Implementation of an Enumeration Protocol Algorithm Using Local Graph Computations *

Mohamed Mosbah and Afif Sellami
LaBRI
Université Bordeaux 1-ENSEIRB
351 Cours de la Libération
33405 Talence Cdx, France

{mosbah|sellami@labri.fr}

Abstract

We present an implementation of Mazurkiewicz algorithm based on local graph transformations. Through this case study, we present a methodology for modeling distributed algorithms coded by graph relabelling systems. We obtain a simple and efficient implementation that makes the algorithm easier to understand. We also provide an experimental evaluation by implementing it in ViSiDiA, a tool for simulating and visualizing distributed algorithms.

1 Introduction

Mazurkiewicz algorithm [6] is one of the recent useful distributed algorithms, as it gives a solution to a family of important problems in distributed computing, including enumeration protocol, leader election, mutual exclusion, termination etc. In particular, it has been used by Métivier and Tel in [8] to characterize a large class of networks for which termination of a distributed computation can be detected. We propose to study and to implement this algorithm by using an approach based on graph relabelling systems.

The model of distributed computation is an asynchronous distributed network of anonymous processors with an arbitrary topology; processors communicate by exchanging messages. The network is represented as a connected graph where vertices represent processors, and two vertices are connected by an edge if the corresponding processors have a direct communication link. A distributed algorithm is encoded by means of a local relabelling system. Labels attached to vertices and edges are modified locally, that is on a subgraph of fixed radius 1 of the given graph, according to certain rules depending only on the subgraph. The relabelling is performed until no more transformation is possible [5, 3]. Examples of distributed algorithms encoded by means of local computations include spanning tree, leader election, detection of stable properties (see [3]).

In this paper, we deal with Mazurkiewicz algorithm[6], which is a distributed algorithm for enumerating nodes in an anonymous *covering-minimal* graph when its size is known. A distributed enumeration algorithm on a graph G is a distributed algorithm that terminates and gives as a result a proper enumeration of vertices; that is, each vertex gets a unique label from the set $\{1, 2, \ldots, n\}$, where n is the number of vertices. This algorithm can be described by a relabelling system, where the rule is some star relabelling; that is, depending on the label of a vertex v and of its neighbours, change the label of v and those of its neighbours. We exploit this encoding to derive an implementation written in ViSiDiA. Since this tool allows the user to visualize the execution of a distributed algorithm, it is possible to observe the animation of this algorithm. Although it seems at first a hard algorithm, its visualization makes it easy to understand.

*This work has been supported by the TMR research network GETGRATS and by the "Conseil Régional d'Aquitaine"

The paper is organized as follows. Section 2 recalls Mazurkiewicz algorithm in its mathematical description. Its implementation is given in Section 3. Experiments of this algorithm are presented in Section 4.

2 Mazurkiewicz algorithm

We say that a graph G is a *covering* of a graph H if there exists a surjective homomorphism γ from G onto H such that for every vertex v of $V(G)$ the restriction of γ to $N_G(v)$ is a bijection onto $N_H(\gamma(v))$. In particular, $\{\gamma(u), \gamma(v)\} \in E(H)$ implies $\{u, v\} \in E(G)$. The covering is proper if G and H are not isomorphic. It is called connected if G (and thus also H) is connected. A graph G is called *covering-minimal* if every covering from G to some H is a bijection.
Mazurkiewicz algorithm is a distributed algorithm for enumerating nodes in an anonymous covering-minimal graph when we know its size. A distributed enumeration algorithm on a graph G is a distributed algorithm that assigns a unique number to each vertex. An application of a transformation rule is done on a subgraph defined by a vertex v and the set of vertices adjacent to v; the application of a rule :

- depends on the label of v and the labels of nodes adjacent to v,
- it changes the label of v and the labels of nodes adjacent to v.

Let G be a graph, to each node is associated a triple (p, N, M) where p is a non-negative integer, N is a finite set of positive integers, and M is a set of couple of the form (p, N).
Let (p, N, M) be the triple associated to the vertex v. The set $K(v)$ denotes the set of integers used on the graph and known by v; more precisely :

$$K(v) = \{p\} \cup N \cup \{p_i | (p_i, N_i) \in M\} \cup \bigcup_{(p_i, N_i) \in M} N_i.$$

We need the following order relation, let N_1 and N_2 two finite sets of non-negative integers :

$$N_1 \prec N_2 \qquad if \qquad Max(N_1 \setminus N_2) < Max(N_2 \setminus N_1).$$

Initially, each triple associated to a vertex of G is equal to $(0, \emptyset, \emptyset)$. Each local computation acts on a vertex and neighbours of this vertex. Let v_0 be a vertex and let $\{v_1, ..., v_d\}$ the set of vertices adjacent to v_0. Let (p_i, N_i, M_i) be the triple associated to the node v_i.

1. If there exist i, j such that $0 \leq i < j \leq d$ and $M_i \neq M_j$ then let :

   $$M = \bigcup_{k=0}^{k=d} M_k$$

   for each $0 \leq i \leq d \quad M_i := M$.

2. We assume that for all i, j such that $0 \leq i < j \leq d$: $\quad M_i = M_j = M$.
   if $(p_0 = 0)$
   or
   if there exists $(p, N) \in M$ such that $p_0 = p$ and $N_0 \prec N$
   then let :

   $$k = Max(\bigcup_{i=0}^{i=d} K(v_i))$$

   for all $0 \leq i \leq d$ add $(k+1, N_0)$ to M_i and for all $0 < i \leq d$ replace p_0 by $k+1$ in N_i and finally replace p_0 by $k+1$ in the first coordinate of the triple associated to v_0.

Let G be a graph, the labelling function obtained after a run of Mazurkiewicz algorithm is denoted by Π_G. If v is a vertex of G, the triple $\Pi_G(v)$ associated to v is denoted $(\pi_1(v), \pi_2(v), \pi_3(v))$.

Thus in the case of covering-minimal graphs, after a run of Mazurkiewicz algorithm, π_1 is a one-to-one correspondence between the set of vertices of G and the set of integers $[1, \cdots, |V(G)|]$.

In fact Mazurkiewicz algorithm computes a set S of labelled stars that allows a graph to be reconstructed.

More precisely, let $(\mathbf{G}_i)_{0\leq i}$ be a relabelling chain associated to a run of Mazurkiewicz algorithm on a covering-minimal graph of size n; let v be a node the triple associated to the node v in $\mathbf{G}_i$ is denoted by $(\pi_{1,i}(v), \pi_{2,i}(v), \pi_{3,i}(v))$.

Let n be the size of G and p an integer such that $1 \leq p \leq n$; we define $S_i(v, p)$ as the labelled star having p as center and known by v in $\mathbf{G}_i$, i.e. : $S_i(v, p) = (p, N) \in \pi_{3,i}(v)$ and if for some $N' \neq N$ $(p, N') \in \pi_{3,i}(v)$ then $N' \prec N$.

Let :

$$S_i(v) = \{S_i(v, p) \mid 1 \leq p \leq n\}.$$

We have:

Proposition

Let G be a graph and let $(\mathbf{G}_i)_{0\leq i}$ be a relabelling chain obtained from G; let i be a positive integer and let v a vertex of G.

If $S_i(v)$ enables the reconstruction of a graph H then G is a quasi-covering of H. Furthermore if $\mathbf{G}_i$ is irreducible then G is a covering of H.

3 Implementation

We use our general approach [1, 3] to implement a distributed algorithm by local computations. The first step is to choose a type of randomized local computations, the second is to define the rules to apply (relabelling of vertices and attribute updates). In other words, each vertex tries randomly to synchronize with its neighbours, and once it does, rules are applied. It has been proved in [7] that randomized local computations are useful to implement distributed algorithms. The primitive **starSynchro()** implements this synchronization and returns the center of a star. Note that, since each processor executes the same algorithm (see below), the synchronization allows each processor to exchange data and labels with its neighbours. In the case of Mazurkiewicz algorithm, the data are mailboxes containing dynamic enumeration of vertices. During a synchronization, the processor being the center of the star, collects all mailboxes of its neighbours, updates its attributes by choosing a *right* number, broadcasts this new number to its neighbours which in turns update their attributes, and finally breaks the synchronization. Algorithm 1 gives a java like program to implement Mazurkiewicz algorithm:

Let us mention that, we use high level primitives to program this algorithm. It can be implemented on any system which provides these functions. We have perfomed experiments on ViSiDiA [2], a tool that we have written for testing distributed algorithms. A graphical user interface allows the end-user to draw a network, and to execute the algorithm. The visual animation of the algorithm can be displayed on the screen (see Figure 1). Particularly, Figure 1 (c) shows the result of the algorithm. The enumeration is given by the second integer.

Compared to the Estelle implementation given in [4], our implementation seems easier to understand and follows a more general approach.

4 Experiments

We made experiments of this algorithm on many classes of graphs. We compute the number of messages and the number of synchronizations. We present in this paper results for the number of messages (Figure 3) and for the number of synchronizations (Figure 2) for grids, rings, trees and other general random graphs.

Acknowledgements We thank Yves Métivier for fruitful discussions on Mazurkiewicz algorithm.

Algorithm 1 Implementation of Mazurkiewicz algorithm

```
while (run) {
 if (endDetected)
   run=false;
 else {
   synchro=starSynchro(); // get a synchronization with neighbours
   if (starCenter) {          // the center executes this part
       ReceiveMailBoxes();
       if ( (myName()==0) ||
          (existMailboxeStronger()))
           changeName(maxName+1);
       sendNewNameToNeigbours();
       broadcastMailboxes();
       breakSynchro();

   }
   else {
           if (neighboursCenter) {  // the neighbours of a center execute this part
               sendMailboxesTo(center);
               receiveNewNameFrom(center);
               changeNeighbours();
               ReceiveMailboxesFrom(center);
           }
   }
 }
}
```

References

[1] M. Bauderon, S. Gruner, Y. Métivier, M. Mosbah, and A. Sellami. Visualization of distributed algorithms based on labeled rewriting systems. In *Second International Workshop on Graph Transformation and Visual Modeling Techniques, Crete, Greece*, July 12-13, 2001.

[2] M. Bauderon, S. Gruner, and M. Mosbah. A new tool for the simulation and visualization of distributed algorithms. Technical Report 1245-00, LaBRI, 2000. Accepted in MFI'01, Toulouse, 21-23 May 2001.

[3] M. Bauderon, Y. Métivier, M. Mosbah, and A. Sellami. From local computations to asynchronous message passing systems. Technical report, LaBRI, 2001. In preparation.

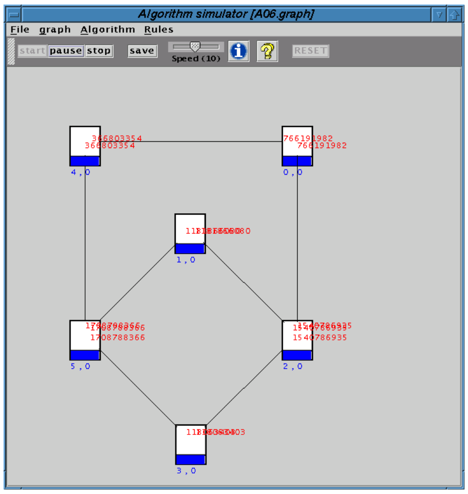
[4] P. Dembinski. Enumeration protocol in estelle: an exercise in stepwise development. In A. Cavalli S. Budkowski and E. Najm, editors, *Formal description techniques and protocol specification, testing and verification*, pages 147–161. Kluwer Academic Publishers, 1998.

[5] I. Litovsky, Y. Métivier, and E. Sopena. Graph relabelling systems and distributed algorithms. In H. Ehrig, H.J. Kreowski, U. Montanari, and G. Rozenberg, editors, *Handbook of graph grammars and computing by graph transformation*, volume 3, pages 1–56. World Scientific, 1999.

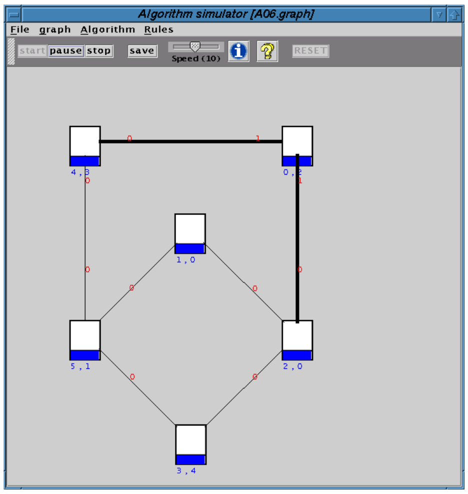
[6] A. Mazurkiewicz. Distributed enumeration. *Inf. Processing Letters*, 61:233–239, 1997.

[7] Y. Métivier, N. Saheb, and A. Zemmari. Randomized local elections : probabilistic and efficiency analysis. Technical report, LaBRI, en préparation.

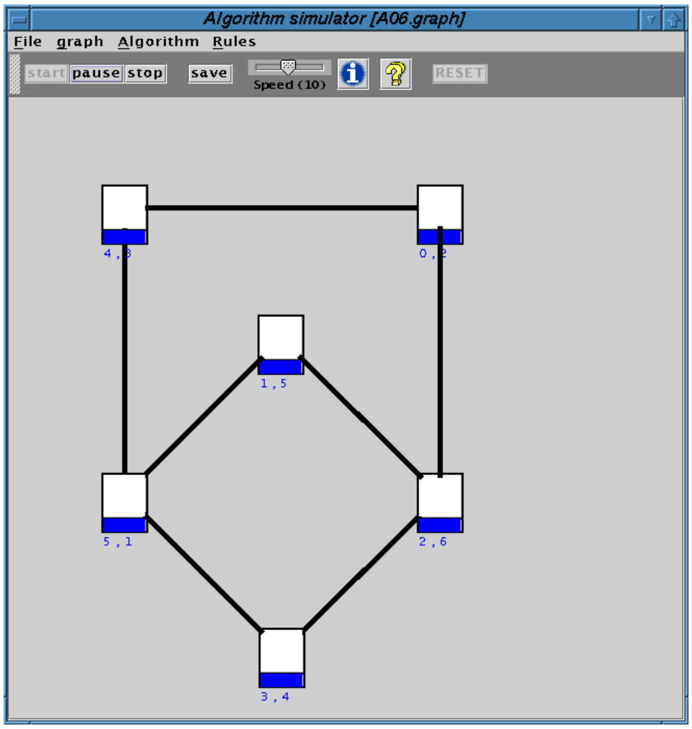
[8] Y. Métivier and G. Tel. Termination detection and universal graph reconstruction. In *International Colloquium on structural information and communication complexity*, pages 237–251, 2000.



(a) Initial step before the execution



(b) During the execution



(c) Result of the execution

Figure 1: Visualization of Mazurkiewicz algorithm

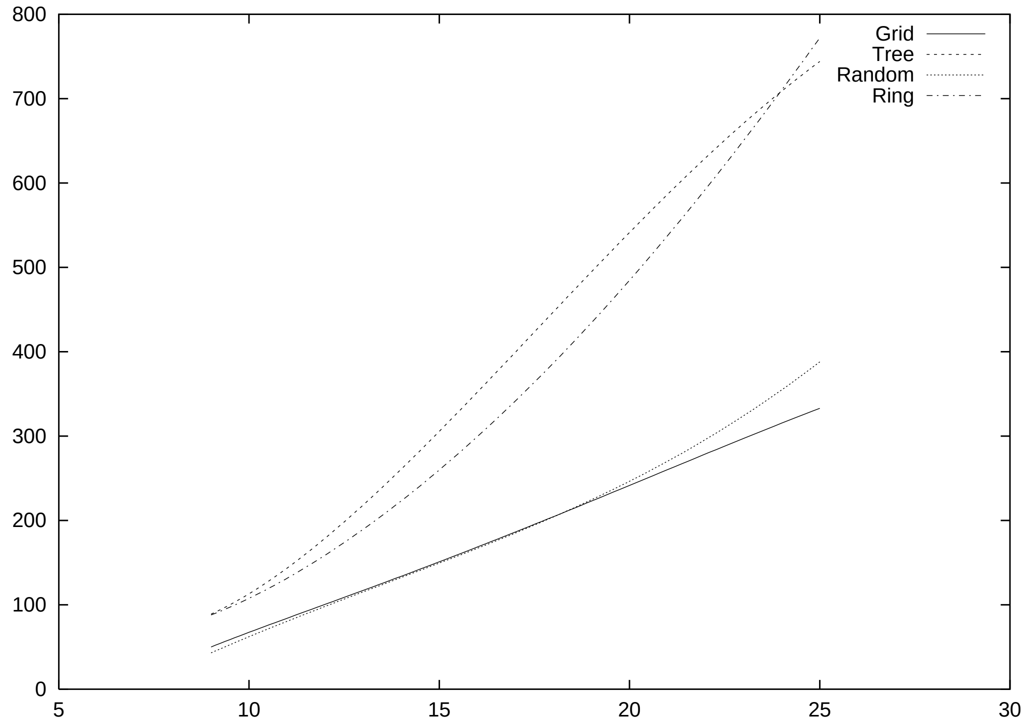


Figure 2: Experimental results for the number of synchronizations. X-axis indicates the number of vertices of the graph.

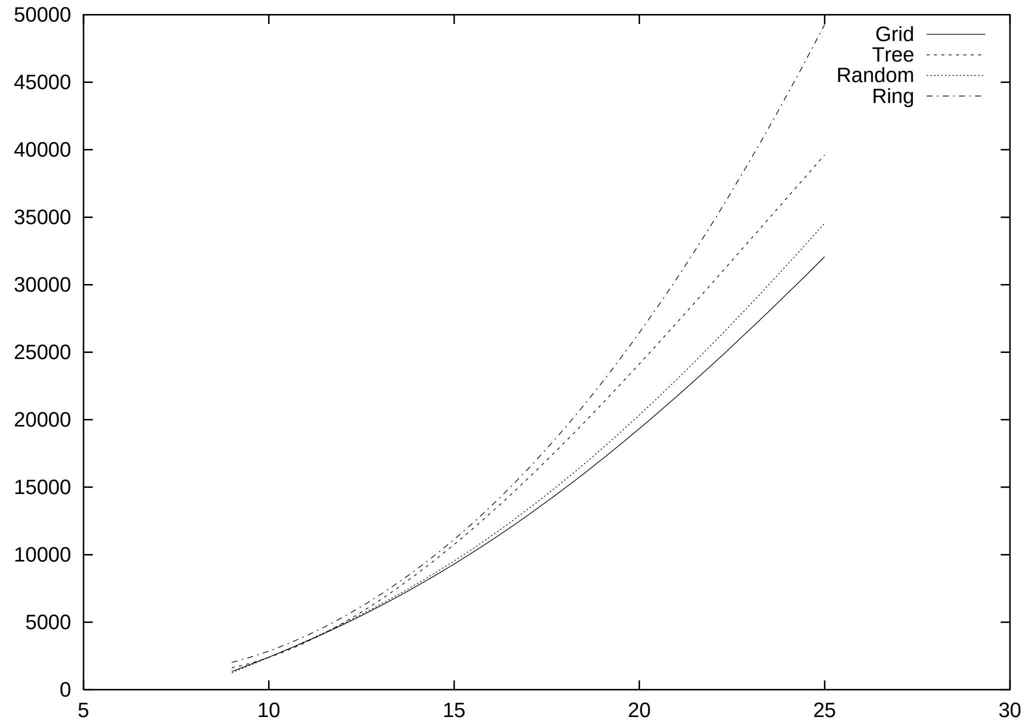


Figure 3: Experimental results for the number of messages. X-axis indicates the number of vertices of the graph.