local view of the other with a stamp o associated to the operation that is different of the other stamps associated to operations involving one of the two nodes. Moreover, when the number $n(v')$ of a neighbour v' of v is inserted in $N(v)$, all the elements (m, o) belonging to $N(v)$ such that $m \leq n(v')$ are deleted from the local view. The rationale behind this deletion step is explained in the next rule $\mathcal{M}_4$ below.

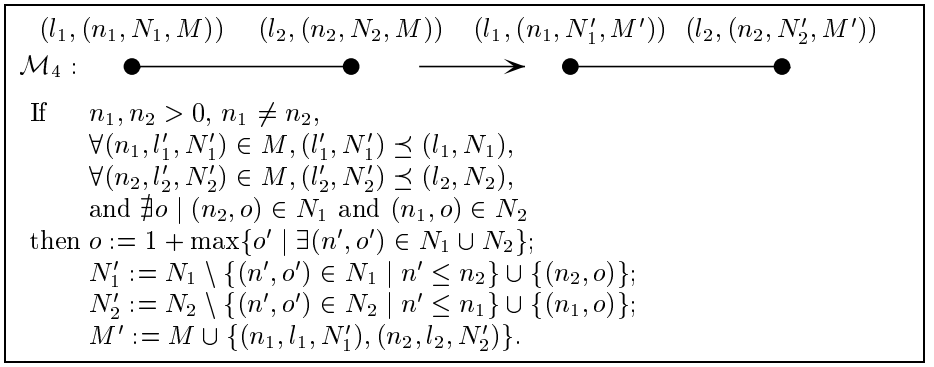
$\mathcal{M}_3$:



If $n > 0$ and $\forall (n, l', N') \in M, (l', N') \preceq (l, N)$
then $k := 1 + \max\{n' \mid \exists (n', l', N') \in M\}$;
$o := 1 + \max\{o' \mid \exists (n', o') \in N\}$;
$N_1 := N \setminus \{(n', o') \in N \mid n' \leq n\} \cup \{(n, o)\}$;
$N_2 := \{(k, o)\}$ and $M' := M \cup \{(k, l, N_1), (n, l, N_2)\}$.

The fourth rule enables two neighbours v and v' to exchange their numbers if an update is needed, i.e., if there does not exist o such that $(n_2, o) \in N_1$ and $(n_1, o) \in N_2$. As for the precedent rule, if the number $n(v')$ of a neighbour v' of v is inserted in $N(v)$, all the elements (m, o) belonging to $N(v)$ such that $m \leq n(v')$ are deleted.

The role of the stamp o associated to the operation is to ensure that at the end of the computation, if the local view of a vertex v_0 contains (n, o), it means that it has a neighbour v_0' such that $n(v_0') = n$, $(n(v_0), o) \in N(v_0')$ and such that the rule $\mathcal{M}_3$ or $\mathcal{M}_4$ was applied to these two vertices; an interesting consequence is that in the final labelling, $|\{v \mid n(v) = n(v_0)\}| = |\{v \mid n(v) = n(v_0')\}|$.



The intuitive justification for the deletion of all the (m, o) is the following. If there is a synchronization between two neighbours v and v', then they should agree on an integer o_0 and add $(n(v), o_0)$ to $N(v')$ and $(n(v'), o_0)$ to $N(v)$. But, it is possible that v synchronized with v' in the past and in the meantime v' has changed its identity number or has synchronized with another vertex w such that $n(w) = n(v)$. In this case, to remain in a consistent state, the vertex v should modify its local view to remove the old identity number of v' and the o associated to this precedent synchronization. The trouble is that v has no means to know which of the pairs (m, o) belonging to its view $N(v)$ should be deleted. However, since our algorithm assures the monotonicity of subsequent identity numbers of each vertex and monotonicity of subsequent o involving the node v', we know that the couple (m, o) to remove is such that $(m, o) <_{Lex} (n(v'), o_0)$ Therefore, by deleting all such (m, o) from the local view $N(v)$, we are sure to delete all invalid information. Of course, in this way we risk to delete also the legitimate current informations about other neighbours of v from its view $N(v)$. However, v can recover this information just by (re)synchronizing with all such neighbours.

Properties. In the following, we consider an execution of the algorithm on a graph $\mathbf{G}$. We will denote by $(\lambda(v), (n_i(v), N_i(v), M_i(v))$ the label of the vertex v after the ith computation step.

We can see that the label of each node can only "increase" during the computation. Indeed, for each step i, for each vertex v, $n_i(v) \leq n_{i+1}(v)$, $N_i(v) \preceq N_{i+1}(v)$ and $M_i(v) \subseteq M_{i+1}(v)$. Moreover, if a vertex v knows the existence of a node with the number m (i.e., $\exists (m, l, N) \in M_i(v)$), then for each $m' \leq m$, there exists a node w such that $n_i(w) = m'$. An immediate corollary of this property is that after each computation step the numbers of the nodes is a set $[1, k]$ with $k \leq |V(G)|$.

We will now prove that each execution of $\mathcal{M}$ on a graph $\mathbf{G}$ is finite. In fact, we just have to prove that the values of $n(v)$, $N(v)$ and $M(v)$ are bounded for

each vertex v . Since we already know that $n(v) \leq |V(G)|$, we just have to prove that the stamps o are also bounded. It will imply that $N(v)$ and $M(v)$ can only take a finite number of values. From the properties described above, there exists a step i_0 such that $\forall i \geq i_0, \forall v \in V(G), n_i(v) = n_{i_0}(v)$ and therefore the rules $\mathcal{M}_2$ and $\mathcal{M}_3$ cannot be applied after the step i_0. Consider two neighbours v and w such that $n_{i_0}(v) > n_{i_0}(w)$ and two steps $j_2 > j_1 > i_0$ where the rule $\mathcal{M}_4$ is applied to the edge $\{v, w\}$. Then, there must exist an edge $\{v', w'\}$ with $(n_{i_0}(v), n_{i_0}(w)) <_{Lex} (n_{i_0}(v'), n_{i_0}(w'))$ and a step $j \in [j_1, j_2]$ where the rule $\mathcal{M}_4$ is applied to $\{v', w'\}$. Consequently, the rule $\mathcal{M}_4$ can only be applied a finite number of time over each edge and we can ensure the termination of the algorithm.

For each execution of the algorithm over $\mathbf{G}$, a graph $\mathbf{H}$ is associated to the final labelling with $V(H) = \{n(v) \mid v \in V(G)\}$ such that $\mathbf{G}$ is a pseudo-covering of $\mathbf{H}$. If $\mathbf{G}$ is pseudo-covering-minimal, then $\mathbf{G} \simeq \mathbf{H}$. Consequently, for every run of the enumeration algorithm, the graph associated to the final labelling is isomorphic to $\mathbf{G}$ and therefore the set of numbers the vertices have is exactly $[1, |V(G)|]$. Moreover, once a vertex gets the number $|V(G)|$, it knows that all the vertices have a different number that will not change any more and therefore it can detect the termination. We can therefore transform the enumeration algorithm into an election algorithm, by choosing to elect the node that gets the number $|V(G)|$. From these results and Lemma 1, we have the following theorem.

Theorem 1. *For every graph* $\mathbf{G}$*, it is equivalent to solve the following problems on* $\mathbf{G}$ *with local computations on closed unlabelled edges: naming, naming with explicit termination and election. These problems can be solved on* $\mathbf{G}$ *if and only if* $\mathbf{G}$ *is a pseudo-covering-minimal graph.*

4 Comparison with Other Models

It is easy to see that each algorithm encoded by local computations on closed unlabelled edges can be translated in the models of Mazurkiewicz [8], Angluin [1] and Chalopin and Métivier [3]. And the algorithms encoded in the model of [4] can be encoded by local computations on closed unlabelled edges.

In the models of Mazurkiewicz [8], Angluin [1], Chalopin and Métivier [3], Yamashita and Kameda [12] and Boldi et al. [2], the election and the naming problems can be solved in the graph $\mathbf{G}_1$ of Figure 4. Nevertheless, this graph $\mathbf{G}_1$ is a pseudo-covering of the graph $\mathbf{H}_1$. Therefore, it is not possible to solve the election problem by using local computations on closed unlabelled edges.

If we consider the pseudo-covering-minimal graphs G_2 and G_3 of Figure 4, we can solve the election problem over these graphs with local computations on closed unlabelled edges. But the election problem cannot be solved over G_3 in the models studied in [2] and in [4]. Moreover there does not exist any election algorithm for the graph G_2 in the model studied in [12].

Consequently, our model is strictly less powerful than the models studied by Mazurkiewicz [8], by Angluin [1] and by Chalopin and Métivier [3], but strictly

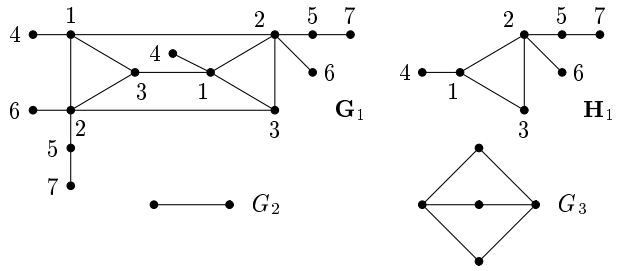


Fig. 4. Different graphs that show the differences between the different models.

more powerful than the model studied by Chalopin et al. [4]. And the power of computation of our model is not comparable to the power of the models of Yamashita and Kameda [12, 13] and Boldi et al. [2].

References

1. D. Angluin. Local and global properties in networks of processors. In *Proc. of the 12th Symposium on Theory of Computing*, pages 82–93, 1980.
2. 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.
3. J. Chalopin and Y. Métivier. Election and local computations on edges (*extended abstract*). In *Proc. of FOSSACS'04*, number 2987 in LNCS, pages 90–104, 2004.
4. J. Chalopin, Y. Métivier, and W. Zielonka. Election, naming and cellular edge local computations (*extended abstract*). In *Proc. of ICGT'04*, number 3256 in LNCS, pages 242–256, 2004.
5. B. Charron-Bost, F. Mattern, and G. Tel. Synchronous, asynchronous and causally ordered communication. *Distributed Computing*, 9(4):173–191, 1996.
6. E. Godard, Y. Métivier, and A. Muscholl. Characterization of classes of graphs recognizable by local computations. *Theory of Computing Systems*, 37(2):249–293, 2004.
7. G. LeLann. Distributed systems: Towards a formal approach. In B. Gilchrist, editor, *Information processing'77*, pages 155–160. North-Holland, 1977.
8. A. Mazurkiewicz. Distributed enumeration. *Inf. Processing Letters*, 61:233–239, 1997.
9. A. Mazurkiewicz. Bilateral ranking negotiations. *Fundamenta Informaticae*, 60:1–16, 2004.
10. K. H. Rosen, editor. *Handbook of discrete and combinatorial mathematics.* CRC Press, 2000.
11. G. Tel. *Introduction to distributed algorithms.* Cambridge University Press, 2000.
12. 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.
13. 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.