Checking Global Graph Properties by Means of Local Computations: the Majority Problem [1]

Igor Litovsky [a], Yves Métivier [b] and Eric Sopena [b]

[a] *I3S URA CNRS 1376, ESSI, BP 145, 06903 Sophia Antipolis Cedex, France*

[b] *LaBRI URA CNRS 1304, Université Bordeaux I, 33405 Talence Cedex, France*

Abstract

This paper is a contribution to the study of the general problem of characterizing those properties which can be computed on a graph or a network by means of local transformations. By using an abstract model based on graph relabelling systems we consider the majority problem : let G be a graph whose vertices have label A or B ; we say that label A has the majority if the number of A-labelled vertices is strictly greater than the number of B-labelled vertices ($|G|_A > |G|_B$). We prove that there exists graph relabelling systems deciding for every connected graph G whether $|G|_A > |G|_B$ (resp. $|G|_A = |G|_B$) or not. On the other hand, we prove that no such system can decide if $|G|_A > |G|_B - m$ (resp. $|G|_A = |G|_B - m$), for any positive integer m.

1 Introduction

One of the main characteristics of distributed systems is the local nature of the computation. A set of processors, connected in some specific way, try to reach a common goal (e.g. computing some function) after a finite number of elementary steps, each involving solely a subset of "near" processors. In this framework, one of the main questions is to characterize those functions, that is those *global* properties of the network, that can be computed by means of local transformations in the network [1,2,6,9,10]. In this paper we investigate that question by using a computational model introduced in [3], which allows to express such computations by means of some graph relabelling systems. More precisely, we will consider graph relabelling systems as recognizers of labelled graphs families based as follows on local relabellings : the labelled graph G to be recognized as a member of a specified set is labelled by some

[1] With the support of the Esprit-Basic Research Working Group "COMPUGRAPH II", the European Basic Action Research Esprit n° 3166 (ASMICS) and the European Community Cooperative Action IC-1000 (ALTEC).

special initial labelling ; labels are then *locally* modified, that is on subgraphs of fixed diameter of the current graph, according to some given *relabelling rules* ; these modifications are iterated until some *irreducible form* is reached, that is until no more transformation is possible. The presence or the absence of some specific final labels decides whether G is accepted or not.

The class of problems which can be solved by local computations is strongly dependent on the assumptions which are made on the initial graph. For instance all problems become easier when the graph has some distinguished vertex (with a special label), or when such a vertex can be elected [1,6,7]. In the same way when every vertex has some knowledge concerning the whole graph (an upper bound on or the exact number of vertices, the whole or partial topology of the graph, etc.) some problems may become solvable. We consider here the more general case, that is no vertex can be distinguished and no vertex has any knowledge concerning the rest of the graph. We are mostly interested in the following paradigm, called the *majority problem* : let A and B be any two labels, G be a graph whose vertices are labelled on $\{A, B\}$, $|G|_A$ (resp. $|G|_B$) be the number of vertices of G labelled with A (resp. with B). To what extent are we able to compare the quantities $|G|_A$ and $|G|_B$? We prove that using such graph recognizers we can decide whether $|G|_A > |G|_B$ (resp. $|G|_A = |G|_B$) or not. Then, using the notion of $k-$*covering*, we prove that it is not possible to decide whether $|G|_A > |G|_B - m$ (resp. $|G|_A = |G|_B - m$) or not, for any $m > 0$.

This paper is organized as follows : in Section 2 we introduce the main notions and notation. We prove in Section 3 our main result and in Section 4 our impossibility result. Due to the lack of space our main result is only established for cycles (the ring is certainly the most commonly studied network) and the main ideas are given for the general case. The complete proofs will be given in the full version of this paper.

2 Basic notions and notation

Let L be a finite set of labels. A *labelled graph* G over L, denoted by (G, λ), is a graph with vertex set $V(G)$ and edge set $E(G)$ equipped with a labelling function $\lambda : V(G) \cup E(G) \longrightarrow L$. We assume that the set L is partitionned into two subsets, the vertex and edge label sets respectively. The graph G is called the *underlying graph*, and the mapping λ is a *labelling* of it. The class of labelled graphs over some fixed alphabet L will be denoted by $\mathcal{G}_L$. Let $c \in L$, a c-labelled vertex (resp. edge) is a vertex v (resp. an edge e) such that $\lambda(v) = c$ (resp. $\lambda(e) = c$).

Let (G, λ) and (G', λ') be two labelled graphs. We say that (G, λ) is a *subgraph* of (G', λ'), denoted by $(G, \lambda) \subseteq (G', \lambda')$, if G is a subgraph of G' and λ is the restriction of λ' to $V(G) \cup E(G)$. An *isomorphism* from (G, λ) to (G', λ') is an isomorphism φ from G to G' which preserves the labelling, that is $\forall\, x \in V(G) \cup E(G)$, $\lambda'(\varphi(x)) = \lambda(x)$. An *occurrence* of (G, λ) in (G', λ') is an isomorphism φ from (G, λ) to a subgraph $(H, \eta) = \varphi(G, \lambda)$ of (G', λ').

A graph relabelling system is given as a 4-tuple $\mathcal{R} = (L, I, P, >)$ where

L is a finite set of labels, $I \subset L$ the set of *initial* labels, P a finite set of relabelling rules and $>$ a partial order over P. Each relabelling rule is given as a triple (R, μ, μ') such that (R, μ) and (R, μ') are two graphs in $\mathcal{G}_L$. Let (G, λ) be a graph in $\mathcal{G}_L$, φ an occurrence of (R, μ) in (G, λ) ; if there is no occurrence ψ of a rule (S, ν, ν'), $S > R$, such that $\psi(S, \nu)$ "intersects" (in an obvious way) $\varphi(R, \mu)$ in (G, λ), we say that (R, μ, μ') is *applicable* on (G, λ). The application of the relabelling rule (R, μ, μ') leads then to the graph (G, λ') obtained by relabelling the components of $\varphi(R, \mu)$ according to the labelling function μ'. We will then write $(G, \lambda)\ \mathcal{R}\ (G, \lambda')$. Note here that the effect of the priority mechanism is strictly local : in order to decide whether a relabelling rule may be applied or not, we only have to check the neighbourhood of the corresponding occurrence.

Let (G, λ) be a graph in $\mathcal{G}_I$, that is a graph with labels in the initial set I. We will denote by $\mathcal{R}(G, \lambda)$ the set of $\mathcal{R}$-*irreducible forms* of (G, λ), that is the set of graphs (G, λ') such that $(G, \lambda)\ \mathcal{R}^*\ (G, \lambda')$ and (G, λ') is irreducible, where $\mathcal{R}^*$ denotes the reflexive and transitive closure of $\mathcal{R}$. This set can be interpreted as the set of possible results of the computation expressed by $\mathcal{R}$ on (G, λ). For that reason we will only consider *noetherian* graph relabelling systems not allowing infinite *derivation sequences* (a derivation sequence is a sequence $(G, \lambda_1), (G, \lambda_2), \ldots, (G, \lambda_i), \ldots$ with $\forall\ i,\ (G, \lambda_i)\ \mathcal{R}\ (G, \lambda_{i+1})$).

A *final condition* over L is any finite propositional formula constructed from variables of the set $\{\Pi_l \mid l \in L\}$ by means of operations $\vee$, $\wedge$ and $\neg$. A labelled graph (G, λ) satisfies a final condition $\wp$ over L, denoted $(G, \lambda) \models \wp$, if the formula $\wp$ where we define Π_l as true if $\lambda^{-1}(l) \neq \emptyset$ is true. Note that this notion is invariant under isomorphism. Thus, such final conditions enable us to check the presence or the absence of some labels in a labelled graph but not to count vertices or edges with given labels, or to express some properties on their relative positions. For intance, it is impossible to specify that there is exactly one **T**-labelled vertex or that there exist two adjacent **T**-labelled vertices. Let $\wp$ be a final condition. We will denote by $\mathcal{K}(\wp)$ the set defined by $\mathcal{K}(\wp) = \{(G, \lambda) \in \mathcal{G}_L \mid (G, \lambda) \models \wp\}$.

A recognizer is a pair $(\mathcal{R}, \wp)$ where $\mathcal{R}$ is a graph relabelling system and $\wp$ a final condition. The class of graphs recognized by $(\mathcal{R}, \wp)$, denoted by $\mathcal{L}(\mathcal{R}, \wp)$, is then defined as those graphs (G, λ) in $\mathcal{G}_I$ such that $\mathcal{R}(G, \lambda) \cap \mathcal{K}(\wp) \neq \emptyset$. A recognizer $(\mathcal{R}, \wp)$ is said to be *deterministic* if for any graph (G, λ) in $\mathcal{G}_I$, either $\mathcal{R}(G, \lambda) \cap \mathcal{K}(\wp) = \mathcal{R}(G, \lambda)$ or $\mathcal{R}(G, \lambda) \cap \mathcal{K}(\wp) = \emptyset$. The class of graphs *deterministically* recognized by $(\mathcal{R}, \wp)$, denoted by $\mathcal{L}_{det}(\mathcal{R}, \wp)$, is then defined as those graphs (G, λ) in $\mathcal{G}_I$ such that $\mathcal{R}(G, \lambda) \subset \mathcal{K}$. In other words, a graph is deterministically recognized if *every* computation leads to a graph satisfying the final condition. A graph is undeterministically recognized if there exists some computation leading to a graph satisfying the final condition. Note here that the term deterministic refers to the recognition procedure (whose result is unique) but that the sets $\mathcal{R}(G, \lambda)$ are in general not singletons. This notion is very similar to the one used in [1].

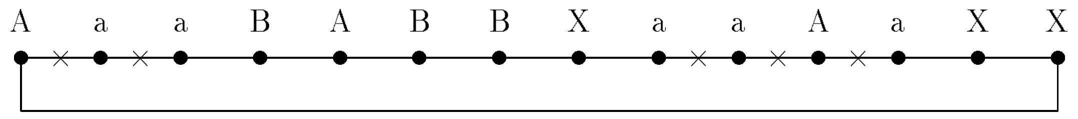

Fig. 1. Construction of A-paths on a cycle.

3 The main result

In this section we prove the following :

Theorem 3.1 *Let A and B be two labels; the class of labelled connected graphs G such that $|G|_A > |G|_B$ (resp. $|G|_A = |G|_B$) is deterministically recognizable by local computations.*

We first illustrate the technique we will use by considering the simple case when the graph G is a cycle. This technique will then be extended in order to capture the general case.

3.1 The cycle case

The main idea can be intuitively described as follows : when a A-labelled vertex has a B-labelled neighbour then they neutralize each other and become X-labelled. By repeating this process it may happen that the graph still contain some A- and B-labelled vertices which have only X-labelled neighbours. The solution is then to build some A-paths (whose edges will be marked) having one A-labelled vertex (the *root* of the A-path) and some X-labelled vertices which will become a-labelled (see Figure 1). In this way, A-labelled vertices will be able to "encounter" some B-labelled vertices not belonging to their immediate neighbourhood.

When the computation stops we have one of the following situations : (1) there are only X-labelled vertices, which means that G was such that $|G|_A = |G|_B$, (2) there are only a- and A-labelled vertices, which means that G was such that $|G|_A > |G|_B$ or (3) there are only X- and B-labelled vertices, which means that G was such that $|G|_B > |G|_A$.

More precisely, this computation can be done by a relabelling system $\mathcal{R}$ using the set of rules depicted on Figure 2. These rules work as follows :

R1, R2 : when a a- or A-labelled vertex has a X-labelled neighbour this neighbour is added to the A-path.

R3 : when a A-labelled vertex has a B-labelled neighbour, this neighbour becomes X-labelled, and the vertex becomes AX labelled. The AX label means that we have to change the labels of all the vertices of its A-path to X (this will be done by rules R10,...,R15).

R4 : when a a-labelled vertex has a B-labelled neighbour it needs to ask the root of its A-path whether the B-labelled vertex can be neutralized or not. The B-labelled vertex is marked as $\overline{B}$ and thus cannot be "attacked" on its other side until the decision is taken. The a-labelled becomes $ax?$-labelled.

R5 : The $ax?$ label is brought along the A-path towards the root.

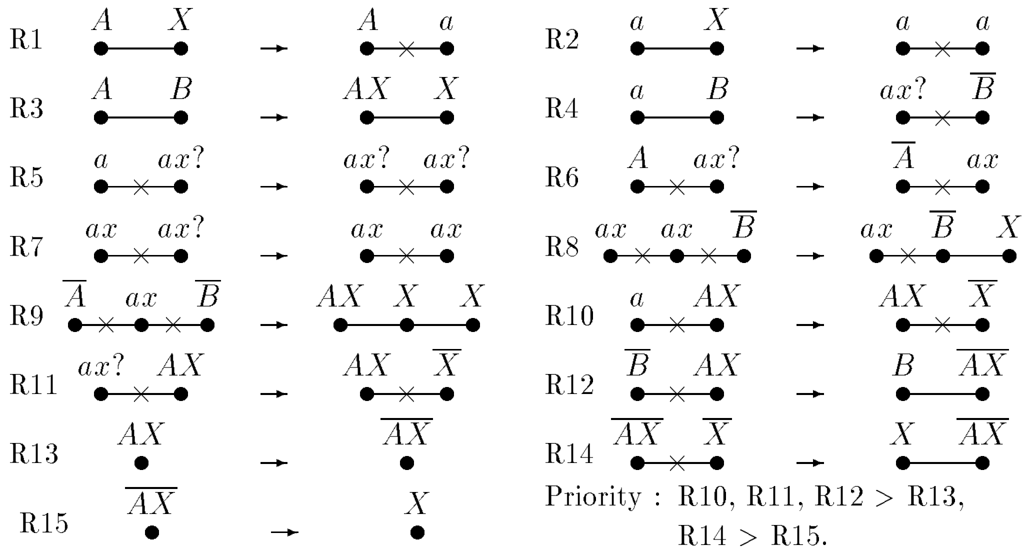

Fig. 2. The set of relabelling rules in the cycle case.

R6 : The $ax?$ label reaches the root which is free (A-labelled). The root then accepts the neutralization and becomes marked as $\overline{A}$ and thus cannot neutralize another B-labelled vertex on its other side. The $ax?$ label becomes ax.
R7 : the ax label return back to the neutralized ($\overline{B}$-labelled) vertex.
R8, R9 : when the ax label reaches the neutralized vertex, the $\overline{B}$ label is brought back to the root. When the $\overline{B}$ label reaches the root the root becomes AX-labelled, in order to change the labels of all the other vertices of the A-path.
R10, R11 : the AX label goes down the A-path (it may encounter only a or $ax?$ labels) and marks as $\overline{X}$ the encountered vertices.
R12, R13 : the AX label reaches the end of the A-path. If a $\overline{B}$-labelled vertex is encountered then the B-label is restored. The AX label can now become $\overline{AX}$. Note that thanks to the priority relation, this is only done when the end of the A-path is reached.
R14, R15 : all the $\overline{X}$-labelled vertices are now unmarked as X-labelled and the whole A-path is thus destroyed.

Note here that by marking with $\overline{X}$ the A-path to be destroyed before effectively destroying it we ensure that the system thus obtain always terminates. Without using that trick we could have such a A-path indefinitely turning around the cycle, growing on one side and being destroyed on the other side.

We will now sketch the proof of Theorem 3.1 for cycles. Due to the lack of space our intent is to illustrate here the proof techniques which are used for the general case. Let $P = x_1 \ldots, x_p$ be a marked path in G (that is whose all edges are marked). Let us call the *label* of P the word $\lambda(x_1).\ldots.\lambda(x_p)$. We denote by $\mathcal{U}^{-1}$ the mirror image of any rational language $\mathcal{U}$.

Claim 3.2 *In every derivation sequence in* $\mathcal{R}$ *the labels of the marked paths are of the form* $\mathcal{U}^{-1}.A.\mathcal{U}$, $\mathcal{U}^{-1}.\overline{A}.\mathcal{V}$, $\mathcal{V}^{-1}.\overline{A}.\mathcal{U}$, $\mathcal{U}^{-1}.AX.\overline{X}^{*}$ *or* $\overline{AX}.\overline{X}^{*}$, *where*

$\mathcal{U} = a^*(\varepsilon + (ax?)^+.\overline{B})$ *and* $\mathcal{V} = (ax)^+.(ax?)^*.\overline{B}$. *Moreover, all the vertices which are not incident to a marked edge have label* A*,* B *or* X*.*

Proof. It suffices to check these invariants for every rule in $\mathcal{R}$. □

Claim 3.3 *The system* $\mathcal{R}$ *is noetherian.*

Proof. Let (G,λ) be a graph whith n vertices labelled on $\{A,B\}$. For every rational language $\mathcal{U}$ let $\Upsilon(\mathcal{U})$ denote the total number of vertices of all (maximal) paths in G whose label is in $\mathcal{U}$. The tuple :

$$(\ |G|_B + |G|_{\overline{B}} + |G|_A,\ |G|_{AX},\ n - \Upsilon(AX.\overline{X}^*),\ \Upsilon(\overline{AX}.\overline{X}^*),$$
$$\Upsilon(\overline{A}.(ax)^+.(ax?)^*.\overline{B}),\ n - \Upsilon(\overline{A}.(ax)^*),\ n - \Upsilon((ax?)^*),\ n - \Upsilon(a^*) \quad)$$

is then a noetherian order compatible with the system $\mathcal{R}$ [3] : every component is positive and if we consider the usual lexicographic order on tuples, every rule in $\mathcal{R}$ decreases this quantity. The following table gives for every rule the component of this tuple which is decreased (in every case the previous components are unchanged) :

Rule :	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Component :	8	8	1	7	7	1	6	5	1	3	3	2	2	4	4

Thus every derivation sequence in $\mathcal{R}$ starting from a graph G labelled on $\{A,B\}$ is finite. □

Claim 3.4 *If* (G,λ) *is an irreducible graph then either* (1) *all its vertices are* X*-labelled or* (2) *all its vertices are* X*- or* B*-labelled or* (3) *all its vertices are* a*- or* A*-labelled.*

Proof. Using Claim 3.2 it is not difficult to deduce the following : if G has a $\overline{A}$-labelled vertex then the rule R9 is applicable ; if G has a $ax?$-labelled vertex then one of the rules R5, R6, R7 or R11 is applicable ; if G has a ax-labelled vertex then one of the rules R7, R8 or R9 is applicable ; if G has a AX-labelled vertex then one of the rules R10, R11, R12 or R13 is applicable ; if G has a $\overline{AX}$-labelled vertex then one of the rules R14 or R15 is applicable ; if G has a $\overline{X}$-labelled vertex the rule R14 is applicable ; if G has a $\overline{B}$-labelled vertex then one of the rules R8, R9 or R12 is applicable . Moreover, if G has some B-labelled vertices together with some a- or A-labelled vertices then one of the rules R3 or R4 is applicable. □

Claim 3.5 *Let* G *and* G' *be two labelled graphs such that* $G\ \mathcal{R}\ G'$*. Then*

$$|G|_A + |G|_{\overline{A}} - |G|_B - |G|_{\overline{B}} = |G'|_A + |G'|_{\overline{A}} - |G'|_B - |G'|_{\overline{B}}$$

Proof. This quantity is clearly preserved by every rule in $\mathcal{R}$. □

Let us now define the following final conditions : $\wp_X = \neg\Pi_A \wedge \neg\Pi_B$, $\wp_A = \Pi_A$. By using the previous claims one can prove that the two recognizers $(\mathcal{R},\wp_X)$ and $(\mathcal{R},\wp_A)$ satisfy the requirements of Theorem 3.1 : let (G,λ) be any graph whose vertices are labelled on $\{A,B\}$ and (G,λ') be any $\mathcal{R}$-irreducible form of (G,λ). By Claim 3.4 and Claim 3.5 we know that either

(G, λ') has only X-labelled vertices (in this case $|(G, \lambda)|_A = |(G, \lambda)|_B$) or (G, λ') has only X- and B-labelled vertices (in this case $|(G, \lambda)|_A < |(G, \lambda)|_B$) or (G, λ') has only X-, a- and A-labelled vertices (in this case $|(G, \lambda)|_A > |(G, \lambda)|_B$). Note that in this latter case we know by Claim 3.2 that (G, λ') has at least one A-labelled vertex. Moreover, the final number of A-paths is exactly the difference between the number of initially A- and B-labelled vertices.

3.2 The general case

For the general case we simply use A-trees instead of A-paths. Those trees will be directed (the orientation of any tree can be simulated by using three additional labels, see [3]) and rooted at a A-, $\overline{A}$- or AX-labelled vertex. The relabelling system is quite more complex but the basic idea is still the same : every A-tree try to neutralize a B-labelled vertex among those which are neighbours of its vertices. When such a neutralization occurs, the whole A-tree is destroyed and all its vertices become X-labelled.

4 Impossibility result

Let (G, λ) be a labelled graph and x a vertex of (G, λ). The *centered ball* $B_G(x, k)$ of radius k is the subgraph of (G, λ) induced by those vertices which are at distance at most k from x. Let k be a positive integer. We say that a graph G is a $k-$*covering* of a graph G' via a mapping γ from $V(G)$ onto $V(G')$ if γ is a surjective homomorphism such that for every vertex v of $V(G)$, the restriction of γ to $B_G(v, k)$ is an isomorphism between $B_G(v, k)$ and $B_{G'}(\gamma(v), k)$. In [4] the following is proved :

Theorem 4.1 [4] *Every class of connected graphs recognizable by local computations is closed under coverings.*

Using that, we easily obtain :

Theorem 4.2 *Let A and B be two labels, let $m > 0$ be an integer ; the class of labelled connected graphs G such that $|G|_A > |G|_B - m$ (resp. $|G|_A = |G|_B - m$) is not recognizable by local computations, even in a non deterministic way.*

Proof. It suffices here to consider the case of cycles : if $C = (x_0 x_1 \ldots x_{p-1}, \lambda)$ is a labelled cycle on $p > k$ vertices, the labelled cycle $C' = (y_0 y_1 \ldots y_{2p-1}, \lambda')$, with $\lambda'(y_i) = \lambda(x_{i \ mod \ p})$, is a k-covering of C. Suppose that there exists a recognizer for the family of graphs G such that $|G|_A > |G|_B - m$ (resp. $|G|_A = |G|_B - m$). By Theorem 4.1, if this recognizer accepts C then it also accepts C', a contradiction since $|C'|_A - |C'|_B = 2(|C|_A - |C|_B)$. □

5 Concluding remarks and open questions

By slightly modifying our system (we mean by using B-trees instead of isolated B-labelled vertices) we obtain a new system such that in any irreducible graph every vertex knows the result of the computation (if a vertex has a A- or a-label (resp. B- or b-) then label A (resp. B) has the majority and if a vertex has a X-label then there is no majority). But no vertex is able to detect the termination of the computation. Whether a system with such a local termination detection property exists or not is still an open question.

Our main concern here was the existence or non-existence of systems solving the majority problem. The design of systems achieving a better time complexity has not been yet considered (this complexity can be measured by the average length of a derivation sequence). This complexity could maybe be improved by using A-, B- and X-trees, leading then to more complicated systems.

Consider a finite set $\mathcal{C} = \{A_1, \ldots, A_k\}$ of labels. By combining several copies of our system (that is by using tuples of labels) we can decide for every graph G whether $|G|_{A_1} > Max\{|G|_{A_i}; 2 \leq i \leq k\}$ (resp. $|G|_{A_1} = |G|_{A_i}$, $\forall\ i,\ 2 \leq i \leq k$) or not. However, we do not know whether it is possible or not to recognize those labelled graphs G satisfying $|G|_A > k \times |G|_B$ (resp. $|G|_A = k \times |G|_B$). Note that in this case the k-covering argument fails.

References

[1] D. Angluin, *Local and global properties in networks of processors*, Proc. 12th ACM STOC (1980), 82–93.

[2] B. Awerbuch, A.V. Goldberg, M. Luby and S.A. Plotkin, *Network decomposition and locality in distributed computation*, Proc. 30th IEEE Symp. on Foundations of Computer Science (1989), 364–369.

[3] M. Billaud, P. Lafon, Y. Métivier and E. Sopena, *Graph rewriting systems with priorities*, Lecture Notes in Comput. Sci. **411** (1989), 94–106.

[4] B. Courcelle and Y. Métivier, *Coverings and minors : application to local computations in graphs*, Europ. J. Combinatorics **15** (1994), 127–138.

[5] R.E. Johnson and F.B. Schneider, *Symmetry and similarity in distributed systems*, Proc. 4th Symp. on Principles of Distributed Computing (1985), 13–22.

[6] N. Linial, *Locality in distributed graph algorithms*, Siam J. Comput. **21-1** (1992), 193–201.

[7] I. Litovsky and Y. Métivier, *Computing with graph rewriting systems with priorities*, Theoret. Comput. Sci. **115** (1993), 191–224.

[8] I. Litovsky, Y. Métivier and E. Sopena, *Different local controls for graph relabelling systems*, Math. System Theory **28** (1995), 41–65.

[9] I. Litovsky, Y. Métivier and W. Zielonka, *The power and limitations of local computations on graphs and networks*, Lecture Notes in Comput. Sci. **657** (1993), 333–345.

[10] M. Naor and L. Stockmeyer, *What can be computed locally ?*, Proc. 25th ACM STOC (1993), 184–193.