ELSEVIER

Information Processing Letters 82 (2002) 313–320

Information Processing Letters

www.elsevier.com/locate/ipl

Randomized local elections

Yves Métivier *, Nasser Saheb, Akka Zemmari

LaBRI, Université Bordeaux I – ENSEIRB, 351 Cours de la Libération, 33405 Talence Cedex, France

Received 21 June 2001; received in revised form 11 September 2001
Communicated by L. Boasson

Abstract

We propose and analyze two randomized local election algorithms in an asynchronous anonymous graph. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Analysis of algorithm; Graph; Performance analysis; Local election; Randomized algorithm

1. Introduction

The problem of election is linked to distributed computations in a network. It aims to choose a unique vertex, called leader, which subsequently is used to make decisions or to centralize some information (see [8]). In this paper we introduce and study randomized *local* elections. For a fixed given positive integer k, a k-local election problem requires that, starting from a configuration where each process is in the same state, the network reaches a configuration $\mathcal{C}$ such that for this configuration there exists a nonempty set of vertices, denoted $\mathcal{E}$, verifying:

- each vertex v of $\mathcal{E}$ is in a special state called *leader* and
- for each vertex $v \in \mathcal{E}$ and for each vertex w ($w \neq v$) such that $d(v, w) \leqslant k$ then w is in the state *lost* (i.e., $w \notin \mathcal{E}$).

As for the election problem, we assume that each process has the same local algorithm. This problem is considered under the following assumptions:

- the network is anonymous: unique identities are not available to distinguish the processes,
- the system is asynchronous: processes have no access to a common clock,
- processes communicate by asynchronous message passing: a process sends a message to another by depositing the message in the corresponding channel and there is no fixed upper bound on how long it takes for the message to be delivered,
- each process knows from which channel it receives a message.

The study of the local election problem is motivated by the implementation of local computations [2]. We consider a network of processors with arbitrary topology. It is represented as a connected, undirected graph where vertices denote processors and edges direct communication links. A distributed algorithm is encoded by means of local relabeling: labels attached to vertices and edges are modified locally, that is on a bounded subgraph of the given graph according to certain rules depending on the subgraph only. The

* Corresponding author.
E-mail addresses: metivier@labri.u-bordeaux.fr (Y. Métivier), saheb@labri.u-bordeaux.fr (N. Saheb), zemmari@labri.u-bordeaux.fr (A. Zemmari).

0020-0190/01/$ – see front matter
PII: S0020-0190(01)00289-7

relabeling is performed until no more transformation is possible. The corresponding configuration is said to be in normal form. We consider more particularly two kinds of local computations:

LC_1: in a computation step, the label attached to the centre of a ball of radius 1 is modified according to some rules depending on the labels in the ball, labels of the other vertices in the ball are not modified.

LC_2: in a computation step, labels attached to the vertices in a ball of radius 1 may be modified according to some rules depending on the labels in the ball.

The implementation of LC_1 (respectively LC_2) may be done by using 1-local election (respectively 2-local election). Once a vertex v is locally elected a local computation can be done on the ball centered on v.

From the work of Angluin [1], we deduce that, under the above assumptions, there is no deterministic algorithm to solve the k-local election problem for $k \geqslant 1$. Thus we have no choice but to consider randomized algorithms. From [1] we deduce also that there is no Las Vegas algorithm to solve the k-local election problem for $k \geqslant 3$.

Here we propose and analyze two Las Vegas algorithms to realize a k-local election for $k = 1, 2$. Although they are *totally distributed* their analysis is based on the consideration of *rounds*: in order to measure the performance of the algorithm in terms of the number of local elections taking place, we assume that at some instant each node sends and then receives messages. Thus this parameter of interest, which is the (random) number of local elections, is the maximal number authorized by the algorithm. Except for the case of simple graphs, modeling the network, a simple characterization of the distribution of this number does not seem possible. Simple expressions for the probability of being elected for vertices are available. Let v be a vertex such that a local computation may be done on $B(v, k)$; then the probability of being elected for v corresponds to the probability that a local computation is done in $B(v, k)$. We use the expected number of elected vertices in a round to define the efficiency of the algorithms. This parameter can roughly be interpreted as the expected number of local computations taking place simultaneously and, thus, can be considered as the degree of parallelism authorized by the algorithms. In both algorithms, we provide a lower bound for the expected number of locally elected vertices. The efficiency of each randomized algorithm is introduced in terms of this parameter. We get finally lower bounds for the efficiency of our randomized election algorithms applied to trees.

Many problems have no solution in distributed computing [5]. The introduction of randomization makes possible tasks that admit no deterministic solutions or simplifies algorithms. General considerations about randomized distributed algorithms may be found in [8] and some techniques used in the design and analysis of randomized algorithms are presented in [4,6].

The paper is organized as follows. In Section 2 we give definitions, notation and general remarks. We present also two randomized algorithms to solve 1-local and 2-local elections denoted RL_1 and RL_2. Section 3 is devoted to the study of the expected number of local elections realized by RL_1 and RL_2. In Section 3 we study their efficiencies; lower bounds are obtained over trees.

2. Definitions, notation and general remarks

We use a standard terminology of discrete and combinatorial mathematics [7]. A simple graph $G = (V, E)$ is defined as a finite set V of vertices together with a set E of edges which is a set of pairs of different vertices,

$$E \subseteq \big\{\{v, v'\} \mid v, v' \in V,\ v \neq v'\big\}.$$

If $e = \{v, v'\} \in E$ we say that e is incident with v, and v and v' are neighbors. Let $N_G(v)$ denote the set of neighbors of v. The degree $d_G(v)$ of the vertex v is the cardinal of $N_G(v)$. For a nonnegative integer r and a vertex v in G, $B_G(v, r)$ denotes the ball of centre v and radius r. It can be defined by induction over r as follows. $B_G(v, 0) = \{v\}$, and

$$B_G(v, r+1) = B_G(v, r) \cup \bigcup_{w \in B_G(v,r)} N_G(w).$$

We drop the subscript G from N_G, d_G, B_G whenever the graph is understood in the context.

2.1. *Randomized* 1- *and* 2-*local elections*

Randomization algorithms use random functions or random-number generators. Algorithms which do not use such function are referred to as deterministic algorithms. A Las Vegas algorithm is an algorithm that always produces correct output, whose running time is a random variable. We introduce in this subsection two randomized procedures: RL_1 and RL_2 to solve 1-local election and 2-local election we will study later. In the sequel, K will denote a nonempty set equipped with a total order.

RL_1: Randomized 1-local election

Each vertex v repeats forever the following actions. The vertex v selects an element $rand(v)$ randomly and uniformly from the set K. The vertex v sends to its neighbors the value $rand(v)$. The vertex v is elected in $B(v,1)$ if for each neighbor w of v: $rand(v) > rand(w)$.

RL_2: Randomized 2-local election

Each vertex v repeats forever the following actions. The vertex v selects an element $rand(v)$ randomly and uniformly from the set K. The vertex v sends to its neighbors the value $rand(v)$. When it has received from each neighbor an element, it sends to each neighbor w the maximum of the set of elements it has received from neighbors different from w. The vertex v is elected in $B(v,2)$ if $rand(v)$ is strictly greater than $rand(w)$ for any vertex w in the ball centred on v of radius 2.

Remark 1. The network is anonymous but each process knows exactly from which channel it can receive a message and thus, in some sense, it has a local knowledge of its neighbors.

Remark 2. The aim of RL_1 and RL_2 is the implementation of local computations. Thus RL_1 and RL_2 contain infinite loops: the global termination depends on the termination of the algorithm encoded by local computations.

2.2. *Impossibility results*

As indicated by Tel [8, pp. 316–317], the impossibility of a deterministic election algorithm in a network of two processes that communicate by asynchronous message passing follows from the observation that, in a such system, from a symmetric configuration it is always possible to reach a symmetric configuration. Thus the algorithm does not terminate. From this fact we deduce there does not exist deterministic k-local election for $k \geqslant 1$.

A graph G is a covering of a graph H if there is a surjective homomorphism γ from G onto H such that for every vertex v of G the restriction of γ to $N_G(v)$ is a bijection onto $N_H(\gamma(v))$. In [1, Theorem 4.5] Angluin proves that if G is a proper covering of H then there is no election algorithm for both G and H. This results holds for a deterministic election algorithm and for a Las Vegas algorithm. If we consider the case where G is the hexagon and H the triangle we deduce that there is no k-local election algorithm (deterministic or Las Vegas) for $k \geqslant 3$.

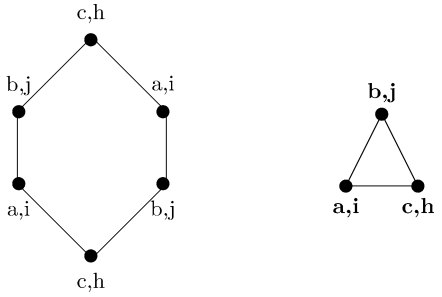
In algorithms RL_1 and RL_2, a vertex v detects if another vertex w ($w \neq v$) has selected the same element in $B(v,1)$ or in $B(v,2)$ ($rand(w) = rand(v)$): this is termed a *coincidence*. The problem of detecting coincidence is weaker than the local election problem. We prove in the next proposition it is not possible to detect coincidence in $B(v,k)$ for $k > 2$.

First we recall our model. In each step, a vertex, depending on its current state, either changes its state, sends a message to a neighbor, or receives a message from a neighbor. It is assumed that graphs are asynchronous and anonymous. Each process has the same local algorithm. Inspired by proofs of [1], we can state:

Proposition 1. *There is no deterministic or Las Vegas algorithm to detect if there is a coincidence in a ball of radius k for $k > 2$.*

Proof. By contradiction, we assume there exists an algorithm. We consider an hexagon G and a triangle H with couples of labels on vertices as indicated in Fig. 1: a, b, c are 3 different labels of a set different from K and i, j, h are 3 different elements of K. We consider the morphism φ from G onto H which maps vertices in G of label x, y into the vertex of label x, y in H.

We assume that each vertex having the label x, y selects the element y of K. Thus in G each element of K appears twice (there are coincidences) and in H



The labelled hexagon G The labelled triangle H

Fig. 1. The graph G is a covering of the graph H.

each element appears once (there is no coincidence). We define a sequence of steps in G, denoted by $(G_i)_{i\geqslant 0}$, and in H by $(H_i)_{i\geqslant 0}$, with the property that the states of the vertices labeled x, y in G_i are the same as the state of the vertex of label x, y in H_i, let $\mathcal{R}$ be this property.

We define inductively $(G_i)_{i\geqslant 0}$ and $(H_i)_{i\geqslant 0}$. Graphs are anonymous and thus initially all vertices in G and in H are in the same state q_0. Let G_0 and H_0 be the graphs G and H with this initial state. Clearly, G_0 and H_0 verify $\mathcal{R}$. Let G_i and H_i be the graphs G and H with states verifying $\mathcal{R}$. Let P be a step in H_i.

- If P is a communication over the edge e, then this communication is possible over the two edges of $\varphi^{-1}(e)$. Perform both communications in G_i, and then obtain H_{i+1} and G_{i+1} which verify $\mathcal{R}$.
- If during P the vertex x, y in H_i changes its state (the new state of x, y is q), then the two vertices in $\varphi^{-1}(x, y)$ change their states; the new state of vertices in $\varphi^{-1}(x, y)$ is q. Thus we have proved that H_{i+1} and G_{i+1} verify $\mathcal{R}$.

Now, if a vertex v of H has a state which indicates that there is no coincidence then vertices of $\varphi^{-1}(v)$ will be labeled by a state which indicates that there is no coincidence. Hence we get a contradiction and the proposition is proved. □

3. Probabilistic analysis of RL_1 and RL_2

We first introduce and study the probability distribution for vertices to be locally elected in a round for each one of the algorithms. We then investigate a uniform lower bound for the expected number of locally elected vertices for each of them.

Procedures RL_1 and RL_2 are used for the implementation of local computations. The first important property we require is: if a local computation may be done, does a local computation will be done somewhere. Thus we study the probability for vertices to be locally elected in a round and the expected number of rounds for a given vertex to be elected.

From now on, we assume that $K = [0, 1]$.

3.1. Probability for a vertex to be locally elected

In order to compute the probability for a vertex to be locally elected according to RL_1 or RL_2, we recall shortly their probabilistic nature. Each vertex $v \in V$ selects at random *uniformly* and *independently* an element $rand(v)$ from K. Then, according to RL_1 (respectively RL_2), the vertex v will be locally elected if for all $w \in B(v, 1) \setminus \{v\}$ (respectively $w \in B(v, 2) \setminus \{v\}$), we have $rand(w) < rand(v)$.

All vertices choose at random uniformly and independently a *real* from the interval $[0, 1]$, under this assumption we have the following facts.

Fact 1. *The probability for a vertex v to be locally elected in an RL_1 round is given by*

$$p_1(v) = \frac{1}{d(v)+1}.$$

Recall that $d(v) + 1$ is the cardinal of $B(v, 1)$.

For a given vertex v we denote by $N_2(v)$ the number of vertices of distance less than or equal to 2 from v (v included). $N_2(v)$ is therefore the cardinality of the ball $B(v, 2)$ of centre v and radius 2.

Fact 2. *The probability for v to be locally elected in an RL_2 round is*

$$p_2(v) = \frac{1}{N_2(v)}.$$

From these two facts we derive:

Fact 3. *Let v be a vertex of G. The expected time (number of rounds) between two successive elections of v, in RL_1 (respectively RL_2), is given by $d(v) + 1$ (respectively $N_2(v)$).*

Consider now the following classes of graphs.

Example 1. If G is a cycle graph of size n, then for any vertex v, we have

$$p_1(v) = \tfrac{1}{3} \quad \text{and} \quad p_2(v) = \tfrac{1}{5}.$$

The expected number of rounds for v to be elected is 3 in RL_1 and 5 in RL_2.

Example 2. If G is a complete graph of size n, then for any vertex v, we have

$$p_1(v) = p_2(v) = \frac{1}{n}.$$

The expected number of rounds for v to be elected is n in both RL_1 and RL_2.

Remark 3. Procedures RL_1 and RL_2 can be modified to stop as soon as a vertex is locally elected (by sending a stop signal), let RL_1' and RL_2' be the new procedures obtained in this way. From Fact 2 and from the detection of coincidence, we deduce that RL_1' and RL_2' are Las Vegas algorithms solving the 1-local election problem and the 2-local election problem.

Remark 4. If we assume that each vertex $v \in V$ selects at random *uniformly* and *independently* an integer $rand(v)$ from $\{1, \ldots, N\}$. Let $X \subseteq V$ be a set of vertices containing a given vertex v. Let $|X| = h$. Then under the above assumptions on $rand$,

$$\Pr\big(rand(v) > rand(w),\ \forall w \in X \setminus \{v\}\big) = \frac{1}{N}\sum_{i=2}^{N}\left(\frac{i-1}{N}\right)^{h-1}. \tag{1}$$

We obtain the expressions for RL_1 or RL_2 by considering balls of radius 1 or 2.

In order to simplify this expression involved in the probability for a vertex to be elected we have assumed that the integer N, which is the range of selection for vertices, is large, so that probability of coincidence of $rand$ in $B(v, 1)$ or in $B(v, 2)$ becomes small. This assumption is equivalent to the one supposing that all vertices choose at random uniformly and independently a *real* from the interval $[0, 1]$.

3.2. Expected number of vertices locally elected

A parameter of interest is the expected number of vertices locally elected. In both algorithms this expected number is the average number of actions taking place simultaneously in a round.

Let $\overline{M_1}(G)$ (respectively $\overline{M_2}(G)$) denote the expected number of vertices locally elected by RL_1 (respectively RL_2) in the graph G. Then, summing $p_i(v)$, $i = 1, 2$, over all $v \in V$, in the previous facts, we get:

Fact 4. *We have*

$$\overline{M_1}(G) = \sum_{v \in V} \frac{1}{d(v)+1},$$

and

$$\overline{M_2}(G) = \sum_{v \in V} \frac{1}{N_2(v)}.$$

Let us consider again the simple class of cycle graphs and that of complete graphs.

Example 3. If G is a cycle graph of size $n \geqslant 3$, then we have

$$\overline{M_1}(G) = \frac{n}{3} \quad \text{and} \quad \overline{M_2}(G) = \frac{n}{5}.$$

Example 4. If G is a complete graph then we have

$$\overline{M_1}(G) = \overline{M_2}(G) = 1.$$

For a given positive integer k, we define the *k-density* of a graph $G = (V, E)$ by

$$\mathcal{D}_k(G) = \frac{\sum_{v \in V} d(v)^k}{|V|}.$$

It should be noted that this definition is slightly different from the usual one, see for instance [7]. In particular, for $k = 1$, we have

$$\mathcal{D}_1(G) = \frac{\sum_{v \in V} d(v)}{|V|} = 2\frac{|E|}{|V|}.$$

This shows that the 1-density of a graph is twice the ratio between the number of edges and that of vertices.

We end the section by some results leading to uniform lower bounds on the mathematical expectation of the number of vertices locally elected in a round of each algorithm.

Lemma 1. *Let $G = (V, E)$ be a connected graph. We have*
- $\sum_{v\in V} N_2(v) \leqslant \sum_{v\in V} d(v)^2 + |V|$.
- *Moreover if G is a tree then the above inequality becomes an equality.*

Proof. Let $G = (V, E)$ be a graph. We start by proving the second assertion of the lemma. So, let G be a tree. The proof is by induction on the size n of the tree G. For $n = 2$, a simple verification yields the result. Suppose the assertion is true for any tree of size $n \geqslant 2$. Let $T = (V, E)$ be a tree of size $n + 1$ and u a leaf of T and $T' = (V', E')$ the tree obtained from T by deleting u and the unique incident edge. Let δ denote the degree of the father of u in T'. By denoting the number of vertices of distance less than or equal to 2 from v by $N'_2(v)$ in T', we have

$$\begin{aligned}\sum_{v\in V} N_2(v) &= \sum_{v\in V'} N'_2(v) + 2\delta + 3\\ &= \sum_{v\in V'} d(v)^2 - 2(\delta+1) + n + 2\delta + 3\\ &= \sum_{v\in V} d(v)^2 + n + 1.\end{aligned}$$

If the connected graph G is not a tree, we use induction on the cardinality m of the set of edges E. Since the claim holds for a spanning tree in G, to this end, it suffices to show that if we add a non-existing edge $\{u, w\}$ to the graph $G = (V, E)$, then its total contribution to the sum $S_1 = \sum_{v\in V} N_2(v)$ is not greater than its contribution to the sum $S_2 = \sum_{v\in V} d(v)^2$. For S_1 we have clearly $S_1 \leqslant 2d(u) + 2d(w) + 2$. For S_2 we can write $S_2 = 2d(u) + 1 + 2d(w) + 1$. This ends the proof of the lemma. □

Back to the expectations $\overline{M_1}(G)$ and $\overline{M_2}(G)$, defined above, Lemma 1 can be used to show the following theorem.

Theorem 1. *Let $G = (V, E)$ be a connected graph of size $n \geqslant 2$ with $m = |E|$ edges. Then we have*

$$\overline{M_1}(G) \geqslant \frac{n}{\mathcal{D}_1(G)+1} = \frac{n^2}{2m+n},$$

and

$$\overline{M_2}(G) \geqslant \frac{n}{\mathcal{D}_2(G)+1}.$$

Proof. For the given graph G, a simple computation yields

$$\sum_{v\in V}\big[d(v)+1\big] = n\mathcal{D}_1(G) + n.$$

Hence

$$\frac{1}{n}\sum_{v\in V}\big[d(v)+1\big] = \mathcal{D}_1(G) + 1.$$

On the other hand, we have

$$\frac{1}{n}\sum_{v\in V}\big[d(v)+1\big] \geqslant \Big(\prod_{v\in V}\big[d(v)+1\big]\Big)^{1/n},$$

and therefore

$$\Big(\prod_{v\in V}\frac{1}{d(v)+1}\Big)^{1/n} \geqslant \frac{1}{\mathcal{D}_1(G)+1}.$$

But we have

$$\Big(\prod_{v\in V}\frac{1}{(d(v)+1)}\Big)^{1/n} \leqslant \frac{1}{n}\sum_{v\in V}\frac{1}{(d(v)+1)},$$

and hence

$$\overline{M_1}(G) = \frac{1}{n}\sum_{v\in V}\frac{1}{(d(v)+1)} \geqslant \frac{1}{\mathcal{D}_1(G)+1}.$$

This ends the proof of the first assertion of the theorem.

To prove the second assertion, by Lemma 1, and the definition of $\mathcal{D}_2(G)$, we have

$$\frac{1}{n}\sum_{v\in V} N_2(v) \leqslant \mathcal{D}_2(G) + 1.$$

Since $\frac{1}{n}\sum_{v\in V} N_2(v) \geqslant (\prod_{v\in V} N_2(v))^{1/n}$, we have

$$\mathcal{D}_2(G) + 1 \geqslant \Big(\prod_{v\in V} N_2(v)\Big)^{1/n},$$

and therefore

$$\Big(\prod_{v\in V}\frac{1}{N_2(v)}\Big)^{1/n} \geqslant \frac{1}{\mathcal{D}_2(G)+1}.$$

Using again the fact that

$$\frac{1}{n}\sum_{v\in V}\frac{1}{N_2(v)} \geqslant \Big(\prod_{v\in V}\frac{1}{N_2(v)}\Big)^{1/n},$$

the second assertion follows. □

In the case of graphs with degrees bounded by the integer Θ the lower bounds for $\overline{M_1}(G)$ and $\overline{M_2}(G)$ have very simple expressions. Indeed, we have:

Corollary 1. *Let G be a graph whose vertices are of degrees less than or equal to Θ. Then*

$$\overline{M_1}(G) \geqslant \frac{n}{\Theta+1}$$

and

$$\overline{M_2}(G) \geqslant \frac{n}{\Theta^2+2\Theta+1}.$$

Also in the case of trees, the first assertion of the theorem reduces to a simple expression for the lower bound.

Corollary 2. *Let T be a tree of size $n \geqslant 2$. Then*

$$\overline{M_1}(T) \geqslant \frac{n^2}{3n-2} > \frac{n}{3}.$$

4. Performance analysis

Throughout this section $G = (V, E)$ is a connected graph. We denote by $\alpha(G)$ the maximal cardinality of a set of independent vertices contained in G, see [7]. The computation of this number for general graphs is NP-hard. In the same way, let $\beta(G)$ denote the maximal cardinality of a set of vertices of G of pairwise distance at least 3. This is the maximal number of pairwise disjoint balls of radius 1. These numbers are intimately linked to the contexts of RL_1 and RL_2. Let us call these types of local elections RL_1- and RL_2-*type* elections, respectively. We assume that in an RL_1-type election no two vertices of distance less than 2 can be elected simultaneously and that in RL_2-type election two elected vertices in a round must be of distance greater than or equal to 3.

Following definitions for approximation algorithms [3]: given *any* randomized algorithm $\mathcal{A}$ for an RL_1-type election, its *efficiency* $\Delta_{\mathcal{A}}(G)$ over a graph G is the ratio

$$\Delta_{\mathcal{A}}(G) = \frac{\overline{M}_{1\mathcal{A}}(G)}{\alpha(G)},$$

where $\overline{M}_{1\mathcal{A}}(G)$ is the expected number of vertices locally elected by a round of $\mathcal{A}$. In a similar way, the efficiency of a randomized algorithm $\mathcal{B}$ for an RL_2-type election over G is given by

$$\Lambda_{\mathcal{B}}(G) = \frac{\overline{M}_{2\mathcal{B}}(G)}{\beta(G)},$$

where $\overline{M}_{2\mathcal{B}}(G)$ is the expected number of vertices locally elected by a round of $\mathcal{B}$. Clearly the efficiencies are upper-bonded by 1. Their values are the ratio between the average simultaneity realized by the considered algorithms and the simultaneity by an idealistic algorithm (which generally cannot be designed in a *distributed* way).

We conclude the section by finding a lower bound on the efficiency of the RL_1 and RL_2 algorithms over the sparsely linked graphs of trees. The study seems extensible to more general classes of graphs.

Theorem 2. *The efficiency of RL_1 over any tree T is strictly greater than $\frac{1}{3}$.*

Proof. By Corollary 2, the numerator of the ratio defining the efficiency is strictly greater than $\frac{1}{3}n$ (for a tree of size $n \geqslant 2$) and clearly the denominator, α, is less than or equal to n. □

We now state the last theorem providing a slightly smaller lower bound for the efficiency of the algorithm RL_2 over the family of trees.

Theorem 3. *The efficiency of RL_2 over any tree $T = (V, E)$ is strictly greater than $\frac{1}{4}$.*

Proof. To prove the theorem we show that $\overline{M_2}(T) > \frac{1}{4}\beta(T)$. To prove this we use a technique similar to that of Theorem 2 of Section 2. We use an induction on the size of T. The theorem holds obviously for the star graphs. Let $\overline{M_2}(T) > \frac{1}{4}\beta(T)$ hold for trees T of size less than n and prove it for trees of size n. Suppose T be a tree of size n which is not a star. It is easy to consider in the sequel the tree T rooted with one of its vertices. Consider a vertex a of degree $d \geqslant 2$ whose neighbors except one, say b, are leaves. Let $d' = d(b) \geqslant 2$. Let S be the tree obtained from T by removing the vertex a and all neighboring leaves and incident edges. Since we have by induction $\overline{M_2}(S) > \frac{1}{4}\beta(S)$ and $\beta(T) \leqslant \beta(S)+1$, it suffices to prove that $\overline{M_2}(T) - \overline{M_2}(S) \geqslant \frac{1}{4}$. A simple computation on the effect of adjoining a and its sons

on the sum defining $\overline{M_2}$, allows to get a lower bound on the $\overline{M_2}(T) - \overline{M_2}(S)$. Indeed we have

$$\begin{aligned}\overline{M_2}(T) - \overline{M_2}(S) &\geqslant \frac{d-1}{d+1} + \frac{1}{d+d'} - \frac{1}{(d'+1)(d+d'+1)} \\ &\quad - \frac{d'-1}{d'(d'+1)}.\end{aligned}$$

Now taking into account the inequalities $d \geqslant 2$ and $d' \geqslant 2$, if we develop the last expression, a straightforward formal calculation allows to prove that $\overline{M_2}(T) - \overline{M_2}(S) - \frac{1}{4} > 0$. The theorem follows. □

Remark 5. It is possible to tighten the lower bound of the theorem at the cost of introducing a greater complexity. This can be done if we impose a sharper lower bound in estimating $\overline{M_2}(T) - \overline{M_2}(S)$. Moreover it seems that the lower bound $\frac{1}{3}$ holds also for the efficiency of RL_2 (acting on trees), but the authors do not know any proof for it.

Remark 6. In the case of general graphs both algorithms may have a low efficiency. Indeed, it is possible to prove that, for any $\varepsilon > 0$, there is a graph G such that $\Delta_{L_1}(G) < \varepsilon$. To prove this consider a graph G_n with $2n$ vertices $v_1, \ldots, v_n, u_1, \ldots, u_n$, constructed as follows. The set $\{u_1, \ldots, u_n\}$ forms a clique in G_n, and there is an edge $\{u_i, v_j\}$ for any $1 \leqslant i \leqslant n$ and any $1 \leqslant j \leqslant n$. Thus $v_1, \ldots, v_n$ are independent and $\alpha(G_n) = n$. On the other hand $M_1(G_n) = n/2n + n/(n+1)$. As, $n \to \infty$, $\Delta_{L_1}(G_n) \to 0$. A similar reasoning with a slight modification shows that $\Lambda_{L_2}(G_n) \to 0$ and cannot be lower-bounded by a positive real uniformly for all graphs.

Acknowledgements

The authors are grateful to André Raspaud for stimulating discussions and to the anonymous referees for constructive comments.

References

[1] D. Angluin, Local and global properties in networks of processors, in: Proceedings of the 12th Symposium on Theory of Computing, 1980, pp. 82–93.

[2] M. Bauderon, S. Gruner, Y. Métivier, M. Mosbah, A. Sellami, Visualization of distributed algorithms based on labelled graph relabelling systems, in: 2nd International Workshop on Graph Transformation and Visual Modeling Techniques, Electronic Notes in Theoret. Comput. Sci. 50 (3) (2001).

[3] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, McGraw-Hill, New York, 1990.

[4] C. Lavault, Évaluation des algorithmes distribués, Hermes, Paris, 1995.

[5] N. Lynch, A hundred impossibility proofs for distributed computing, in: Proceedings of the 8th Annual ACM Symposium on Principles of Distributed Computing, 1989, pp. 1–28.

[6] R. Motwani, P. Raghavan, Randomized Algorithms, Cambridge University Press, Cambridge, 1995.

[7] K.H. Rosen, Handbook of Discrete and Combinatorial Mathematics, CRC Press, 2000.

[8] G. Tel, Introduction to Distributed Algorithms, Cambridge University Press, Cambridge, 2000.